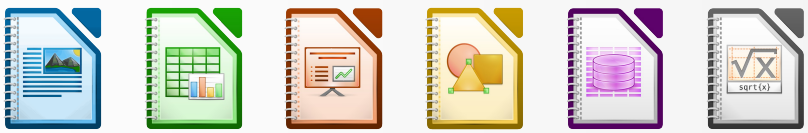
LibreOffice

Équipe documentation de LibreOffice

Guide Base

6.4



Droits d’auteur

Contributeurs

De cette édition

Pulkit Krishna
Dan Lewis
Jean-Pierre Ledure
Jean Hollis Weber
Jenna Sargent

Des éditions précédentes

Pulkit Krishna
Jean Hollis Weber
Dan Lewis
Peter Scholfield
Alain Romedenne
Jean-Pierre Ledure
Jochen Schiffers
Robert Großkopf
Jost Lange
Hazel Russman
Andrew Pitonyak
Jean Hollis Weber
Martin Fox
Randolph GAMO

Traduction

Jean-Michel COSTE

Relecteurs

Philippe Clément
Francis Lecher
Patrick Auclair

Retour d'information

Veuillez adresser tout commentaire ou suggestion concernant ce document à la liste de diffusion de l’Équipe de Documentation : doc@fr.libreoffice.org

Note :

Tout ce que vous envoyez à la liste de diffusion, y compris votre adresse mail et toute autre information personnelle incluse dans le message, est archivé publiquement et ne peut pas être effacé.

Date de publication et version du logiciel

Publié en juin 2021. Basé sur LibreOffice 6,4.

Table des matières

Guide Base

Préface

À qui est destiné ce document ?

Quiconque souhaite se familiariser rapidement avec LibreOffice Base trouvera ce livre précieux. Que vous n'ayez jamais travaillé avec des bases de données auparavant, ou que vous ayez travaillé avec elles dans un SGBD (système de gestion de base de données), ou que vous soyez habitué à un autre système de base de données à partir d'une suite bureautique ou d'un système de base de données autonome tel que MySQL, ce livre est pour vous. Vous souhaiterez peut-être commencer par lire le chapitre 8, Débuter avec Base, dans le *Guide de mise en route*.

Qu'y a-t-il dans ce document ?

Ce livre présente Base, le composant de base de données de LibreOffice. Base utilise le moteur de base de données HSQLDB pour créer des documents de base de données. Il peut accéder aux bases de données créées par de nombreux programmes de base de données, notamment Microsoft Access, MySQL, Oracle et PostgreSQL. Base comprend des fonctionnalités supplémentaires qui vous permettent de créer des applications entièrement basées sur les données.

Note

Un moteur Firebird intégré est également disponible. Il est disponible en v6.4.1, v6.4.2 et v6.4.3 même sans mode expérimental. Dans la v6.4.4, il a été déplacé vers des fonctionnalités expérimentales en raison du grand nombre de bogues qu'il présentait. Son utilisation est déconseillée. Ce guide est basé sur le moteur HSQLDB et donne quelques exemples sur Firebird. Veuillez noter que le moteur Firebird est très bon, c'est son implémentation dans Base qui comporte quelques bogues.

Ce livre présente les fonctionnalités et les fonctions de Base à l'aide d'un ensemble d'exemples de bases de données.

- Créer une base de données
- Accéder aux bases de données externes
- Créer et utiliser des tables dans des bases de données relationnelles
- Créer et utiliser des formulaires pour la saisie de données
- Utiliser des requêtes pour rassembler les données de différentes tables, calculer des résultats si nécessaire et filtrer rapidement un enregistrement spécifique à partir d'une masse de données
- Création de rapports à l'aide du Générateur de rapports
- Lier des bases de données à d'autres documents et formulaires externes, y compris l'utilisation dans des publipostages
- Filtrer et rechercher des données
- Utiliser des macros pour éviter les erreurs de saisie, simplifier les tâches et améliorer la convivialité des formulaires
- Maintenance des bases de données

Exemples de bases de données

Un ensemble d'exemples de bases de données a été créé pour accompagner ce livre. Vous pouvez les trouver ici :

https://wiki.documentfoundation.org/images/d/da/Exemples_Base_64.zip

- Media_sans_macros.odb.....Plusieurs chapitres
- Media_avec_macros.odb.....Chapitre 9, “Macros”
- Exemple_Sport.odb .....Chapitre 1, “Introduction à Base”
- Exemple_CSV_inclus.odb .....Chapitre 2, “Créer une base de données”
- Exemple_Saut_Curseur_Formulaires.odb .....Chapitre 4, “Formulaires”
- Exemple_Rapport_Affichage_Images_Conditionnel.odb .....Chapitre 6, “Rapports”
- Exemple_Rapport_Facturation.odb .....Chapitre 6, “Rapports”
- Exemple_Rapport_Lignes_Colorées_Alternées.odb .....Chapitre 6, “Rapports”
- Exemple_Import_Formulaire_PDF.odb .....Chapitre 7, “Connexions”
- Exemple_Autotexte_Recherche_Orthographe.odb.....Chapitre 8, “Trucs et astuces”
- Exemple_Connexions_Images.odb .....Chapitre 8, “Trucs et astuces”
- Exemple_Activer_Fichier_Courriel.odb .....Chapitre 9, “Macros”
- Exemple_BaseDeDonnees_LettreType_Directe.odb.....Chapitre 9, “Macros”
- Exemple_Calcul_Direct_Formulaire.odb.....Chapitre 9, “Macros”
- Exemple_Cherche_et_Filtre.odb.....Chapitre 9, “Macros”
- Exemple_Controles de saisie.....Chapitre 9, “Macros”
- Exemple_CopieDonnees_Source_Cible.odb.....Chapitre 9, “Macros”
- Exemple_DateFormulaire_en_Variable.odb.....Chapitre 9, “Macros”
- Exemple_Defiler_Enregistrements.odb.....Chapitre 9, “Macros”
- Exemple_Dialogues.odb.....Chapitre 9, “Macros”
- Exemple_Formulaire_a_formulaire.odb.....Chapitre 9, “Macros”
- Exemple_Import_Donn,es_Calc..odbodb.....Chapitre 9, “Macros”
- Exemple_Inserer_Date_Actuelle.....Chapitre 9, “Macros”
- Exemple_InsertUpdateDelete_SQL.odb.....Chapitre 9, “Macros”
- Exemple_Liste_Selection_Multiple.odb.....Chapitre 9, “Macros”
- Exemple_Tableau_de_Champs.odb.....Chapitre 9, “Macros”
- Exemple_ZoneCombinee_ChampListe.odb.....Chapitre 9, “Macros”

Où obtenir plus d’aide

Ce livre, les autres guides de l’utilisateur de LibreOffice, le système d’aide intégré et les systèmes de support utilisateur supposent que vous êtes familiarisé avec votre ordinateur et les fonctions de base telles que le démarrage d’un programme, l’ouverture et l'enregistrement de fichiers.

Système d'aide

LibreOffice est livré avec un système d’aide complet. Ceci est votre première forme d’assistance pour l'utilisation de LibreOffice. Les utilisateurs Windows et Linux peuvent choisir de télécharger et d’installer l'aide hors ligne pour l'utiliser lorsqu’ils ne sont pas connectés à Internet ; l’aide hors ligne est installée avec le programme sur macOS.

Pour afficher le système d’aide, appuyez sur *F1* ou sélectionnez **Aide de LibreOffice** dans le menu Aide. Si l’aide hors ligne n’est pas installée sur votre ordinateur et que vous êtes connecté à Internet, votre navigateur par défaut ouvrira les pages d’aide en ligne sur le site Web de LibreOffice. Le menu Aide comprend également des liens vers d’autres informations et services d’assistance de LibreOffice.

Vous pouvez placer le pointeur de la souris sur l’une des icônes de la barre d’outils pour voir une petite boîte (« info-bulle ») avec une brève explication de la fonction de l’icône. Pour une explication plus détaillée, sélectionnez **Aide> Qu’est-ce que c’est ?** dans la barre de menus et placez le pointeur sur l'icône. De plus, vous pouvez choisir d’activer ou non les infoballons (en utilisant **Outils> Options> LibreOffice> Général**).

Assistance en ligne gratuite

La communauté LibreOffice développe non seulement des logiciels, mais fournit un support gratuit basé sur des bénévoles. En plus des liens du menu Aide ci-dessus, d'autres options de support de la communauté en ligne sont disponibles, voir le tableau ci-dessous.

Support gratuit LibreOffice	
FAQs	Réponses aux questions fréquemment posées. https://wiki.documentfoundation.org/Faq/fr
Listes de diffusion	Un support communautaire gratuit est assuré par un réseau d'utilisateurs expérimentés https://wiki.documentfoundation.org/Language/LocalMailingLists#French
Questions, réponses et Base de connaissances	Une assistance communautaire gratuite est fournie dans un service Web au format Question & Réponse. Recherchez des sujets similaires ou ouvrez-en un nouveau dans : https://ask.libreoffice.org/fr/questions/ Le service est disponible dans plusieurs autres langues; remplacez simplement /fr/ par en, de, es, fr, ja, ko, nl, pt, tr et bien d'autres dans l'adresse Web ci-dessus.
Prise en charge de la langue native	Le site Web LibreOffice en différentes langues https://www.libreoffice.org/community/nlc/ Listes de diffusion pour les langues natives https://wiki.documentfoundation.org/Language/LocalMailingLists/fr Informations sur les réseaux sociaux https://wiki.documentfoundation.org/Website/Web_Sites_services
Options d'accessibilité	Informations sur les options d'accessibilité disponibles https://www.libreoffice.org/get-help/accessibility/
Forum OpenOffice & LibreOffice	Un autre forum qui fournit un support pour LibreOffice, parmi d'autres suites bureautiques open source https://forum.openoffice.org/fr/forum/

Support et formation rémunérés

Vous pouvez également payer le support via des contrats de service auprès d'un fournisseur ou d'un cabinet de conseil spécialisé dans LibreOffice. Pour plus d'informations sur l'assistance professionnelle certifiée, consultez le site Web de The Document Foundation :
https://www.documentfoundation.org/gethelp/support/

Ce que vous voyez peut être différent

Illustrations

LibreOffice fonctionne sur les systèmes d'exploitation Windows, Linux et macOS, dont chacun a plusieurs versions et peut être personnalisé par les utilisateurs (polices, couleurs, thèmes, gestionnaires de fenêtres). Les illustrations de ce guide proviennent de divers systèmes d'exploitation. Par conséquent, certaines illustrations ne ressembleront pas exactement à ce que vous voyez sur l'écran de votre ordinateur.

De plus, certaines boîtes de dialogue peuvent être différentes en raison des paramètres sélectionnés dans LibreOffice. Sur certains systèmes, vous pouvez utiliser des boîtes de dialogue

du système d'exploitation de votre ordinateur (par défaut) ou des boîtes de dialogue fournies par LibreOffice. Pour modifier les boîtes de dialogue utilisées, allez dans **Outils> Options> LibreOffice> Général** et sélectionnez ou désélectionnez l'option **Utiliser les boîtes de dialogue LibreOffice**.

Icônes

Les icônes utilisées pour illustrer certains des nombreux outils disponibles dans LibreOffice peuvent différer de celles utilisées dans ce guide. Les icônes de ce guide proviennent d'une installation LibreOffice qui a été configurée pour afficher le jeu d'icônes *Colibre*.
Pour modifier le jeu d'icônes utilisé, allez dans **Outils> Options> LibreOffice> Affichage**. Dans la section *Style d'icône*, choisissez dans la liste déroulante.

Utiliser LibreOffice sur macOS

Certaines touches et éléments de menu sont différents sur macOS de ceux utilisés sous Windows et Linux. Le tableau ci-dessous donne quelques substitutions courantes aux instructions de ce chapitre. Pour une liste plus détaillée, consultez l'aide de l'application.

Windows ou Linux	***Equivalent macOS***	***Effet***
Tools > Options	**LibreOffice > Préférences**	Accès aux options de configuration
Clic-Droit	Contrôle + clic et/ou clic droit selon la configuration de l'ordinateur	Ouvre un menu contextuel
Ctrl (Contrôle)	*⌘ (Command)*	Utilisé avec d'autres touches
F5	*Maj+ ⌘+F5*	Ouvre le Navigateur
F11	*⌘+T*	Ouvre le panneau Styles dans la barre latérale

Comment toutes ces choses sont-elles nommées ?

Les termes utilisés dans LibreOffice pour la plupart des parties de l'interface utilisateur (les parties du programme que vous voyez et utilisez, contrairement au code en arrière-plan qui le fait réellement fonctionner) sont les mêmes que pour la plupart des autres programmes.

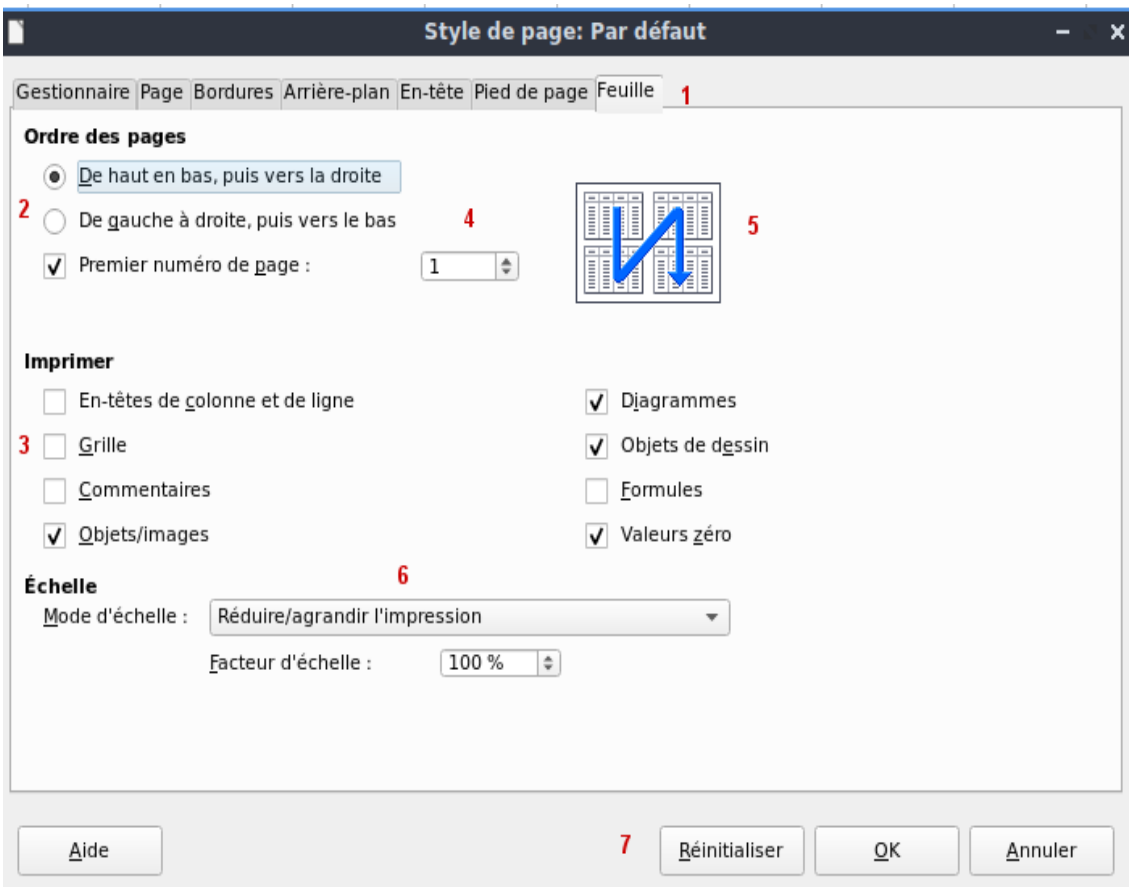


Figure 1: Dialogue (dans Calc) montrant les contrôles communs

1. Page à onglets (pas à proprement parler un contrôle).
2. Boutons radio (un seul peut être sélectionné à la fois).
3. Case à cocher (plusieurs peuvent être sélectionnés à la fois).
4. Zone de sélection rotative (cliquez sur les flèches haut et bas pour modifier le nombre affiché dans la zone de texte à côté, ou saisissez dans la zone de texte).
5. Vignette ou aperçu.
6. Liste déroulante pour la sélection d'un élément.
7. Boutons poussoir.

Une boîte de dialogue est un type spécial de fenêtre. Son but est de vous informer de quelque chose, ou de demander votre contribution, ou les deux. Il fournit des commandes que vous pouvez utiliser pour spécifier comment effectuer une action. Les noms techniques des commandes courantes sont indiqués dans la figure 1. Dans la plupart des cas, nous n'utilisons pas les termes techniques dans ce manuel, mais il est utile de les connaître car l'aide et d'autres sources d'informations les utilisent souvent.

Dans la plupart des cas, vous ne pouvez interagir qu'au niveau de la boîte de dialogue (pas sur le document lui-même) tant qu'elle reste ouverte. Lorsque vous fermez la boîte de dialogue après utilisation (généralement, cliquer sur OK ou un autre bouton enregistre vos modifications et ferme la boîte de dialogue), vous pouvez à nouveau travailler avec le document.

Certaines boîtes de dialogue peuvent rester ouvertes pendant que vous travaillez, de sorte que vous pouvez basculer entre la boîte de dialogue et le document. Un exemple de ce type est la boîte de dialogue **Rechercher et Remplacer**.

Questions fréquemment posées

Quelle est la licence de LibreOffice ?

LibreOffice est distribué sous la licence publique Mozilla (MPL) approuvée par l'Open Source Initiative (OSI). Voir https://www.libreoffice.org/about-us/licenses/

Il est basé sur le code d'Apache OpenOffice mis à disposition sous la licence Apache 2.0, mais comprend également des logiciels qui diffèrent d'une version à l'autre sous une variété d'autres licences Open Source. Le nouveau code est disponible sous LGPL 3.0 et MPL 2.0.

Puis-je distribuer LibreOffice à n'importe qui ? Puis-je le vendre ? Puis-je l'utiliser dans mon entreprise ?

Oui.

Sur combien d'ordinateurs puis-je l'installer ?

Autant que vous le souhaitez.

LibreOffice est-il disponible dans ma langue ?

LibreOffice a été traduit (localisé) dans plus de 40 langues, donc elle est probablement prise en charge. En outre, il existe plus de 70 dictionnaires d'orthographe, de césure et de thésaurus disponibles pour les langues et les dialectes qui n'ont pas d'interface de programme localisée. Les dictionnaires sont disponibles sur le site Web de LibreOffice à l'adresse:www.libreoffice.org.

Pourquoi ai-je besoin de Java pour exécuter LibreOffice ? Est-il écrit en Java ?

LibreOffice n'est pas écrit en Java; il est écrit en langage C ++. Java est l'un des nombreux langages pouvant être utilisés pour étendre le logiciel. Le JDK / JRE Java n'est requis que pour certaines fonctionnalités. Le plus notable est le moteur de base de données relationnelle HSQLDB.

Comment puis-je contribuer à LibreOffice ?

Vous pouvez aider au développement et au support utilisateur de LibreOffice de plusieurs manières, et vous n'avez pas besoin d'être programmeur. Pour commencer, consultez cette page Web: https://www.libreoffice.org/community/get-involved/

Puis-je distribuer le PDF de ce livre ou imprimer et vendre des copies ?

Oui, tant que vous répondez aux exigences de l'une des licences dans la déclaration de copyright au début de ce livre. Vous n'êtes pas obligé de demander une autorisation spéciale. Nous vous demandons de partager avec le projet une partie des bénéfices que vous faites de la vente de livres, compte tenu de tout le travail que nous avons mis à leur production.

Faire un don à LibreOffice : https://fr.libreoffice.org/donate/

LibreOffice

Guide Base

Chapitre 1
Introduction à Base

Introduction

Dans les opérations quotidiennes de bureau, les feuilles de calcul sont régulièrement utilisées pour agréger des ensembles de données et pour effectuer des analyses sur ces derniers. Comme les données d'une feuille de calcul sont présentées dans une vue de tableau, clairement visibles et peuvent être modifiées ou ajoutées, de nombreux utilisateurs se demandent pourquoi ils devraient utiliser une base de données au lieu d'une feuille de calcul. Ce manuel explique les différences entre les deux.

Note

Dans le langage technique, le "Fichier base de données" est utilisé pour une base de données à partir d'une seule interface et le "système de base de données" englobe le système de gestion de base de données (SGBD) et la base de données proprement dite.

Base offre un accès à divers systèmes de base de données via une interface utilisateur graphique et fonctionne par défaut avec le moteur de base de données intégré HSQLDB.

Ce chapitre présente deux exemples de bases de données et tout ce manuel est construit autour d'eux : Media_sans_macros.odb et Media_avec_macros.odb, étendus avec l'inclusion de macros. Les deux bases de données sont destinées à l'exploitation d'une bibliothèque : acquisition de média, emprunt par l'utilisateur, location de média et tout ce qui y est lié, comme le rappel pour les lecteurs.

Un exemple plus détaillé est donné plus loin dans ce chapitre (à partir de la page 29). Il utilise la base de données Exemple_Sport.odb pour organiser une compétition sportive.

Note

Comme tout logiciel, LibreOffice Base n'est pas totalement exempt d'erreurs. Particulièrement ennuyeuses sont les régressions, qui réintroduisent un bogue d'une ancienne version dans la version actuelle. Le lien suivant mène aux régressions actuellement en suspens :

https://bugs.documentfoundation.org/buglist.cgi?bug_status=UNCONFIRMED&bug_status=NEW&bug_status=REOPENED&bug_status=VERIFIED&bug_status=NEEDINFO&component=Base&keywords=regression&keywords_type=allwords&list_id=1162928&product=LibreOffice&query_format=advanced

Un regard sur la liste des bogues peut donc vous aider à comprendre les différences entre la documentation et sa propre version du programme.

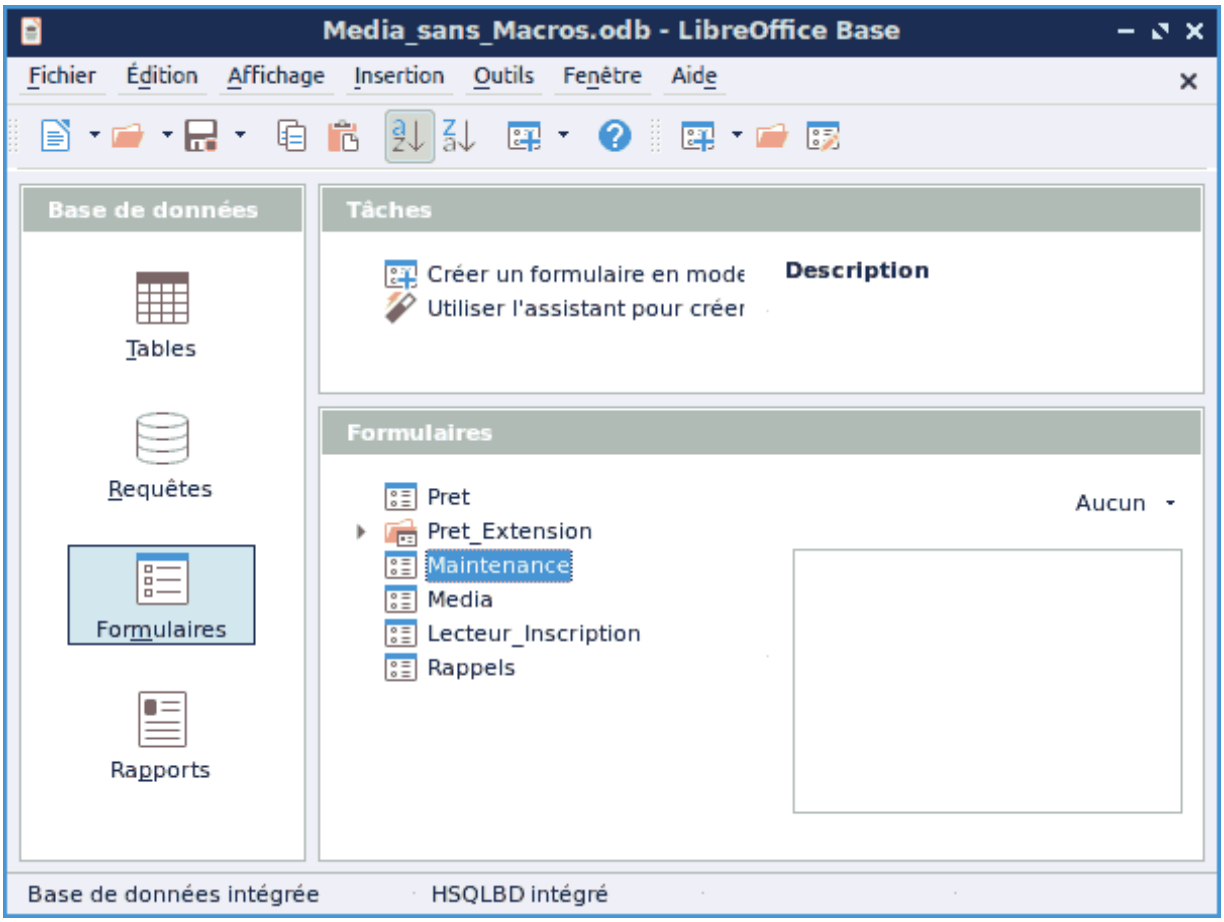
Base – un conteneur pour les éléments de la base de données

Un fichier Base est un dossier compressé contenant des informations sur les différentes zones de travail (composants) de Base. En utilisation quotidienne, Base s'ouvre initialement avec la vue illustrée dans la Figure 2.

L’environnement de base contient quatre zones de travail : Tables, Requêtes, Formulaires et Rapports. En fonction de la zone sélectionnée, diverses tâches (création de nouveau contenu ou appel d’éléments existants) peuvent être effectuées.

Dans les zones de travail Formulaires et Rapports, les éléments respectifs sont organisés dans une structure de dossiers (Figure 3). Cela se fait soit directement lors de l’enregistrement dans la boîte de dialogue Enregistrer, soit par la création de nouveaux dossiers en utilisant **Insertion > Dossier**

Bien que le fondement d’une base de données soit constituée de tables, similaires à des tableaux, Base commence par le mode Formulaire, car ils sont les éléments les plus couramment utilisés lors de l’utilisation de bases de données. Avec les formulaires, vous pouvez effectuer des entrées dans les tableaux et analyser le contenu des tableaux.



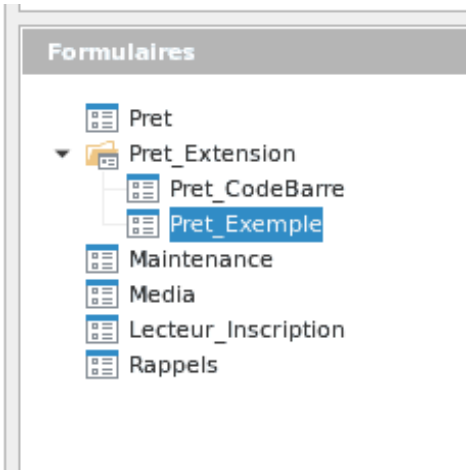


Figure 3: Structure de répertoire dans la zone de travail formulaires

Saisie de données à l'aide de formulaires

Les formulaires simples affichent un seul tableau, comme dans la partie supérieure du formulaire de prêt. Le formulaire de prêt (Figure 4) a été étendu pour afficher des informations supplémentaires : L'éventail des personnes affichées peut être filtré sur le nom de famille pour limiter les détails affichés. Si un utilisateur entre la lettre "G" dans le champ Filtre (nom de famille) à droite du tableau des prêts, seules les personnes dont le nom commence par "G" seront affichées[1].

- Les informations sur les nouveaux emprunteurs peuvent être saisies directement dans les champs du tableau du formulaire.
- Les détails des éléments à emprunter sont saisis et affichés dans la zone centrale du formulaire. Le nom de l'utilisateur est également clairement souligné. Si un article déjà emprunté est en retard et doit être retourné, cette zone est bloquée (aucune saisie possible) et le titre indiquera "Prêt temporairement verrouillé !". Les articles en prêt sont affichés dans la partie inférieure du formulaire.
- La date d'emprunt est définie comme date du jour. Dans le champ déroulant à gauche du bouton Actualiser se trouvent les éléments multimédias qui peuvent être empruntés. Les articles déjà prêtés à l'emprunteur sélectionné ne peuvent pas être sélectionnés.
- Les éléments multimédias sélectionnés pour le prêt sont ajoutés aux détails du prêt en cours en cliquant sur le bouton Actualiser.
- Dans la partie inférieure du formulaire (Retours), il n'est pas possible de supprimer une ligne de données. Seuls les champs Date de retour et Extension peuvent être modifiés. Si un emprunteur a été précédemment verrouillé et a ensuite renvoyé le ou les articles en retard, la zone de prêt peut être déverrouillée en cliquant sur le bouton Actualiser.

Toutes ces fonctions peuvent être effectuées sans utiliser de macros, lorsque le formulaire est configuré et rempli de la manière décrite.

1 Ce formulaire est visible dans le fichier exemple Media_sans_Macros.odb

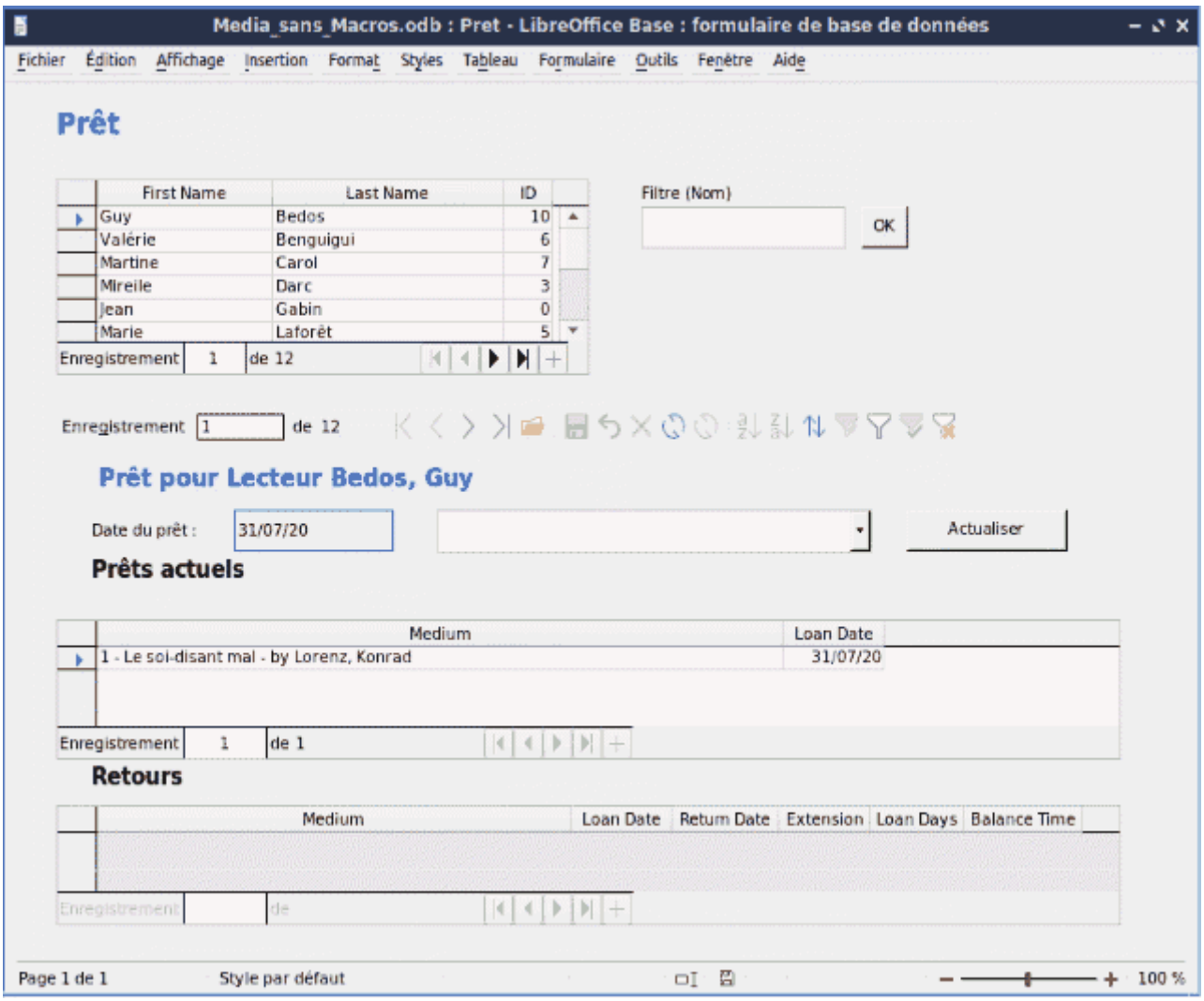


Figure 4: Le formulaire de prêt

Entrée directe de données dans une table – principes de base pour la saisie de données

Les tables d'une base de données sont liées un peu comme dans un réseau. Une table reçoit des informations d'autres tables ou leur en fournit. Ceci est appelé la relation et est indiqué par une ligne entre les tables reliant un champ de chacune. Ces relations sont visibles dans **Outils>Relations**.

Note

La table Lecteurs a une relation avec une autre table qui implique ID_Genre. De même, la table Medias a une relation avec quatre autres tables, chacune impliquant l'un de ces champs : ID_Categorie, ID_TypeMedia, ID_Ville et ID_Editeur.

La table des prêts est directement liée aux tables Media et Lecteurs, comme illustré à la figure 5.

Lorsqu'un livre est emprunté, au lieu que son titre soit enregistré dans la table Prets, un seul numéro est enregistré dans le champ ID_Media. Le champ ID de la table Media stocke l'identifiant

unique de chaque enregistrement de cette table. Ce champ est un champ clé de la table Media : la clé primaire.

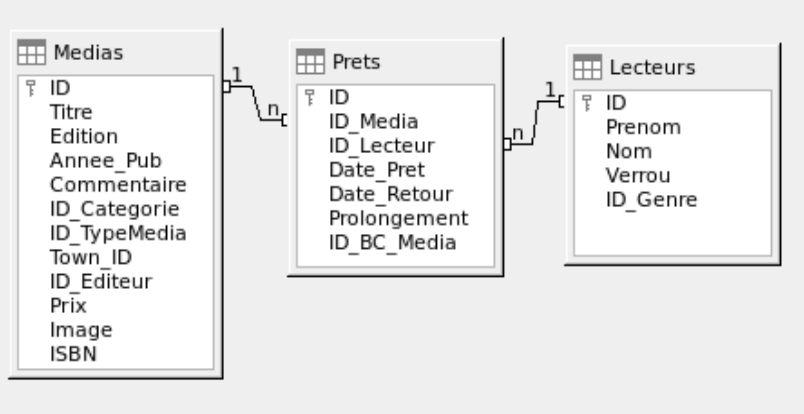


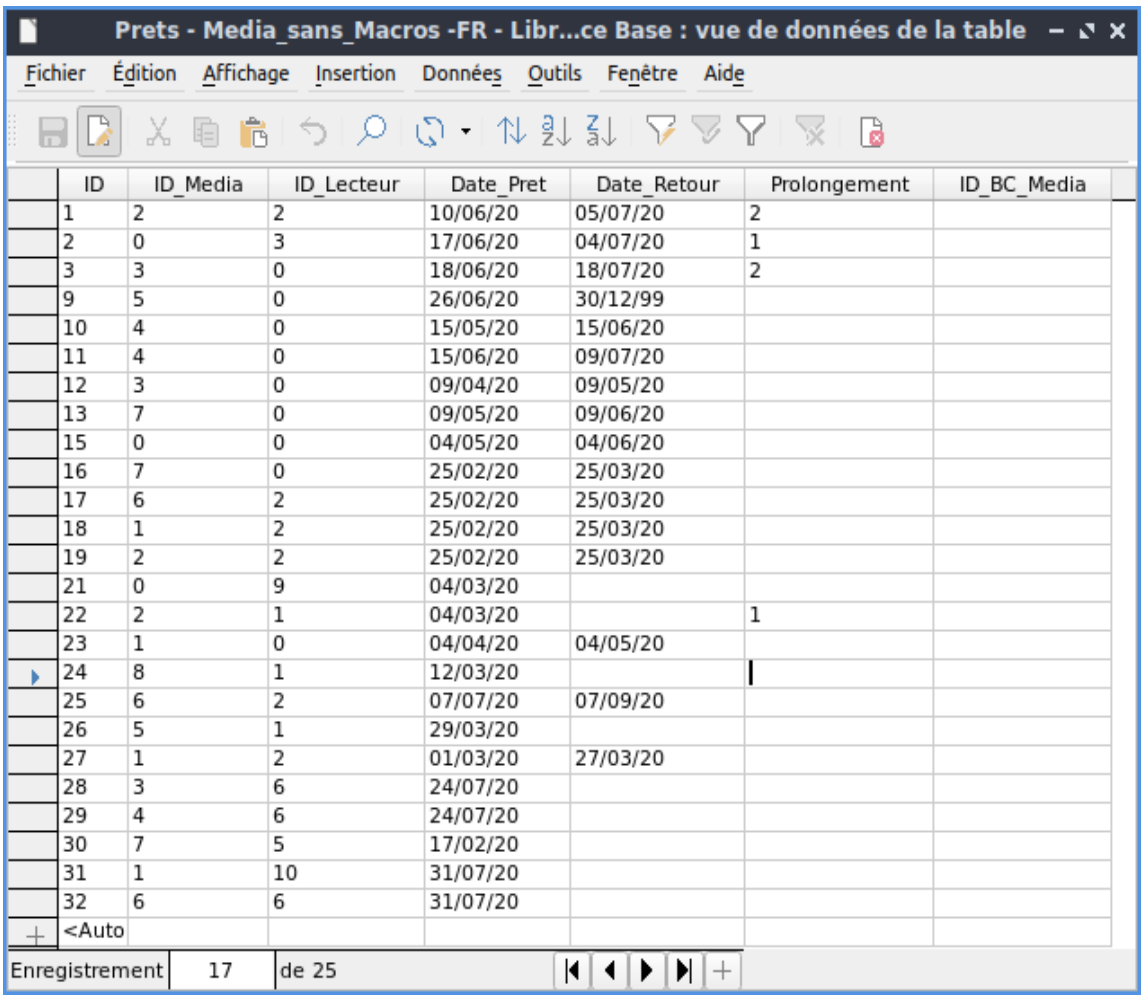
Figure 5: Relation entre la table Prets et la table Lecteurs

Conseil

La clé primaire détermine de manière unique les valeurs de chaque champ dans chaque enregistrement d'une table. Ainsi, lorsqu'un élément est emprunté, le numéro entré dans le champ ID-Media correspond au numéro du champ ID de la table Media qui identifie l'enregistrement contenant les informations sur l'élément emprunté.

Le nom du lecteur n'est pas saisi à chaque fois dans la table des prêts. Ces informations sont enregistrées dans la table Lecteurs. Il a également un champ de clé primaire qui identifie chaque personne qui emprunte un article. La valeur de ce champ peut ensuite être entrée dans la table Prets avec le champ ID_Lecteur identifiant la personne spécifique.

Les relations entre les tables présentent l'avantage de réduire considérablement le travail de bureau sur le formulaire. Au lieu d'avoir à entrer le titre du média ainsi que le prénom et le nom sans aucune erreur, ceux-ci peuvent être entrés en sélectionnant les bons numéros pour les champs ID_Media et ID_Lecteur, ce qui permet de sélectionner les bons éléments multimédias ainsi que les nom et prénom. Enfin, le même support peut être emprunté à nouveau plus tard et le même lecteur peut emprunter plusieurs supports supplémentaires lors de toute action de prêt.



ID	ID_Media	ID_Lecteur	Date_Pret	Date_Retour	Prolongement	ID_BC_Media
1	2	2	10/06/20	05/07/20	2	
2	0	3	17/06/20	04/07/20	1	
3	3	0	18/06/20	18/07/20	2	
9	5	0	26/06/20	30/12/99		
10	4	0	15/05/20	15/06/20		
11	4	0	15/06/20	09/07/20		
12	3	0	09/04/20	09/05/20		
13	7	0	09/05/20	09/06/20		
15	0	0	04/05/20	04/06/20		
16	7	0	25/02/20	25/03/20		
17	6	2	25/02/20	25/03/20		
18	1	2	25/02/20	25/03/20		
19	2	2	25/02/20	25/03/20		
21	0	9	04/03/20			
22	2	1	04/03/20		1	
23	1	0	04/04/20	04/05/20		
24	8	1	12/03/20			
25	6	2	07/07/20	07/09/20		
26	5	1	29/03/20			
27	1	2	01/03/20	27/03/20		
28	3	6	24/07/20			
29	4	6	24/07/20			
30	7	5	17/02/20			
31	1	10	31/07/20			
32	6	6	31/07/20			
<Auto						

Figure 6: Table Prets, structure et données

La structure de table pour un tel formulaire est relativement basique et facile à mettre en place. Dans le tableau illustré à la Figure 6, (table prêt) les mêmes données peuvent être entrées directement dans les lignes et les colonnes de la table que lors de l'utilisation du formulaire, à condition de connaître les ID Media et lecteur correspondants. Les relations de cette table avec les autres tables de la base de données sont utilisées dans le formulaire.

- Le champ le plus important, la clé primaire (ID) qui est ajoutée automatiquement, montre le contenu unique et indispensable pour la plupart des bases de données. Pour plus d'informations sur ce sujet, reportez-vous à la section "Relations entre les tables" du chapitre 3, Tables.
- Le deuxième champ, ID_Media, stocke les valeurs de la clé primaire de la table Medias. Il fait référence au numéro dans le champ correspondant, ID, dans la table Media. Une telle référence à une clé primaire est appelée clé étrangère. Dans le formulaire, le titre et l'auteur seront affichés à la place de la clé étrangère dans une zone de liste. La zone de liste transmet la valeur en arrière-plan à la clé étrangère de la table.
- Le troisième champ, ID_Lecteur, stocke les valeurs de clé primaire de la table Lecteurs. Dans cet exemple, cette clé est uniquement un nombre qui fait référence au lecteur qui emprunte des éléments multimédias. Dans le formulaire, le nom et le prénom du lecteur sont indiqués. Comme le montre le tableau, le lecteur avec le numéro de clé primaire "0" a emprunté beaucoup de supports. La table peut enregistrer plusieurs fois la clé primaire unique de la table Lecteurs en tant que clé étrangère ID_Lecteur. Mais en aucun cas un lecteur, qui est répertorié dans la clé étrangère de la table Prets, ne peut être supprimé dans la table Lecteur. Sinon, il ne serait plus possible de savoir qui a emprunté des médias

à un instant “t”. La base de données définit les paramètres par défaut de sorte qu’une suppression est impossible. Le terme technique pour cela est l’exigence d’intégrité référentielle.

- La date du prêt est stockée dans le quatrième champ. Si cette date est présente et est postérieure à la date actuelle, l’ensemble des données correspondant pour le lecteur est affiché dans le tableau du bas du formulaire sous le bouton Retour. Le prêt est différé.
- Le champ Prolongement contient des informations sur les extensions du prêt pour un article. La signification des valeurs 1, 2… est détaillée ci-après. La base de données contient une table séparée avec le nom Preferences.

L’entrée de ces données permet la gestion d’une bibliothèque simple.

Requêtes – obtenir des informations à partir de données dans des tables

Requêtes affiche une vue des tables. Elles rassemblent le contenu de plusieurs tables dans une vue d’ensemble. Les requêtes sont stockées uniquement dans le langage de requête SQL. Ce ne sont donc pas des tables, même si dans Base elles nous semblent apparaître comme des tables.

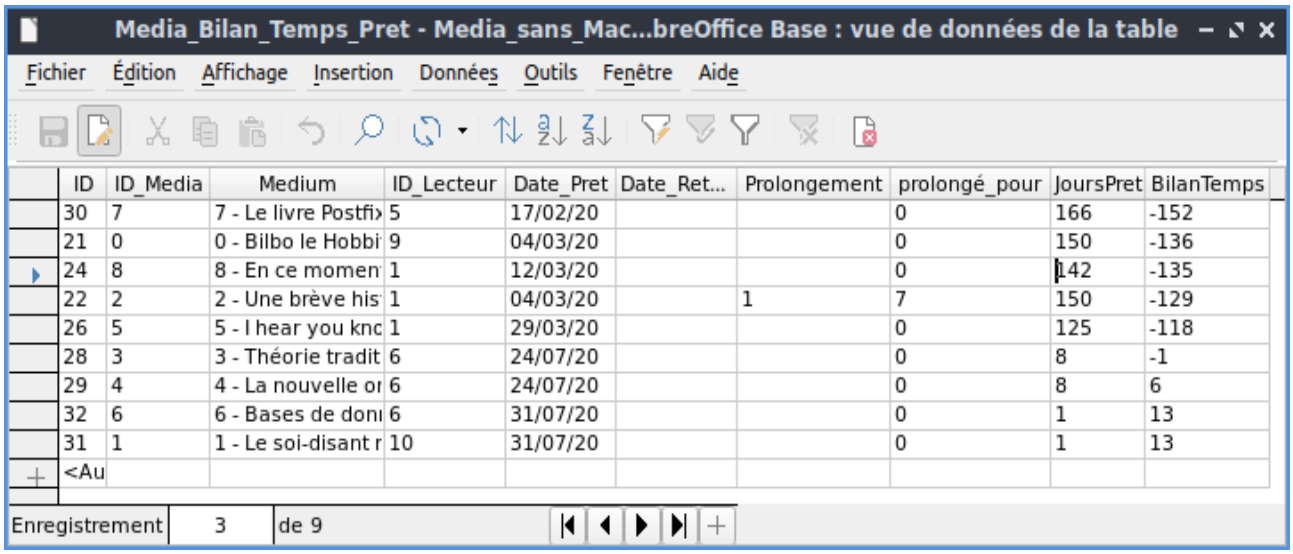


Figure 7: Exemple de requête

La requête illustrée à la figure 7 répertorie tous les médias actuellement prêtés. Elle calcule la durée du prêt de chaque article et le solde de la période de prêt. Lorsque ID_Media, le champ de clé étrangère, lit la clé primaire de la table Medias, le titre et l’auteur seront combinés en un seul texte dans le champ Medium. Ce champ sera nécessaire dans le formulaire sous le sous-titre “Retour”. Les champs combinés de la requête servent également de champs de connexion entre le formulaire de prêt réel et la table de prêt, à savoir via les champs ID_ Media et ID_Lecteur.

- Tous les supports sont répertoriés pour lesquels la date de retour n’est pas entrée dans la table Prets. À titre de présentation supplémentaire, le nom du média est inclus dans la requête avec le champ ID_Media.
- La référence au lecteur est établie avec la clé primaire de la table Lecteurs.
- La différence en jours entre la date du prêt (Date_Pret) et la date actuelle est spécifiée sous la forme de JoursPret
- Le nombre de jours de prêt est soustrait de la date du jour pour donner le nombre de jours restants dans la période de prêt. La durée du prêt peut varier selon les types de supports.

- Dans la table Parametres, une valeur de “1” pour le champ Prolongement correspond à une extension de la période de prêt de 7 jours. Dans l’ensemble de données ci-dessus, la ligne avec ID_Media '2' montre un prolongement de 7 jours.

Rapports – présentation des données

Dans les rapports, les données sont traitées de manière à pouvoir être imprimées dans un format utile. Les formulaires tels que celui de la figure 8 ne conviennent pas pour une lettre convenablement formatée.

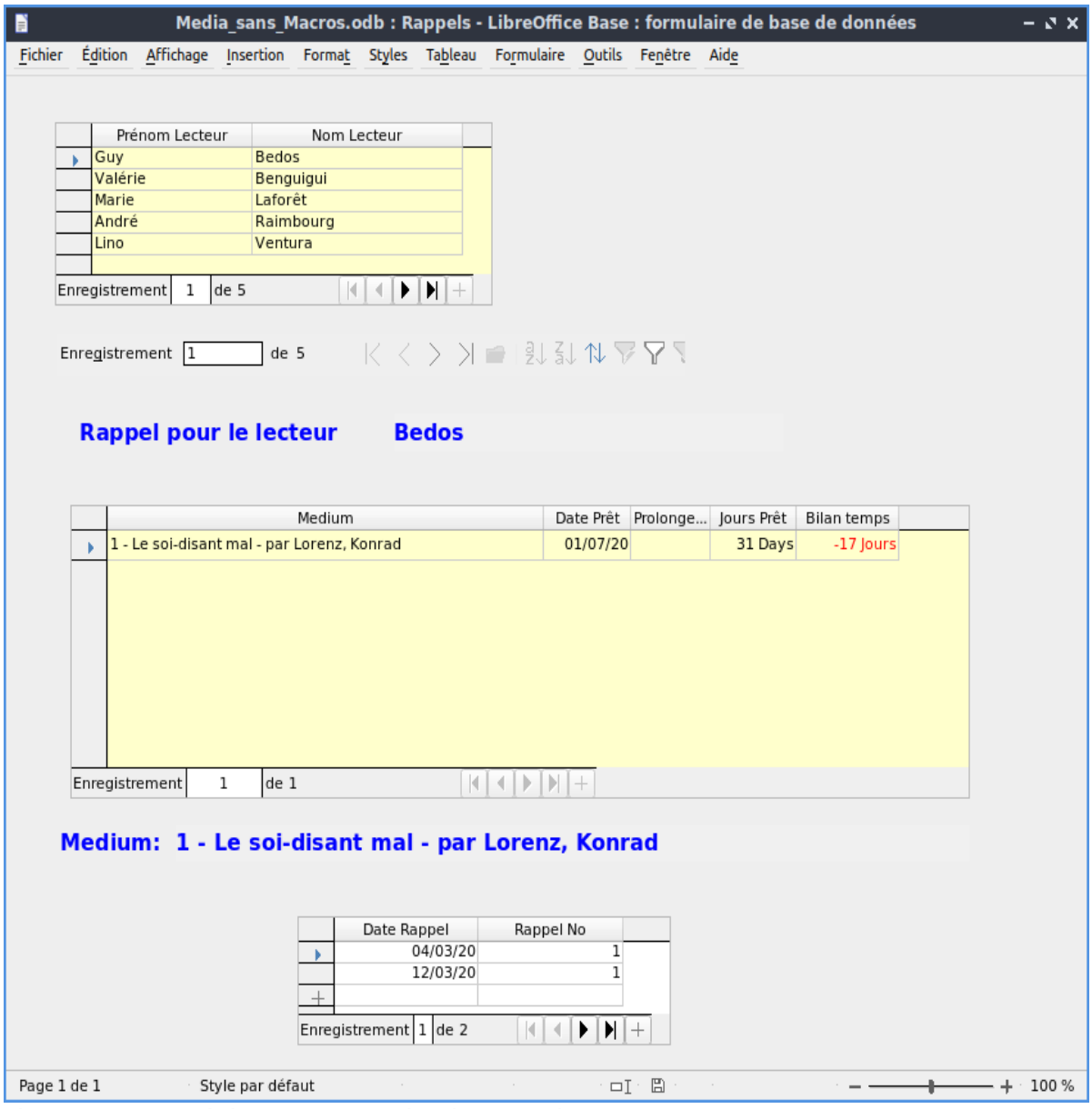


Figure 8: Formulaire contenant les informations pour un avis de rappel

Avant qu’un rapport réel, sous la forme d’un avis de rappel, puisse être imprimé, les informations doivent être saisies dans le formulaire de rappel. Le tableau du formulaire montre toutes les personnes qui ont emprunté des articles avec une durée de prêt résiduelle négative.

La date de rappel et le numéro de l’avis de rappel sont saisis pour chaque élément multimédia à rappeler. Il peut s’agir de la date actuelle de traitement des avertissements. La date de rappel est par défaut la date actuelle. Le numéro de rappel est un entier incrémenté de 1 à chaque avis de rappel successif pour un emprunt/média particulier.

Ce formulaire, dans l'exemple de base de données actuel sans macros, nécessite l'entrée de l'utilisateur pour créer des avis de rappel. Dans la version macro, la date est automatiquement saisie et l'avis de rappel imprimé.
L'avis de rappel (Figure 9) est généré au moyen d'une requête à partir des données précédemment entrées. L'utilisateur de la base de données n'a qu'à sélectionner le rapport de rappel et une lettre de rappel peut être imprimée et envoyée à toutes les personnes qui ont une entrée de rappel effectuée dans le formulaire de la page précédente.

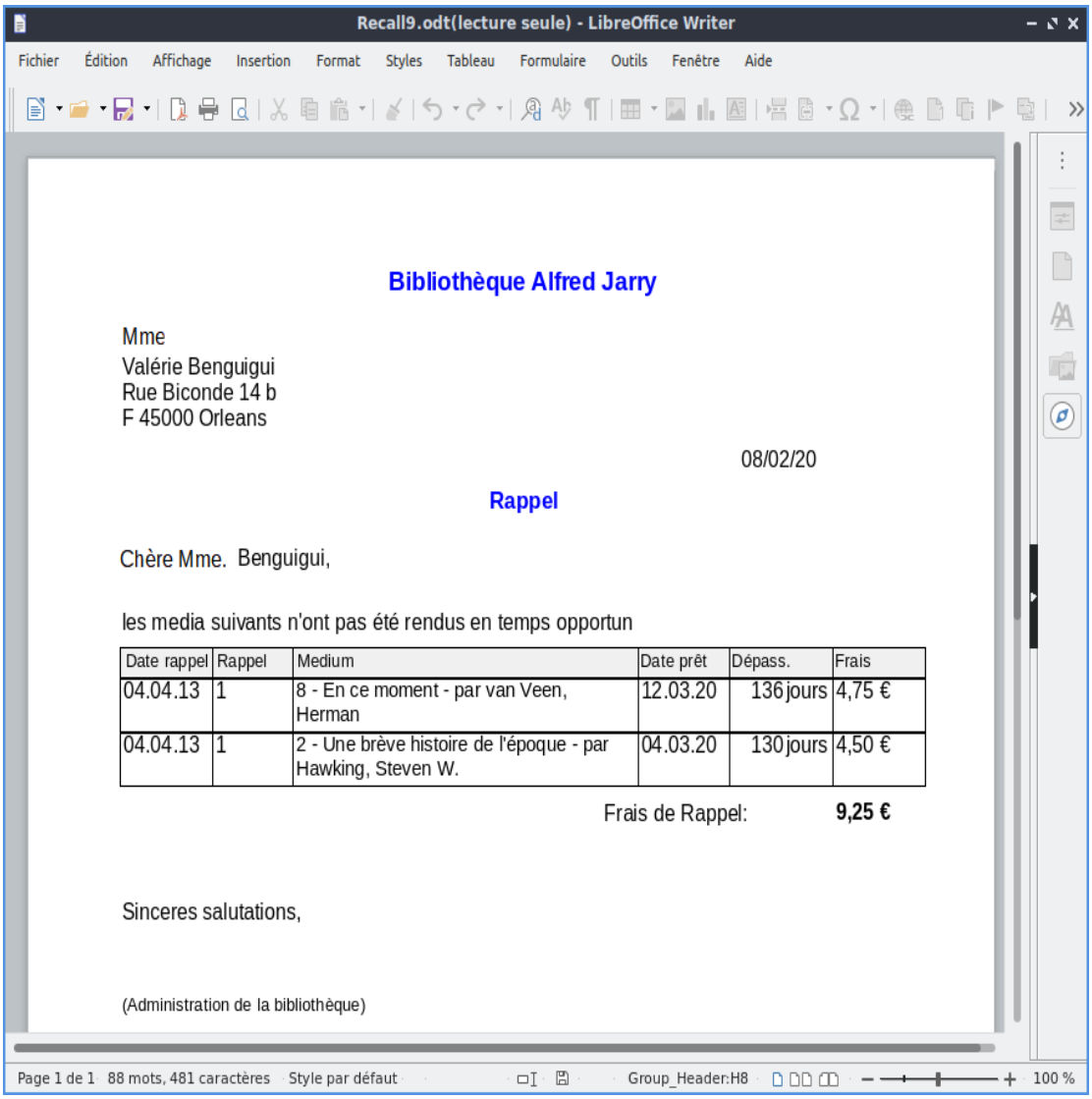


Figure 9: Exemple d'avis de rappel

Dans un tel rapport, il peut y avoir plusieurs entrées (éléments en retard) pour une personne en particulier. Si le tableau contenant les éléments pour cette personne dépasse l'espace d'une page, il est étendu pour inclure une page suivante.

Un tel rapport est plus complet qu'une lettre de publipostage produite avec Writer. Il rassemble automatiquement les ensembles de données pour l'impression et organise le texte d'accompagnement nécessaire en conséquence.

Une lettre similaire à celle de la figure ci-dessus ne peut être implémentée qu'avec des macros, ou le générateur de rapports, comme décrit dans "Créer un rapport" à la page 57.

Manipulation sûre d’un fichier Base

Les tables, requêtes, formulaires et rapports de la base de données interne HSQLDB sont stockés dans un fichier Base. Étant donné que le fichier de base de données est écrit en mémoire, les multiples objets qu’il contient vous obligent à le traiter avec soin. Les rapports de bogue indiquent clairement qu’un fichier de base de données nécessite un traitement un peu plus soigneux que, par exemple, un fichier texte écrit dans Writer.

Les instructions suivantes doivent donc être prises en compte lors du traitement d’un fichier de base :

- Un fichier de base de données ouvert ne doit pas être enregistré sous un nom différent en utilisant **Enregistrer sous**. Lorsqu’il n’y a pas d’autre choix, les tables, requêtes, formulaires et états doivent d’abord être fermés. Il est préférable de fermer le fichier de base de données et de créer ensuite une copie du fichier.
- Le Générateur de rapports est un module complémentaire. Bien qu’il ne soit désormais plus visible en tant qu’extension distincte, il fonctionne en grande partie indépendamment du fichier de base de données. Renommer le fichier supprime le Générateur de rapports de sa fondation.
- Lorsqu’une table, une requête, un formulaire ou un rapport est enregistré, il ne s’ensuit pas que le fichier de base de données entier a été enregistré. Cette sauvegarde doit être effectuée séparément. Lorsqu’un objet (table, requête, formulaire, rapport) est enregistré, les informations sont écrites dans le fichier de base de données en mémoire. Lorsque le fichier de base de données est enregistré, tout ce qui est contenu dans le fichier de base de données en mémoire est écrit dans le fichier *.odb.
- Ce comportement de mémoire est particulièrement vrai pour l’utilisation du Générateur de rapports. La préparation d’un rapport est toujours le composant le plus instable du fichier Base. Par conséquent, après chaque étape, le rapport et le fichier *.odb doivent tous deux être enregistrés. Une fois le rapport créé, il fonctionne par lui-même sans problème particulier.
- Une fois qu’une action est terminée, les données ajoutées à la base de données sont uniquement écrites dans le fichier de base de données en mémoire, mais pas dans le fichier *.odb. Ce n’est que lorsque vous fermez le fichier *.odb que vous y enregistrez les données. Le contenu de la base de données HSQLDB sera réécrit dans le fichier. Un crash à ce stade peut entraîner une perte de données. Par conséquent, une stratégie doit être développée pour que les copies de sauvegarde soient effectuées à temps. Le chapitre 9, Macros, comprend une macro pour effectuer une copie de sauvegarde lorsque vous ouvrez un fichier de base de données. De même, un moyen est montré pendant que le fichier Base est ouvert aussi bien que possible.

  Un niveau de sécurité nettement plus élevé peut enfin être atteint en utilisant des bases de données de serveurs externes comme MySQL / MariaDB ou PostgreSQL. Pour cela, Base peut alors servir de frontal avec les requêtes, formulaires et états de la base de données.

Une base de données simple – exemple de test en détail

La création d’une base de données est abordée au chapitre 2, Création d’une base de données. L’exemple suivant est basé sur la base de données par défaut HSQLDB qui est installée avec LibreOffice en tant que base de données interne. Il s’agit donc d’une base de données embarquée créée au préalable sans aucune inscription dans LibreOffice. Divers autres systèmes de bases de données peuvent être connectés en plus de la HSQLDB interne.
La première étape à faire après avoir trouvé un emplacement pour le fichier de base de données

et l'avoir enregistré consiste à fermer l'assistant Base de données en cliquant sur le bouton **Terminer**[2].

La base de données est destinée à organiser une compétition sportive dans différentes disciplines. Par conséquent, Exemple_Sport a été choisi comme nom du fichier.

Créer des tables

Dès que la base de données a été enregistrée, la fenêtre principale de la base de données apparaît. Par défaut, Tables est sélectionné dans la section Base de données sur le côté gauche de la fenêtre (Figure 10). Les tables sont le stockage central des données ; sans tables, il n'y a pas de base de données.

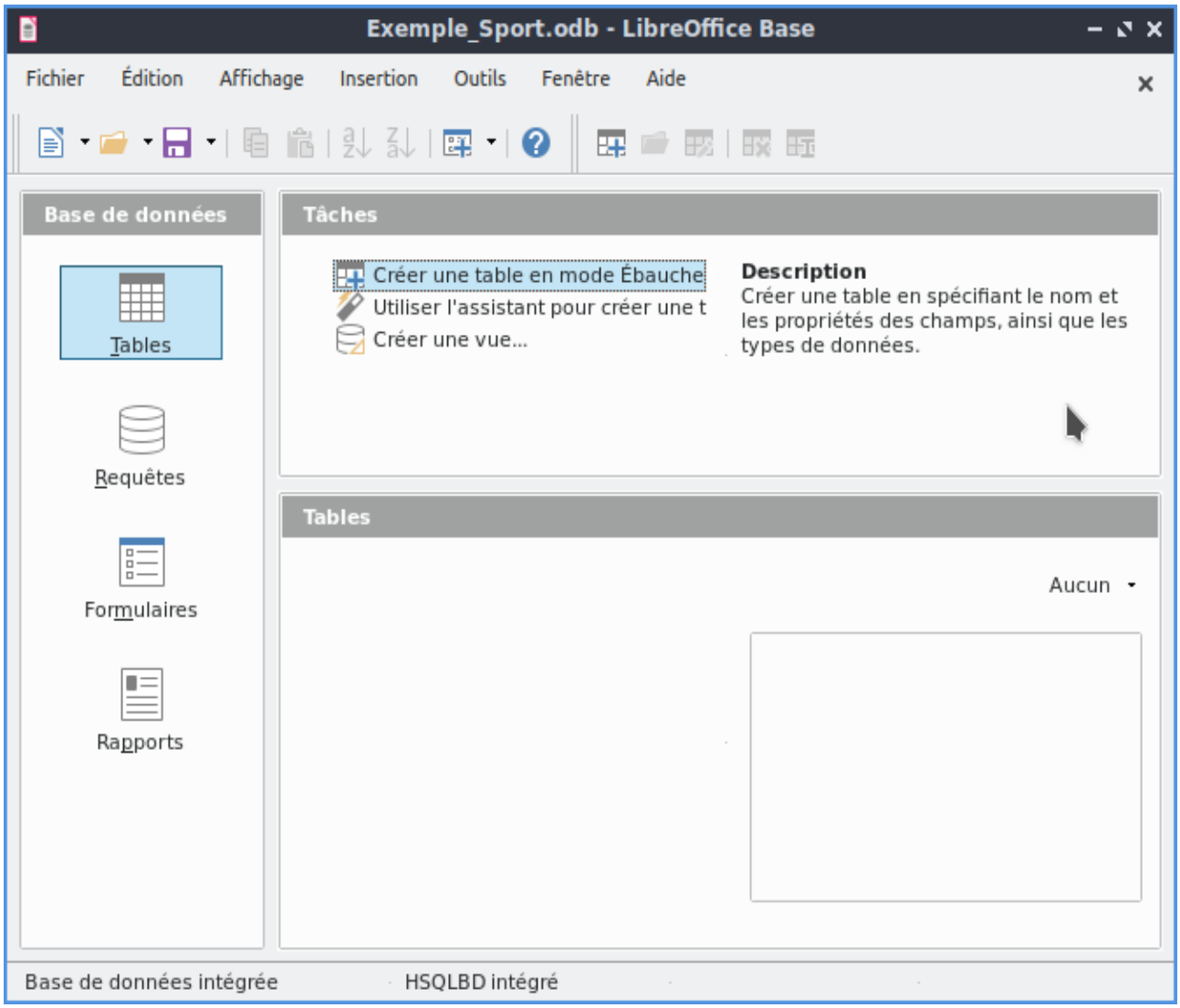


Figure 10: Fenêtre principale de la base de données exemple Exemple_Sport.odb

Cliquez sur **Créer une table en mode Ébauche** pour ouvrir la fenêtre illustrée à la Figure 11.

2 Cet exemple est réalisé dans le fichier Exemple_Sport.odb, disponible avec ce guide

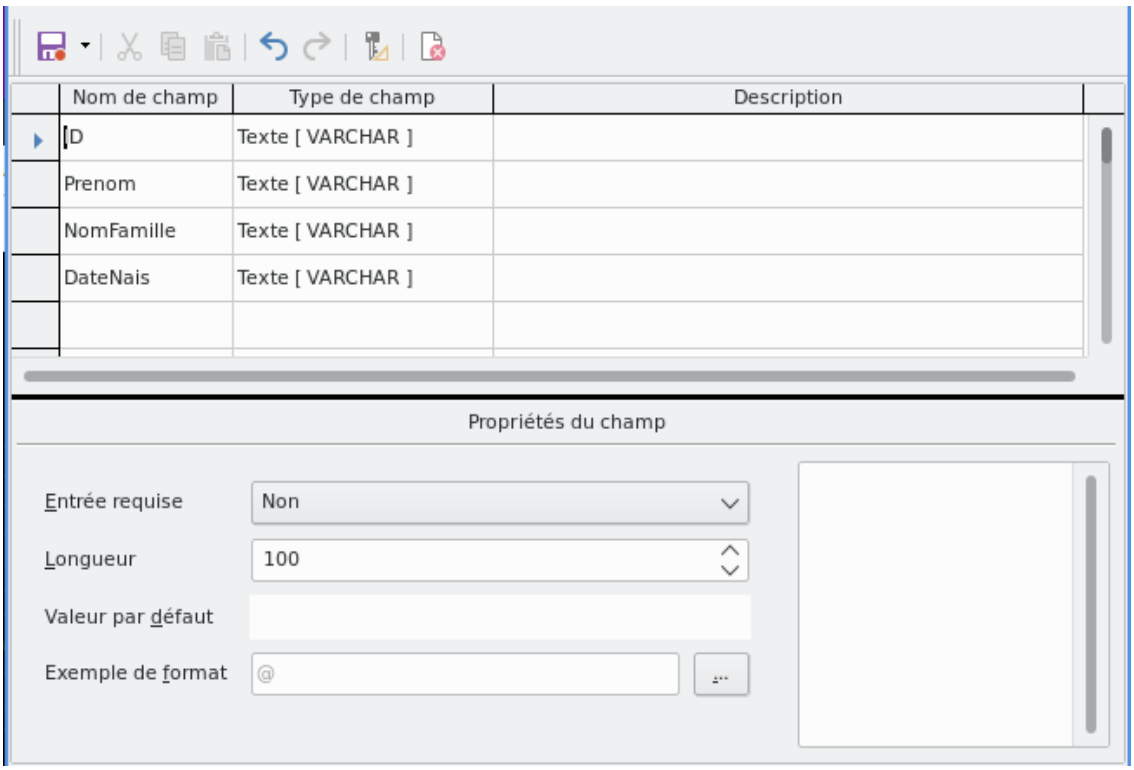
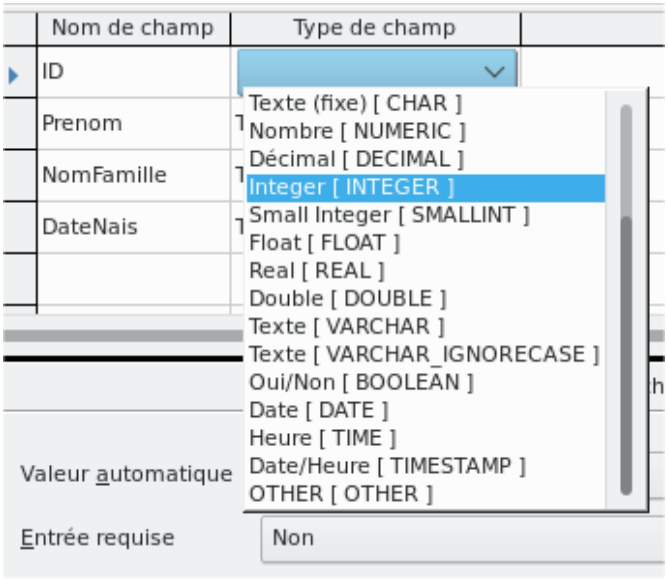


Figure 11: Vue en mode ébauche d'une table

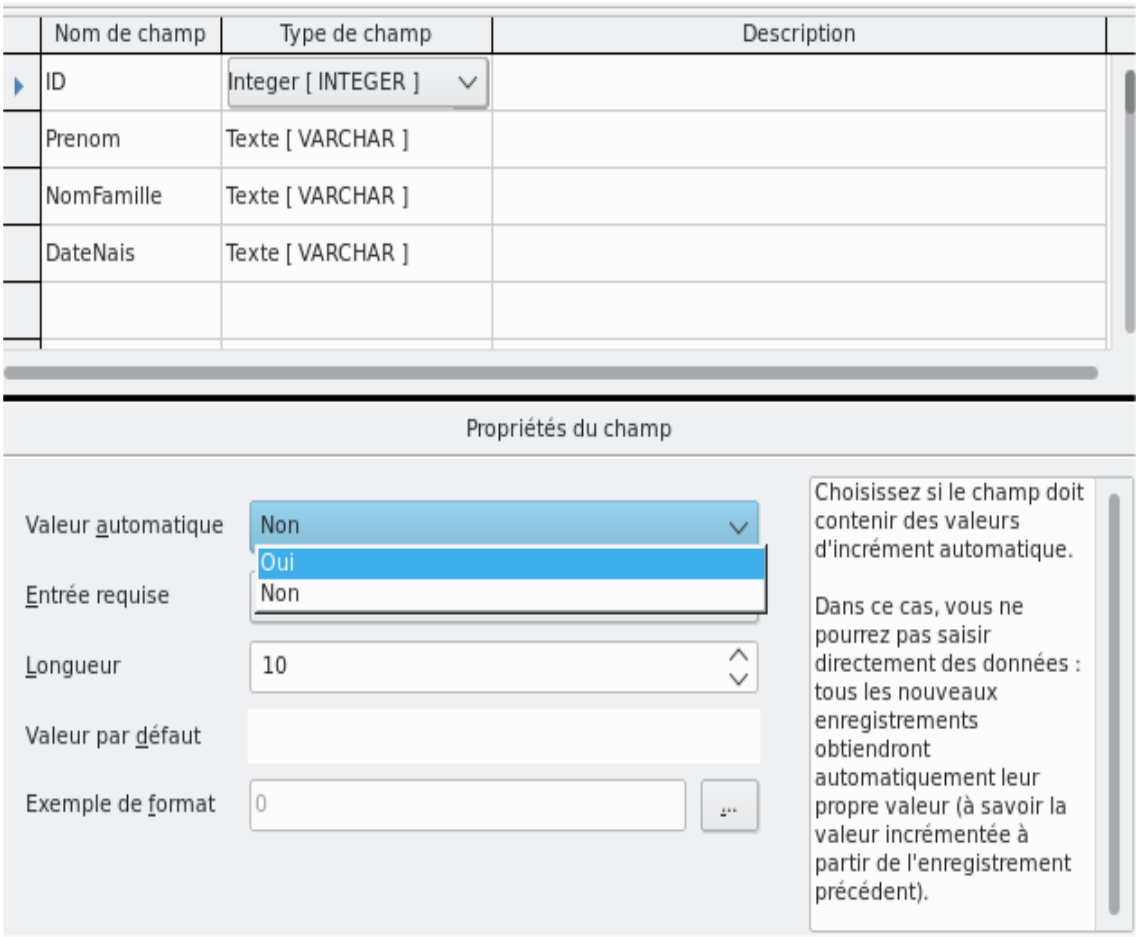
Les noms des champs de la première table seront saisis ici. Le tableau devrait inclure les candidats hommes et femmes. Ici, les noms de champ sont d'abord réduits aux composants clés.

Les noms de champ prénom (Prenom), nom (NomFamille) et date de naissance (DateNais) sont susceptibles d'être sans ambiguïté. De plus, un champ appelé ID a été ajouté. Ce champ prendra plus tard une valeur unique pour chaque enregistrement. Un champ clé unique est nécessaire pour la base de données intégrée. Sinon, aucun enregistrement ne peut être saisi dans la table. Ce champ de clé est appelé *clé primaire* dans les bases de données.

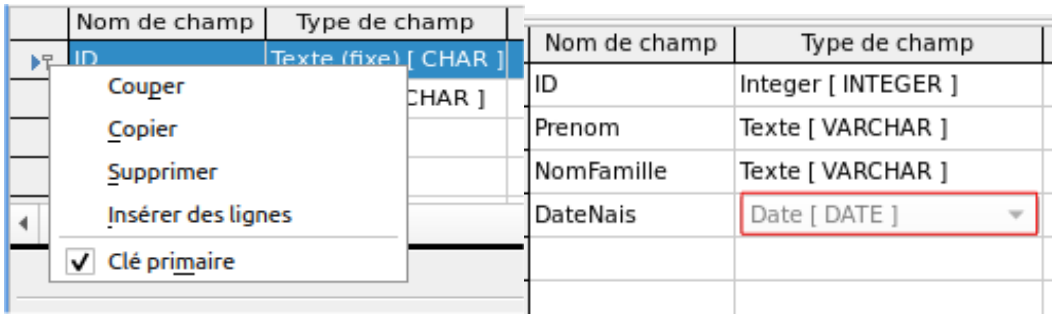
Un autre champ pourrait être utilisé pour cette propriété. Cependant, si par exemple le nom de famille était utilisé seul pour cela, deux personnes portant le même nom de famille ne pouvaient pas être enregistrées. Dans ce cas, il peut être utile de déclarer deux champs ensembles dans une clé primaire partagée. Il n'y a aucune garantie que cela fonctionne à long terme, ce qui n'est pas le cas. Ici, la version simple est préférée.



Dans la deuxième étape, sélectionnez les types de champs pour les champs déjà nommés dans les listes. Définissez le champ ID sur le type de champ Integer. Ce type de champ a l'avantage de pouvoir être automatiquement fourni par la HSQLDB intégrée avec l'entier supérieur suivant à chaque nouvel enregistrement.



Modifiez les propriétés de champ ID. Pour ce champ, activez l'incrémentation automatique des valeurs numériques croissantes : **Propriétés du champ > Valeur automatique > Oui**.



Après avoir sélectionné **Valeur automatique**, ou indiqué que ce champ était une clé primaire,une icône en forme de clé apparaît sur l'en-tête de ligne lorsque vous quittez la sélection du type de champ. Cela indique que ce champ est la clé primaire de la table. Si **Valeur automatique** n'est pas sélectionné, la clé primaire peut également être sélectionnée dans le menu contextuel (clic droit> clé primaire).

Sélectionnez le type de champ Date pour **DateNais** . Cela garantit que seules les entrées de date valides sont ajoutées. Il est également utilisé pour trier les dates ou, par exemple, calculer l'âge.

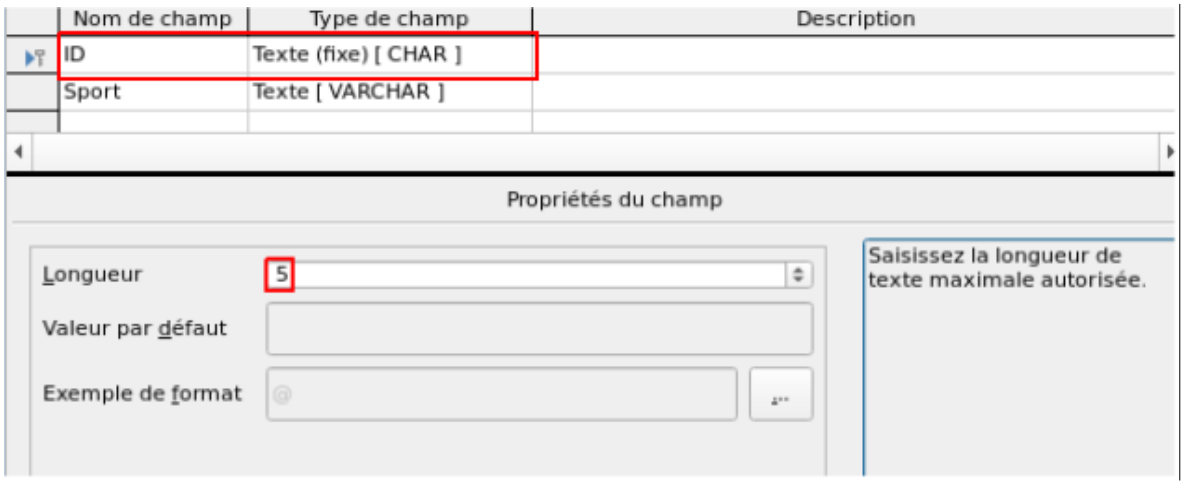
La table peut maintenant être enregistrée sous le nom **Candidats**. Par la suite, les données peuvent être saisies. La saisie dans le champ ID n'est pas nécessaire. Cela se fait automatiquement lorsque vous sauvegardez l'enregistrement

Note

Le fichier de base de données est un dossier compressé de fichiers individuels. Le stockage d'un seul objet comme table n'est donc pas directement écrit dans le fichier de base de données lui-même. C'est pourquoi il faut cliquer sur le bouton Enregistrer du fichier de base de données lui-même, même après la création de tables, requêtes, formulaires et rapports. Ce n'est qu'en quittant la ligne de données d'une table que les données saisies sont enregistrées automatiquement.

Les candidats (athlètes) peuvent désormais être inscrits. Cependant, les informations suivantes manquent à première vue :

- Une liste de sports dans lesquels les candidats veulent concourir.
- Pour les compétitions, la distinction entre les candidats masculins et féminins.



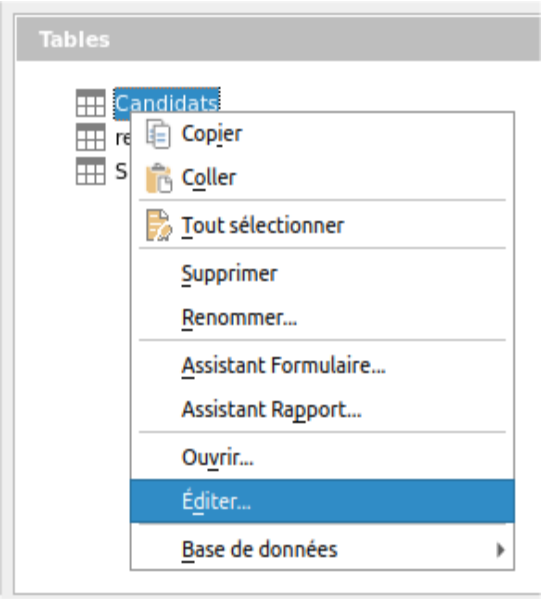
Créez une table Sports avec 2 champs comme indiqué sur la figure. Comme il n'y a pas beaucoup de sports différents, Valeur automatique n'est pas sélectionné pour la clé primaire. Au lieu de cela,

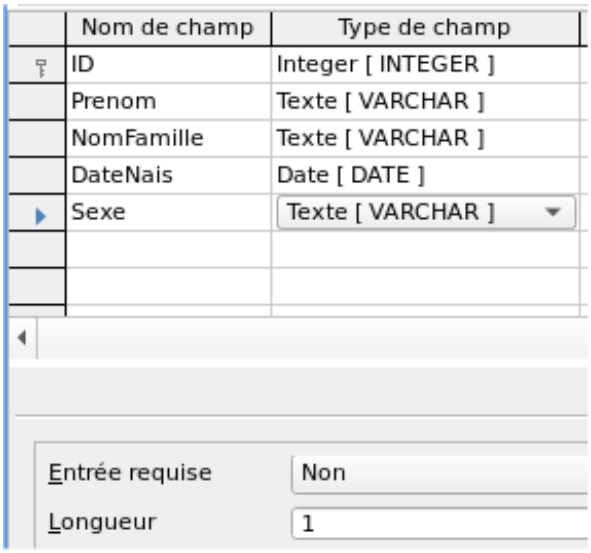
le type de champ (champ ID) est laissé sous forme de texte, mais limité à 5 caractères. Les 5 caractères suffisent, afin de trouver une abréviation appropriée pour les différents sports.

Ouvrez à nouveau la table Candidats pour la modifier (**Éditer**), à l'aide du menu contextuel de la table.

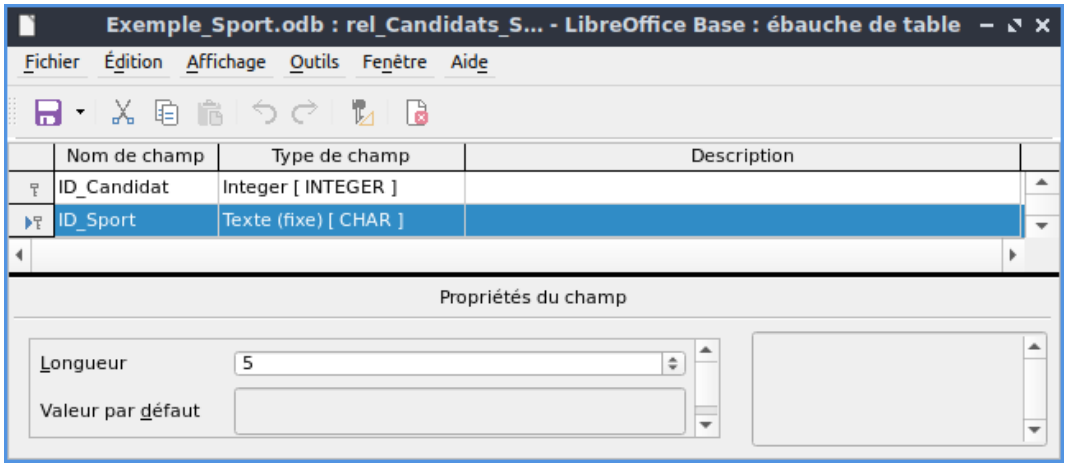
Ajoutez le champ "Sexe" à la table. Un nouveau champ peut être ajouté à la boîte de dialogue de conception de la table, uniquement à la fin de la définition de la table. Il est également possible d'utiliser SQL pour ajouter de nouveaux champs à certaines positions.
La longueur du texte dans ce champ est limitée à un caractère, suffisant pour "m" et "f" comme entrée.

D'une manière ou d'une autre, les deux tables doivent être liées afin que chaque Candidat puisse être inscrit dans plusieurs sports et que plus de candidats puissent être inscrits pour n'importe quel sport. Cela se fait via une table dans laquelle les valeurs des deux clés primaires des tables Candidats et Sport sont enregistrées. Étant donné que seule la combinaison de ces champs sera enregistrée ensemble, ces champs sont la clé primaire de cette table.



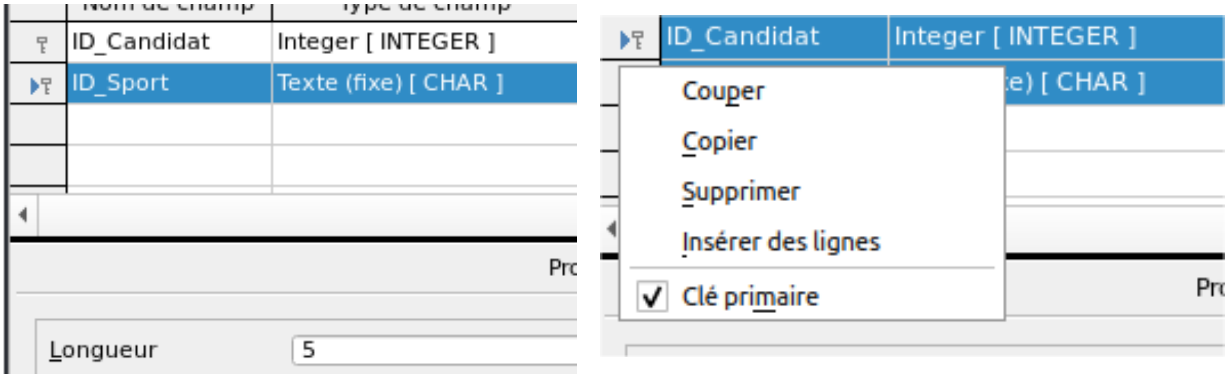


Nous allons créer cette table sous le nom rel_Candidat_Sport, avec deux champs, ID_Candidat et ID_Sport, ***du même type*** que les champs ID des tables Candidats et Sports.



Les valeurs appropriées peuvent être extraites de Candidats et Sports, car les champs doivent correspondre exactement aux types de champs que vous souhaitez enregistrer. ID_Candidat doit donc avoir le type de champ Integer. ID_sport a le type de champ Texte et est également limité à 5 caractères, comme le champ ID de la table Sport.

Pour attribuer la clé primaire aux deux champs, cliquez sur l'en-tête de ligne du premier champ, puis sur Maj - clic sur l'en-tête de ligne du deuxième champ ; ceci sélectionne les deux champs. Cliquez avec le bouton droit sur l'un des en-têtes de ligne, puis cliquez sur Clé primaire dans le menu contextuel pour spécifier la clé primaire.



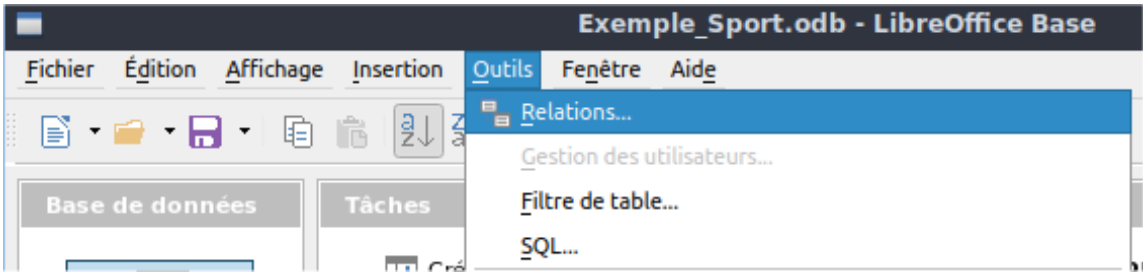
Enregistrez la table sous le nom **rel_Candidats__Sports.**

Conseil

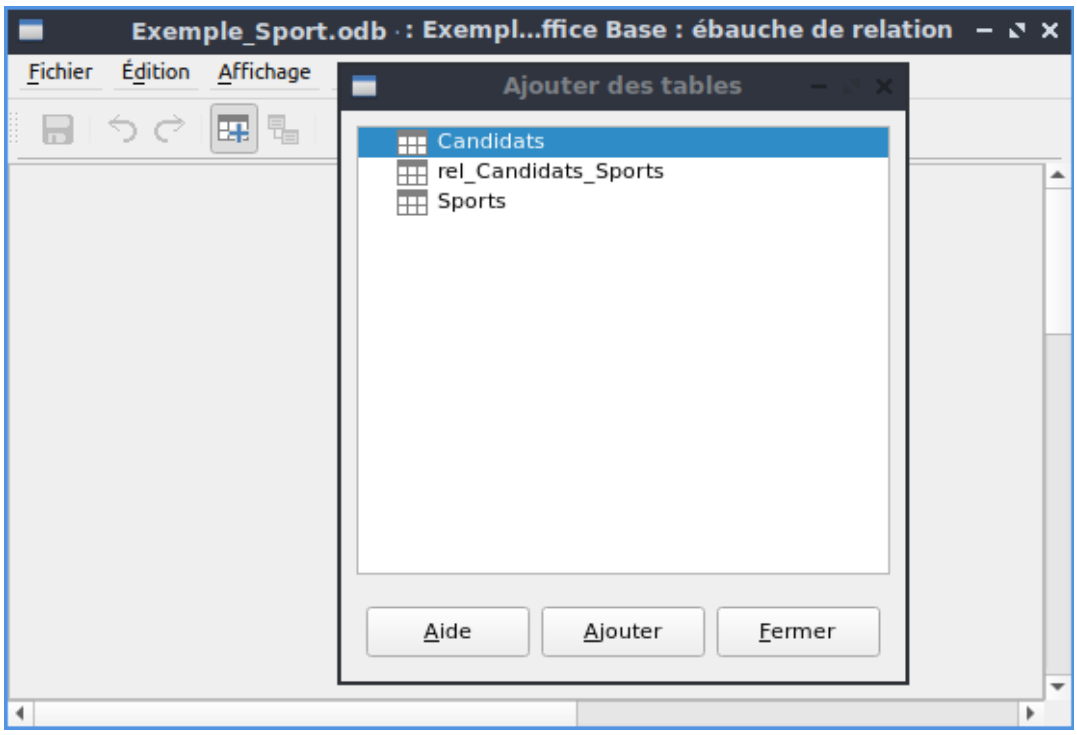
Les résultats d'un concours pourraient également être inclus dans ce tableau. Cependant, si plusieurs compétitions sont organisées, une date de course doit être attachée à la clé primaire commune.

Au fur et à mesure que les tables sont complétées, une relation entre les tables doit être définie. Cela peut empêcher un numéro pour un Candidat d'apparaître dans la table rel_Candidats_Sports qui n'est pas répertorié dans la table Candidats, par exemple.

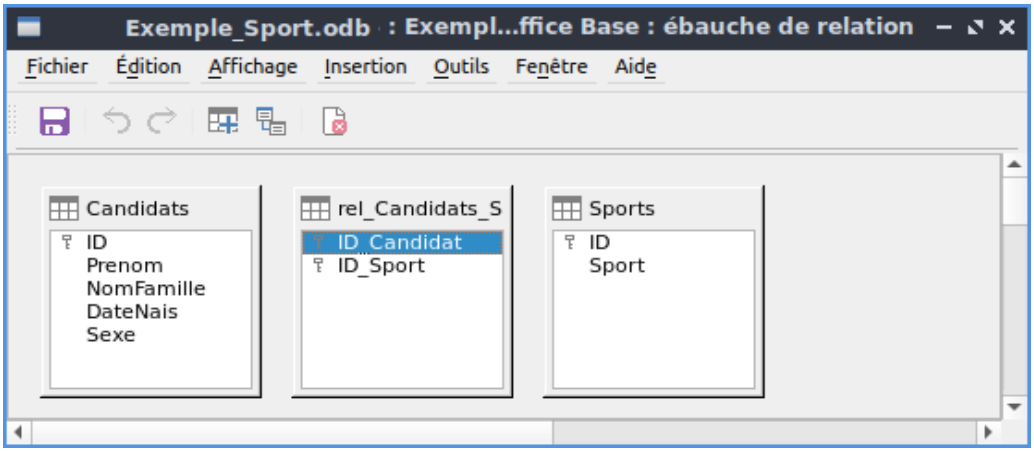
Outils > Relations ouvre la fenêtre de définition des relations



Toutes les tables créées jusqu'à présent sont nécessaires pour la définition de la relation. Cliquez sur chaque table individuelle et cliquez sur le bouton **Ajouter** (où double-cliquer) pour les ajouter à la conception de relations. Fermez ensuite la boîte de dialogue **Ajouter des tables**.

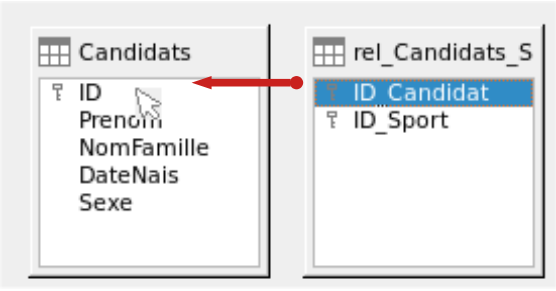


Tous les champs sont répertoriés dans chacune des tables ajoutées. Les champs de clé primaire sont marqués d'un symbole de clé. Les rectangles des tables peuvent être déplacés et redimensionnés.

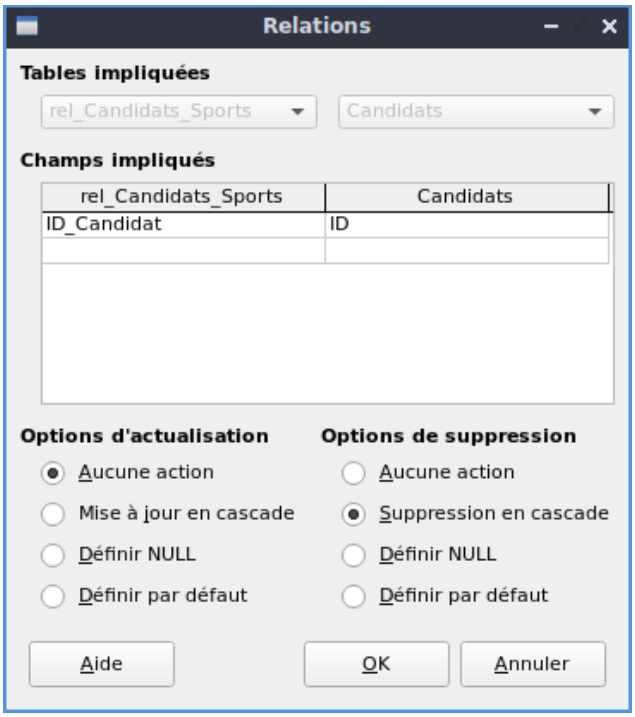


Note

Dorénavant, pour distinguer les champs en fonction des tables auxquels ils appartiennent, nous les désignerons par "Table"."Champ" (Syntaxe SQL dans HSQL).



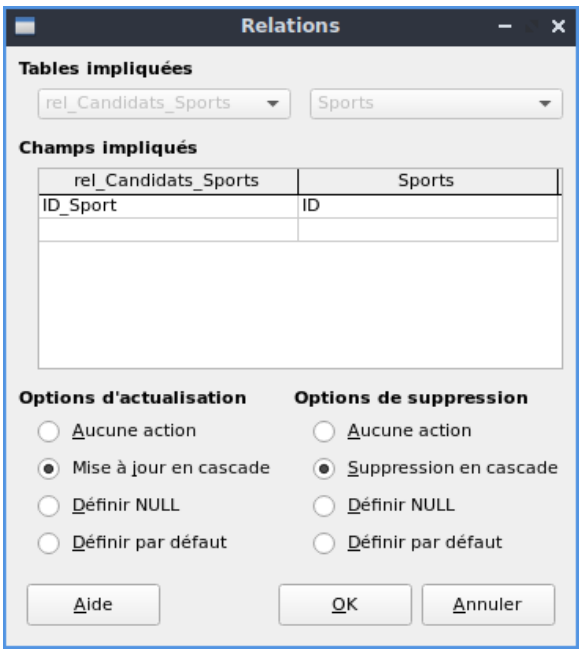
Nous allons créer une relation entre les tables. Faites un clic gauche sur “Candidats”. “ID” (table Candidats). Maintenez le bouton de la souris enfoncé et placez le pointeur sur “rel_Candidats_Sports”. “ID_Candidat”. Le curseur indique un lien. Relâchez le bouton de la souris. La boîte de dialogue suivante apparaîtra pour définir la relation.



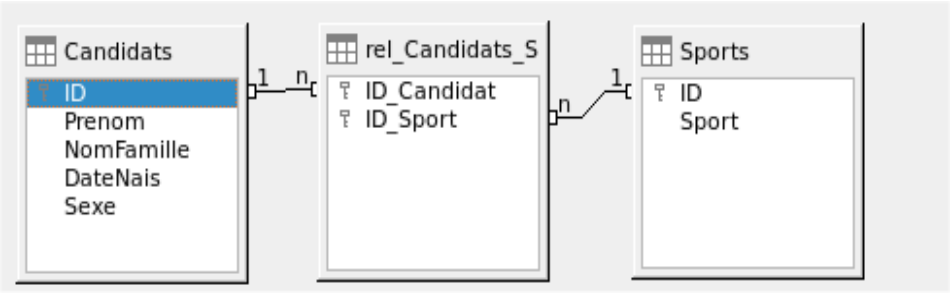
Le champ “rel_Candidats_Sports”.“ID_Sport” ne doit pas être modifié lorsque “Candidats”.“ID” est modifié. C’est la valeur par défaut. “ID” n’est pas modifié, car il s’agit d’un champ auto-incrémenté et aucune entrée n’est nécessaire.

L’enregistrement dans la table rel_Candidats_Sports doit être supprimé lorsque “ID_Candidat” est égal à “Candidats”.“ID” et “Candidats”.“ID” est supprimé. Donc, si un candidat est supprimé de la table Candidats, tous les enregistrements pertinents de la table rel_Candidats_Sports seront supprimés. Cette procédure s’appelle **Suppressions en cascade**.

À l’étape suivante “Sports”. “ID” et “rel_Candidats_Sports”.“ID_sport” sont connectés en faisant glisser la souris depuis rel_Candidats_Sports vers Sports, tout en maintenant le bouton gauche de la souris. Ici aussi, un enregistrement sera supprimé lorsque le sport correspondant sera supprimé.



Cependant, pour un changement de données dans “Sports”.“ID”, une autre variante a été sélectionnée. Si “Sports”.“ID” est modifié, “rel_Candidats_Sports”.“ID_sport” le sera également. Ainsi, l’abréviation d’un sport de 5 caractères maximum peut être facilement ajustée bien qu’il existe déjà de nombreux enregistrements dans la table rel_Candidats_Sports. Cette procédure s’appelle **Mise à jour en cascade**.

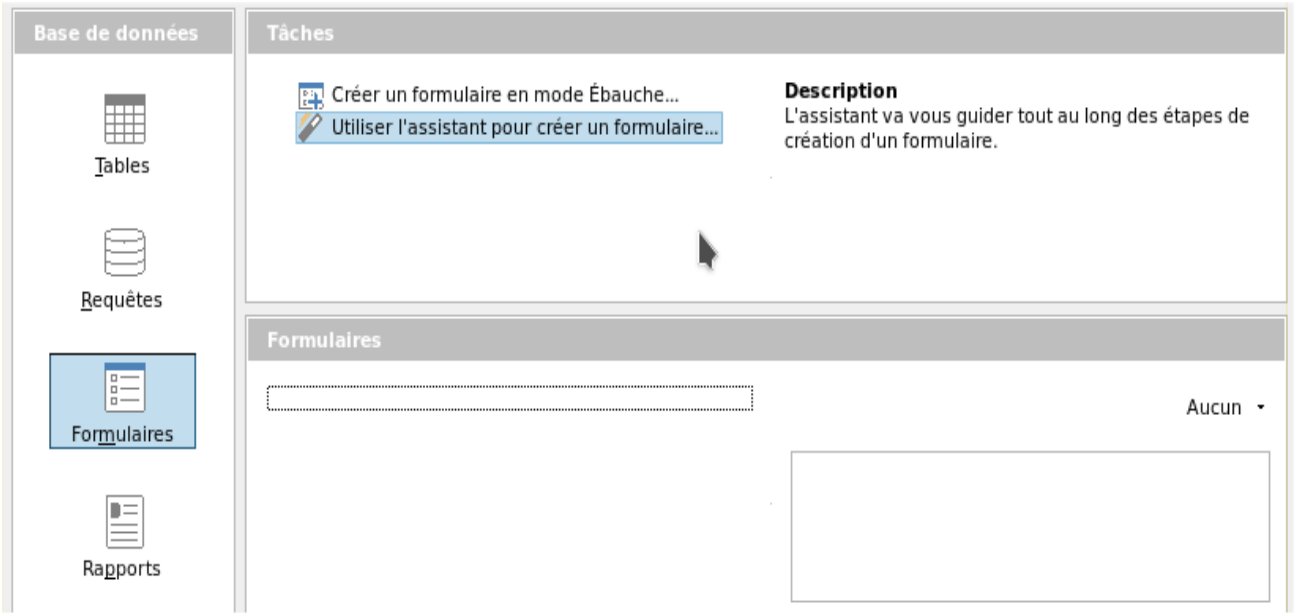


Les champs sont maintenant complètement connectés. Chaque 1:n apparaît aux extrémités des connexions. Un candidat peut apparaître à plusieurs reprises dans la table rel_Candidats_sport. Un sport peut également apparaître à plusieurs reprises dans la table rel_Candidats_Sports. Une combinaison donnée de Candidat et Sport ne peut apparaître dans la table qu’une seule fois. À partir de deux relations 1:n, une relation n:m existe désormais entre Candidats et Sport via la table rel_Candidats_Sports intermédiaire.

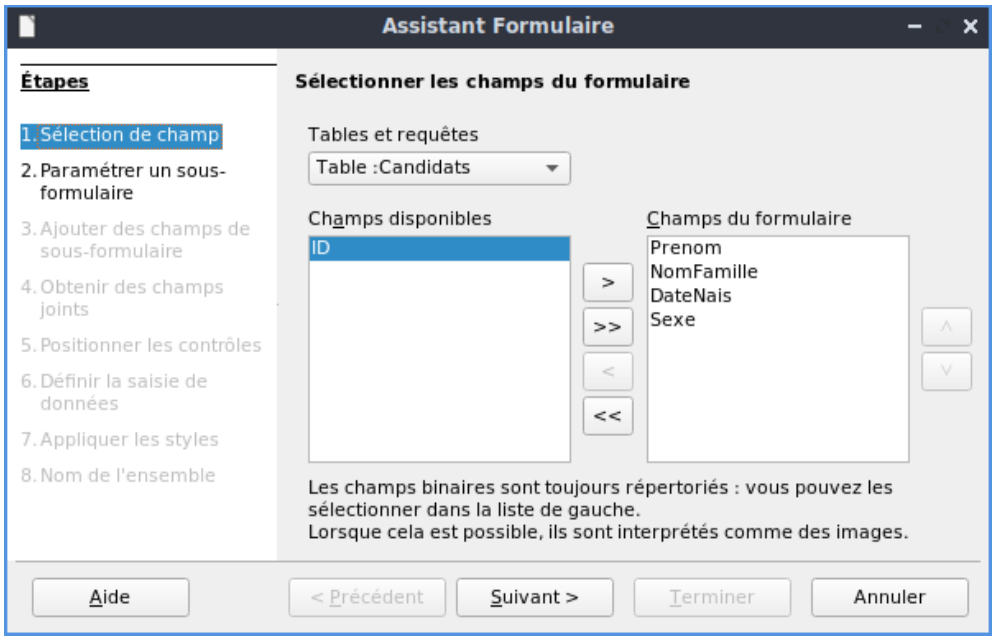
Une telle conception de table peut être mauvaise lorsqu’elle est remplie en tapant du contenu directement dans des tables. Cela nécessite que les trois tables soient ouvertes lorsqu’un candidat est affecté à un sport. “Candidats”.“ID” doit être recherché dans la table Candidats et transmis à “rel_Candidats_Sports”. “ID_Candidat” et “Sports”.“ID” doit être recherché dans la table Sports et transmis à “rel_Candidats_Sports”. C’est trop compliqué. Un formulaire résout ce problème de manière plus élégante.

Créer un formulaire de saisie de données

Les formulaires peuvent être créés directement dans la vue du mode ébauche ou à l'aide de l'assistant. Même les personnes expérimentées peuvent utiliser rapidement l'assistant, puis personnaliser ce qu'il crée pour produire ce qu'elles veulent. C'est souvent le moyen le plus rapide.



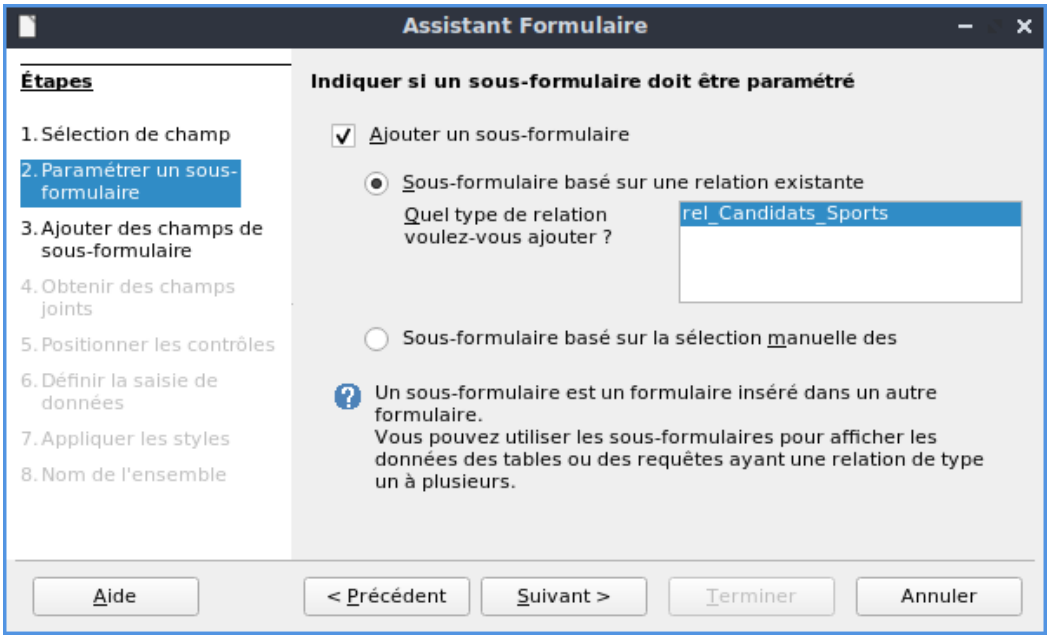
Sélectionnez d'abord Formulaires dans la section Base de données. Ensuite, dans la section **Tâches**, sélectionnez **Utiliser l'assistant pour créer un formulaire**.



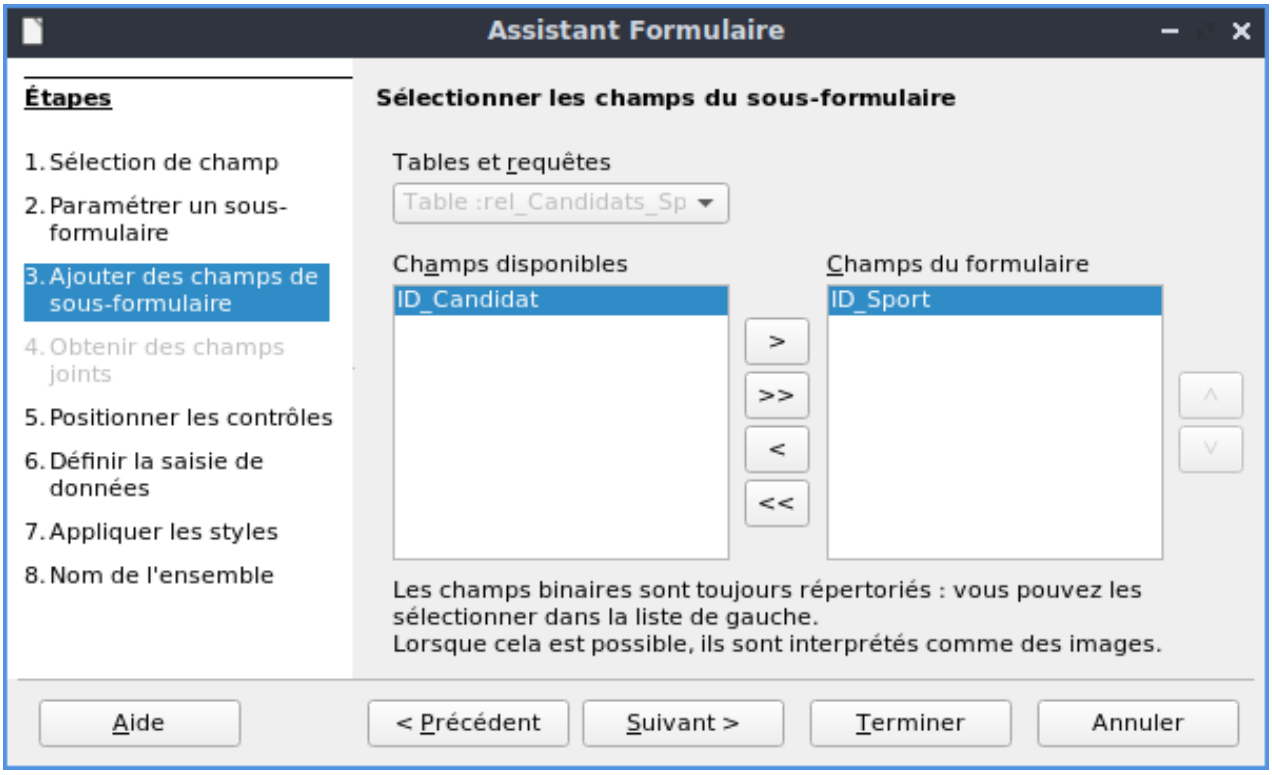
Les données de la table Candidats doivent être écrites dans le formulaire principal. Les données du tableau Sport sont chargées directement avec les quelques sports nécessaires et seront rarement mises à jour.

Tous les champs à l'exception du champ de clé primaire "ID" sont nécessaires à partir de la table Candidats. Le champ de clé primaire est rempli automatiquement avec une valeur distinctive correspondante.

Sélectionnez les champs dans la liste *Champs disponibles* et utilisez les boutons fléchés pour les déplacer vers la liste des *Champs du formulaire*. Cliquez sur **Suivant**.

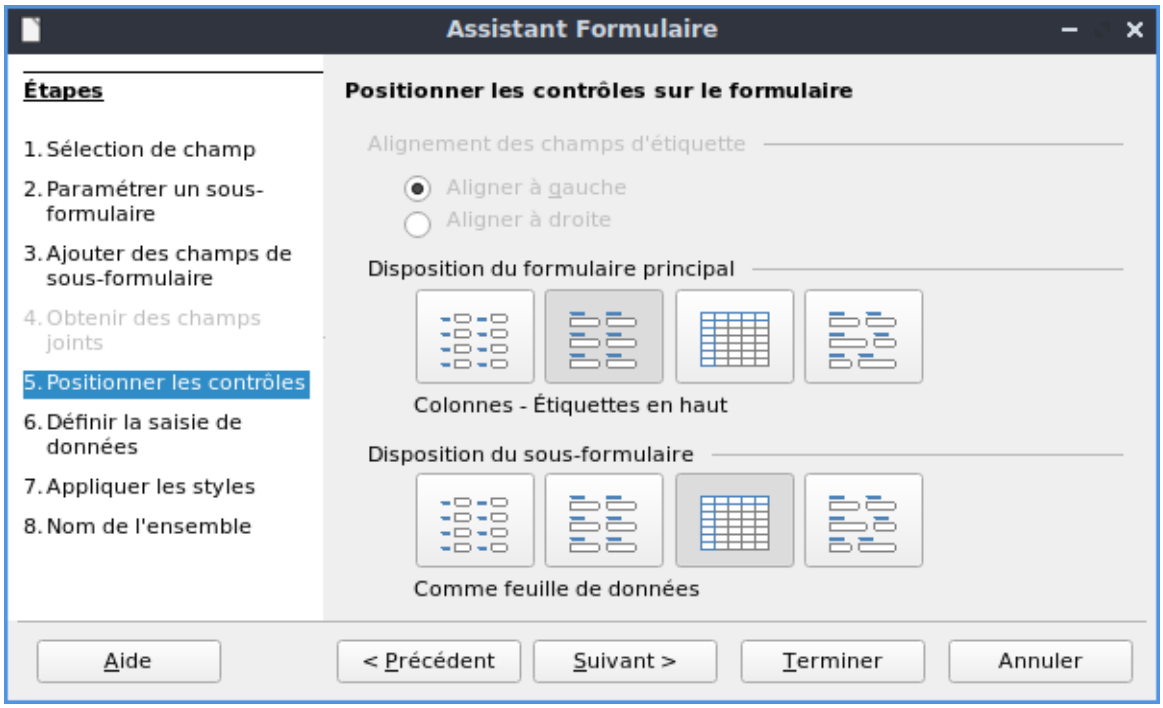


Un sous-formulaire doit être mis en place où un sport peut être attribué à un Candidat. À l'étape 2, sélectionnez Ajouter un sous-formulaire en fonction de la relation existante. Pour confirmer la relation précédemment définie, sélectionnez rel_Candidat_Sports. Cliquez sur **Suivant**.



Seul le champ ID_Sport est requis dans la table rel_Candidats_Sports. La clé primaire dans la table Candidats fournit la valeur du champ ID_Sport pour l'enregistrement en cours par la connexion du formulaire principal au sous-formulaire.

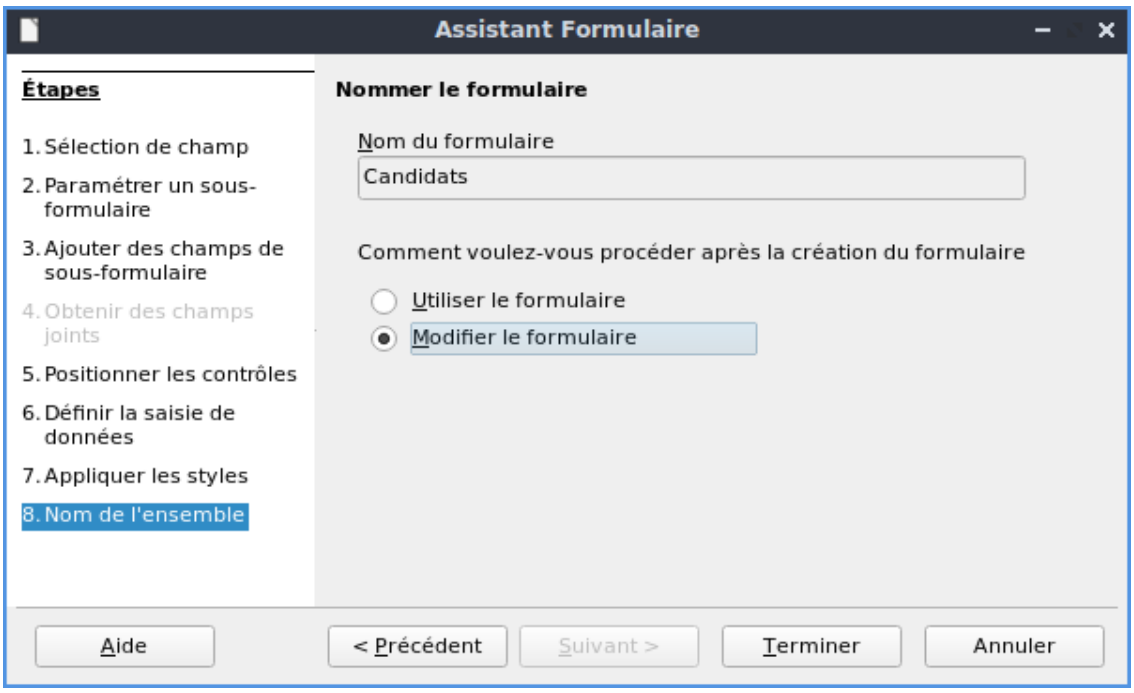
Cette connexion est déjà réglée, car le sous-formulaire est basé sur une relation existante. L'étape 4 de l'assistant : *Obtenir les champs joints* est donc inactif. Si le sous-formulaire est basé sur la sélection manuelle des champs, l'étape 4 est alors active et permet de réaliser la connexion manuellement (choix à déconseiller dans le présent exemple).



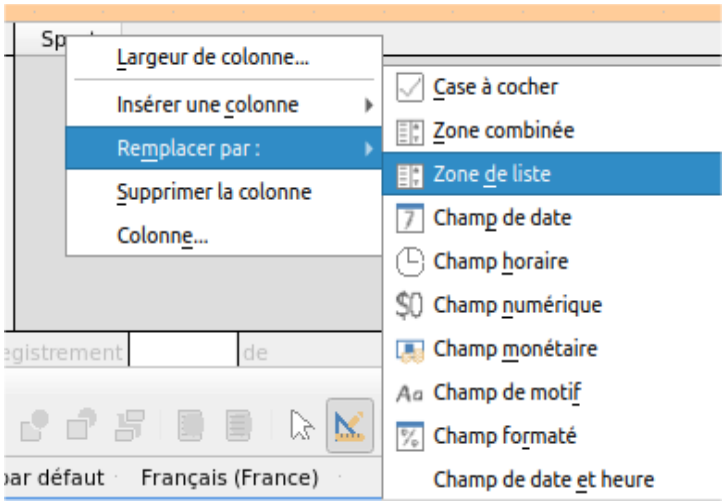
La manière dont les éléments du formulaire principal et du sous-formulaire sont organisés n'a pas d'importance en fin de compte. Cependant, l'attribution des données doit également être simple pour les inexpérimentés. Dans notre exemple, **Comme feuille de données** ne doit pas être sélectionné pour le formulaire principal, à la place choisissez **Colonnes – Étiquettes en haut**. Les champs du sous-formulaire afficheront plus tard tous les sports des Candidats, il est donc préférable de laisser la disposition du sous-formulaire telle quelle : **Comme feuille de données**.

Étape 6 (Définir la saisie des données) doit rester telle quelle : **Le formulaire doit afficher toutes les données**. Ainsi, une nouvelle entrée est possible, ainsi qu'une modification des données existantes.

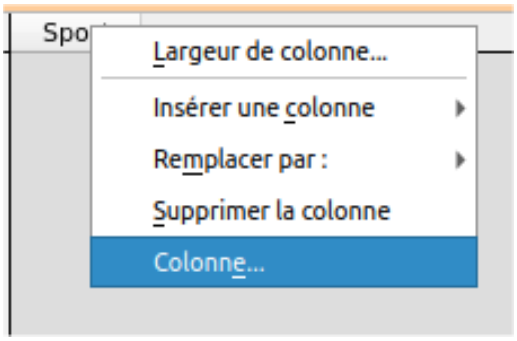
Étape 7 (Appliquer les styles). C'est une question de goût. Attention : certains styles impliquent des images inattendues à faible contraste, en particulier dans les champs de contrôle de table. Ici, la couleur de police des champs de la feuille de données doit être réajustée si nécessaire.



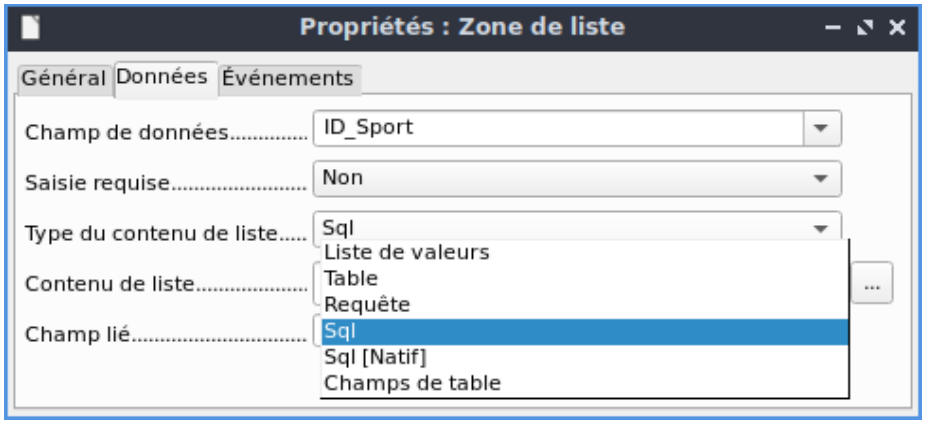
Le formulaire ne fonctionnera pas encore, car le sous-formulaire vous oblige toujours à entrer l'abréviation pour les sports. Ce qu'il faut, c'est une sélection de sports après le nom complet souhaité. Par conséquent, à l'Étape 8 (Définir le nom), nommez le formulaire **Candidats** et sélectionnez **Modifier le formulaire**. Cliquez sur **Terminer**.



Sur la feuille de données, cliquez droit sur l'en-tête du tableau, ID_Sport. Dans le menu contextuel, sélectionnez **Remplacer par > Zone de liste**.

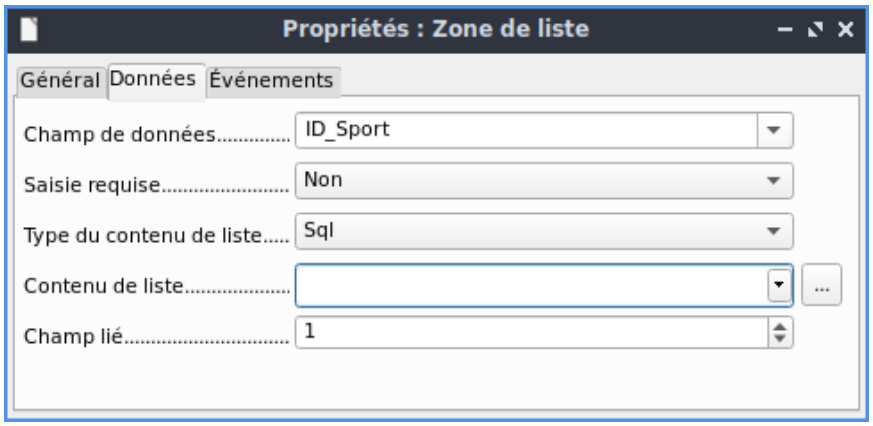


Ensuite, la liste doit être traitée afin qu'elle puisse également afficher les données correspondantes. Cliquez à nouveau avec le bouton droit sur ID_Sport et sélectionnez **Colonne**.



Cela ouvre les propriétés de la zone de liste sélectionnée. Sélectionnez **Données > Type du contenu de la liste > Sql**. Avec l'aide de SQL (Structured Query Language – langage de requête standard pour les bases de données), le champ doit obtenir son contenu de la table Sport.

Lorsque Sql est sélectionné, le *Contenu de liste* a un bouton d'ellipse sur la droite (…). Cliquez sur ce bouton pour ouvrir l'éditeur de création de requêtes. La requête en cours de création sera rassemblée et finalement enregistrée dans la zone de liste elle-même.



Dans la boîte de dialogue Ajouter une table ou une requête, sélectionnez **Tables > Sports**, puis cliquez sur **Ajouter** et **Fermer**.

Dans la première colonne de la partie inférieure de l'éditeur de requêtes, cliquez dans la case en regard de Champ et sélectionnez Sports.Sport dans la liste déroulante.

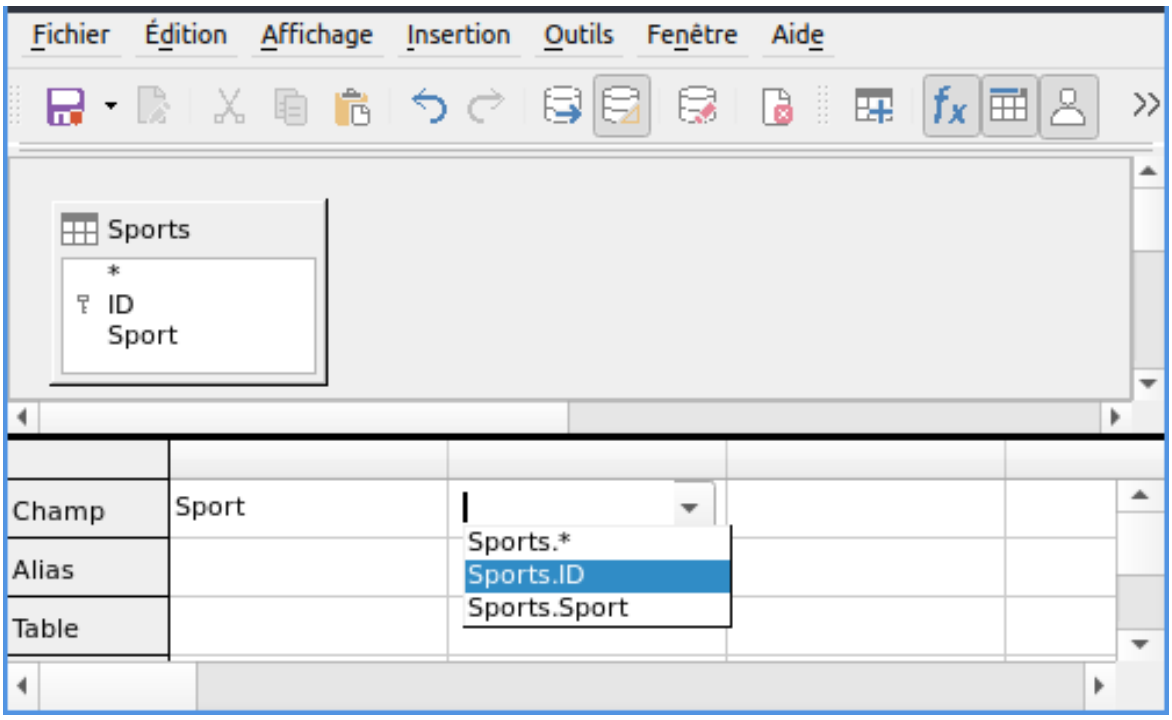
Dans la deuxième colonne, sélectionnez le champ "Sports"."ID". Ce champ transmet ensuite sa valeur à la table, qui est la source de données du sous-formulaire. Les mots inscrits sont donc affichés et les raccourcis appropriés stockés.

Enregistrez la requête (Icône disquette), qui est ensuite transmise aux propriétés de la zone de liste. Fermez l'éditeur de requêtes.

Le champ *Contenu de liste* de la boîte de dialogue **Propriétés** affiche maintenant le code SQL qui a été créé dans l'éditeur de requête :

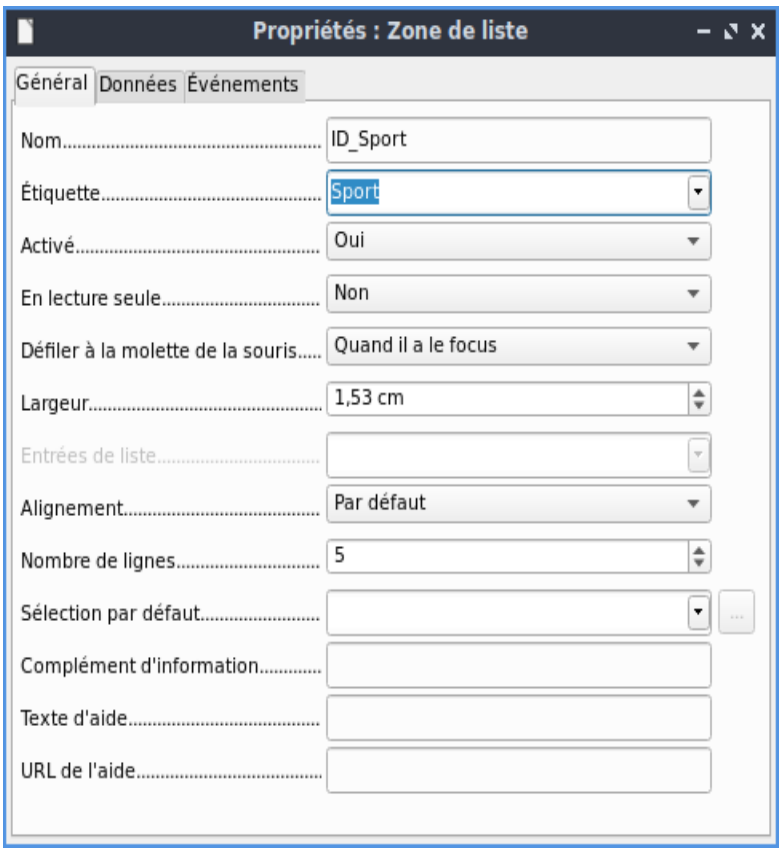
```
SELECT "Sport","ID" FROM "Sports"
```

Ce code dit : Dans la table "Sports", sélectionner le champ "Sport" et la valeur clé associée "Sports"."ID".

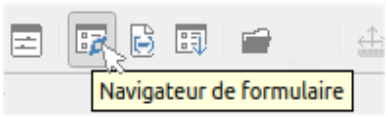


Cette requête illustre le minimum qui doit être sélectionné. Bien entendu, le tri pourrait être intégré. La sauvegarde des abréviations avec une sélection habile donne une liste utile des sports stockés dans “ID”. Si les enregistrements ne sont pas triés d’une manière spécifiée, le tri est toujours effectué par le champ de clé primaire. Afin de voir les sports plus tard dans la zone de liste, ce contenu doit être entré dans la table Sports en conséquence.

Nous devons maintenant changer le libellé du champ ID_Sport. Il est toujours stocké sous ID_Sport, mais nous voulons qu’il soit visible en tant que sport. Par conséquent : **Général > Étiquette > Sport**. Fermez la boîte de dialogue Propriétés.



Si vous souhaitez modifier les noms d'autres champs, il est préférable d'utiliser le Navigateur de formulaires. Si vous cliquez sur des champs, non seulement les champs, mais également les étiquettes sont sélectionnées. Grâce à l'assistant, ils ont été regroupés. Cela nécessite ensuite une action supplémentaire du menu contextuel de la sélection.



Le Navigateur de formulaire est lancé à partir de la barre d'outils de conception de formulaire en bas de la fenêtre.

Conseil

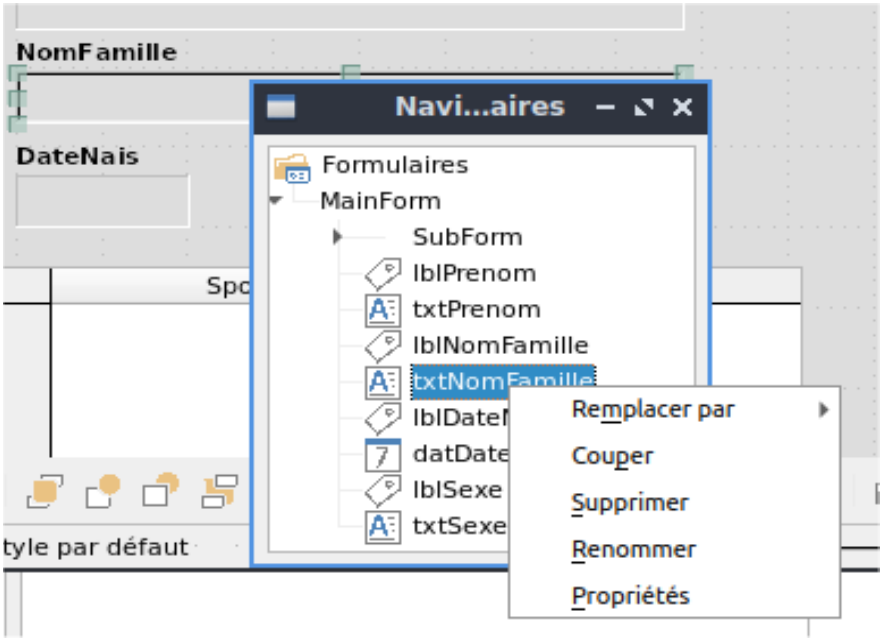
Parfois, la barre d'outils ne s'ouvre pas correctement pour créer des formulaires lors du traitement des formulaires. Le navigateur de formulaire est introuvable dans ce cas.

Pour afficher les barres d'outils, choisissez **Affichage > Barres d'outils > Ébauche de formulaire** et **Contrôles de formulaire**. S'ils sont ensuite visibles lors de la saisie des données, la vue doit à nouveau être modifiée en conséquence.

L'usage du Navigateur de Formulaires est fortement conseillé car, car il permet de naviguer facilement entre les contrôles et d'accéder aux paramètres des formulaires.

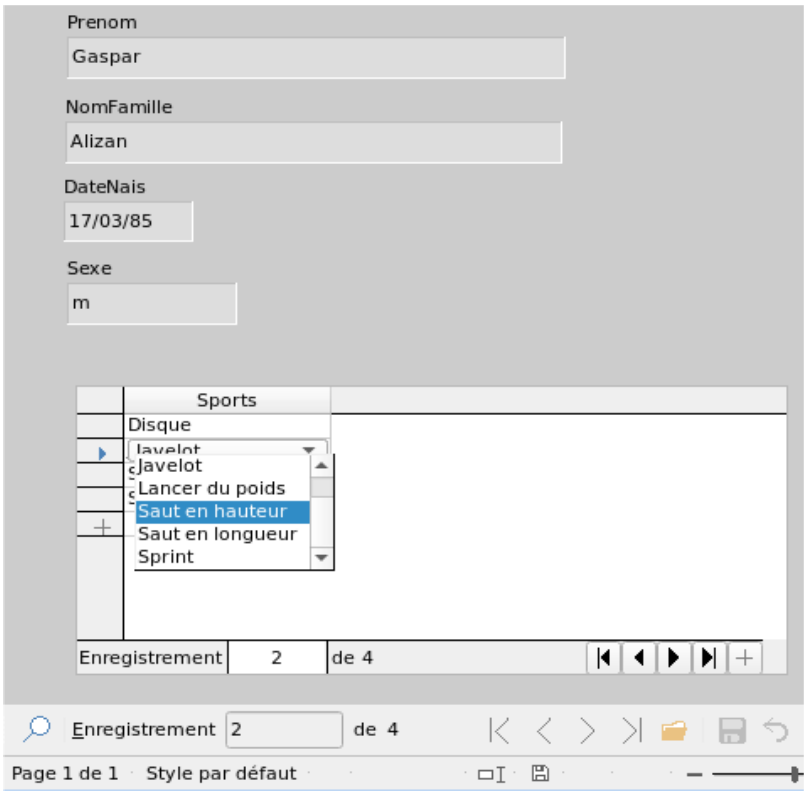
Chaque champ peut être examiné individuellement avec le Navigateur de formulaires. Les propriétés du champ sont alors accessibles dans le menu contextuel. Une entrée de propriété est automatiquement enregistrée après avoir accédé à une autre propriété. Il est possible de passer

d’un champ à un autre même lorsque la boîte de dialogue des propriétés est ouverte. Ici aussi, le niveau intermédiaire respectif est stocké.



Si la conception est maintenant terminée, enregistrez et fermez le formulaire. Puis enregistrez à nouveau le fichier de base.

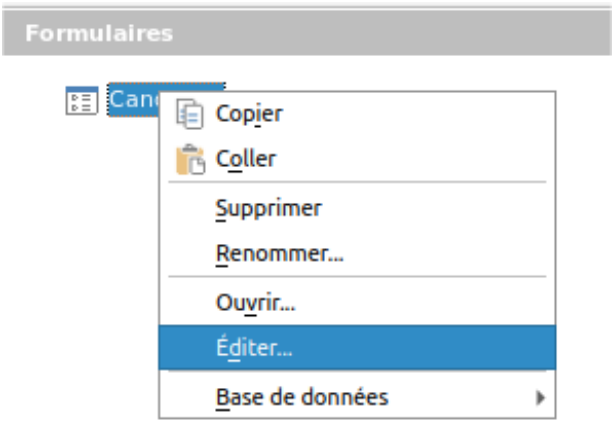
Maintenant, si vous souhaitez saisir des données dans le formulaire, cela peut ressembler à ceci. Dans le formulaire suivant, un enregistrement a été saisi pour le test. Après avoir entré le sexe, cliquez sur le bouton **Enregistrer**. Bien entendu, le tri pourrait être intégré.



Lors de l’utilisation du formulaire, vous remarquerez quelques inconvénients :

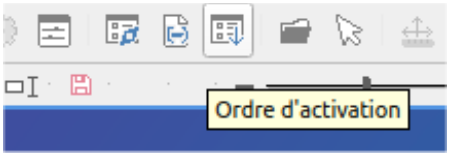
- Le sous-formulaire avec Sports n'est pas directement accessible. Si vous naviguez avec la touche Tab, après avoir entré le sexe, l'onglet passe directement au début de l'enregistrement suivant.
- La barre de navigation, si on se trouve dans le sous-formulaire, affiche le numéro d'enregistrement du sous-formulaire. Vous devriez naviguer uniquement dans le formulaire principal.
- Le sexe est entré en fonction de la préférence de l'utilisateur trouvée en mémoire. Dans le tableau, la longueur du champ a été limitée à un seul caractère. Donc, ici, une entrée plus sûre doit être garantie.

Pour résoudre ces problèmes, ouvrez le formulaire pour le modifier, non pour la saisie de données.

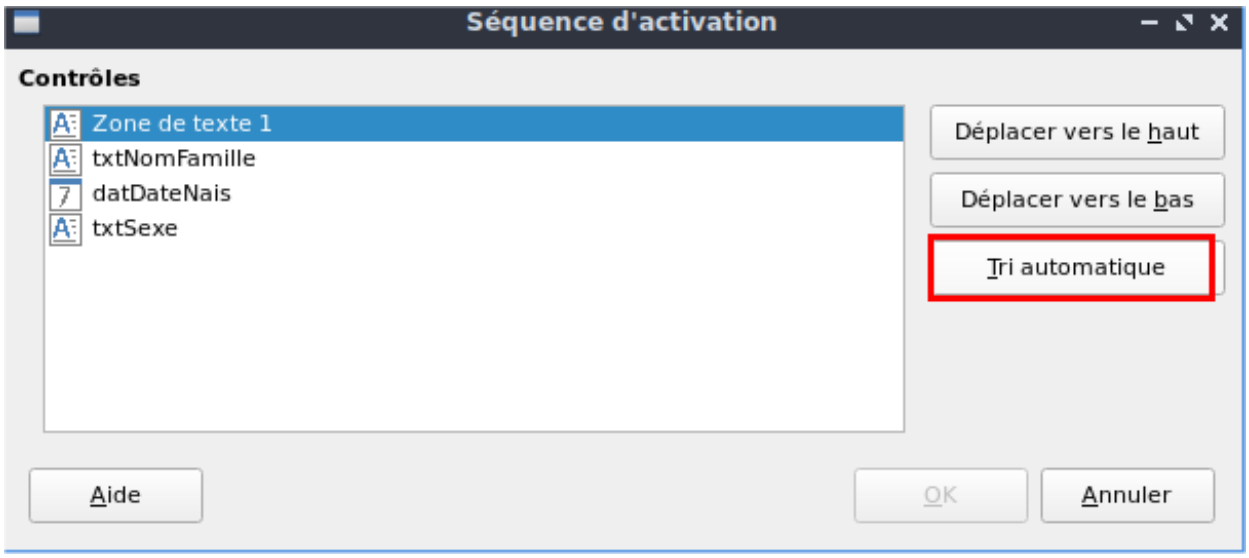


Tabulation dans le sous-formulaire

Afin de ne pas passer directement à l'enregistrement de démarrage suivant en utilisant la touche Tab après avoir saisi le sexe, il est nécessaire de modifier la séquence d'activation. Cliquez sur Ordre d'activation dans la barre d'outils Ébauche de formulaire.

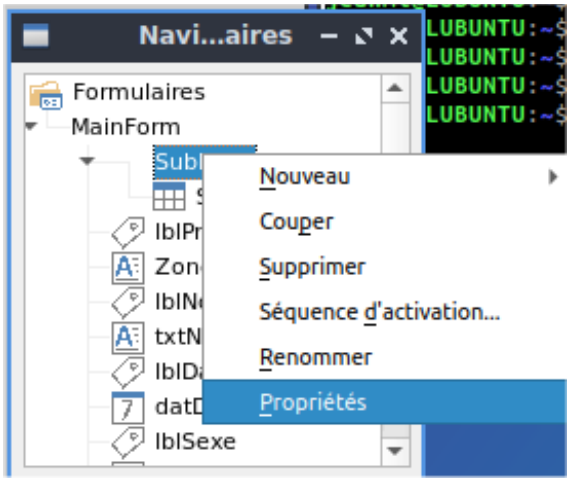


Dans la boîte de dialogue Ordre de tabulation, sélectionnez Tri automatique, qui affecte non seulement le tri des contrôles affichés, mais également la redirection automatique dans le sous-formulaire. Bien que cela ne soit pas visible dans la boîte de dialogue, il est réglé de cette manière en arrière-plan.

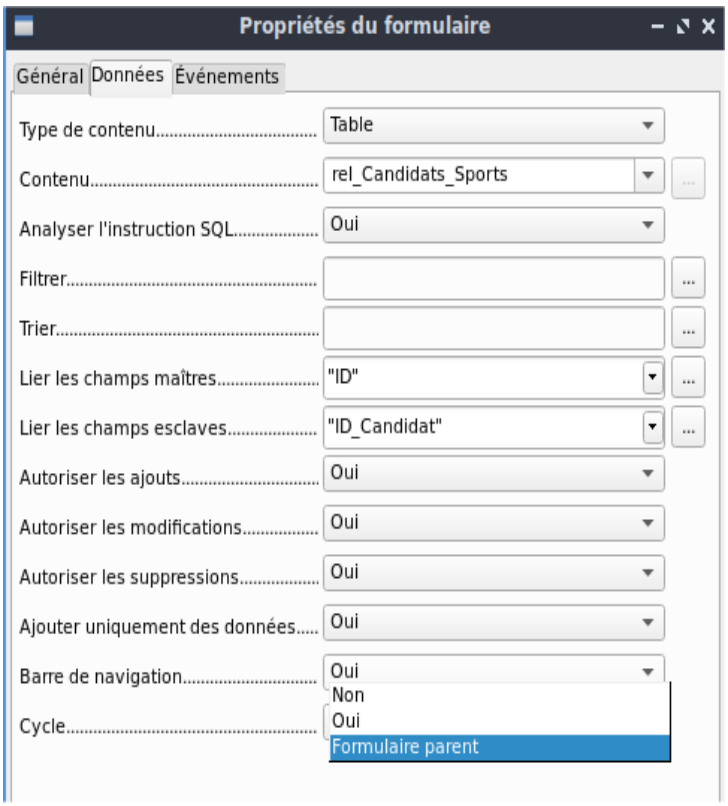


Activer la barre de navigation du formulaire principal dans le sous-formulaire

Ouvrez le navigateur de formulaires pour afficher les propriétés du sous-formulaire.



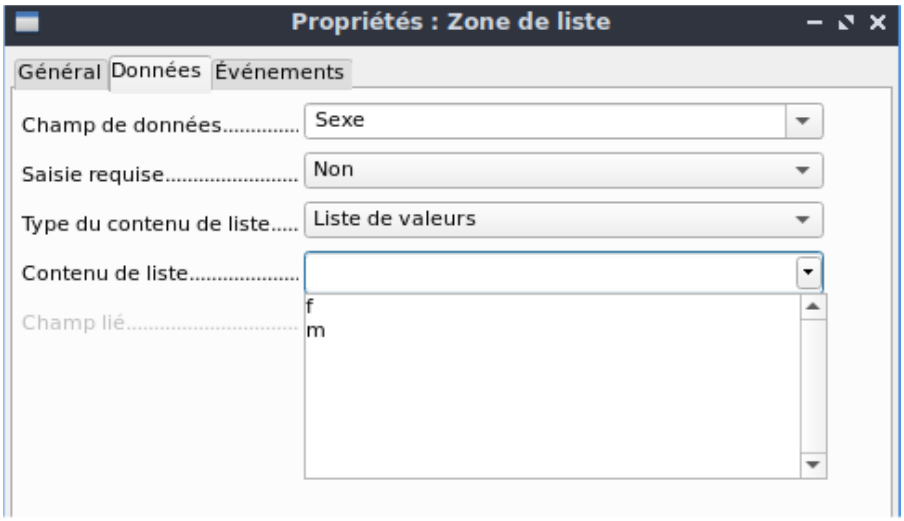
Sous l'onglet **Données > Barre de navigation,** modifiez la valeur de *Oui* à *Formulaire parent*. Désormais, la barre d'outils de navigation affiche toujours le numéro de l'enregistrement de la table Candidats.



Restreindre la saisie dans un contrôle

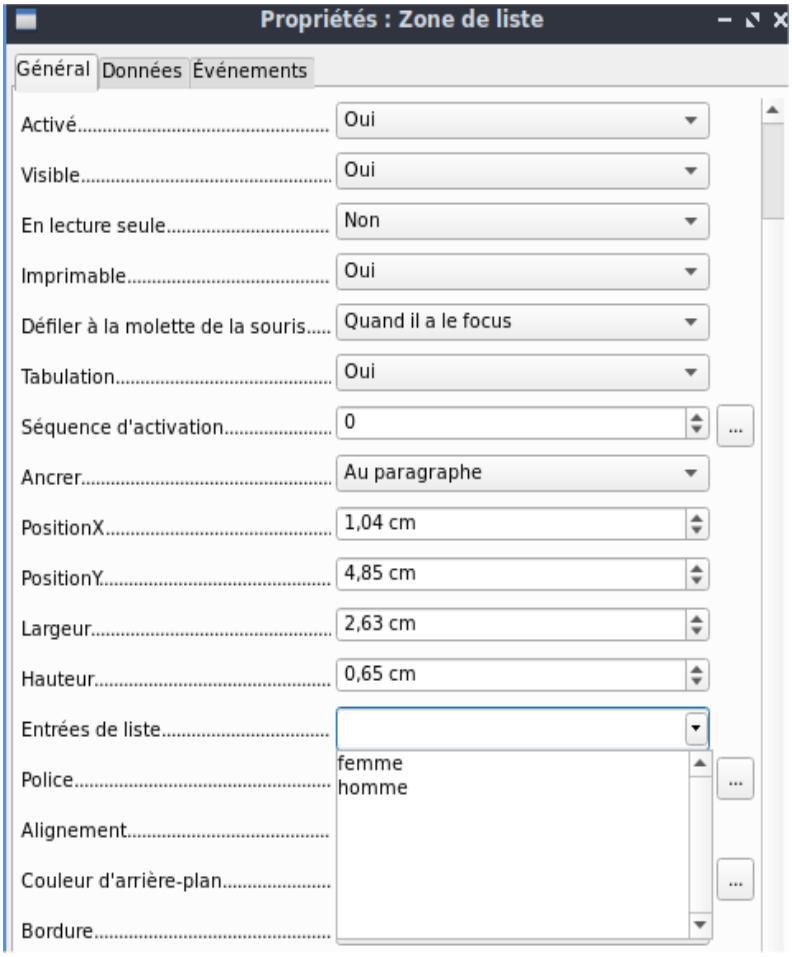
Pour limiter l'entrée aux valeurs spécifiées, le contrôle ne peut pas être une simple zone de texte. Dans le formulaire principal, la solution peut être trouvée en saisissant le sexe avec des boutons radio dans une boîte de groupe. Une autre solution consiste à présenter les choix dans une zone de liste.

Tout d'abord, mettez en surbrillance le champ Sexe sur le navigateur. Cliquez avec le bouton droit sur ce champ pour ouvrir le menu contextuel. Sélectionnez **Remplacer par> Zone de liste**. Les propriétés seront sélectionnées à l'aide du menu contextuel.



Onglet **Données > Type de contenu de liste > Liste de valeurs** est par défaut dans cette propriété. Dans **Données > Contenu de liste**, entrez les caractères f et m l'un en dessous de

l’autre (en utilisant Maj + Entrée). Ces abréviations sont les données qui seront affectées à la table Candidats, ou récupérées dans cette table pendant la lecture des enregistrements.



Dans l’onglet **Général > Entrées de liste**, spécifiez ce qui sera affiché dans la zone de liste. Cela peut également être f et m, ou il peut s’agir de femme et homme comme indiqué sur l’illustration. Il doit en tout cas avoir le même ordre que les éléments de l’onglet **Données**.

Sélectionnez ensuite **Oui** pour la propriété **Général > Liste déroulante**. Regardez vers le bas de l’onglet Général pour cette propriété.

Les inconvénients les plus frappants ont jusqu’à présent été éliminés. Maintenant, vous devez à nouveau enregistrer le formulaire et le fichier de base de données. La saisie pour les candidats masculins et féminins peut commencer ainsi que leur affectation aux sports.

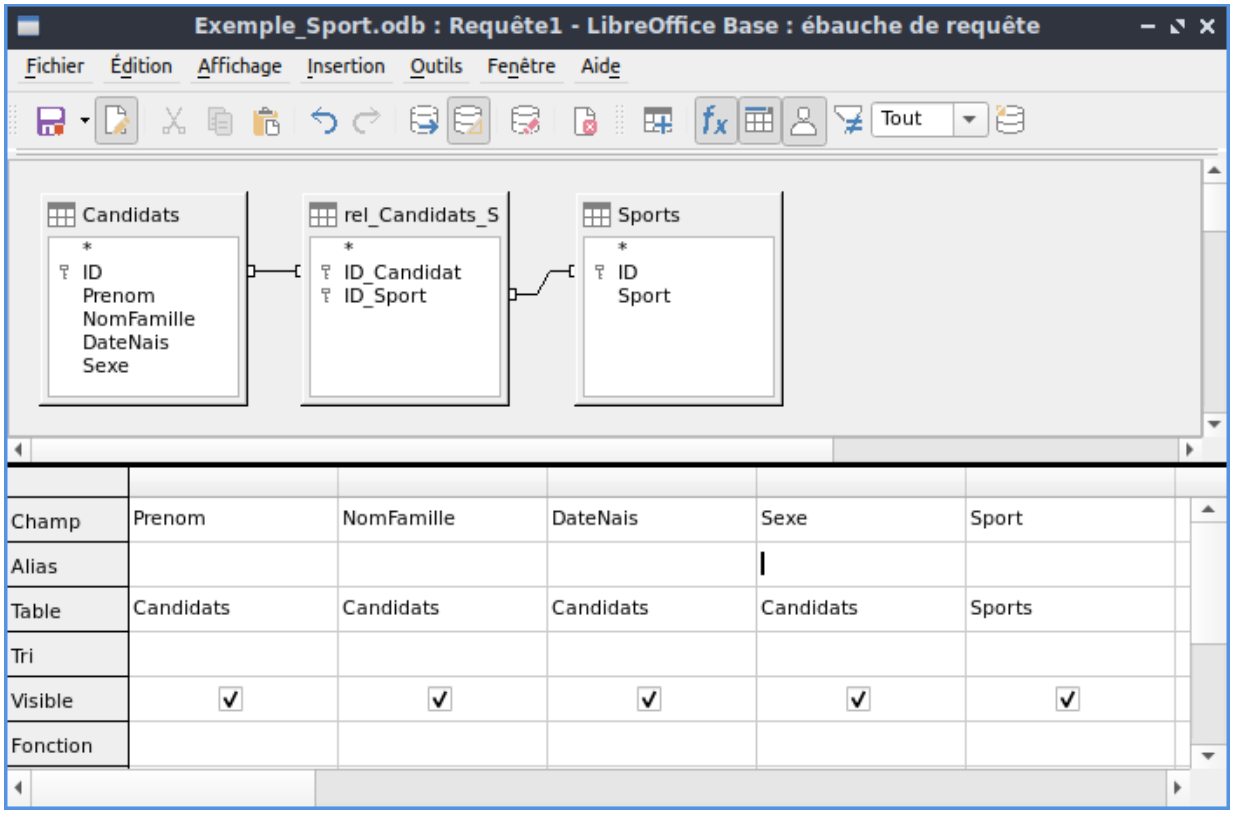
Ainsi, l’étape suivante est utile : les enregistrements ne doivent être saisis qu’une seule fois. Veillez à ce que les candidats puissent également s’affronter par âge et par sport. Sinon, les requêtes et rapports suivants n’ont pas de sens.

Créer une requête

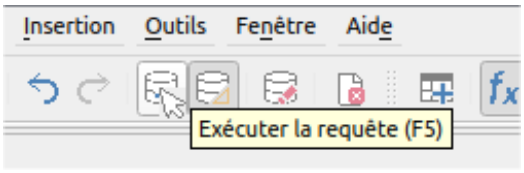
Dans une requête, les contenus de différentes tables peuvent être regroupés. Chacun des candidats doit être affiché avec les sports auxquels il a participé.

Cliquez sur **Requêtes > Créer une requête en mode Ébauche**. Une boîte de dialogue, listant toutes les tables de la base données, apparaît et dans laquelle nous sélectionnerons, une à une, toutes les tables utiles pour la requête. Cela sera très clair lorsque la table rel_Candidats_Sports

est sélectionnée comme deuxième table. Ensuite, on voit que les connexions des relations sont également bien reconnues.



Les champs qui doivent être affichés dans la requête peuvent être ajoutés soit en double-cliquant sur les noms de champs dans la table, soit en sélectionnant une case dans la ligne Champ. Une liste déroulante apparaît, composée des noms de champ et de leurs noms de table correspondants indexés par nom de table. Afin de faire correspondre correctement les champs des tables à leurs tables, ils sont étiquetés “Nom de table”.“Nom de champ” dans les requêtes. Si “*” est utilisé à la place du nom de champ, cela signifie que tous les champs de la table correspondante sont affichés.



En principe, une requête doit être exécutée une fois avant l’enregistrement pour voir si elle produit réellement le résultat souhaité. À cet effet, trois possibilités sont offertes :

- cliquez sur le bouton **Exécuter la requête** illustrée ci-dessus ;
- appuyez sur *F5*
- ou choisissez **Édition > Exécuter la requête**.

	Prenom	NomFamille	DateNais	Sexe	Sport
▶	Gaspar	Alizan	17/03/85	m	Disque
	Gaspar	Alizan	17/03/85	m	Javelot
	Gaspar	Alizan	17/03/85	m	Saut en hauteur
	Gaspar	Alizan	17/03/85	m	Sprint
	Amélie	Oration	07/04/78	f	Disque
	Amélie	Oration	07/04/78	f	Lancer du poids
	Amélie	Oration	07/04/78	f	Saut en longueur
	Amélie	Oration	07/04/78	f	Sprint
	César	Yennes	28/09/87	m	Course de fond

Enregistrement 1 de 35

La requête affiche désormais toutes les combinaisons de candidats et de sports valides. Si les candidats ont plusieurs sports, ils ont autant d'enregistrements. Les candidats sans sport n'apparaissent pas.

Afin de rassembler différentes tranches d'âge dans un concours, l'année de naissance est déterminante. Cela dépend de l'âge d'une personne cette année-là.

L'année de naissance peut être utilisée avec différentes fonctions. Par exemple, considérez les données suivantes.

Champ	Prenom	NomFamille	DateNais	Sexe	Sport	YEAR(NOW( )) - YEAR("DateNais")
Alias						
Table	Candidats	Candidats	Candidats	Candidats	Sports	

```
YEAR(NOW())-YEAR("DateNais")
```

NOW () signifie "maintenant", cette fonction retourne la date et l'heure système. L'année en cours est lue avec YEAR (NOW ()).
YEAR ("DateNais") extrait l'année de naissance. Une différence est calculée entre les deux, indiquant l'âge ou plus exactement l'âge de la personne dans l'année en cours.
Ces fonctions, et bien d'autres, qui fonctionnent avec le moteur HSQLDB intégré, sont décrites dans l'annexe A de ce manuel.

	Prenom	NomFamille	DateNais	Sexe	Sport	YEAR(NOW( )) - YEAR("DateNais")
▶	Gaspar	Alizan	17/03/85	m	Disque	35
	Gaspar	Alizan	17/03/85	m	Javelot	35
	Gaspar	Alizan	17/03/85	m	Saut en ha	35
	Gaspar	Alizan	17/03/85	m	Sprint	35
	Amélie	Oration	07/04/78	f	Disque	42
	Amélie	Oration	07/04/78	f	Lancer du	42
	Amélie	Oration	07/04/78	f	Saut en lo	42
	Amélie	Oration	07/04/78	f	Sprint	42
	César	Yennes	28/09/87	m	Course de	33

Enregistrement 1 de 35

Lorsque la requête est exécutée au cours d'une année donnée (par exemple 2020), les résultats appropriés apparaissent pour cette même année.

L'ensemble du code que nous avons saisi dans le champ apparaît également dans l'en-tête de colonne. Cela est résolu en entrant un alias pour le code sous lequel le résultat apparaîtra.

Champ	YEAR(NOW( )) - YEAR("DateNais")
Alias	AgeSport
Table	

Dans la ligne Alias, le terme “AgeSport” est entré. Pour une personne qui n’a pas encore fêté son anniversaire cette année, l’âge calculé indique 1 an en trop. Voir plus bas pour un remède à ce défaut

Sport	AgeSport
Disque	35
Javelot	35
Saut en hauteur	35
Sprint	35
Disque	42
Lancer du poids	42

Si la requête est exécutée à nouveau, l’en-tête de colonne ne contient plus le code, mais le terme AgeSport.

Ainsi, la requête sera enregistrée sous le nom AgeSport pour éviter la confusion liée à l’utilisation du code comme nom.

Pour remédier à l’erreur citée précédemment (Age erroné pour une personne n’ayant pas encore fêté son anniversaire), on pourra remplacer le code du champ par le suivant :

```
CASEWHEN
      ( MONTH( "DateNais" ) > MONTH( CURDATE( ) )
      OR ( MONTH( "DateNais" ) = MONTH( CURDATE( ) )
      AND DAY( "DateNais" ) > DAY( CURDATE( ) ) ) )
THEN DATEDIFF( 'yy', "DateNais", CURDATE( ) ) - 1
ELSE DATEDIFF( 'yy', "DateNais", CURDATE( ) )
```

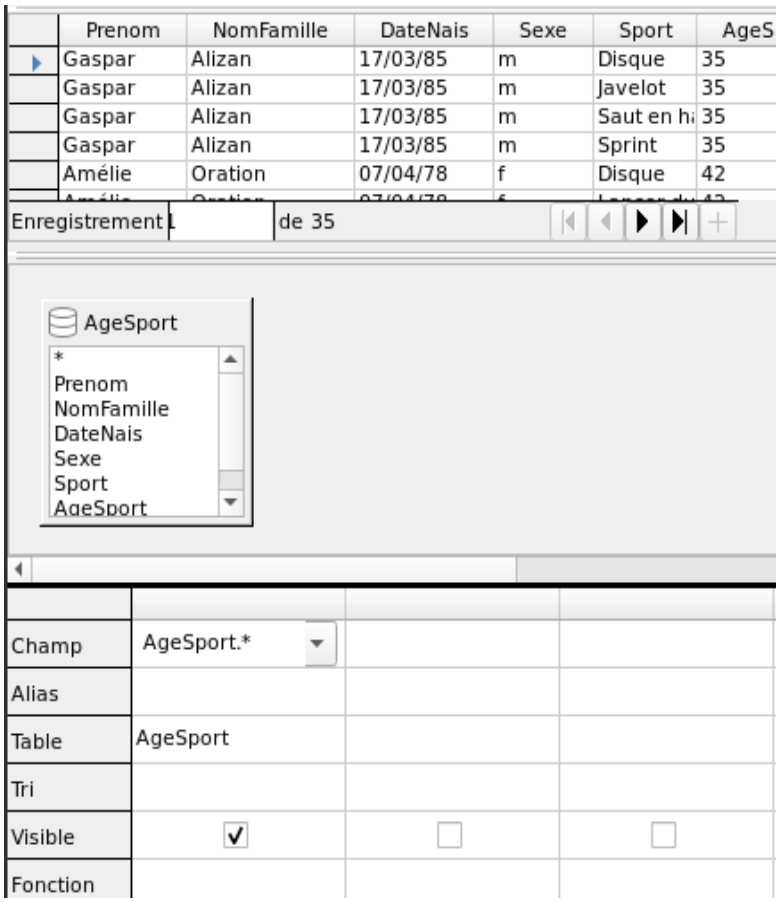
Cette requête est ensuite utilisée comme base pour la requête suivante, que le AgeSport attribue désormais à un Groupe_Age.

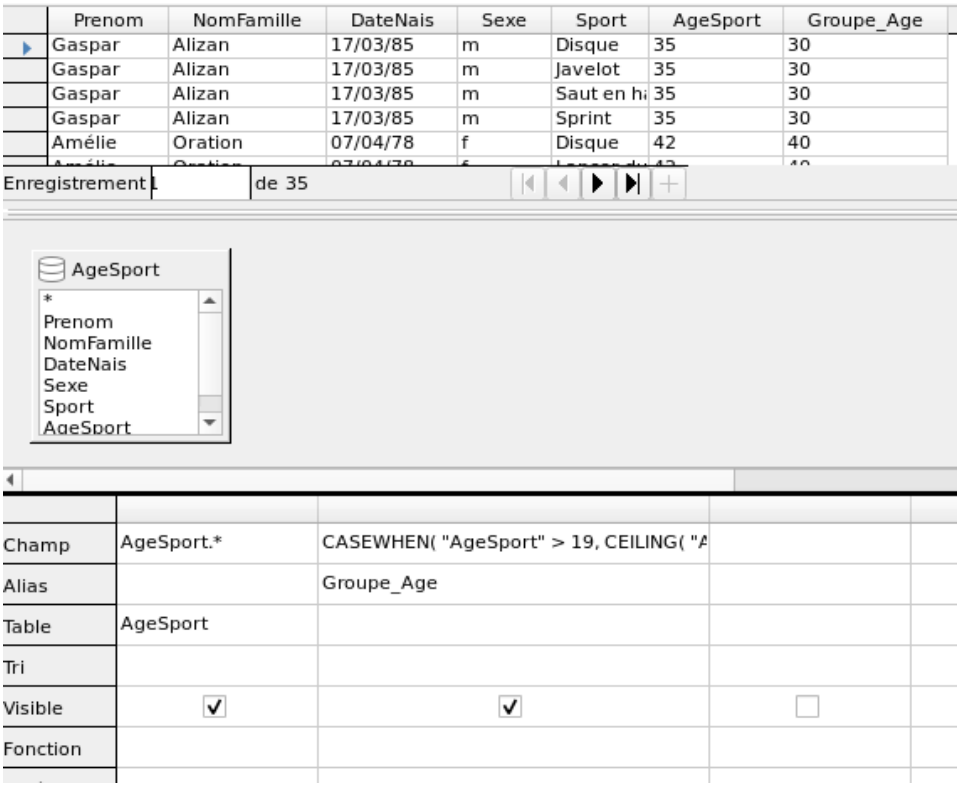
La requête AgeSport a été choisie comme base de la 2e requête. Avant le terme AgeSport dans le conteneur de table, il y a une icône différente de celle des tables de la première requête. Ce symbole indique que la base de la nouvelle requête est une requête et non une table.

Double-cliquez sur * ou sélectionnez *AgeSport.** pour sélectionner tous les champs. La requête suivante renvoie donc le même résultat, mais la formule n’est plus visible.

Accédez maintenant au champ AgeSport. Il va permettre de déterminer dans quelle tranche d’âge la personne concernée participe.

Afin que le calcul ne soit pas trop complexe, il convient de créer les classifications suivantes : pour les moins de 20 ans, les candidats sont divisés en groupes d'âge de 2 ans. À partir de 20 ans et plus, les groupes sont formés par tranche de 10 ans, par exemple, 20–29 ans.





Ces formules n'appartiennent plus vraiment au groupe de compétences utilisateur débutant. Pour des affectations plus sophistiquées, veuillez vous référer à l'annexe A de ce manuel.

La formule dans le second champ de la requête ;

```
CASEWHEN( [AgeSport] > 19,
      CEILING( [AgeSport] / 10 ) * 10,
      [AgeSport] - MOD( [AgeSport], 2 ) )
```

(Les guillemets sont remplacés par des crochets, que Base traduit ensuite dans la cellule de champ)

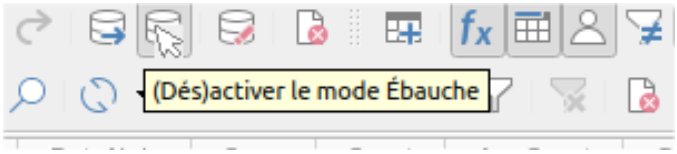
CASEWHEN(condition, valeur1, valeur2). Si condition est vraie, valeur 1 est retournée, autrement valeur2 est retournée. Valeur peut être un nombre ; une chaîne de caractère ; le résultat d'un calcul.. En français, cela signifie :

- Si AgeSport est supérieur à 19 ans, il est utilisé pour calculer l'âge inférieur suivant qui se termine par un zéro pour obtenir le Groupe_Age. CEILING("AgeSport" / 10) signifie : arrondi de la division de "AgeSport" par 10 (ex : 25/10=2,5 → Ceiling (2,5) = 2). Ce résultat est ensuite multiplié par 10, et on obtient 20.
- Si AgeSport est inférieur à 19 ans, calculer MOD ("AgeSport", 2) et le retrancher à la valeur de AgeSport. MOD("AgeSport", 2) signifie : Reste de la division (modulo) de AgeSport par 2 (ex : pour 17 / 2 le reste est 1).

Tous les candidats de plus de 19 ans sont ainsi affectés à une tranche d'âge tous les dix ans. Tous les candidats jusqu'à 19 ans sont attribués à un groupe d'âge d'amplitude 2 ans.

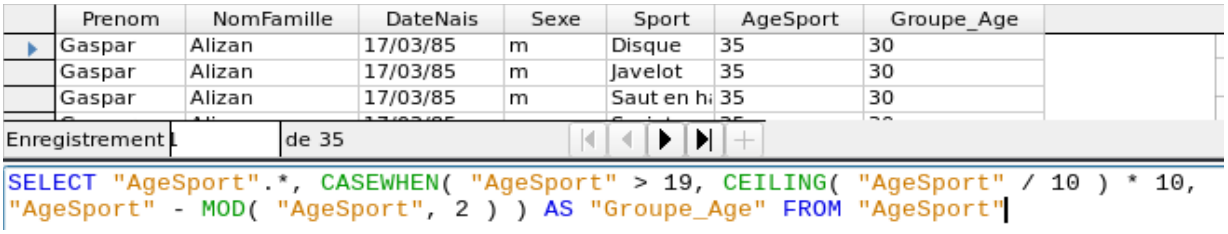
Cette formule a aussi reçu un alias, dans ce cas l'alias est Groupe_Age.

Enregistrez la requête sous le nom *Inscription*.



La désactivation du mode ébauche n'est pas vraiment nécessaire, car toutes les entrées sont possibles dans la vue de conception sans problème majeur. Cependant, regarder de plus près le code dans une requête ne peut jamais faire de mal.

Pour terminer, il existe des expressions SQL qui s'intègrent mal dans le mode Ébauche ou qui n'y sont pas réalisables. Dans ce cas on utilise l'entrée directe du code SQL.



Note

Les noms des champs et des tables sont entourés de guillemets doubles et affichés en ocre. Les mots-clés du code SQL sont en bleu, les fonctions en vert.

```
SELECT "AgeSport".*, CASEWHEN("AgeSport" > 19, CEILING("AgeSport" / 10) * 10,
"AgeSport" – MOD("AgeSport", 2)) AS "Groupe_Age" FROM "AgeSport"
```

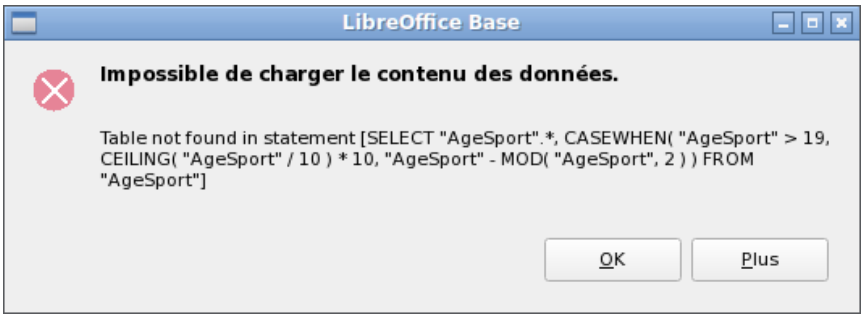
Les parties de la formule ont déjà été mentionnées. Voici maintenant la structure :

```
SELECT "AgeSport".*,... AS "Groupe_Age" FROM "AgeSport"
```

Sélectionnez (SELECT), dans `(FROM)` la table `AgeSport,` tous les champs de la table ("AgeSport".*), plus ce qui est retourné par la formule, dont l'alias (AS) est "`Groupe_Age`".

Le code ne fait pas la distinction entre les tables et les requêtes comme base des données. Il ne fonctionne donc que dans l'interface utilisateur graphique de Base. Une requête ne peut pas avoir le même nom qu'une table et réciproquement.

Lorsque l'utilisateur clique sur le bouton SQL (Exécuter directement l'instruction SQL) en mode Vue SQL , la base de données répond avec un message d'erreur. Le HSQLDB embarqué ne connaît pas la requête "AgeSport" à laquelle se réfère la requête en cours.



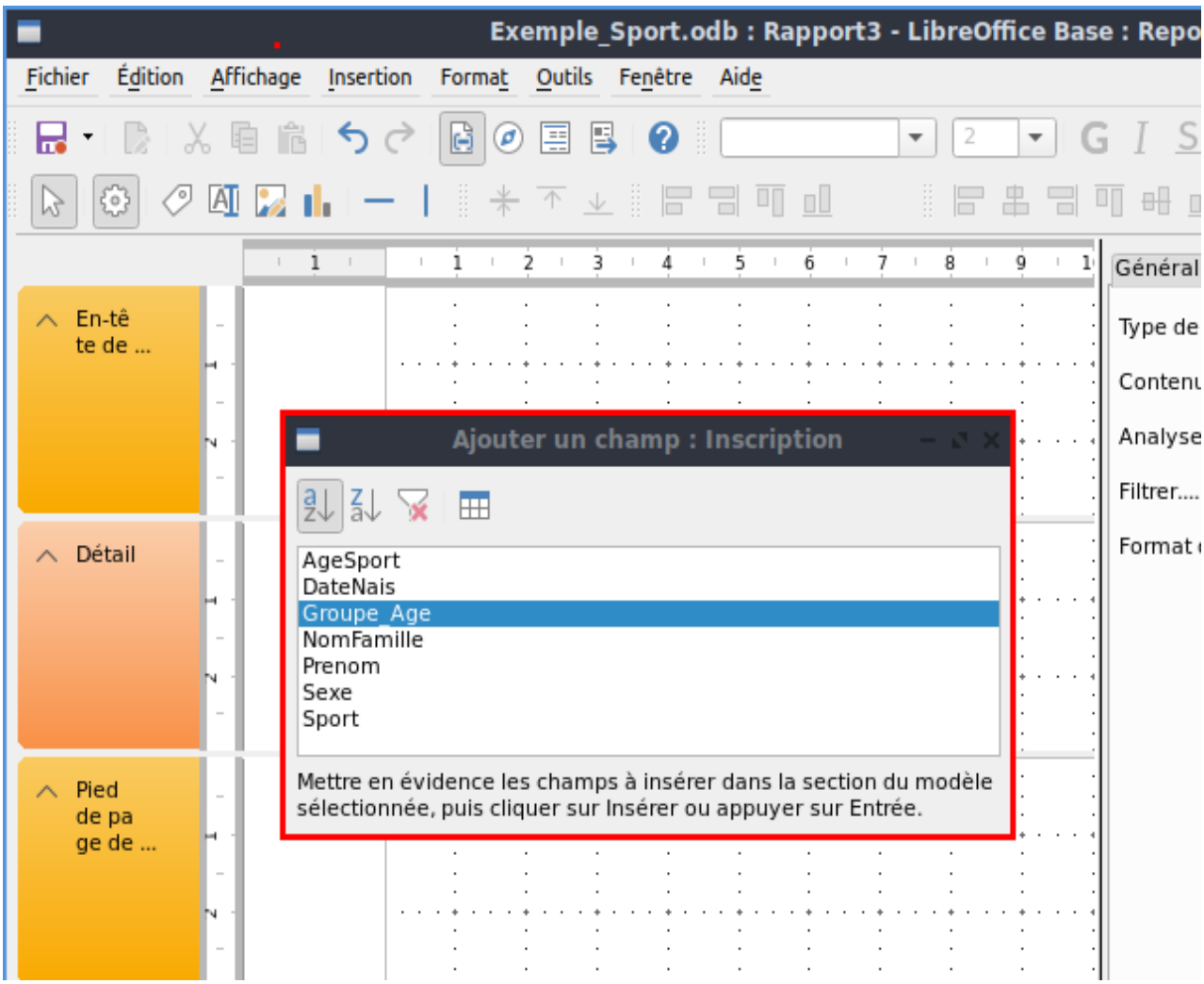
Créer un rapport

Utilisez **Rapports > Créer un rapport en mode Ébauche** pour créer un rapport avec une liste claire des entrées, triées par sport, sexe et tranche d'âge.

Au démarrage de l'éditeur, la boîte de dialogue **Ajouter un champ** apparaît d'abord au premier plan. Utilisez cette boîte de dialogue pour prendre comme base les champs de la table triée par ordre alphabétique. La requête doit être choisie comme source de données.

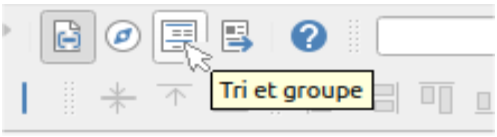
Dans la fenêtre de rapport, le côté droit montre un aperçu des propriétés de l'objet actuellement actif. S'il n'est pas visible, sélectionnez **Affichage > Propriétés**.

Dans Propriétés, sélectionnez l'onglet **Données > Type de contenu > Requête**. Pour la propriété Contenu sélectionnez la requête *Inscription*, qui était la requête récapitulative finale.



Le Générateur de rapports affiche désormais tous les champs de la requête sélectionnée.

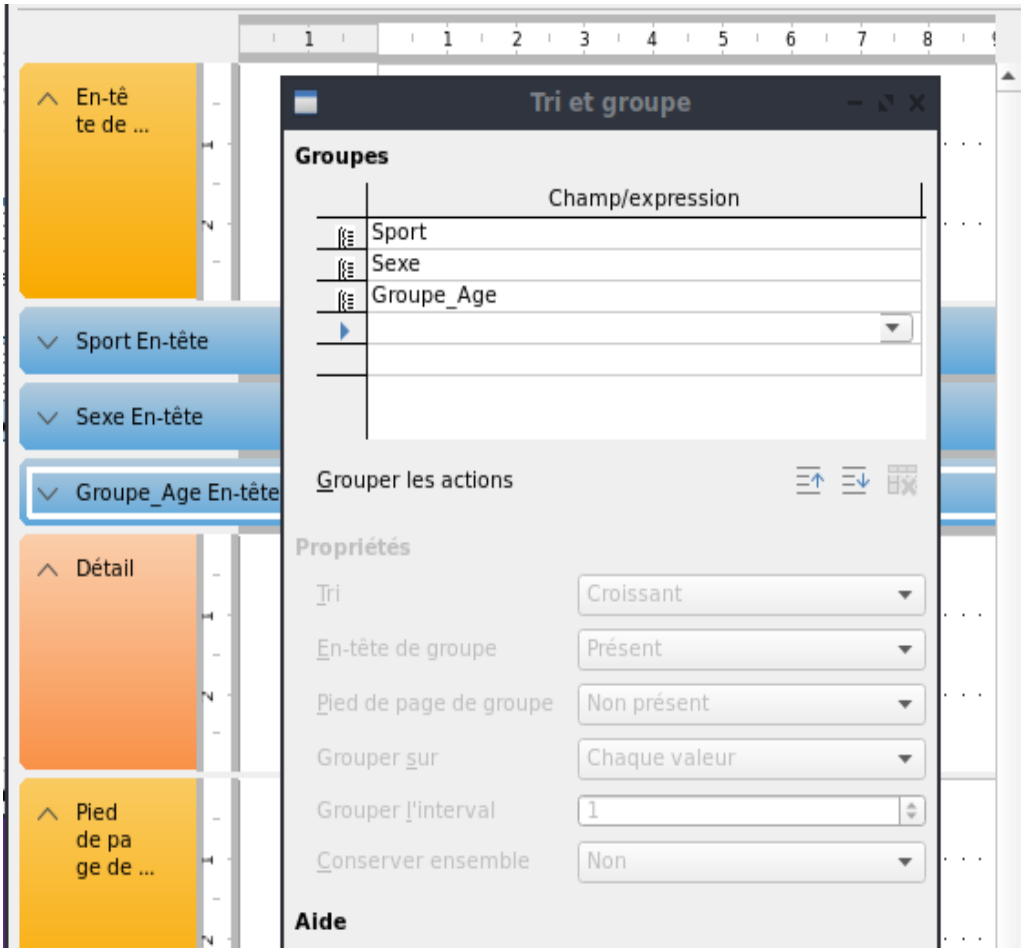
À l'étape suivante, ouvrez la boîte de dialogue **Tri et groupe** à l'aide du bouton de la barre d'outils ou en sélectionnant **Affichage > Tri et groupe**.



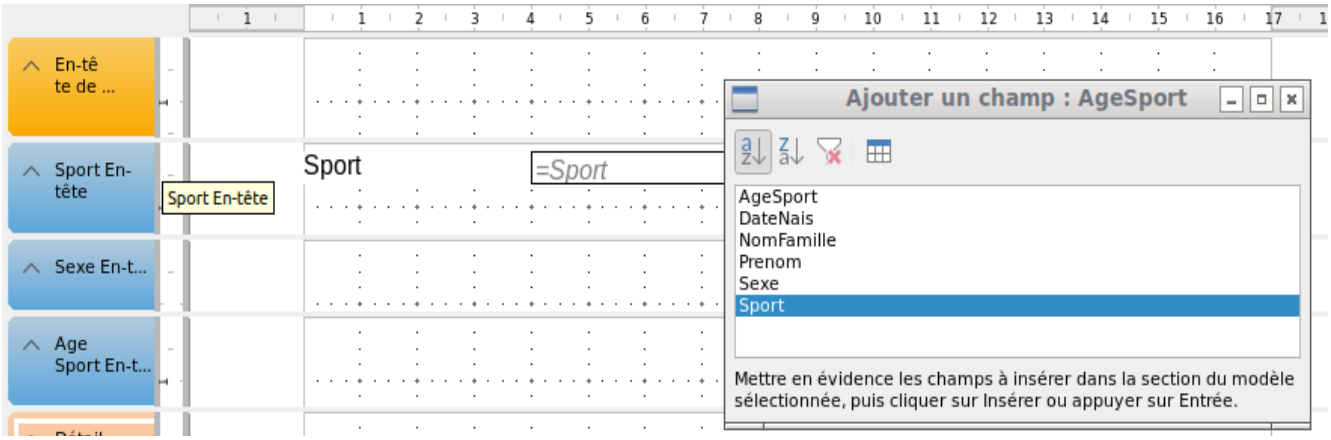
Dans la boîte de dialogue Tri et groupe, sélectionnez les champs Sport, Sexe et Groupe_Age.

Un en-tête de groupe pour le tri apparaît à chaque sélection de tri sur le côté gauche du rapport. Utilisez les paramètres par défaut pour les propriétés des sélections de tri et de regroupement.

Fermez la boîte de dialogue Tri et groupe.



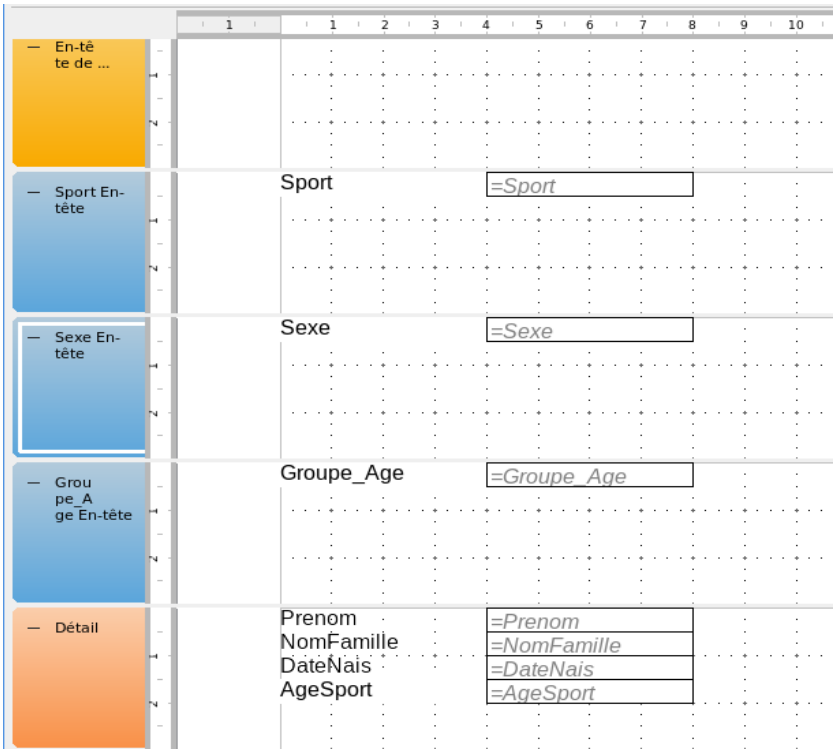
Sélectionnez l'en-tête de groupe, Sport. Ceci est visible dans la bordure blanche de l'en-tête. À l'aide de la boîte de dialogue Ajouter un champ, cliquez sur le champ de départ pour insérer un champ d'étiquette et un champ de texte qui afficheront le contenu du champ de candidat.



De la même manière, affectez les champs Sexe et Groupe_Age aux en-têtes de groupe appropriés.

Insérez tous les champs restants dans la zone Détails.

Le projet de rapport devrait maintenant ressembler à ceci :



Enregistrez le rapport. Le nom pourrait être quelque chose comme *Liste des participants*.

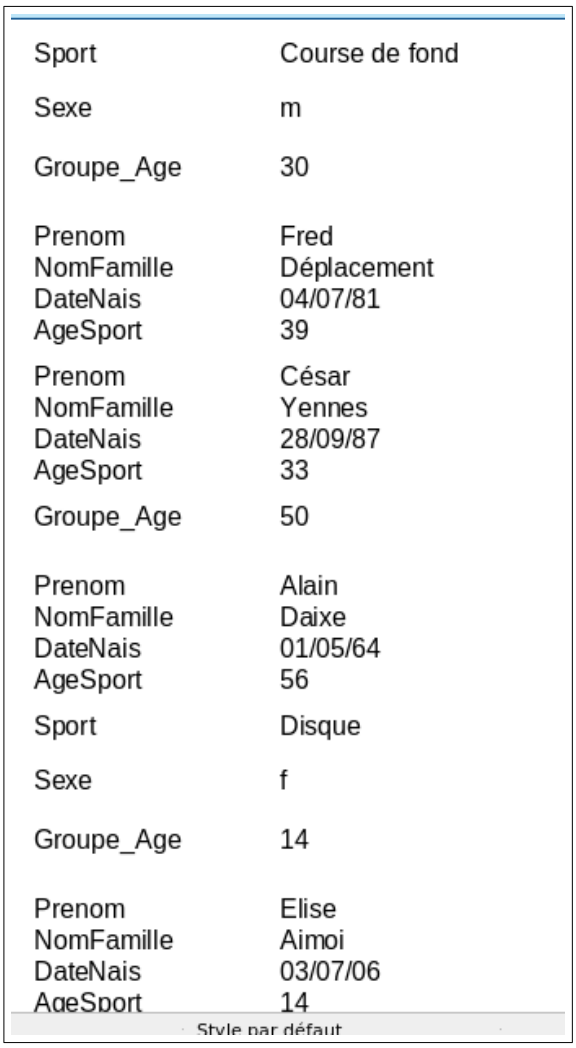
Ensuite, enregistrez le fichier de base de données lui-même ; sinon, le rapport n'est stocké que temporairement.

Note

Surtout lors de la conception d'un rapport à l'aide du Générateur de rapports, il échoue souvent en raison d'instabilités du programme. Il est important de sauvegarder régulièrement le rapport et le fichier de base de données.

Heureusement, l'exécution ultérieure d'un rapport n'est pas affectée par ces instabilités.

Si ce rapport est exécuté avec les données appropriées, le résultat ressemble à l'image suivante.



Le début du rapport montre deux candidats qui souhaitent se lancer dans la course de fond et appartiennent au groupe d'âge 30.

Lors de l'exécution du rapport, des défauts de conception apparaissent :

- Les distances entre les en-têtes de groupe et le contenu de la section Détail sont beaucoup trop grandes.
- Le sexe est indiqué par m et f. Mieux vaut des noms tels que Hommes et Femmes.
- Probablement en raison de l'opération de division dans la requête, le champ Groupe_Age contient des nombres avec une décimale.
- Les champs d'étiquette de la section Détail (prénom, nom, ...) seraient mieux placés ensemble horizontalement comme en-têtes de tableau sous l'étiquette et le champ Groupe_Age dans l'en-tête Groupe_Age.

Définition des distances entre les champs du rapport

Pour réduire les distances verticales entre les champs, utilisez la souris pour faire glisser la bordure inférieure de l'en-tête Groupe_Age vers le bas des en-têtes de tableau. Vous pouvez également mettre en évidence une section et régler la hauteur de l'en-tête de groupe dans **Propriétés**. Par conséquent, aucun champ mais uniquement l'en-tête de groupe ne doit être étiqueté.

Il n’est pas possible qu’une section soit plus petite que les étiquettes et les champs qu’elle contient.

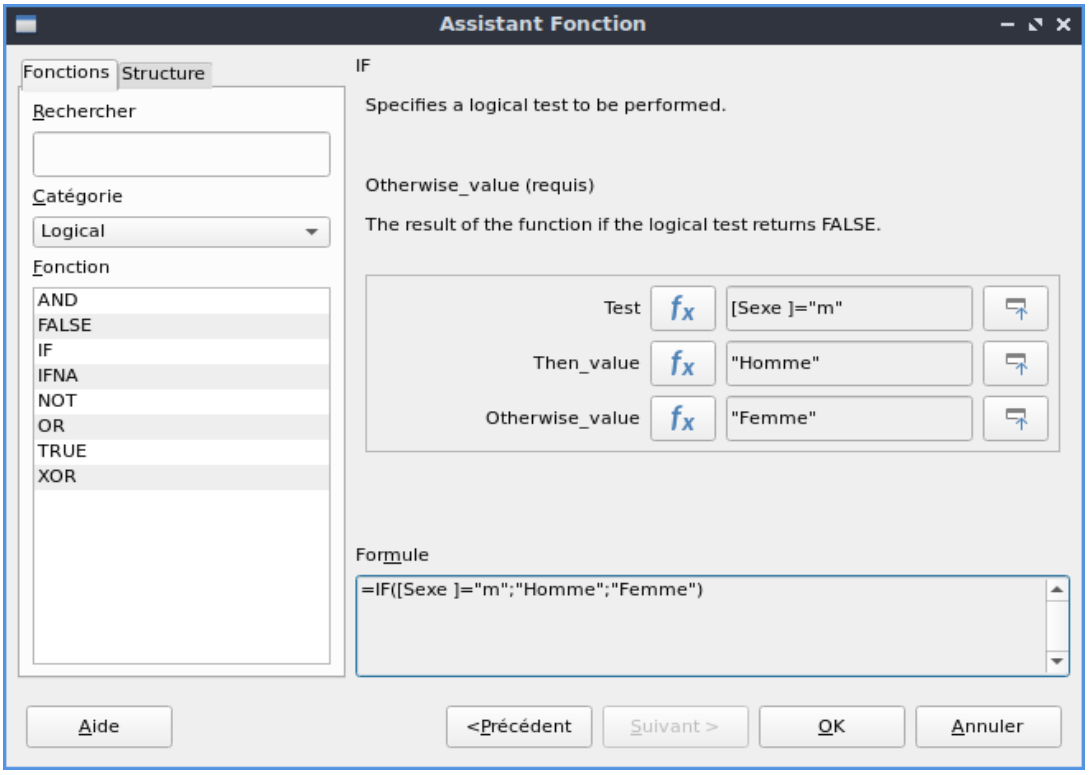
La zone d’en-tête Groupe_Age doit être suffisamment grande pour prendre les champs d’étiquette de la section Détail.

Tenez également compte, lors de la sélection des distances, qu’un groupe suivant n’apparaît pas trop près du groupe précédent. L’étiquette et les zones de texte peuvent également nécessiter une distance par rapport au haut de la section les contenant. Si cette distance au sommet n’est pas souhaitée, des pieds de page de groupe plutôt que des en-têtes peuvent être affichés pour fournir cette distance. Une telle préférence est possible dans **Affichage > Tri et groupe** pour chaque groupe.

Influencer le contenu d’un champ de texte par une formule

La désignation du sexe dans la table ne suffit pas pour les exigences d’une liste de candidats. Le changement de nom du champ peut être effectué dans la requête. Mais comme la requête a déjà été créée, vous pouvez utiliser des fonctions dans le rapport à la place.

Mettez en surbrillance le champ de texte “= Sexe”. Dans les propriétés sur le côté droit de l’onglet Générateur de rapports, sélectionnez **Données > Champ de données**. Cliquez sur le bouton avec l’ellipse (…).L’assistant de fonction s’affiche.



Dans **Catégorie**, sélectionnez **Logique**, puis double-cliquez sur **IF**.

Saisissez la valeur de test. Il s’agit du champ de la requête à partir duquel les données sont lues et mises entre crochets. Les textes doivent être placés entre guillemets. Lorsque le champ Sexe a la valeur **m**, alors **Homme** est affiché dans le champ du rapport. S’il n’y a pas de m, alors **Femme** apparaît.

Cliquez sur **OK** pour confirmer la saisie.

Ainsi, le libellé du champ est modifié en conséquence.

Modifier la mise en forme d'un champ de texte

Le champ qui reçoit le contenu de la base de données est identifié dans le rapport comme une zone de texte, mais il peut être formaté comme les champs des tables dans Writer ou Calc.

Mettez en surbrillance le champ "= Groupe_Age". Dans les propriétés sur la droite, sélectionnez **Général > Formatage (**dernière ligne). La propriété Mise en forme est définie sur "texte". Cliquez sur le bouton avec l'ellipse (…). Cela ouvre la boîte de dialogue de mise en forme, également utilisée dans Calc, Writer ou lors de la création de formulaires. Sélectionnez **Catégorie > Nombre** et confirmez avec **OK.** Choisissez également le nombre de décimales.

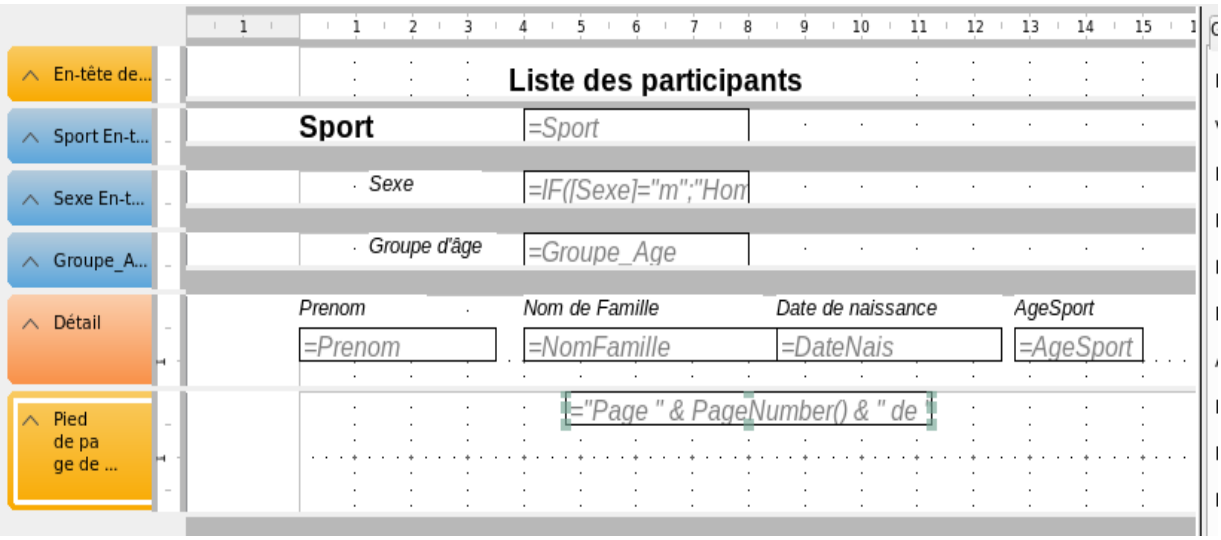
Le format de l'affichage passe désormais du texte aux nombres.

Déplacement des boîtes dans le générateur de rapports

Les champs peuvent également être déplacés au-delà des limites d'une section vers une autre section dans le Générateur de rapports. Cependant, un espace suffisant pour le champ doit être présent dans la section de destination. Aucune partie d'un champ ne peut exister au même emplacement qu'une partie d'un autre champ.

Le positionnement des champs à l'aide d'une souris n'est pas précis, vous devez donc prévoir suffisamment d'espace dans chaque section pour les champs. Dans une section comme l'en-tête de groupe Groupe_Age, les champs peuvent ensuite être positionnés avec précision à l'aide des touches fléchées.

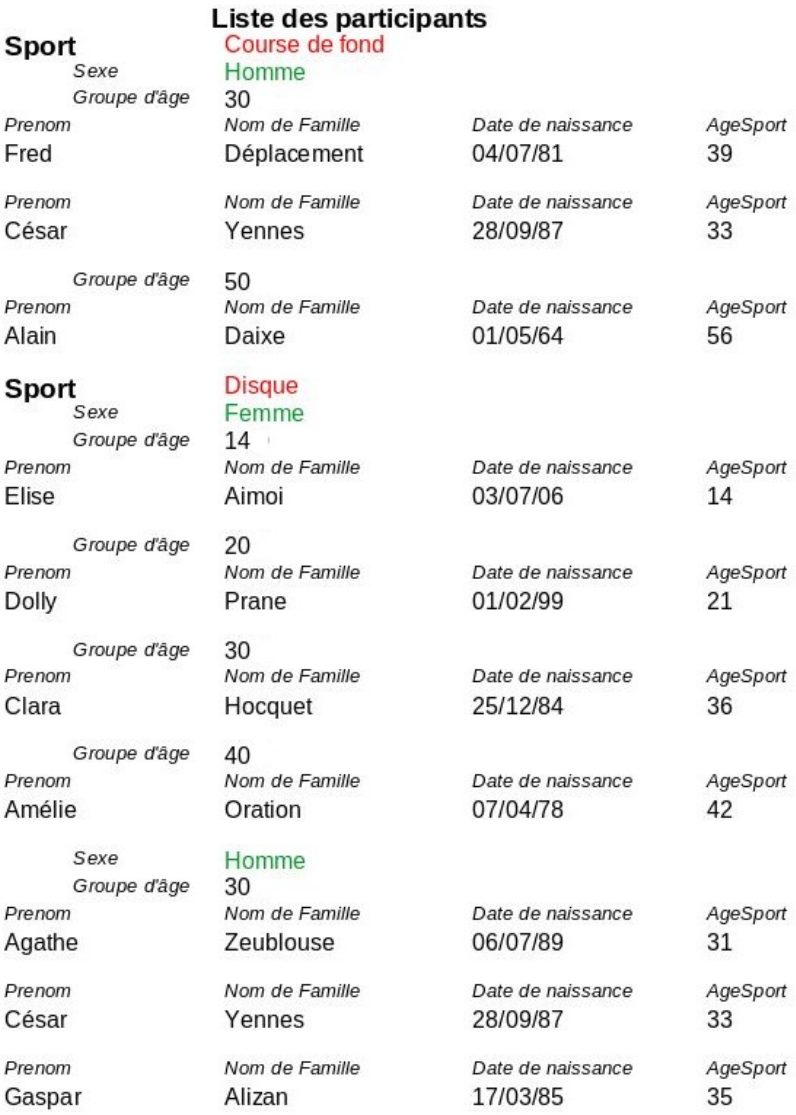
Pour positionner les champs à l'aide du clavier, regardez les propriétés **Général > Position X** et **Général > Position Y**, déplacer le contrôle avec Maj+Flèches (grands déplacements) ou plus finement avec Alt+Flèches (petits déplacements).



Ici, un champ d'étiquette pour l'en-tête a été ajouté. L'entrée du texte pour le champ d'étiquette apparaît dans les propriétés des champs.

Le pied de page est défini dans les propriétés : **Visible > Oui**. La marge inférieure du document contient trop d'espace. N'oubliez pas que le montant disponible est déjà en plus réduit par la taille des marges de la page. Par défaut, toutes les marges de page sont définies pour le format de page lettre à 0,79 pouces (2 centimètres).

Pour plus d’options de mise en forme des rapports, reportez-vous au Chapitre 6, Rapports.



Liste des participants

Sport	Course de fond		
Sexe	Homme		
Groupe d'âge	30		
Prenom	*Nom de Famille*	*Date de naissance*	*AgeSport*
Fred	Déplacement	04/07/81	39
Prenom	*Nom de Famille*	*Date de naissance*	*AgeSport*
César	Yennes	28/09/87	33
Groupe d'âge	50		
Prenom	*Nom de Famille*	*Date de naissance*	*AgeSport*
Alain	Daixe	01/05/64	56
Sport	Disque		
Sexe	Femme		
Groupe d'âge	14		
Prenom	*Nom de Famille*	*Date de naissance*	*AgeSport*
Elise	Aimoi	03/07/06	14
Groupe d'âge	20		
Prenom	*Nom de Famille*	*Date de naissance*	*AgeSport*
Dolly	Prane	01/02/99	21
Groupe d'âge	30		
Prenom	*Nom de Famille*	*Date de naissance*	*AgeSport*
Clara	Hocquet	25/12/84	36
Groupe d'âge	40		
Prenom	*Nom de Famille*	*Date de naissance*	*AgeSport*
Amélie	Oration	07/04/78	42
Sexe	Homme		
Groupe d'âge	30		
Prenom	*Nom de Famille*	*Date de naissance*	*AgeSport*
Agathe	Zeublouse	06/07/89	31
Prenom	*Nom de Famille*	*Date de naissance*	*AgeSport*
César	Yennes	28/09/87	33
Prenom	*Nom de Famille*	*Date de naissance*	*AgeSport*
Gaspar	Alizan	17/03/85	35

Extensions de la base de données exemple

L’exemple présenté ici n’est que la première étape d’une base de données dans le secteur du sport. Maintenant, ajoutez un champ pour chaque élément possible d’informations nécessaires. Un bon endroit pour ajouter un tel champ est dans la table rel_Candidats_Sports.

Cependant, si les entrées s’appliquent à plusieurs compétitions pour le même sport, incluez un champ de date ou un autre champ qui peut être attribué à la compétition respective dans la table rel_Candidats_Sports. Le champ fait alors également partie de la clé primaire de la table.

Peut-être que l’adhésion au club pourrait également être ajoutée. L’ajout d’un champ dans la table Candidats serait suffisant. Lorsque de nombreux clubs ont le même nom, cela suggère d’ajouter une table Club séparée et également la clé étrangère correspondante dans la table Candidats.

Ensuite, bien sûr, il faut déterminer, comme pour toutes les compétitions, qui doit être placé dans quel groupe d’âge et quel sport pertinents. Le tri est nécessaire ici, ce qui peut à nouveau se transformer en un rapport avec une liste de résultats.

Ce serait bien si chaque candidat (encore une fois via un rapport) obtenait un certificat magnifiquement conçu avec la performance personnelle et le classement.

De telles améliorations sont facilement possibles, comme décrit dans d'autres chapitres de ce guide.

Guide Base

Chapitre 2
Créer une Base de données

Introduction

Les bases de la création d'une base de données dans LibreOffice sont décrites dans le chapitre 8 du Guide de mise en route, Débuter avec Base.

Le composant base de données de LibreOffice, appelé Base, fournit une interface graphique pour travailler avec des bases de données. De plus, LibreOffice contient une version du moteur de base de données HSQL. Cette base de données HSQLDB ne peut être utilisée que par un seul utilisateur. L'ensemble de données entier est stocké dans un fichier ODB doté d'un mécanisme de verrouillage de fichier dans le dossier de configuration lorsqu'il est ouvert par un utilisateur. En fait, un fichier portant le même nom est créé dans le répertoire où se trouve le xxx.odb. Il porte l'extension ".lck".

Depuis LibreOffice 4.2, une base de données interne Firebird est disponible en plus de celle interne HSQLDB. La base de données Firebird est incluse dans les « fonctionnalités expérimentales » de la version 6.4 en raison de son instabilité (bien qu'elle se soit beaucoup améliorée). Les exemples de bases de données dans ce livre continuent de faire référence à HSQLDB, mais sont personnalisés de sorte que la plupart des fonctions sont directement transférables à Firebird. Si nécessaire, des alternatives dans Firebird sont affichées.

Création d'une nouvelle base de données à l'aide du moteur HSQL interne

Si une base de données avec plusieurs utilisateurs n'est pas prévue, ou si vous souhaitez acquérir une première expérience avec une base de données, le moteur de base de données interne suffira. Il est possible à un stade ultérieur de transférer la base de données vers un environnement HSQLDB externe, où plusieurs utilisateurs peuvent avoir un accès simultané à la base de données sur le serveur HSQLDB. Ceci est décrit plus loin dans ce chapitre.

Pour créer une base de données interne à partir de l'écran de démarrage de LibreOffice, cliquez sur le bouton Base de données Base ; ou, de n'importe où dans LibreOffice, utilisez **Fichier > Nouveau > Base de données**. L'assistant de base de données (Figure 12 s'ouvre.

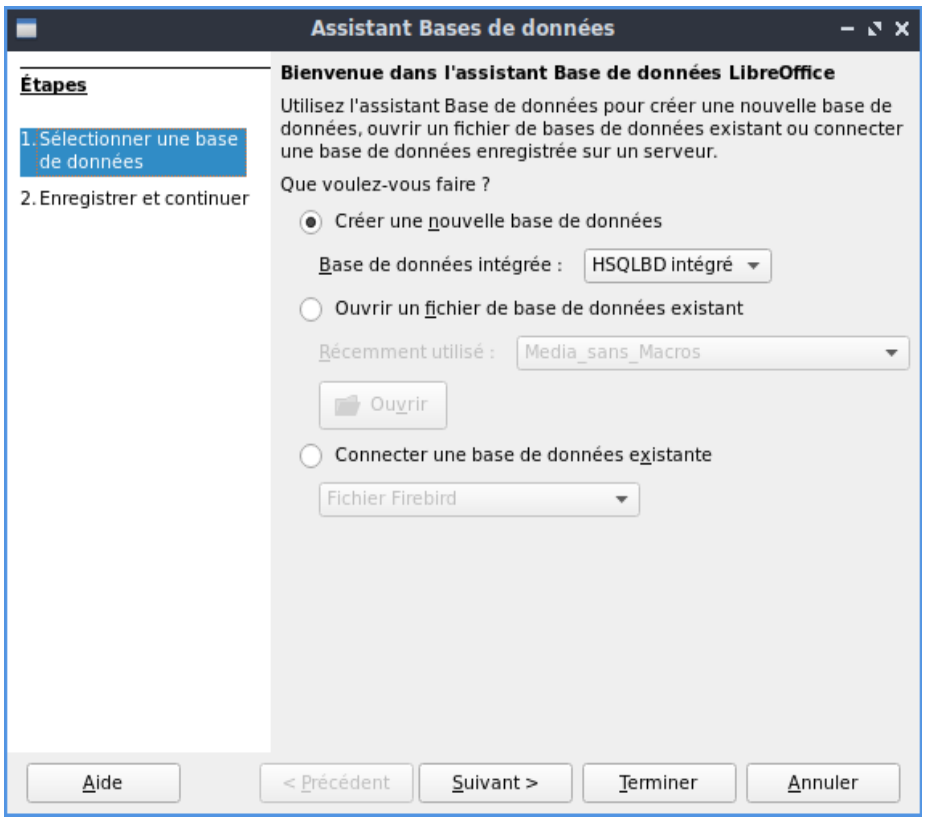


Figure 12: Étape 1 de l'assistant Sélectionner une Base de données

Sélectionnez **Créer une nouvelle base de données**. Par défaut, il s'agit d'une base de données HSQLDB intégrée. Une base de données interne Firebird est également disponible ; voir la note de mise en garde à la page suivante.
Les autres options servent à ouvrir un fichier existant ou à créer une connexion à une base de données externe telle qu'un carnet d'adresses ou une base de données MySQL.

Choisissez **Suivant**>>pour passer à l'étape 2 de l'assistant de base de données (Figure 13).

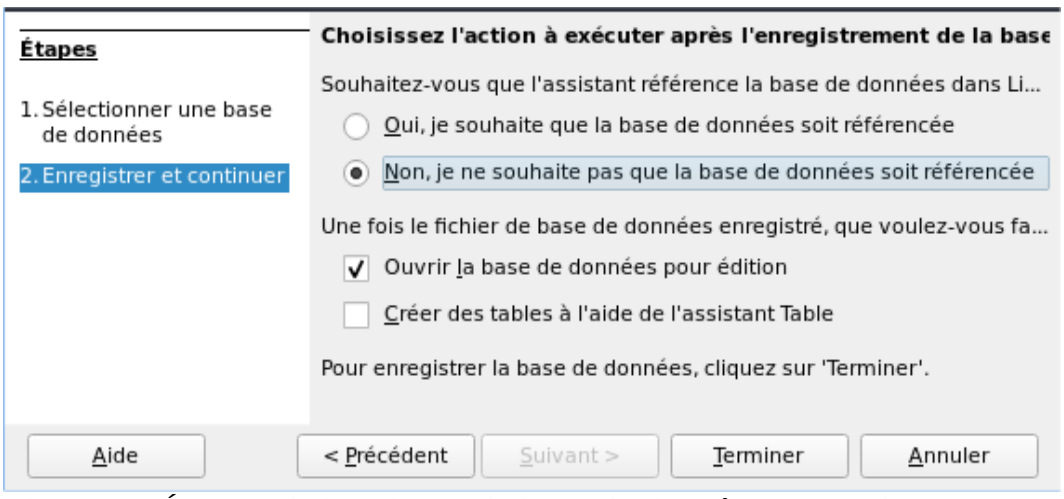


Figure 13: Étape 2 de l'Assistant de base de données : Enregistrer et continuer

Une base de données enregistrée avec LibreOffice peut être utilisée par d'autres composants de LibreOffice, par exemple pour les lettres de publipostage dans Writer. Il est recommandé que toutes les bases de données soient enregistrées lors de leur création, mais le choix final appartient au lecteur.

Sélectionnez **Ouvrir la base de données pour édition** et désélectionnez Créer des tables à l'aide de l'assistant Table. Cette dernière option a été abordée au chapitre 1 ; le reste de ce livre n'utilise pas les assistants pour créer des tables, des requêtes, etc.

Cliquez sur **Terminer** pour enregistrer la base de données. Une boîte de dialogue standard Enregistrer sous s'ouvre, demandant un nom et un emplacement pour le fichier * odb, qui est prêt pour la saisie d'enregistrements de la base de données interne et le stockage des requêtes, des formulaires et des rapports. Contrairement aux autres parties de LibreOffice, le fichier est enregistré avant que vous n'ayez fait des saisies visibles.

Attention

Si les fonctionnalités expérimentales sont activées (**Outils > Options > Avancé**), si une base de données HSQLDB interne existante est ouverte, un message peut s'afficher, vous demandant si vous souhaitez migrer la base de données vers Firebird maintenant.

La prudence est de mise : **ne pas confirmer**. Soyez extrêmement prudent ici ! Cliquez plus tard. Lorsque vous êtes prêt à migrer la base de données, suivez cette procédure :

1. Sauvegardez le fichier de base de données HSQLDB.
2. Adaptez les fonctions autant que possible en fonction de la liste sur cette page wiki: https://wiki.documentfoundation.org/Documentation/HowTo/MigrateFromHSQLDB/fr
3. Copiez les vues qui ne peuvent pas être directement converties à partir du code SQL et enregistrez-les en tant que requêtes.
4. Les noms de table et les noms de colonne dans Firebird peuvent comporter au maximum 31 caractères. Si nécessaire, adaptez-vous à un nom plus court.

5) Les tables de texte pur (table *.csv intégré, etc.) ne sont pas possibles sous Firebird, ils doivent donc être placés ailleurs.

Accéder aux bases de données externes

Une base de données externe doit exister avant de pouvoir y accéder. Si l'accès à une telle base de données est souhaité, la base de données doit être configurée pour autoriser les connexions réseau avec un nom d'utilisateur et un mot de passe spécifiques avant que des programmes externes puissent s'y connecter.

Lorsqu'une telle base de données est correctement configurée, vous pouvez, selon le logiciel de connexion disponible (le pilote de base de données), créer des tables, saisir des données et interroger les données.

Cliquez sur **Fichier > Nouveau > Base de données** pour ouvrir l'assistant de base de données (Figure 12) et sélectionnez **Se connecter à une base de données existante**. La liste des types de bases de données disponibles varie en fonction du système d'exploitation et de l'interface utilisateur, mais les éléments suivants doivent toujours être disponibles :

- Classeur
- dBASE
- Document Writer
- Fichier Firebird
- JDBC
- MySQL
- ODBC
- Oracle JDBC
- PostgreSQL
- Texte
- ainsi que divers types de carnets d'adresses.

Les options de connexion varient en fonction du type de base de données sélectionné. Vous pouvez modifier les options après la création du fichier *.odb.

Certains types de bases de données (par exemple, une connexion à des feuilles de calcul) ne permettent pas la saisie de nouvelles données. Ceux-ci sont utilisés uniquement pour rechercher ou rapporter des données existantes. Les descriptions dans les chapitres suivants traitent exclusivement de LibreOffice Base en utilisant la base de données interne HSQLDB. La plupart des travaux de conception peuvent être étendus aux bases de données qui utilisent MySQL, PostgreSQL, etc.

Voici quelques brefs exemples de la façon dont vous pouvez vous connecter à une base de données externe.

Bases de données MySQL/MariaDB

Base peut se connecter aux bases de données MySQL et MariaDB par trois méthodes. Le moyen le plus simple et le plus rapide est la connexion directe avec le connecteur MySQL. Les deux autres sont la connexion via ODBC ou JDBC.

Note

Dans MySQL et MariaDB, il est possible de saisir et de modifier des données dans des tables sans champ de clé primaire. L'interface graphique de Base affiche ces tables, mais n'offre aucune option d'entrée ou de modification.

Si vous souhaitez utiliser des tables sans clé primaire, vous pouvez utiliser **Outils > SQL** à la place, ou dans des formulaires via des macros, pour fournir des données aux tables.

Créer un utilisateur et une base de données

Une fois MySQL ou MariaDB installé, effectuez les étapes suivantes dans l'ordre décrit.

1. Le compte administrateur dans MySQL s'appelle root. Les utilisateurs Linux doivent noter qu'il ne s'agit pas de l'utilisateur root du système d'exploitation Linux. L'utilisateur root doit se voir attribuer un mot de passe directement après l'installation, si cela n'a pas été fait avant l'installation.

2) `mysql –u root –p`

   Pour commencer, aucun mot de passe n'est défini, appuyez simplement sur Entrée. Une invite de saisie apparaît:
   `mysql>`

   Toutes les entrées suivantes sont effectuées sur la console MySQL. Les mots de passe peuvent être différents selon que l'invite provient de l'ordinateur local (localhost) ou d'un autre ordinateur qui fait office de serveur MySQL (hôte).

```
SET PASSWORD FOR 'root'@'localhost'=PASSWORD('MotdePasse') ;
SET PASSWORD FOR 'root'@'host'=PASSWORD('MotdePasse')
```

   Pour les utilisateurs de Windows, la deuxième ligne se lit comme suit:

```
SET PASSWORD FOR root@'%'=PASSWORD('MotdePasse') ;
```

3. Par mesure de sécurité, tous les utilisateurs anonymes actuels sont supprimés.

```
DELETE FROM mysql.user WHERE User='';
```

```
DELETE FROM mysql.db WHERE User='';
FLUSH PRIVILEGES ;
```

4. Une base de données appelée ***libretest*** est créée.
```
CREATE DATABASE libretest ;
```

5) Un compte ***lotest*** est créé pour accéder à la base avec le mot de passe ***libre***
```
CREATE USER 'lotest' IDENTIFIED BY 'libre'
```

6) Tous les droits sur la base de données libretest sont accordés à l'utilisateur ***lotest***,
```
GRANT ALL ON libretest.* TO lotest;
```
On aurait pu grouper ces deux dernères commandes en une seule ;
```
GRANT ALL ON libretest.* TO lotest IDENTIFIED BY 'libre';
```

La base de données est maintenant disponible et peut être connectée comme suit.

Connexion MySQL directe à l'aide d'une extension

Depuis LibreOffice 6.2, la connexion directe de Base à MySQL/MariaDB a été intégrée à LibreOffice. L'installation d'une extension n'est plus nécessaire.

Connexion MySQL via JDBC

L'accès général à MySQL peut se faire via JDBC ou ODBC. Pour pouvoir utiliser JDBC, il est nécessaire d'installer mysql-connector-java.jar. Il est préférable de copier ce fichier d'archive Java dans le même dossier où se trouve la version java actuelle utilisée dans LibreOffice. Il s'agit probablement d'un sous-dossier comme… javapath… /lib/ext pour une installation Linux.

Alternativement, le dossier approprié contenant l'archive Java peut être défini via **Outils > Options > LibreOffice > Avancé > Options Java > Chemin de Classe**, ou par macro comme décrit page 97.

Connexion MySQL via ODBC

Pour vous connecter via ODBC, vous devez bien sûr avoir installé le logiciel ODBC. Les détails sur la façon de procéder ne sont pas donnés ici.

Après l'installation du logiciel, il peut arriver que LibreOffice refuse le service car il ne trouve pas la bibliothèque libodbc.so. Dans la plupart des systèmes, libodbc.so.2 sera présent. Vous devrez créer un lien vers ce fichier dans le même dossier avec le nom libodbc.so.

Dans les fichiers odbcinst.ini et odbc.ini, qui sont nécessaires pour le système, vous devez faire des entrées similaires à ce qui suit :

odbinst.ini

```
[MySQL ODBC 8.0 Unicode Driver]
Driver=/usr/lib/x86_64-linux-gnu/odbc/libmyodbc8w.so
UsageCount=1

[MySQL ODBC 8.0 ANSI Driver]
Driver=/usr/lib/x86_64-linux-gnu/odbc/libmyodbc8a.so
UsageCount=1
```

odbc.ini

```
mon-connecteur]
Description = MySQL connection to database
Driver = MySQL
Database = libretest
```

```
Server = localhost
User = lotest
Password = libre
Port = 3306
Socket = /var/run/mysqld/mysqld.sock
Option = 3
Charset = UTF8
```

Dans les systèmes Linux, ces deux fichiers se trouvent dans le dossier /etc/UnixODBC. Si vous n'entrez pas le jeu de caractères que vous allez utiliser, il peut y avoir des problèmes avec les caractères accentués même si la configuration est la même dans MySQL/MariaDB et dans Base.

Les détails des paramètres de connexion peuvent être trouvés dans le manuel MySQL (*MySQL Handbook*.).

Connexion à une base de données MySQL avec l'assistant de base de données

Pour accéder à une base de données MySQL existante avec une connexion directe, procédez comme suit.

À l'étape 1 de l'assistant de base de données, sélectionnez **Se connecter à une base de données existante**. Dans la liste des formats de base de données dans le menu déroulant (Figure 14), sélectionnez **MySQL**. Cliquez sur **Suivant>>**.

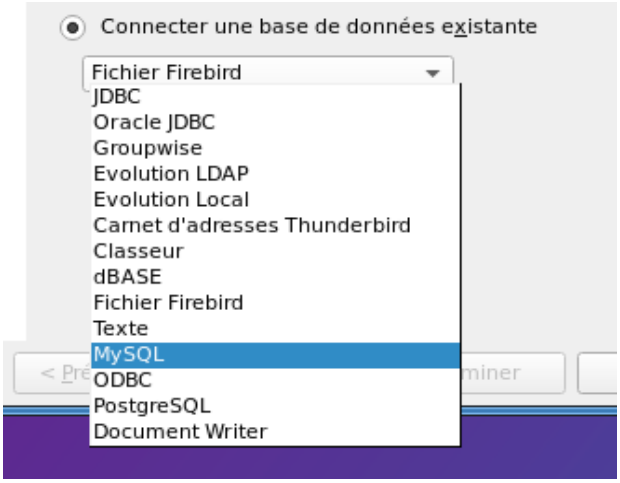


Figure 14: Connexion à une base MySQL existante

À l'étape 2 de l'assistant de base de données (figure 4), sélectionnez pour vous connecter en utilisant ODBC ou JDBC ou directement.

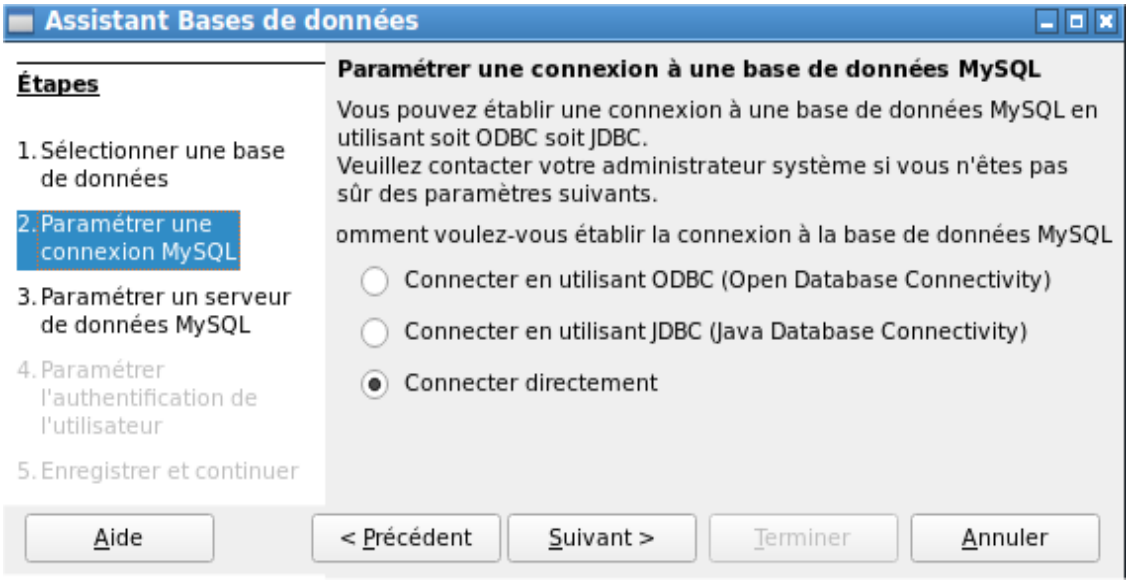


Figure 15: Etape 2 de l'assistant : configurer la connexion MySQL

Connexion directe

La connexion directe est la meilleure pour la vitesse et la fonctionnalité. À l'étape 3 (figure 16), remplissez les informations requises.

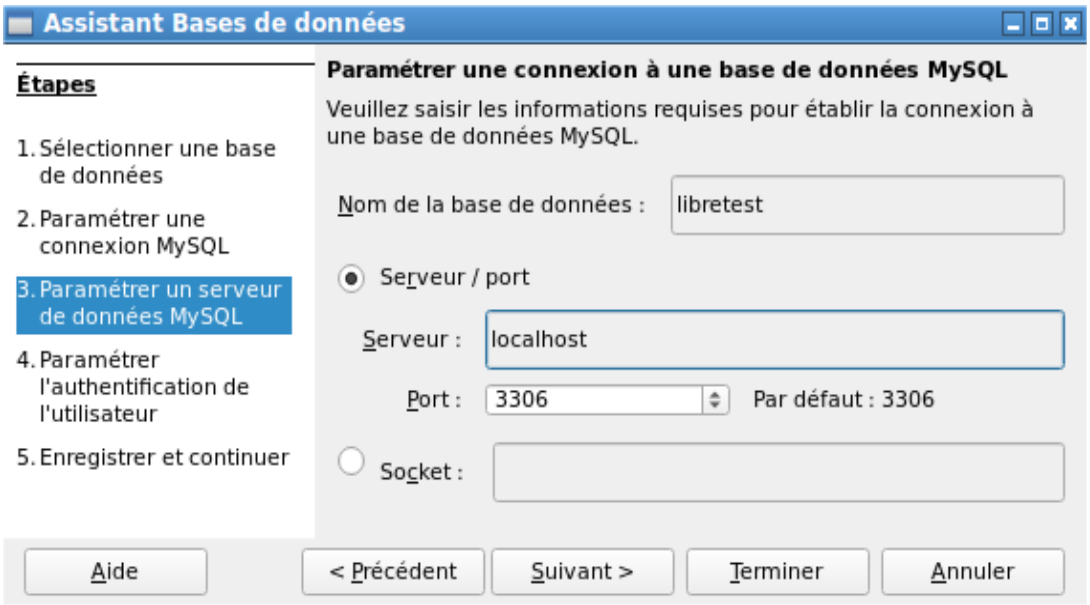


Figure 16: Etape 3 de l'assistant : entrez les informations requises pour la connexion

Le nom de la base de données doit être connu. Si le serveur se trouve sur le même ordinateur que l'interface utilisateur dans laquelle la base de données doit être créée, vous pouvez sélectionner localhost comme serveur. Sinon, vous pouvez utiliser une adresse IP ou, selon la structure du réseau, le nom de l'ordinateur ou même une adresse Internet. Il est ainsi possible pour Base d'accéder à une base de données qui se trouve sur la page d'accueil de quelqu'un d'autre.

Lorsque vous travaillez avec Base sur Internet, vous devez savoir comment la connexion est configurée. La connexion est-elle sécurisée ? Comment le mot de passe est-il transmis ?

Toute base de données accessible sur Internet doit être protégée par un nom d'utilisateur spécifique avec un mot de passe. Cela fournit un moyen direct de tester si la connexion doit continuer. Un utilisateur correspondant doit être configuré dans MySQL ou MariaDB pour le serveur défini.

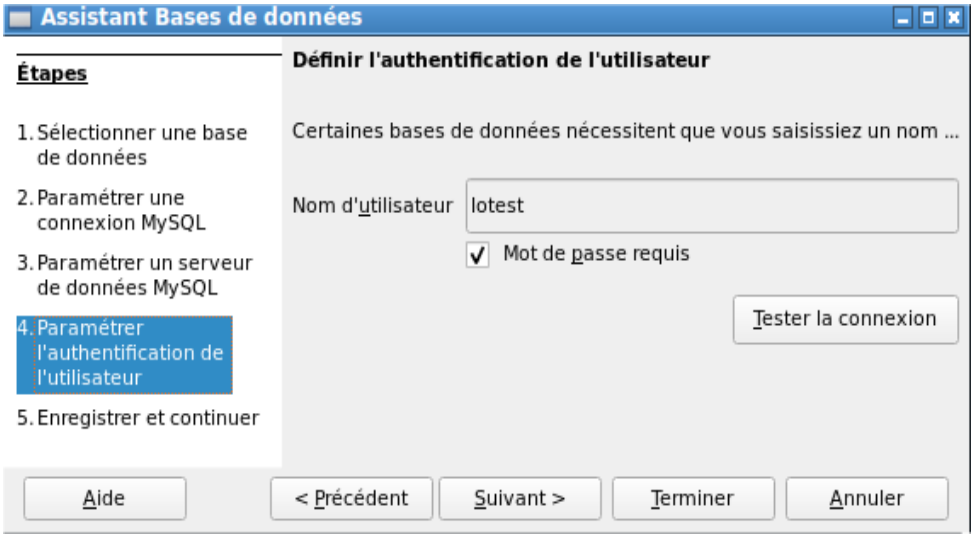
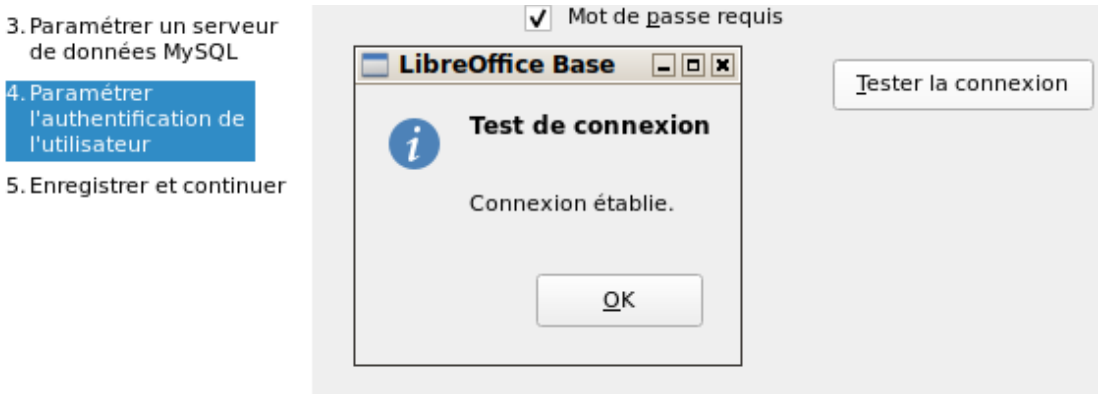


Figure 17: Etape 4 de l'assistant Configurer l'authentification des utilisateurs

À l'étape 4, indiquez un nom d'utilisateur et sélectionnez **Mot de passe requis**. Cliquez sur le bouton **Tester la connexion** pour lancer l'authentification avec le nom d'utilisateur donné. Après avoir entré le mot de passe, vous êtes informé si la connexion a réussi. Si, par exemple, MySQL n'est pas en cours d'exécution, vous recevrez un message d'erreur.

Note

À chaque fois que vous accédez au fichier de base de données, la boîte de dialogue ci-dessous apparaît lorsque vous accédez pour la première fois à la base de données MySQL.



Cliquez sur **Suivant>>** pour afficher l'étape 5 de l'assistant de base de données (Figure 18). Sélectionnez **Non, je ne souhaite pas que la base de données soit référencée** et **Ouvrir la base de données pour édition**. Cliquez sur **Terminer**.

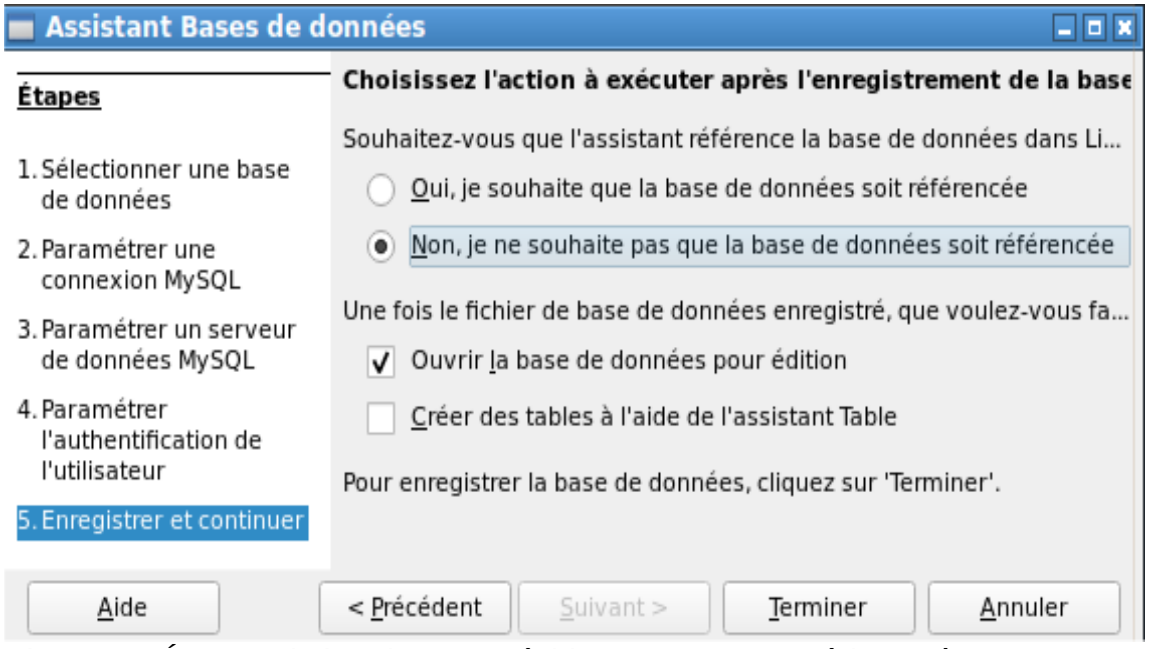


Figure 18: Étape 5 de l'assistant : Décidez comment procéder après l'enregistrement de la base de données.

Dans cet exemple, la base de données ne sera pas référencée, car elle n'est construite que pour les tests. L'enregistrement n'est nécessaire que si d'autres programmes tels que Writer doivent accéder aux enregistrements, par exemple pour un publipostage.

L'assistant termine la configuration de la connexion en enregistrant la connexion à la base de données. Le fichier de base est créé et une vue des tables de la base de données MySQL est ouverte (Figure 19). Les tables de la base de données sont affichées sous le nom de la base de données elle-même.

À ce stade, le fichier *.odb contient uniquement les informations de connexion qui seront lues à chaque lancement de la base de données, de sorte que les tables de la base de données MySQL soient accessibles.

Certains pilotes afficheront uniquement la base de données libretest pour laquelle la connexion a été commandée. D'autres pilotes affichent également d'autres bases de données MySQL ou

MariaDB sur le même serveur. Ce qui sera vu dépend totalement du pilote spécifique et du système d'exploitation utilisé.

Même avec les pilotes affichant une seule base de données, l'accès à d'autres tables pour les requêtes est possible si l'utilisateur de la base de données (le plus grand dans l'exemple ci-dessus) peut accéder aux enregistrements avec son mot de passe. Contrairement aux précédents pilotes natifs de LibreOffice, celui-ci ne fournit pas d'accès en écriture aux autres bases de données MySQL sur le même serveur.

Note

Les instructions du paragraphe ci-dessus dépendent du système d'exploitation et du pilote utilisés. Open SUSE donnera les résultats vus. Ubuntu, en utilisant le dernier pilote MySQL ou en se connectant directement au serveur MySQL, permettra un accès visuel et en écriture à toutes les bases de données pour lesquelles l'utilisateur a des droits d'accès. D'autres systèmes d'exploitation et pilotes peuvent donner des résultats différents.

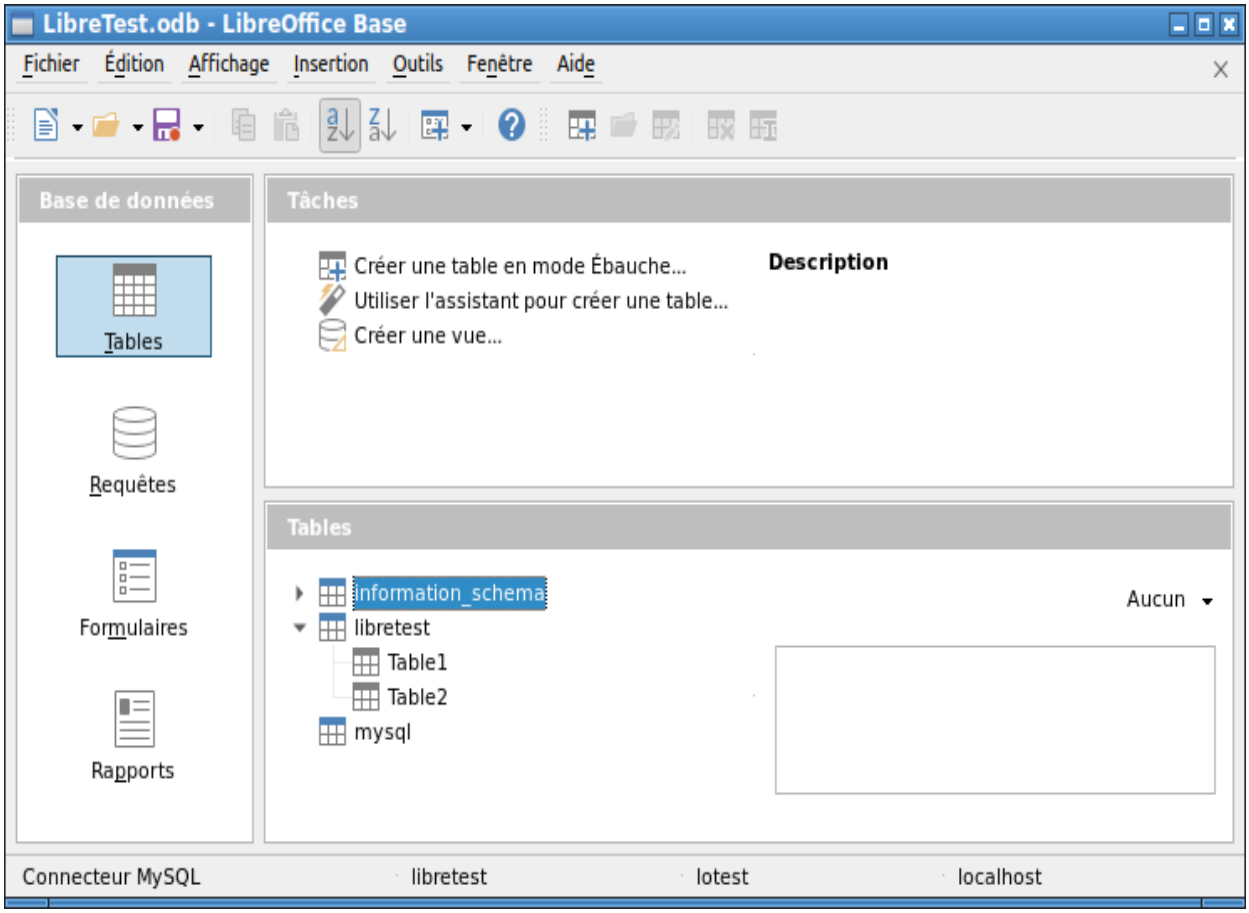


Figure 19: Vue du fichier de base de données ouvert avec vue d'ensemble des tables et dans le pied de page de la désignation du pilote utilisé MySQL (Natif), le nom de la base de données libretest, l'utilisateur de la base de données lotest et le serveur sur lequel la base de données s'exécute, localhost. Réalisé avec Lubuntu 20,04

Contrairement à la base de données interne de Base, les requêtes dans MySQL nécessitent le nom de la base de données pour définir les tables. Par exemple :

```
… FROM "lotest"."Classe" AS "Classe",…
```

Il était nécessaire de donner à la combinaison du nom de la base de données et du nom de la table un autre nom (alias) en utilisant « AS ». Depuis plusieurs versions, l'utilisation d'AS a été abandonnée avec l'alias écrit sans lui. Par exemple :

```
… FROM "lotest"."Classe" "Classe",…
```

Les tables peuvent être créées et supprimées dans la base de données. Les valeurs incrémentées automatiquement (AutoChamps) fonctionnent et peuvent être sélectionnées au moment de la conception de la table. Dans MySQL, les valeurs commencent à 1.

Connexion à l'aide d'ODBC

Les premières étapes pour établir une connexion ODBC sont les mêmes que pour une connexion directe. Si une connexion ODBC à MySQL est sélectionnée à la deuxième étape, la page illustrée à la figure 20 apparaît dans l'assistant de base de données.

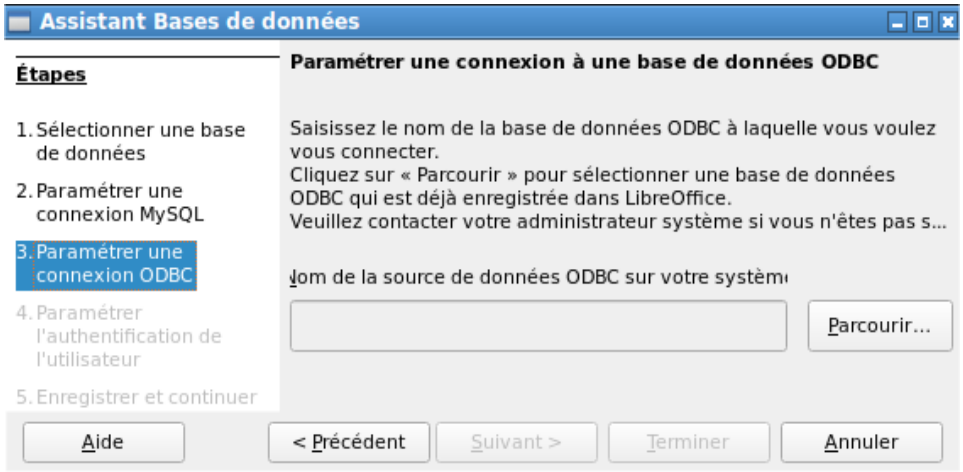


Figure 20: Configuration d'une connexion à une base de données ODBC

La source de données ODBC n'a pas besoin d'avoir le même nom que la base de données dans MySQL elle-même. Ici, vous devez entrer le nom qui est répertorié dans le fichier `odbc.ini`. La façon la plus simple de le faire est de lire le nom directement dans `odbc.ini` en utilisant le bouton **Parcourir**.

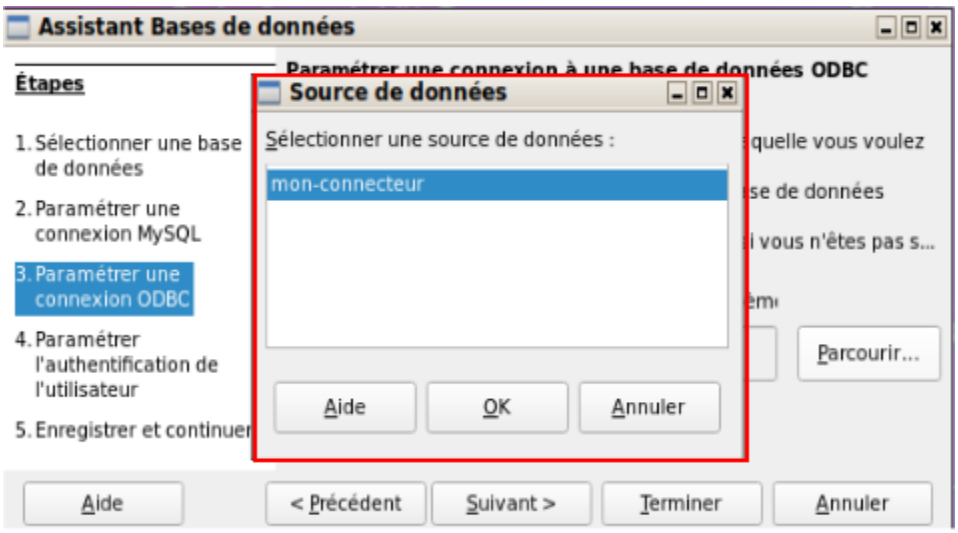


Figure 21: Choisir une source de données

Le nom du fichier défini dans odbc.ini apparaît. Ici aussi, lorsque vous vous connectez à une base de données, les autres tables du serveur MySQL peuvent être lues assez facilement.

Les étapes 4 et 5 sont identiques à celles d'une connexion directe.

Connexion à l'aide de JDBC

Pour une connexion JDBC, les premières étapes sont les mêmes. La différence n'apparaît qu'à l'étape 3 (Figure 22).

L'assistant demande les mêmes informations que pour une connexion directe. Le nom de la base de données est celui utilisé par MySQL lui-même.

Utilisez le bouton **Tester la classe** pour tester si l'archive mysql-connector-java.jar est accessible par Java. Cette archive doit être soit sur le chemin de la version Java choisie, soit directement intégrée dans LibreOffice.

Toutes les étapes suivantes sont identiques aux connexions précédentes. Les connexions à d'autres bases de données sur le même serveur MySQL sont accessibles par tout utilisateur disposant des droits d'accès sur celles-ci.

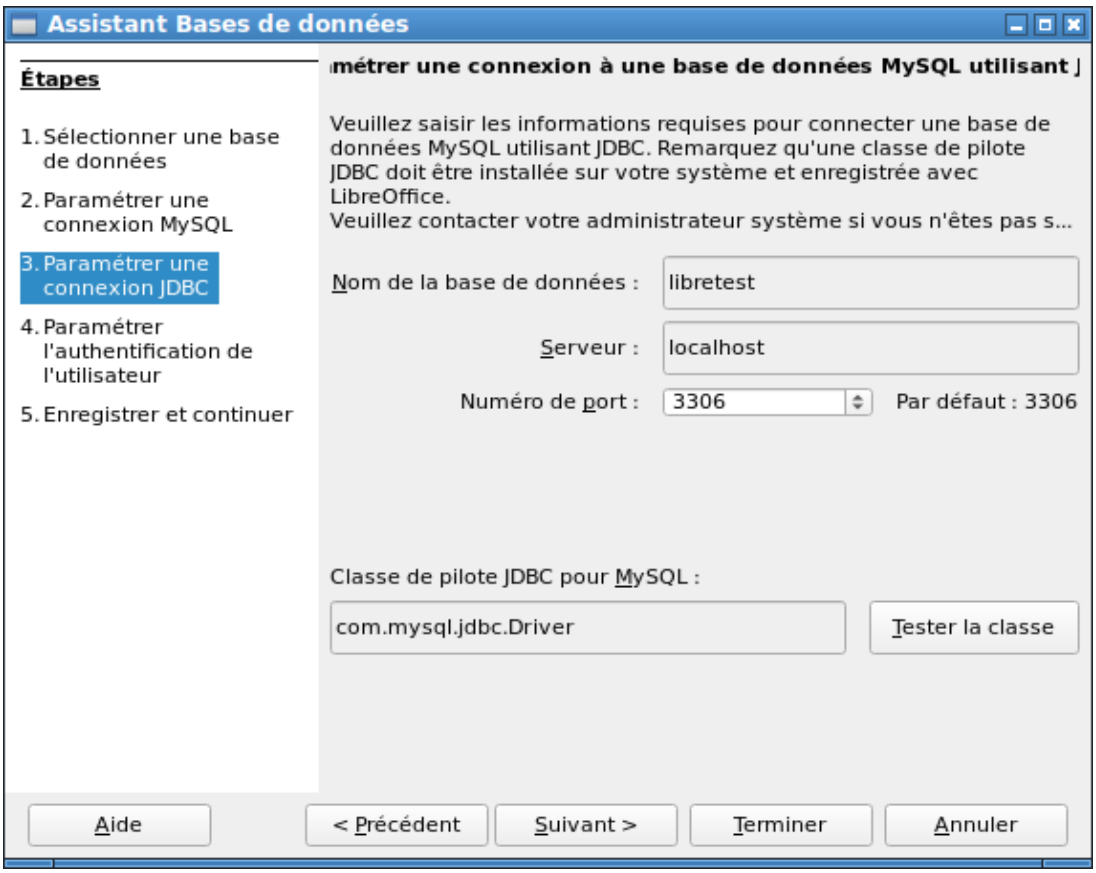


Figure 22: Configuration d'une connexion à l'aide de JDBC

PostgreSQL

LibreOffice a un pilote direct pour les bases de données PostgreSQL, qui est préinstallé. Pour garantir une connexion sécurisée, suivez ces brèves instructions pour les premières étapes après l'installation de PostgreSQL.

Créer un utilisateur et une base de données

Les étapes suivantes sont nécessaires après l'installation à l'aide du gestionnaire de packages d'OpenSUSE. Vous pouvez en supposer des similaires dans d'autres systèmes d'exploitation.

1. L'utilisateur postgres doit recevoir un mot de passe. Cela peut être fait à l'aide de l'utilitaire du système d'exploitation.
2. Le serveur Postgres doit être lancé par l'administrateur:
   `service postgresql start` ou `rcpostgresql start`.

3) L'utilisateur se connecte en tant que postgres à la console avec:
   `sudo –i –u postgres`

4. Un utilisateur de base de données non privilégié, ici appelé lotest, est créé avec un mot de passe:
   `create user –P lotest`
5. Afin de permettre à l'utilisateur de la base de données de se connecter à la base de données qui doit être créée, une entrée dans le fichier **var/lib/pqsql/data/pg_hba.conf** doit être modifiée. Ce fichier comprend les méthodes utilisées pour identifier les utilisateurs à différents niveaux. La méthode utilisée par LibreOffice pour communiquer est la méthode du « password » et non la méthode « ident » telle que définie initialement dans le fichier.

6. L'utilisateur système « postgres » se connecte avec « psql » :
   `psql –d template1 –U postgres`
7. L'utilisateur système crée la base de données « libretest » :
   `CREATE DATABASE libretest ;`

Connexion directe à la base

Choisissez l'entrée PostgreSQL à l'étape 1 de l'assistant. Pour établir la connexion, indiquez le nom de la base de données (dbname) et l'hôte. Dans certaines circonstances, il est nécessaire de donner le nom d'hôte complet, y compris le nom de domaine.

Figure 23: Configurer une connexion directe

L'authentification de l'utilisateur (étape 3) est exactement la même que pour MySQL.

La boîte de dialogue **Enregistrer sous** (Figure 24) montre les différents schémas de PostgreSQL. Le seul qui peut réellement être enregistré est le schéma public, sauf si des droits étendus ont été accordés à cet utilisateur.

Note

Si des tables de la base de données interne HSQLDB sont copiées vers PostgreSQL, l'assistant d'importation utilise les noms de table simples de HSQLDB, par exemple Table1. Cependant, l'importation sous ce nom entraînera des erreurs. Au lieu de cela, le nom du schéma doit être ajouté au début pour que Table1 devienne public. Table1.

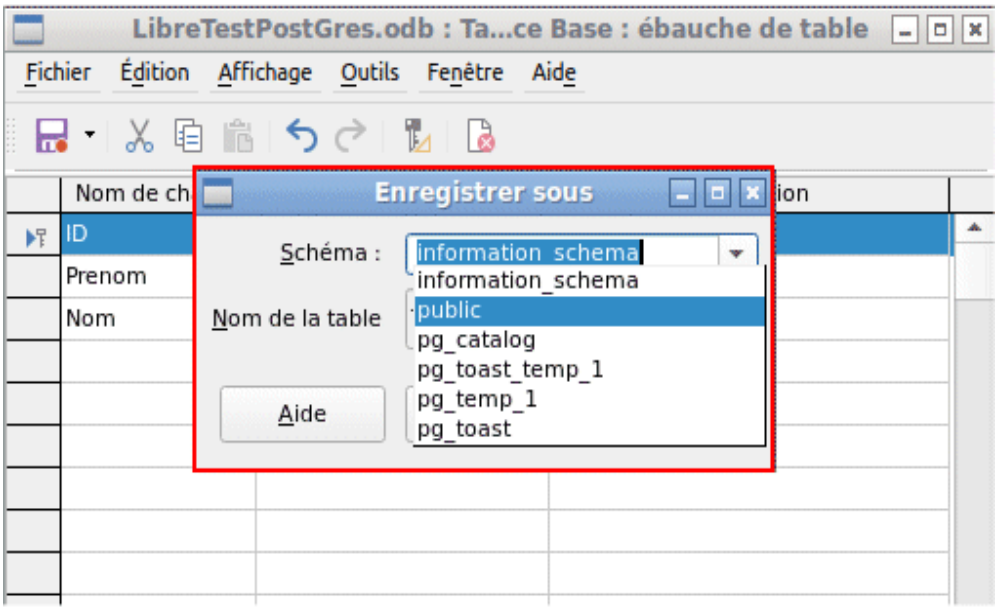


Figure 24: Enregistrement dans le schéma public

Lors de la création des tables, Base peut suggérer des types de données que le moteur intégré ne connaît pas.

Les différents schémas apparaissent dans la vue de table de PostgreSQL (Figure 26). Dans le schéma public, vous pouvez voir une table appelée Table1.

Si la base de données est ouverte pour la première fois, vous verrez un grand nombre de tables dans la zone Schéma d'informations. Ces tables, comme celles de la zone pg_catalog, ne peuvent pas être lues ou écrites par l'utilisateur ordinaire. Ces différents domaines sont appelés schémas dans PostgreSQL. Les utilisateurs créent de nouvelles tables dans le schéma public.

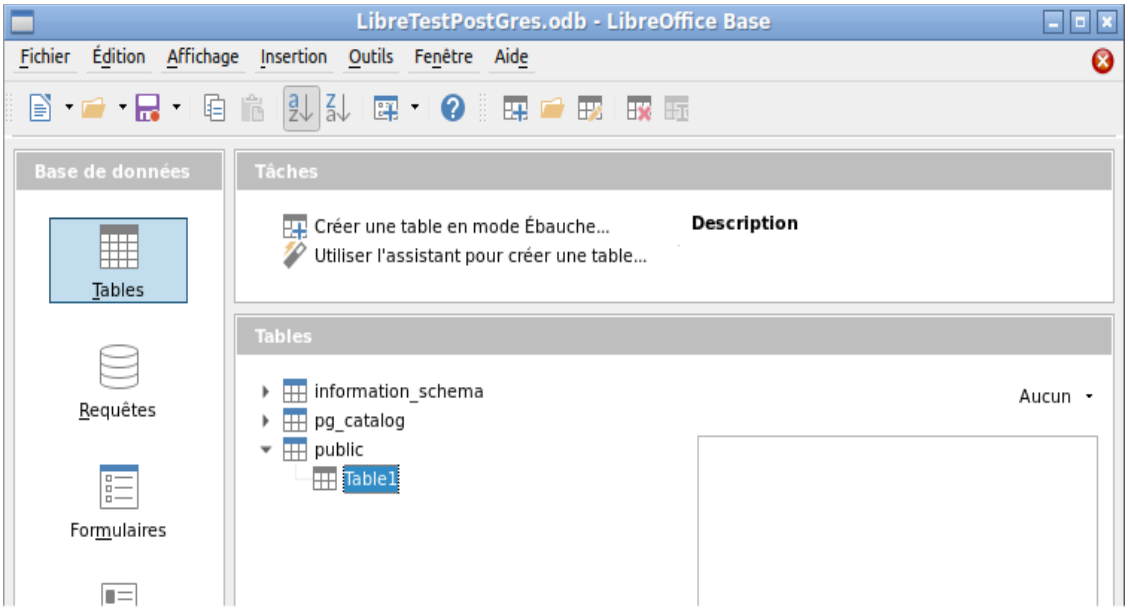


Figure 25: Vue de table de PostgreSQL dans LibreOffice Base

Bases de données dBase

Les bases de données dBase ont un format dans lequel toutes les données sont contenues dans des tables distinctes préalablement initialisées. Les liens entre les tables doivent être réalisés par programme. Les relations ne sont pas prises en charge.

Le format dBase est particulièrement adapté à l'échange et à l'édition complète des données. De plus, les feuilles de calcul peuvent accéder directement aux tables dBase.

dBase n'a aucun moyen d'empêcher la suppression, par exemple, des supports d'une base de données de bibliothèque qui continuent à être référencés dans la table des prêts de supports.

Note

À l'heure actuelle, les seules bases de données dBase reconnues sont celles contenues dans des fichiers dont la terminaison *.dbf est en minuscules. Les autres extensions de fichiers, telles que *. DBF ne sont pas reconnues. (Bogue 46180)

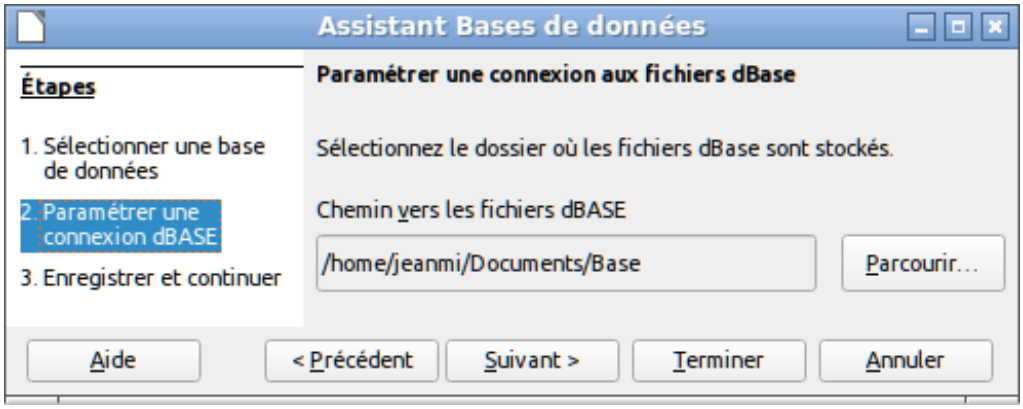


Figure 26: Configuration d'une connexion aux fichiers dBase

La connexion est établie avec un dossier spécifique. Tous les fichiers *.dbf de ce dossier seront inclus et affichés dans la base de données *.odb et peuvent être liés ensemble à l'aide de requêtes.

Les tables dans dBase n'ont pas de clé primaire. Ils peuvent en principe être décrits comme correspondant aux feuilles de calcul dans Calc.

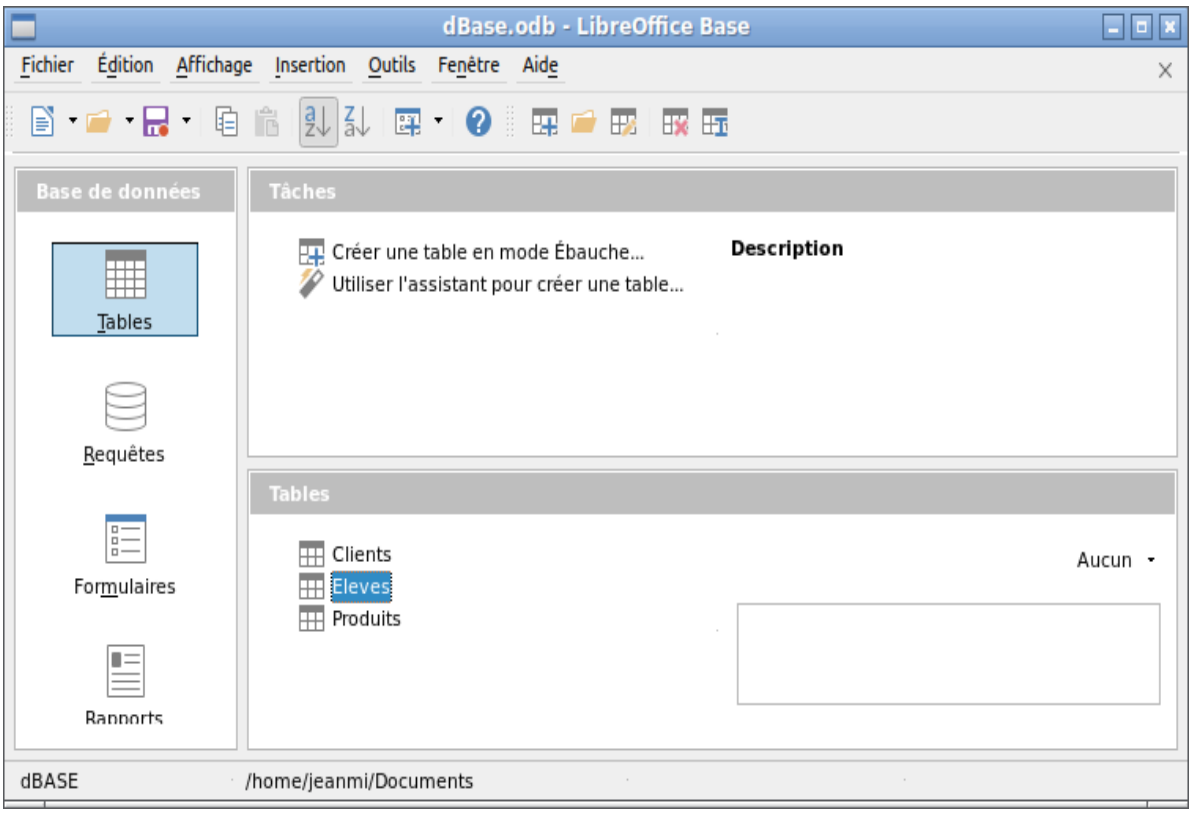


Figure 27: Tables dans fichier dBase

Des tables peuvent être créées et seront ensuite copiées en tant que nouveaux fichiers dans le dossier précédemment sélectionné (Figure 27).

Le nombre de types de champs différents pour une nouvelle table dBase est nettement inférieur à celui du format HSQLDB interne. Dans la figure 28 suivante, il existe encore des types de champs avec le même nom de type .

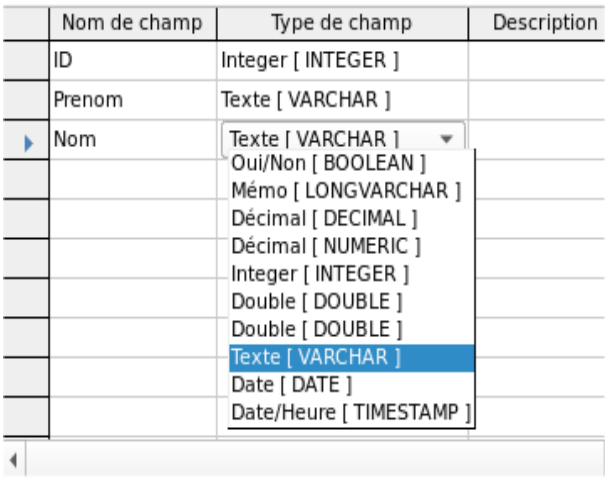


Figure 28: Types de champs dans une nouvelle table dBase

Base prend en charge le codage du système d'exploitation. Par conséquent, les anciens fichiers dBase développent facilement des erreurs lorsque des caractères spéciaux sont importés. Le jeu de caractères peut être corrigé ultérieurement en utilisant **Edition > Base de données > Propriétés > Paramètres supplémentaires** (Figure 29).

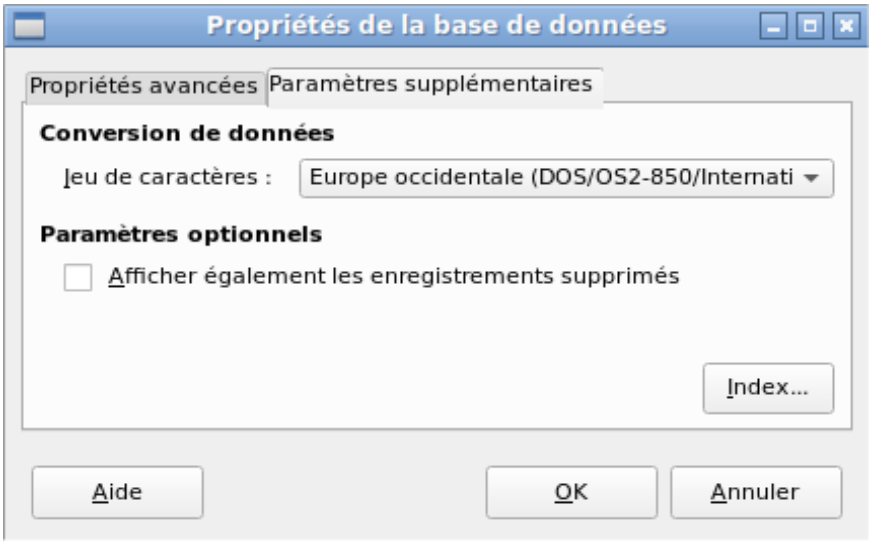


Figure 29: Correction du jeu de caractères

Note

L'assistant d'importation pour dBase a des problèmes avec la reconnaissance automatique des types de champs numériques et des champs Oui/Non (bogue 53027). Cela peut nécessiter des corrections ultérieures.

Feuilles de calcul

Les classeurs Calc ou Excel peuvent également être utilisés comme source de tables pour les bases de données. Si, toutefois, un classeur Calc est utilisé, aucune modification des données de la table n'est possible. Si le document Calc est toujours ouvert, il sera protégé en écriture.

Les seules questions auxquelles il faut répondre sont l'emplacement du fichier de feuilles de calcul et s'il est protégé par mot de passe ou non. Base ouvre alors le fichier et inclut toutes les feuilles de calcul dans le document. La première ligne est utilisée pour les noms de champ et les noms de feuille de calcul deviennent les noms de table.

Les relations entre les feuilles de calcul ne peuvent pas être configurées dans Base, car Calc ne convient pas pour une utilisation en tant que base de données relationnelle.

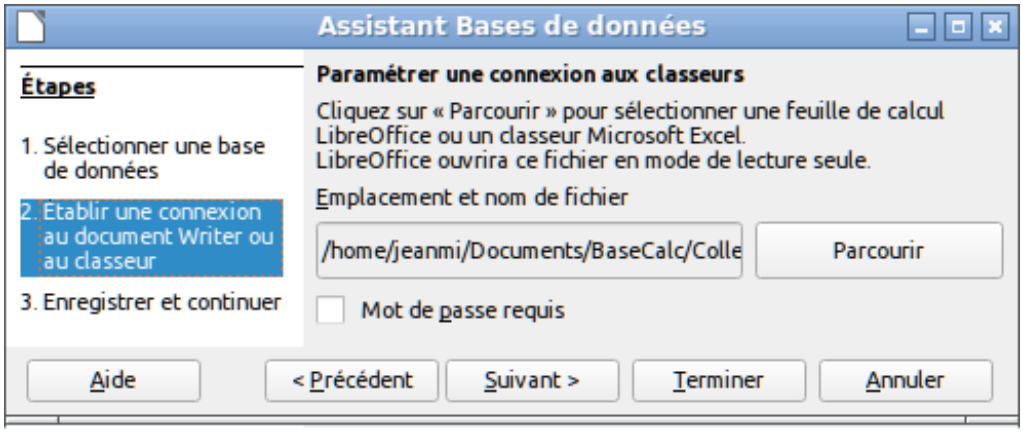


Figure 30: Configurer une connexion à un fichier de feuilles de calcul

Carnet d’adresses Thunderbird

L’assistant cherchera automatiquement une connexion à un carnet d’adresses, par exemple tel qu’il est utilisé dans Thunderbird. L’Assistant vous demandera l’emplacement du fichier ODB qui sera produit.

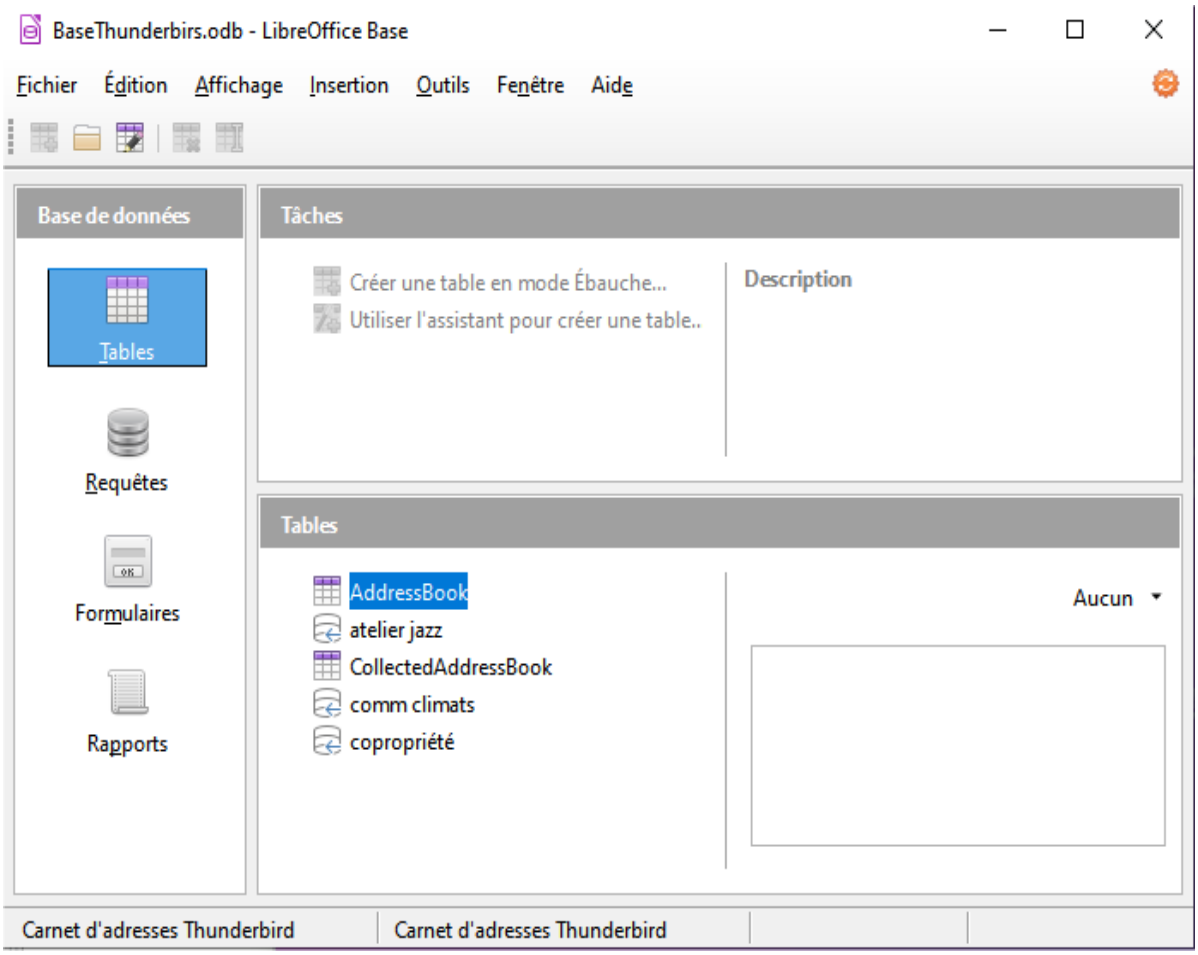


Figure 31: Tables dans un carnet d’adresses Thunderbird

Toutes les tables sont affichées. Comme pour les feuilles de calcul Calc, les tables ne sont pas modifiables. Base utilise les données de la table uniquement pour les requêtes et les applications de publipostage.

Note

Seul le fichier d’adresses personnelles est lu comme carnet d’adresses sous Linux et macOS. Les groupes collectés ne sont actuellement visibles que dans le cadre des adresses personnelles. Les groupes des adresses collectées ne sont pas affichés.

Tables Texte

Dans Base, vous pouvez créer une base de données complète en accédant aux tables de texte. Les tables de texte sont également accessibles à partir d’une base de données interne.

Tables de texte dans une base de données HSQLDB interne

Le format *.csv est un format d'échange courant entre les bases de données. Les enregistrements sont stockés sous une forme qui peut être lue et modifiée par un simple éditeur de texte. Les champs individuels sont séparés par des virgules. Si un champ contient du texte qui inclut une virgule, le champ de texte est placé entre guillemets. Chaque nouvel enregistrement commence par une nouvelle ligne.

Par exemple, le contenu d'un carnet d'adresses qui est dans un format non pris en charge par un pilote de Base peut être importé via un fichier *.csv (en utilisant Calc comme intermédiaire si nécessaire) ou le fichier est directement importé dans la base de données en tant que table de texte. Pour y être modifiable, le fichier *.csv doit inclure un champ avec des valeurs uniques pouvant servir de clé primaire.

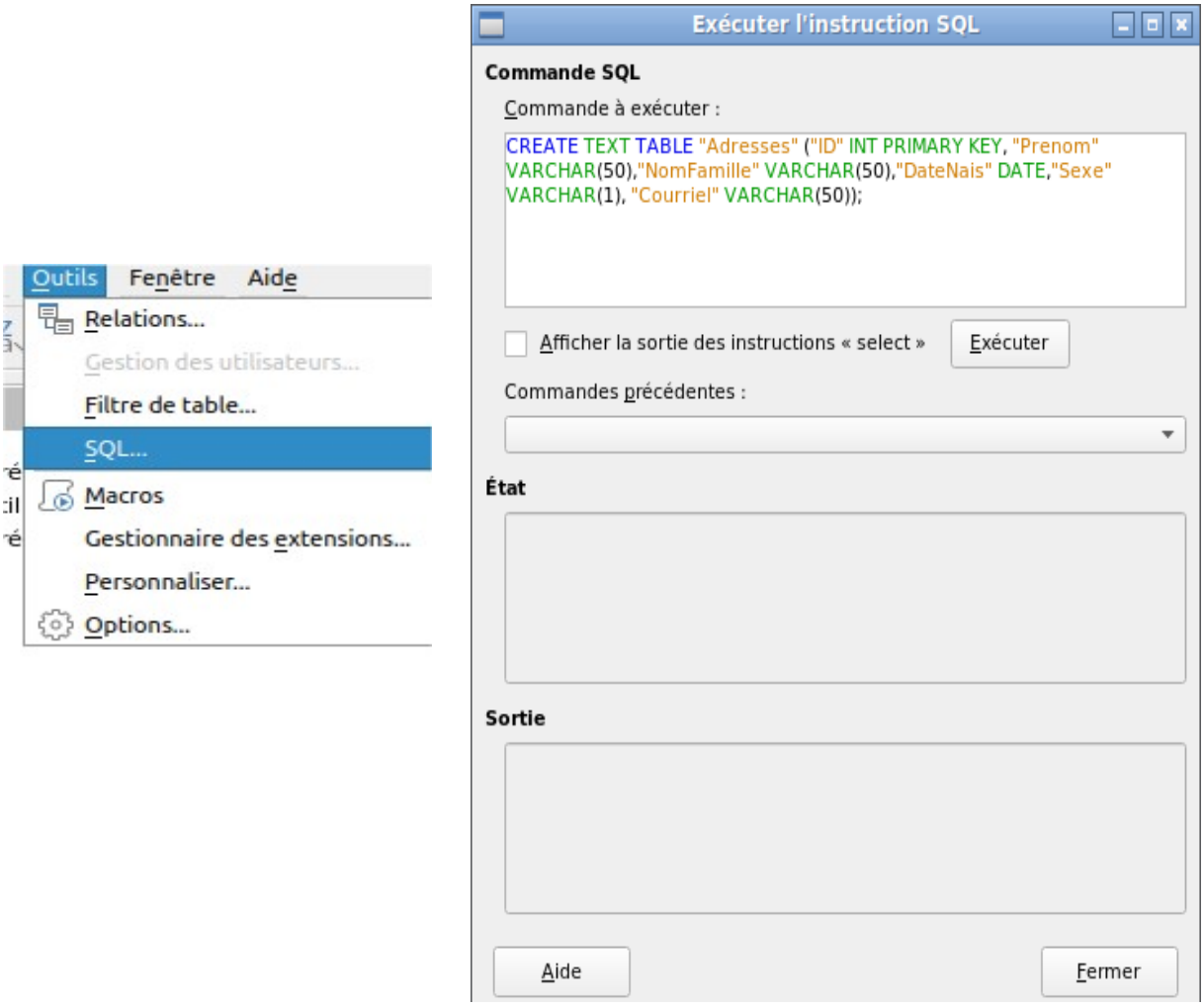


Figure 32: Création d'une table de texte à l'aide du menu Outils > SQL

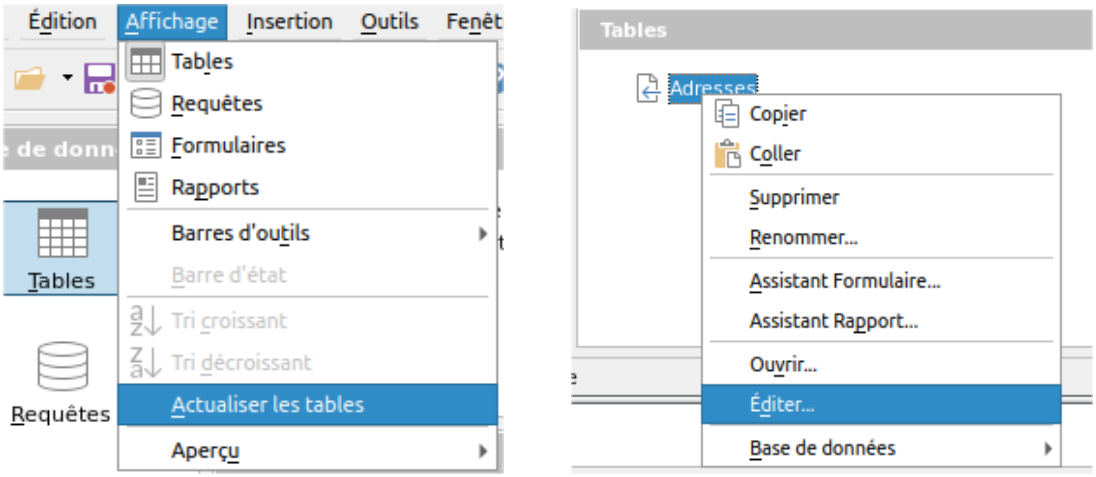
Une table texte ne peut pas être créée à l'aide de l'interface graphique[3]. Au lieu de cela, vous devez utiliser **Outils > SQL** pour créer une table texte (voir Figure 32). Les champs de la table texte doivent correspondre en type et en ordre à ceux que le fichier texte met à disposition. Par exemple, le champ ID doit contenir des entiers positifs et le champ Anniversaire doit contenir des valeurs de date sous la forme **Année-Mois-Jour**.

3 Voir la base de données supplémentaire de ce manuel, Exemple_Import_CSV.odb

La table n’est pas directement visible dans l’interface utilisateur. Utilisez **Affichage > Actualiser les tables** pour rendre les tables disponibles. Le symbole de table indique qu’il ne s’agit pas d’une table de base de données « normale ».

La table est créée dans l’interface de requête SQL avec l’instruction :

```
CREATE TEXT TABLE "Adresses" ("ID" INT PRIMARY KEY, "Prenom" VARCHAR(50),
"NomFamille" VARCHAR(50), "DateNais" DATE, "Sexe" VARCHAR(1), "Courriel"
VARCHAR(50)
```



La table est ouverte pour modification et le champ de clé primaire est remplacé par un champ à incrémentation automatique.

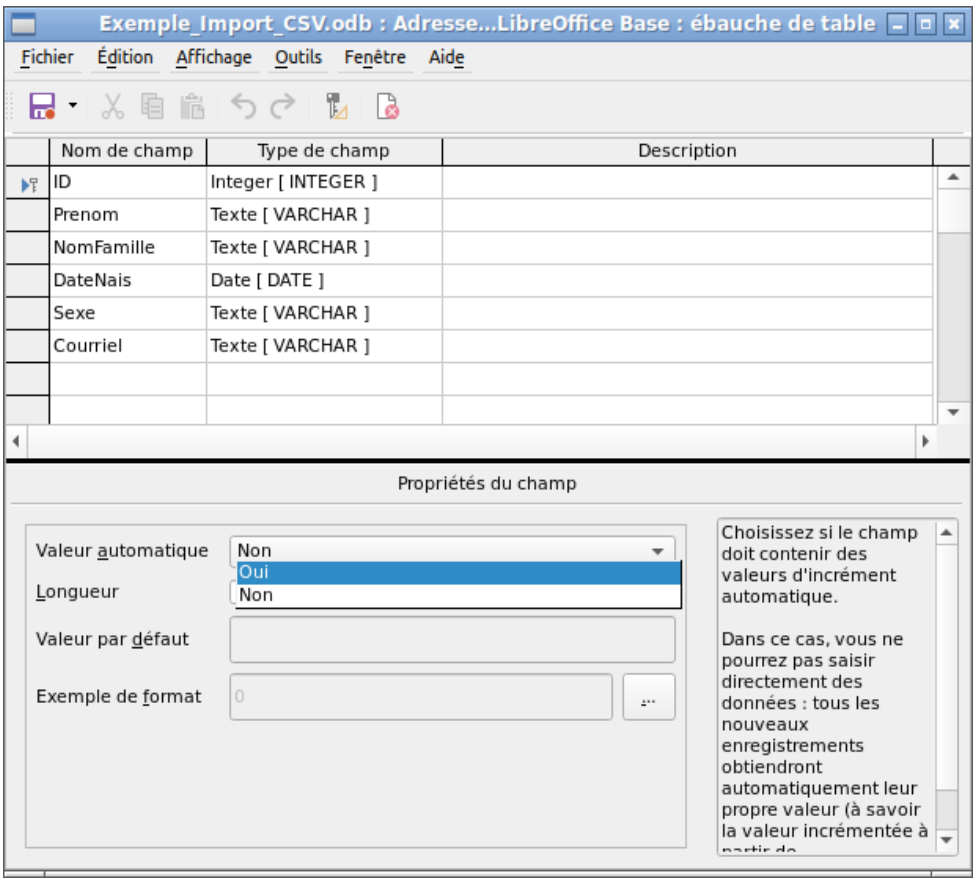
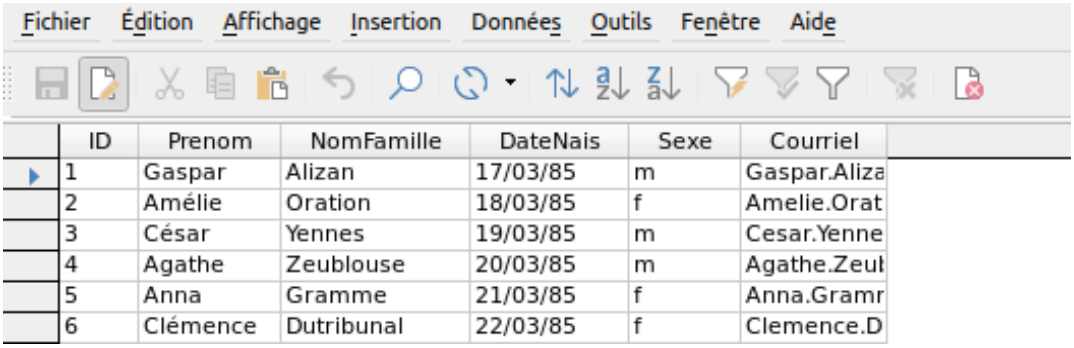


Figure 33: Edition de la table

Nous devons maintenant établir une connexion à une table de texte externe en utilisant **Outils > SQL**. La table de texte se trouve dans le même dossier que la base de données elle-même.

```
SET TABLE "Adresses" SOURCE "Adresses.csv ; encoding=UTF-8";1
```



ID	Prenom	NomFamille	DateNais	Sexe	Courriel
1	Gaspar	Alizan	17/03/85	m	Gaspar.Aliza
2	Amélie	Oration	18/03/85	f	Amelie.Orat
3	César	Yennes	19/03/85	m	Cesar.Yenne
4	Agathe	Zeublouse	20/03/85	m	Agathe.Zeul
5	Anna	Gramme	21/03/85	f	Anna.Gramr
6	Clémence	Dutribunal	22/03/85	f	Clemence.D

Figure 34: Vue du fichier CSV comme table dans Base

Après cela, la table texte est disponible pour saisie de la manière normale. Mais il convient de noter les points suivants :

- Les tables de texte peuvent être ouvertes et modifiées simultanément par des programmes de texte externes. La perte de données ne peut être exclue dans ces circonstances.

- Les changements dans les enregistrements déjà écrits entraînent l'effacement de la ligne correspondante dans le fichier d'origine et l'ajout de la nouvelle version à la fin de la table. Le tableau ci-dessus présente quatre lignes écrites avec des numéros d'identification correctement triés. Dans le fichier d'origine, le deuxième enregistrement a été modifié, conduisant à la séquence d'enregistrements suivante par ID : 1, ligne vide, 3, 4…, 2. à la fin du fichier.

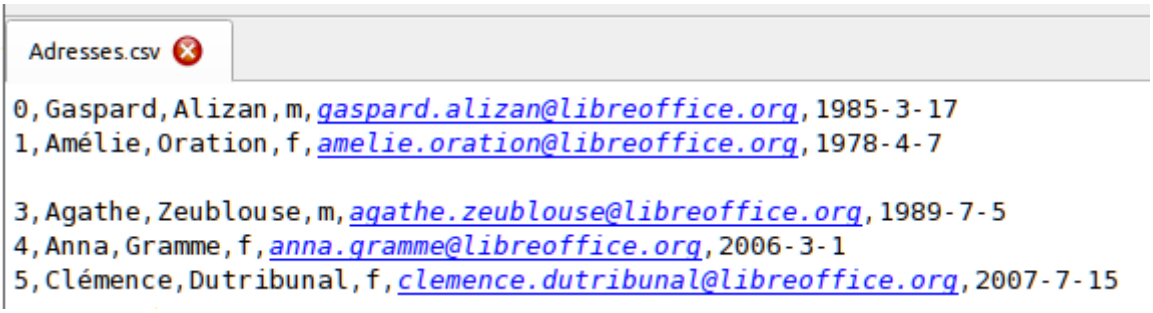


Figure 35: Vue du fichier CSV après modification dans Base

Lors de la connexion à un fichier texte, les paramètres suivants sont disponibles.

```
SET TABLE "Adresses" SOURCE "Adresses.csv ; ignore_first=false ;
all_quoted=true ; encoding=UTF-8";
```

"ignore_first=true" signifierait que la première ligne n'est pas lue. Cela a du sens si la ligne ne contient que des en-têtes de champ. La valeur par défaut interne pour HSQLDB est *false*.

Par défaut, les champs de texte HSQLDB ne sont placés entre guillemets doubles que s'ils contiennent une virgule interne, car la virgule est le séparateur de champ par défaut. Si chaque champ doit être entre guillemets, définissez all_quoted = true.

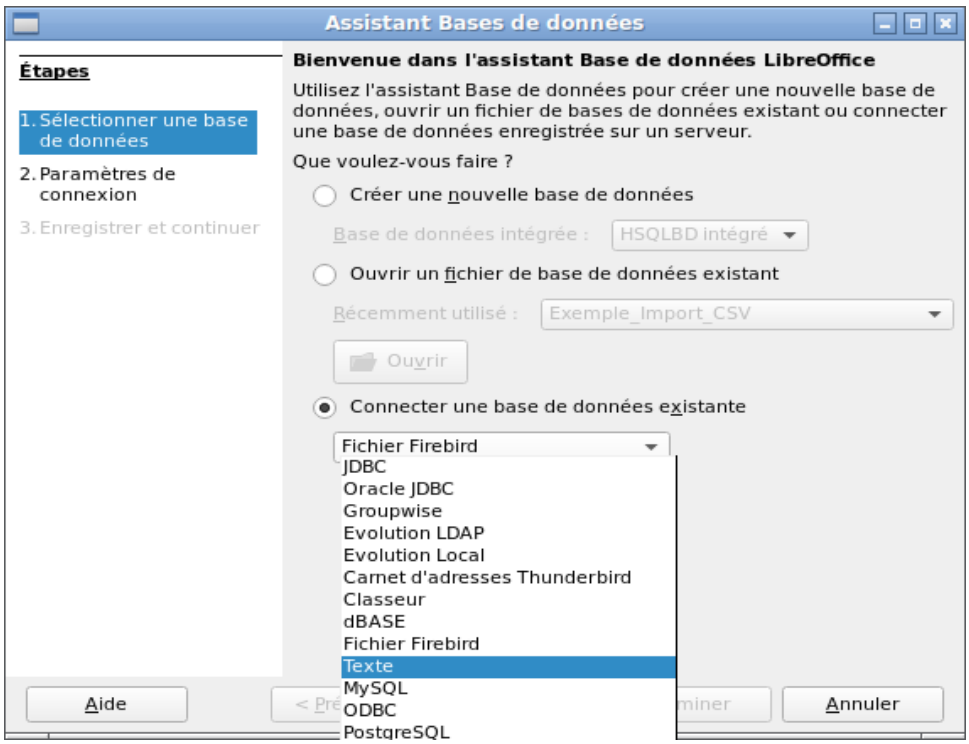
Pour plus de paramètres, voir http://www.hsqldb.org/doc/1.8/guide/guide.html#set_table_source-section

```
SET TABLE "Adresses" READONLY TRUE ;
```

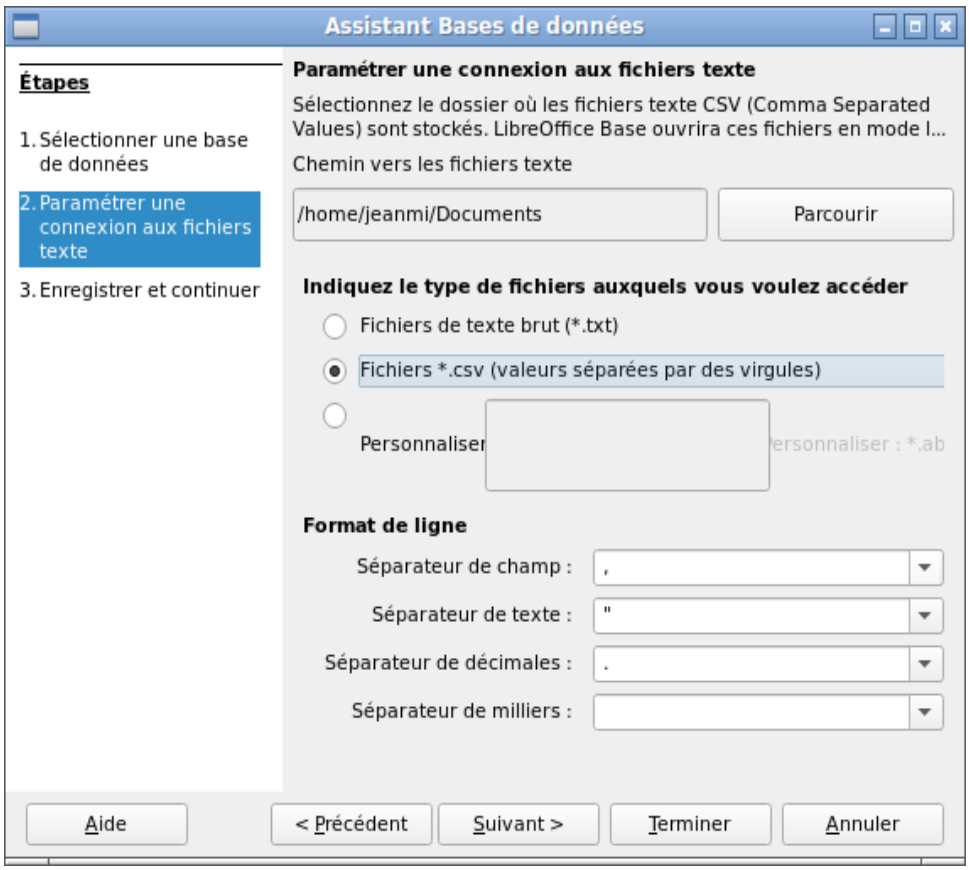
empêche que quoi que ce soit soit écrit dans la table. La table est alors disponible en lecture seule comme un carnet d'adresses à partir d'un programme de messagerie. La définition de la protection en écriture séparément n'est nécessaire que lorsqu'une clé primaire a été définie pour la table.

Les tables texte comme base d'une base de données autonome

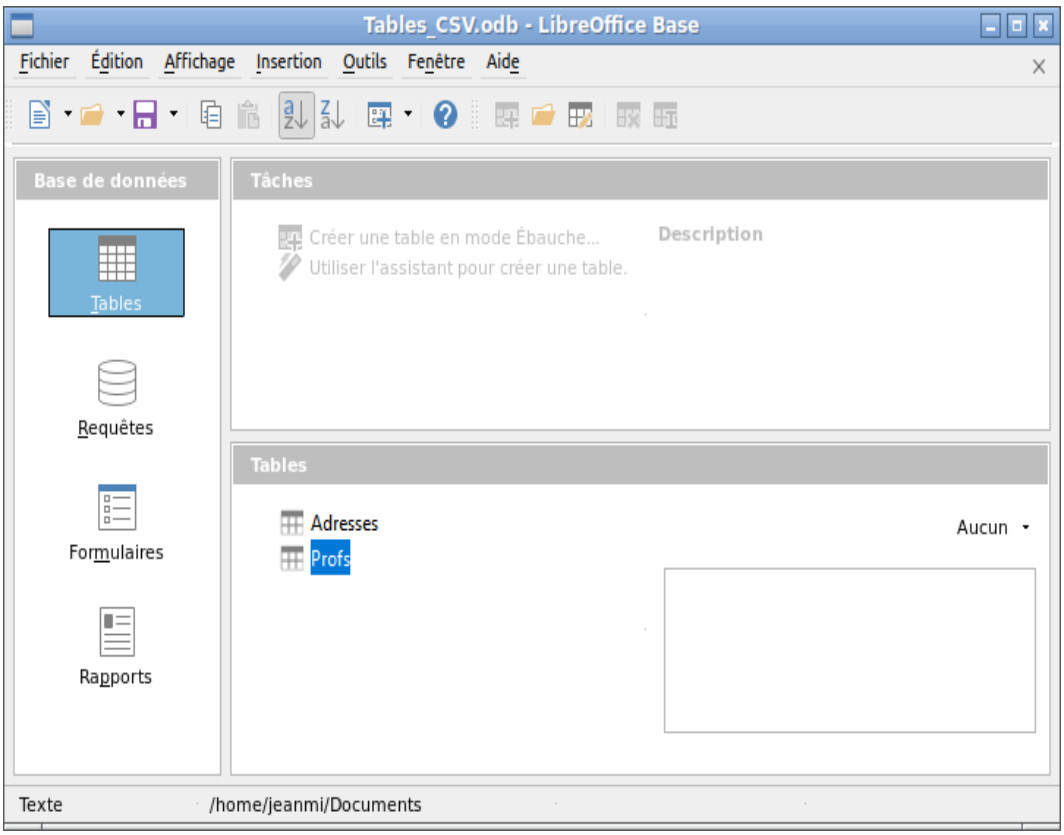
Comme dans l'exemple précédent, les fichiers *.csv sont utilisés comme source de données. En utilisant Base, un dossier contenant les fichiers *.csv est incorporé en tant que dossier de données. Nous commençons par nous connecter à une base de données existante. Ici, le format Texte a été sélectionné.



Le chemin d'accès aux fichiers texte est recherché. Dans le dossier, tous les fichiers du type spécifié seront répertoriés ultérieurement. Pour les fichiers *.csv, choisissez la deuxième option.



À ce stade, vous pouvez déjà voir un avertissement clair indiquant que les fichiers seront ouverts en lecture seule sans accès en écriture.



Dans la vue Table, toutes les tables du dossier spécifié sont affichées par leur nom de fichier mais sans le suffixe de nom de fichier. Les tâches de création de tables ne sont pas actives. Les tables elles-mêmes peuvent être lues, mais pas écrites.

L'accès aux tables par requêtes est également limité à une table à la fois et sans l'utilisation de fonctions. De même, les relations ne sont pas prises en charge/

Lorsqu'une telle base de données est utilisée pour rechercher brièvement des enregistrements dans un fichier *.csv ou pour importer un fichier *.csv dans une autre base de données en utilisant la fonction de copie, elle a rempli son objectif. Le fichier *.csv correspondant est uniquement déplacé dans le dossier spécifié et peut être directement recherché ou copié. Ces bases de données textuelles ne conviennent pas à une utilisation plus générale.

Firebird

À terme, l'ancienne version HSQLDB utilisée pour les bases de données internes devrait être remplacée par une base de données Firebird interne.

Si vous souhaitez voir ce que Firebird peut offrir, voici la procédure de configuration d'une base de données externe.

La documentation n'étant pas aussi complète que pour MySQL ou PostgreSQL, voici les étapes les plus importantes de l'installation

Créer un utilisateur et une base de données

Linux fournit des packages Firebird via ses gestionnaires de packages. Après l'installation, le serveur doit être configuré. Les étapes suivantes sous OpenSUSE 12.3 sont liées à une base de données Firebird fonctionnelle :

1. `Sysdba` est le nom d'utilisateur du compte administrateur. Le mot de passe par défaut est `masterkey`. Cela doit être modifié dans un environnement de production.
2. Pour changer le mot de passe dans un terminal : su
   `gsec –user sysdba –pass masterkey –mo sysdba –pw nouveaumotdepasse`
3. Pour accéder à une base de données fonctionnelle, vous avez besoin des droits d'administrateur sur l'ordinateur. L'utilisateur système Firebird doit avoir un mot de passe qui lui est attribué.
4. L'utilisateur de Firebird se connecte à la console:
   `su firebird`
5. Un nouvel utilisateur est créé, affiché ici avec le mot de passe par défaut d'origine pour sysdba:
   `gsec –user sysdba –pass masterkey –add lotest –pw libre`

   Cela crée un nouvel utilisateur avec le nom *lotest* et le mot de passe *libre*.
6. Ensuite, toujours en tant que superutilisateur, créez une base de données pour l'utilisateur. Pour cela, nous utilisons le programme auxiliaire `isql-fb.`
   `isql-fb`

   Vous voyez le message suivant:
   ```
   Use CONNECT or CREATE DATABASE to specify a database
   followed directly by the SQL> prompt. The appropriate entries are:

   SQL> CREATE DATABASE 'libretest.fdb'
   CON> user 'lotest' password 'libre';
   ```

Si ces tâches sont effectuées en tant qu'administrateur système root, la base de données ne sera pas affectée au bon utilisateur lorsqu'elle est mise en réseau. La base de données doit être saisie en tant qu'utilisateur firebird dans le groupe firebird. Sinon, la connexion ne fonctionnera pas plus tard.

Connexion directe à Firebird

Choisissez l'entrée **Connecter une base de données existante** à l'étape 1 de l'assistant, puis **Paramètres de connexion**. Pour établir la connexion, indiquez le nom de la base de données (dbname) et l'hôte. Dans certaines circonstances, il est nécessaire de donner le nom d'hôte complet, y compris le nom de domaine.

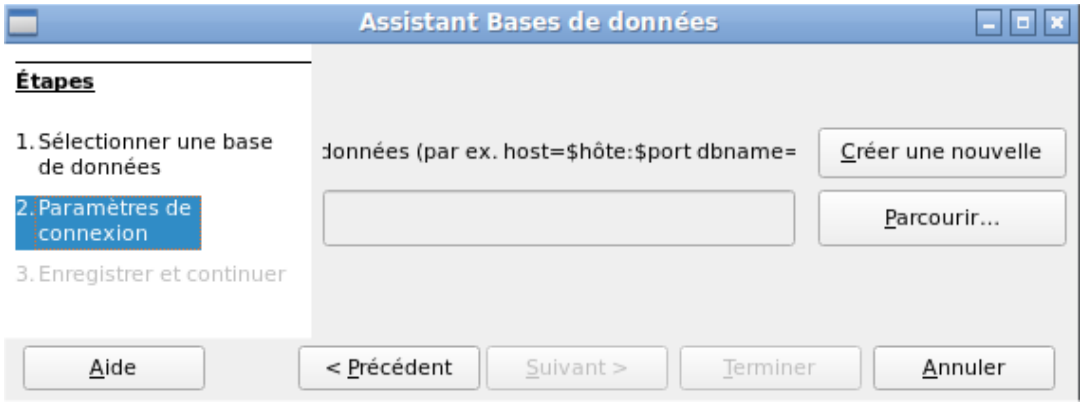


Figure 36: Mettre en place une connexion directe

L'authentification de l'utilisateur (étape 3) est exactement la même que pour MySQL.

Lors de la création des tables, Base peut suggérer des types de données que l'installation actuelle de Firebird ne peut pas gérer.

Connexion à Firebird via JDBC

Vous devrez d'abord intégrer l'archive Jar dans LibreOffice. Cependant, il n'y a pas d'archive appelée `firebird-*.jar`. Le pilote JDBC actuel peut être trouvé à http://www.firebirdsql.org/en/jdbc-driver/. Le nom du pilote commence par Jaybird…

copier l'archive `jaybird-full-x.y.z.jar` (Au moment de l'écriture de ce document, c'est la version 3.0.9) du fichier zip et placez-le dans le chemin Java de l'installation ou importez-le directement dans LibreOffice en tant qu'archive. Voir la section correspondante sur MySQL.

Pour éviter les soucis avec l'accès au bases locales, on peut définir ce chemin par macro, voir la méthode page 97.

Note :

Sur Open Suse, le pilote est installé en /usr/share/java/jaybird-full-x.y.z.jar

Les paramètres suivants sont importants lors de l'installation JDBC :

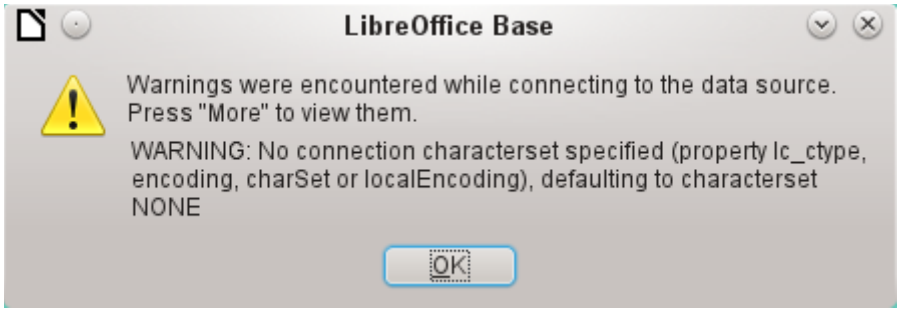
JDBC URL `jdbc:firebirdsql://host[:port]/<chemin_ou_alias>`

Driver bus name : `org.firebirdsql.jdbc.FBDriver`

Dans l'exemple ci-dessus, cela devient l'URL

```
jdbc:firebirdsql://localhost/libretest.fdb?charSet=UTF-8
```

Si vous ne spécifiez pas le jeu de caractères, vous obtiendrez cette erreur :



Lors de la création de tables, veillez à ce que le format des champs correspondants (propriétés des champs) soit conforme dès le début. Sinon LibreOffice définira le format par défaut pour toutes les valeurs numériques, qui, étrangement, est une devise.

La modification ultérieure des propriétés de champ dans les tables n'est pas possible, mais vous pouvez agrandir la table ou supprimer des champs.

Connexion à Firebird via ODBC

Vous devez d'abord télécharger le pilote ODBC approprié depuis le site http://www.firebirdsql.org/en/odbc-driver/. Ce pilote est généralement un simple fichier appelé **libOdbcFb.so**. Ce fichier est généralement placé dans un chemin d'accès généralement accessible dans le système. Il doit être exécutable.

Dans les fichiers `odbcinst.ini` et `odbc.ini` files, nécessaires au système, les entrées suivantes sont obligatoires :

odbinst.ini

```
[Firebird]
Description = Firebird ODBC driver
Driver64 = /usr/lib64/libOdbcFb.so
```

odbc.ini

```
[Firebird-libretest]
Description   = Firebird database libreoffice test
Driver = Firebird
Dbname = localhost:/srv/firebird/libretest.fdb
SensitiveIdentifier = Yes
```

Dans un système Linux, ces deux fichiers se trouvent dans le dossier /etc/unixODBC. La variable *SensitiveIdentifier* doit toujours être définie sur « Oui » pour que l'entrée dans les tables fonctionne lorsque les noms et les définitions de champ ne sont pas en majuscules.

Connexion directe à un fichier Firebird

Si une base de données serveur n'est pas requise, une connexion directe à un fichier Firebird est également possible. Ici seulement, comme pour les feuilles de calcul, le chemin d'accès au fichier Firebird doit être accessible. La connexion sera alors établie automatiquement. Le fichier Firebird doit avoir été créé à partir de LibreOffice 5.3 avec une version 3 de Firebird.

Un fichier de base de données externe peut être créé à partir d'un fichier de base de données interne :

- Le répertoire temporaire défini dans LibreOffice doit être recherché avec le gestionnaire de fichiers.
- La base de données contenant le fichier de base de données Firebird interne doit être ouverte. Ce fichier contient actuellement le fichier "firebird.fbk", un fichier de sauvegarde de données.
- Le conteneur de table doit être ouvert pour que la connexion au fichier de base de données interne soit établie.
- Un répertoire est maintenant automatiquement créé dans le répertoire temporaire, qui à son tour a plusieurs sous-répertoires. Ce répertoire ne peut pas être reconnu par son nom, mais au moment de sa création.
- Dans l'un de ces sous-répertoires, il y a à côté de «firebird.fbk» le «firebird.fdb» créé à partir de celui-ci. Ce fichier peut maintenant être copié et utilisé comme fichier de base de données externe. Il n'y a pas de protection par mot de passe pour ce fichier.
- Après la fermeture du fichier de base de données interne, le répertoire temporaire est automatiquement supprimé.

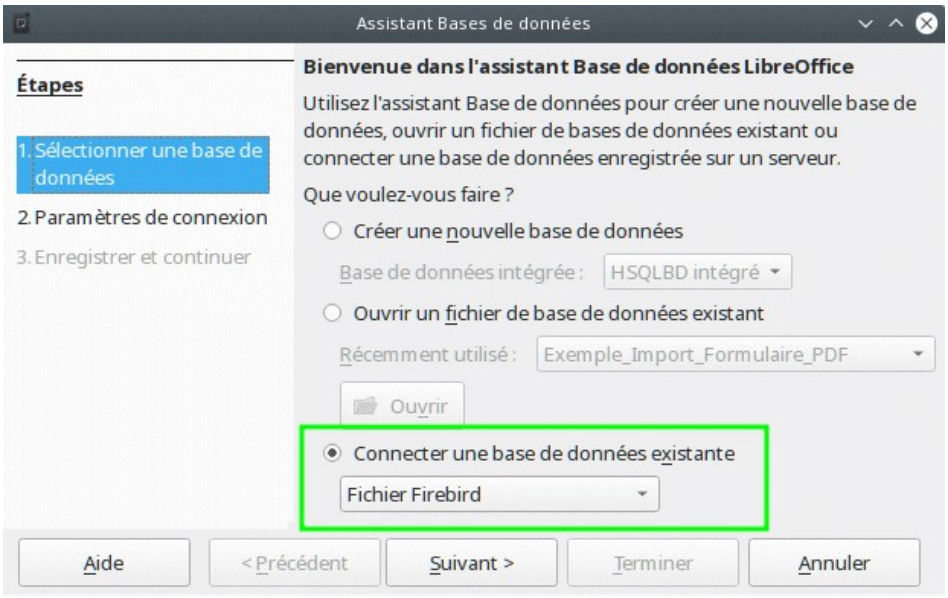


Figure 37: Ouverture d'une base Firebird - Fichier copié

Connexion d'une base de données à un HSQLDB externe

Conseil

Si, pour une raison quelconque, la base de données a été ouverte mais qu'il n'y a pas d'accès aux tables, vous pouvez utiliser **Outils > SQL** pour entrer la commande SHUTDOWN SCRIPT. La base de données peut alors être fermée et rouverte. Cela ne fonctionnera pas s'il y a déjà eu le message d'erreur *Erreur dans le fichier de script.*

Les enregistrements de la base de données sont stockés dans le sous-dossier *database* du fichier *.odb Ici, il y a deux fichiers nommés *data* et *backup*. Si le fichier de données est défectueux, il peut être restauré à partir d'une sauvegarde. Pour ce faire, vous devez d'abord modifier le fichier de propriétés, qui se trouve également dans le dossier de la base de données. Il contient une ligne `modified=no`. Changez cela en `modified=Yes`. Cela informe le système que la base de données n'a pas été correctement fermée. Lorsque vous redémarrez, le fichier de sauvegarde compressé régénérera le fichier de données.

Le HSQLDB interne ne se distingue pas de la variante externe. Si, comme dans la suite de la description, l'accès initial à la base de données se fait de l'extérieur, aucune fonction serveur n'est nécessaire. Vous avez juste besoin du programme d'archivage fourni avec LibreOffice. Vous le trouverez sur le chemin /program/classes/hsqldb.jar. L'utilisation de cette archive est la solution la plus sûre, car vous ne rencontrez alors aucun problème de version.

Le HSQLDB externe est disponible gratuitement pour téléchargement sur http://hsqldb.org/. Lorsque la base de données est installée, les étapes suivantes doivent être effectuées dans LibreOffice :

Si le pilote de base de données ne se trouve pas sur le chemin du Runtime Java, il doit être entré comme chemin de classe sous **Outils > Options > Avancé**.

La connexion à la base de données externe utilise JDBC. Le fichier de base de données doit être stocké dans un dossier particulier. Ce dossier peut être librement choisi. Dans l'exemple suivant, il se trouve dans le dossier de départ. Le reste du chemin du dossier et le nom de la base de données ne sont pas indiqués ici.

Il est important, si les données de la base de données doivent être écrites à l'aide de l'interface graphique, qu'à côté du nom de la base de données, les mots "; default_schema=true" soient écrits. Cela peut être étendu avec "; shutdown=true", de sorte que la base de données puisse être fermée par LibreOffice.

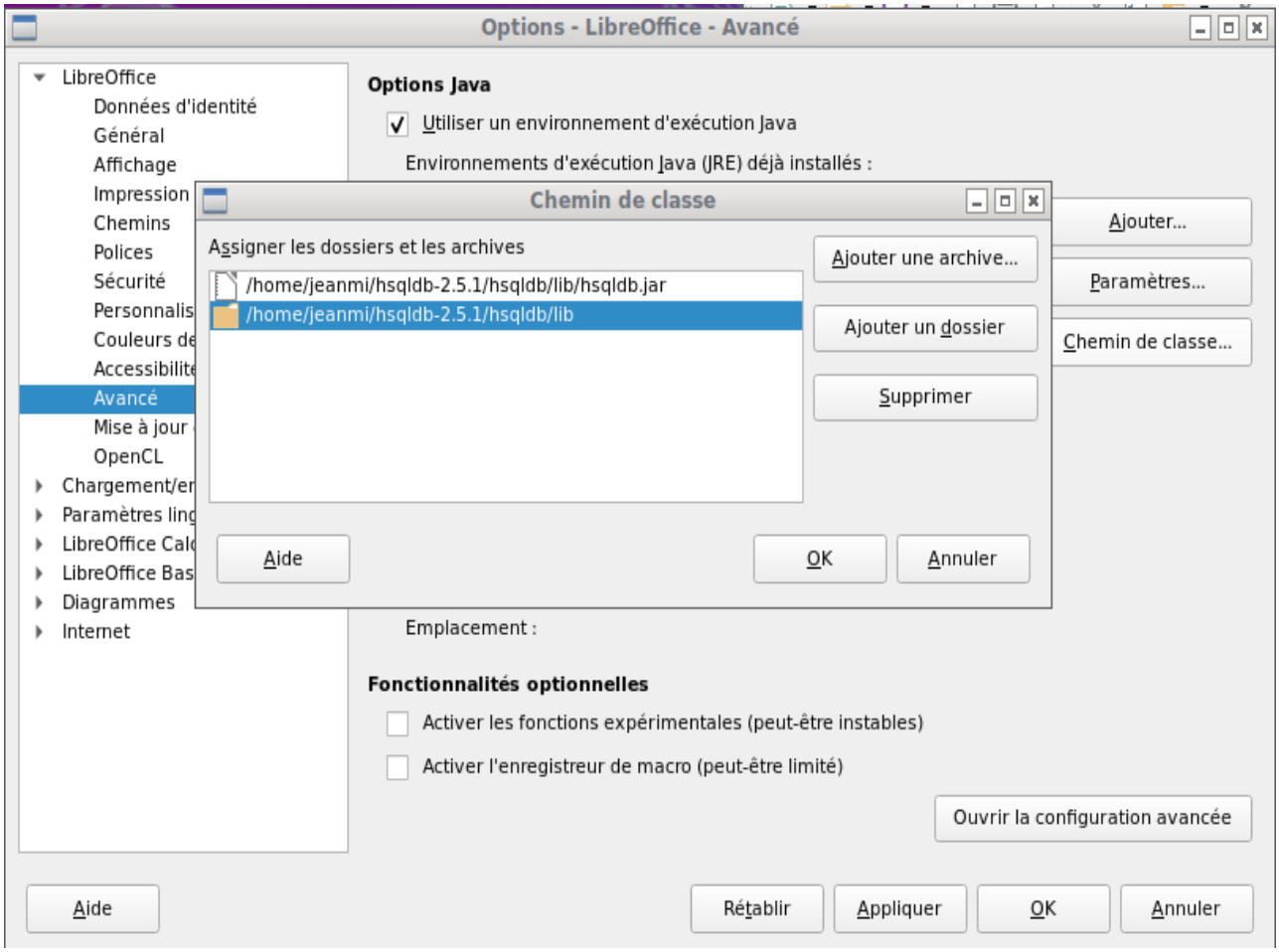


Figure 38: Définition du chemin de classe pour HSQLDB

Soit :

```
jdbc:hsqldb:file:/home/ChemindelaBase/NomdelaBase;
default_schema=true ; shutdown=true
```

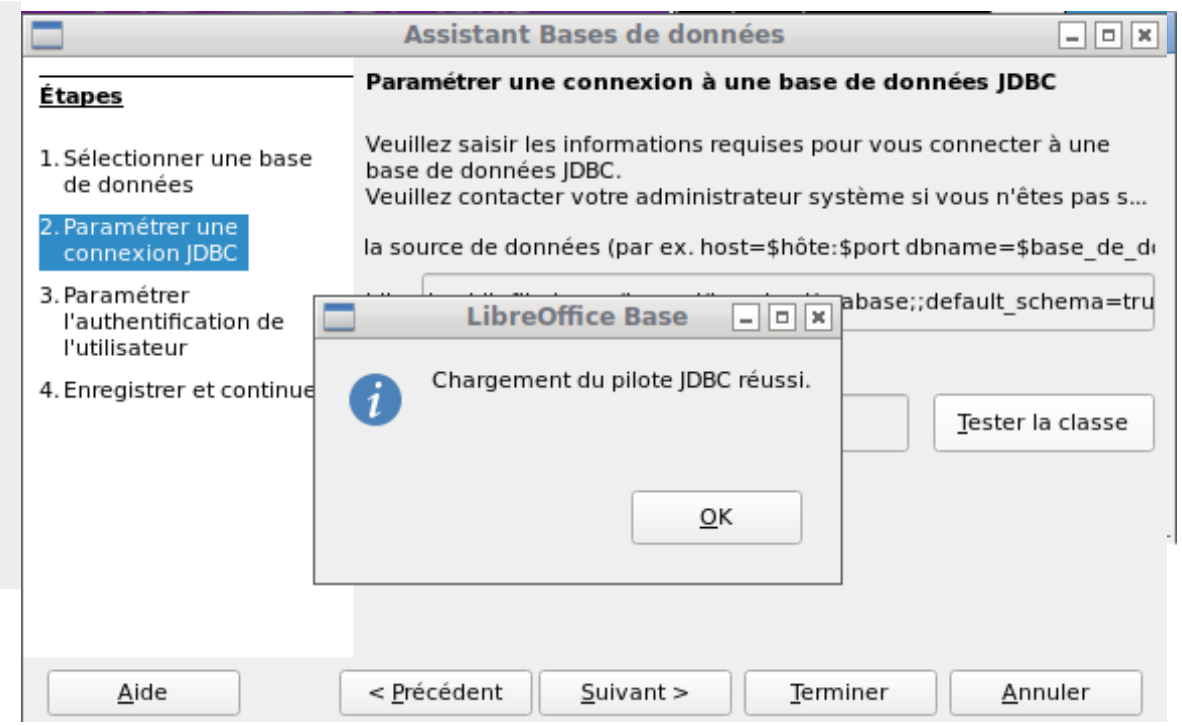
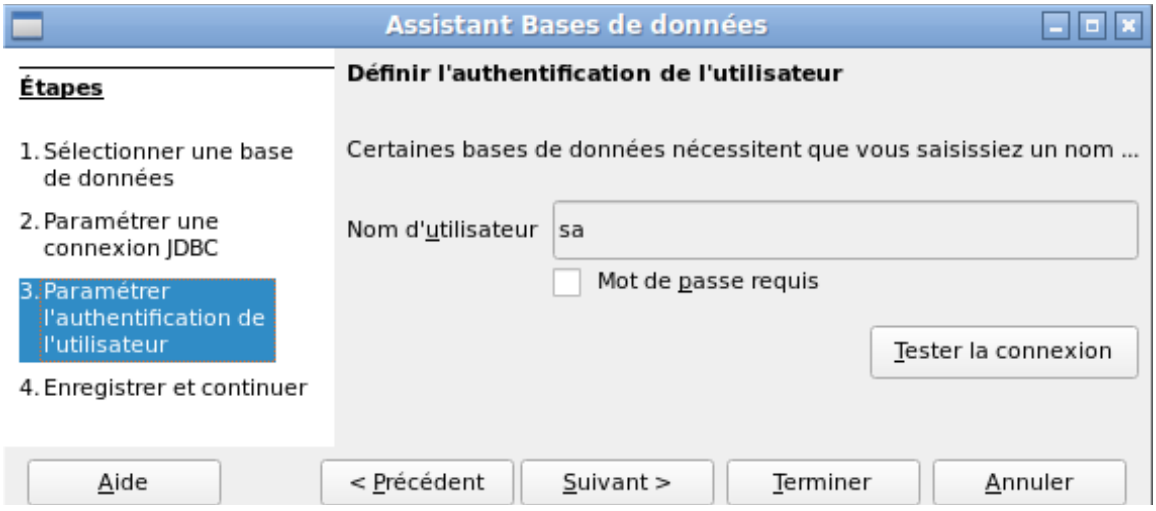


Figure 40: Test de la classe pour le pilote

L'étape suivante consiste à donner l'utilisateur par défaut, si rien dans la configuration HSQLDB ne doit être changé:
Cela crée la connexion et la base de données devient accessible.



Après l'enregistrement, dans le dossier, vous trouverez les fichiers :

```
NomDeLaBase.backup
NomDeLaBase.data
NomDeLaBase.properties
NomDeLaBase.script
NomDeLaBase.log
```

Attention

Si une base de données externe est éditée avec une version de HSQLDB 2.x, elle ne peut plus être convertie en base de données interne sous LibreOffice. Cela est dû à des fonctions supplémentaires qui ne sont pas présentes dans la version 1.8.x. Cela met fin à l'appel dans le cas de la version 1.8.x pendant la lecture du fichier script de la base de données.

De la même manière, une base de données externe qui a déjà été éditée avec une version de la deuxième série ne peut plus être éditée par la suite avec la version 1.8.x, qui est compatible avec LibreOffice.

Installation parallèle de bases de données HSQLDB internes et externes

L'inclusion du fichier externe hsqldb.jar dans le chemin de classe peut, dans certaines versions, empêcher l'ouverture de bases de données internes. La base reste bloquée, car les pilotes portent le même nom et essaient d'utiliser le pilote externe pour la base de données interne. Cela fonctionne la première fois. La deuxième fois, vous recevez un message indiquant que la base de données ne peut pas être ouverte, car elle a été écrite avec une version plus récente de HSQLDB.

Pour résoudre ce problème, ne placez pas le fichier hsqldb.jar, qui est requis pour les bases de données externes, sur le chemin de classe de LibreOffice. Au lieu de cela, le chemin d'accès aux classes de ce fichier doit être défini par une macro, comme illustré ci-dessous. Cette macro sera lancée à l'ouverture de la base.

```
SUB Demarre
       ' Sur UBUNTU, le pilote est en /usr/share/java/hsqldb/-2,5,1.jar
       Const sChemin = "/home/<utilisateur>/hsqldb-2,5,1/hsqldb/lib/hsqldb.jar"
       DIM oSourceDonnees AS OBJECT
       DIM oParametres AS OBJECT
       DIM sURL AS STRING
       sURL = ConvertToURL(sChemin)
       oSourceDonnees = ThisComponent.DataSource
       oParametres = oSourceDonnees.Settings
       oParametres.JavaDriverClassPath = sURL
END SUB
```

Ici, le fichier hsqldb.jar dans un système Linux se trouve sur le chemin indiqué ci-dessus. Ce chemin est passé à la base de données qui vient d'être ouverte en tant que fichier de pilote. Cette macro est appelée une seule fois après l'ouverture du fichier *.odb. Il écrit la connexion correspondante à la classe Java dans le fichier content.xml de l'archive *.odb :

```
<db: data-source-settings>
     <db: data-source-setting
          db: data-source-setting-is-list="false"
          db: data-source-setting-name="JavaDriverClass"
          db: data-source-setting-type="string">
     <db: data-source-setting-value>
          org.hsqldb.jdbcDriver
     </db: data-source-setting-value>
     </db: data-source-setting>
     <db: data-source-setting
          db: data-source-setting-is-list="false"
          db: data-source-setting-name="JavaDriverClassPath"
          db: data-source-setting-type="string">
          <db: data-source-setting-value>
```

```
          </db: data-source-setting-value>
      </db: data-source-setting>
</db: data-source-settings>
```

Enfin, le chemin pourrait être écrit directement dans content.xml sans utiliser de macro. Cependant, cette méthode n'est pas très confortable pour un utilisateur ordinaire.

La seule autre méthode connue nécessite l'installation de deux versions de LibreOffice en parallèle et la modification du fichier bootstrap pour l'une d'entre elles. Cette méthode n'est pas non plus très confortable pour un utilisateur ordinaire.

Modification de la connexion vers une base de données HSQLDB externe

Les bases de données internes HSQL ont l'inconvénient que le stockage des données implique une archive compressée. Ce n'est que lors de la compression que toutes les données sont finalement écrites. Cela peut plus facilement entraîner une perte de données que lorsque vous travaillez avec une base de données externe. La section suivante montre les étapes nécessaires pour réussir à changer une base de données existante d'une archive *.odb à une version externe dans HSQL.

À partir d'une copie de la base de données existante, extrayez le dossier de la base de données. Copiez le contenu dans un dossier arbitraire comme décrit ci-dessus. Préfixez le nom de la base de données aux noms de fichiers résultants :

```
NomDeLaBase.backup
NomDeLaBase.data
NomDeLaBase.properties
NomDeLaBase.script
NomDeLaBase.log
```

Le fichier content.xml doit maintenant être extrait de l'archive *.odb. Utilisez n'importe quel éditeur de texte simple pour trouver les lignes suivantes :

```
<db: connection-data><db: connection-resource xlink: href="sdbc: embedded:
hsqldb"/><db: login db: is-password-required="false"/></db: connection-
data><db: driver-settings/>
```

Ces lignes doivent être remplacées par une connexion à une base de données externe, dans ce cas une connexion à une base de données avec le nom mabase, dans le dossier hsqldb_data.

```
<db: connection-data><db: connection-resource xlink: href="jdbc: hsqldb:
file:/home/jeanmi/basehsql/mabase ; default_schema=true"/><db: login db:
user-name="sa" db: is-password-required="false"/></db: connection-
data><db: driver-settings db: java-driver-class="org.hsqldb.jdbcDriver"/>
```

Si, comme décrit ci-dessus, la configuration de base de HSQLDB n'a pas été endommagée, le nom d'utilisateur et le mot de passe facultatif doivent également correspondre.

Après avoir modifié le code, content.xml doit être remis dans l'archive *.odb. Le dossier de base de données dans l'archive est maintenant excédentaire par rapport aux besoins. Les données seront à l'avenir accessibles via la base de données externe.

Modification de la connexion à la base de données pour l'accès multi-utilisateur

Pour un accès multi-utilisateur, HSQLDB doit être mis à disposition sur un serveur. La manière dont l'installation du serveur est effectuée varie en fonction de votre système d'exploitation. Pour OpenSuSE, il suffit de télécharger le package approprié et de démarrer le serveur de manière centralisée à l'aide de YAST (paramètre de niveau d'exécution). Les utilisateurs d'autres systèmes

d’exploitation et d’autres distributions Linux peuvent probablement trouver des conseils appropriés sur Internet.

Le dossier de départ sur le serveur (dans SuSE, /var/lib/hsqldb) contient, entre autres, un dossier appelé data, dans lequel la base de données doit être archivée, et un fichier appelé server.properties, qui contrôle l’accès aux bases de données de ce dossier.

Les lignes suivantes reproduisent le contenu complet de ce fichier sur un exemple d’ordinateur. Il contrôle l’accès à deux bases de données, à savoir la base de données par défaut d’origine (qui peut être utilisée comme nouvelle base de données) et la base de données extraite du fichier *.odb.

```
# Hsqldb Server cfg file.
# See the Advanced Topics chapter of the Hsqldb User Guide.

server.database.0 file: data/db0
server.dbname.0 firstdb
server.urlid.0 db0-url

server.database.1 file: data/mabase
server.dbname.1 mabase
server.urlid.1 mabase-url

server.silent true
server.trace false

server.port 9001
server.no_system_exit true
```

la `database.0` est adressée avec le nom `firstdb`, bien que les fichiers individuels dans le dossier de données commencent par db0. Une autre base de données a été ajoutée en tant que `database.1`. Ici, le nom de la base de données et celui du fichier commencent de la même manière.

Les deux bases de données sont adressées de la manière suivante :

```
jdbc: hsqldb: hsql://localhost/firstdb ; default_schema=true
username sa
password
jdbc: hsqldb: hsql://localhost/mabase ; default_schema=true
username sa
password
```

Le suffixe `default_schema=true` dans l’URL, qui est nécessaire pour l’accès en écriture à l’aide de l’interface graphique de base, est inclus en permanence

Si vous avez réellement besoin de travailler sur le serveur, vous voudrez vous demander si la base de données doit être protégée par mot de passe pour des raisons de sécurité.

Vous pouvez maintenant vous connecter au serveur en utilisant LibreOffice. Avec ces données d’accès, le serveur peut être chargé sur son propre ordinateur.

Sur un réseau avec d’autres ordinateurs, vous devez donner le nom d’hôte ou l’adresse IP du serveur.

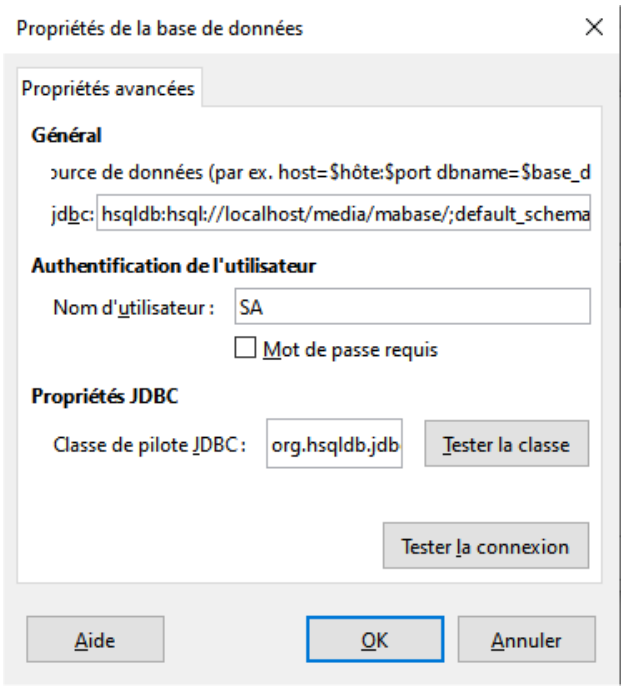
Exemple : Un ordinateur a l’IP 192.168.0.20 et est connu sur le réseau sous le nom **lin_serv**. Supposons maintenant qu’il y ait un autre ordinateur à déclarer pour la connexion à la base de données, il faut, dans ce cas, saisir :

```
jdbc: hsqldb: hsql://192.168.0.20/mabase ; default_schema=true
```

ou :

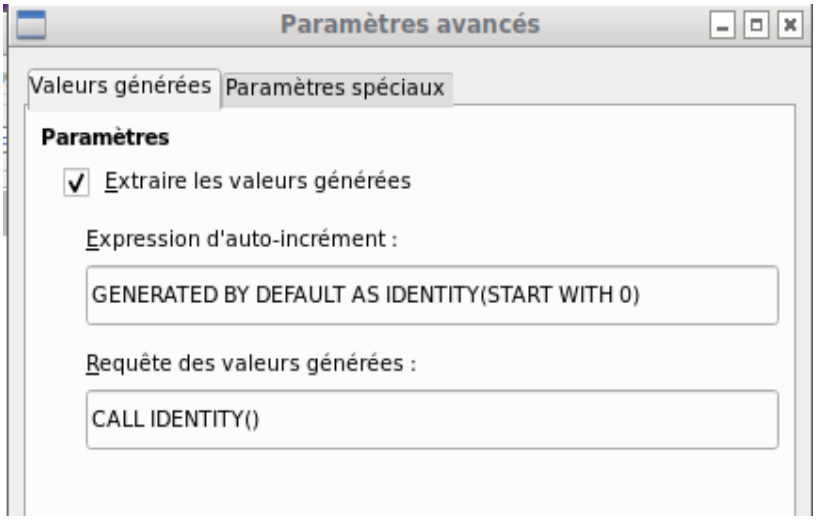
```
jdbc: hsqldb: hsql://lin_serv/mabase ; default_schema=true
```

La base de données est maintenant connectée et nous pouvons y écrire. Rapidement, cependant, un problème supplémentaire apparaît. Les valeurs précédemment générées automatiquement ne sont soudainement plus incrémentées. Pour cela, nous avons besoin d'un paramètre supplémentaire.



Incrémentation automatique des valeurs avec HSQLDB externe

Pour utiliser les valeurs automatiques, différentes procédures de configuration de table sont nécessaires selon la version de LibreOffice. L'entrée suivante sous **Édition > Base de données > Paramètres avancés** est commune à tous :



L'ajout de `GENERATED BY DEFAULT AS IDENTITY (START WITH 0)` entraîne la définition de la fonction d'incrémentation automatique des valeurs de la clé primaire. L'interface graphique de LibreOffice reprend cette commande, mais malheureusement préfixe l'instruction avec NOT NULL, de sorte que la séquence de commandes pour HSQLDB n'est pas lisible. Ici, vous devez vous assurer que HSQLDB reçoit la commande ci-dessus afin que le champ correspondant contienne la clé primaire.

Dans toutes les versions de LO, la lecture de la dernière valeur et l'incrémentation à la suivante utilise la commande `CALL IDENTITY ()`. Cela vous permet de créer un fichier mini – *. Odb, de le tester minutieusement, puis de supprimer simplement les tables.

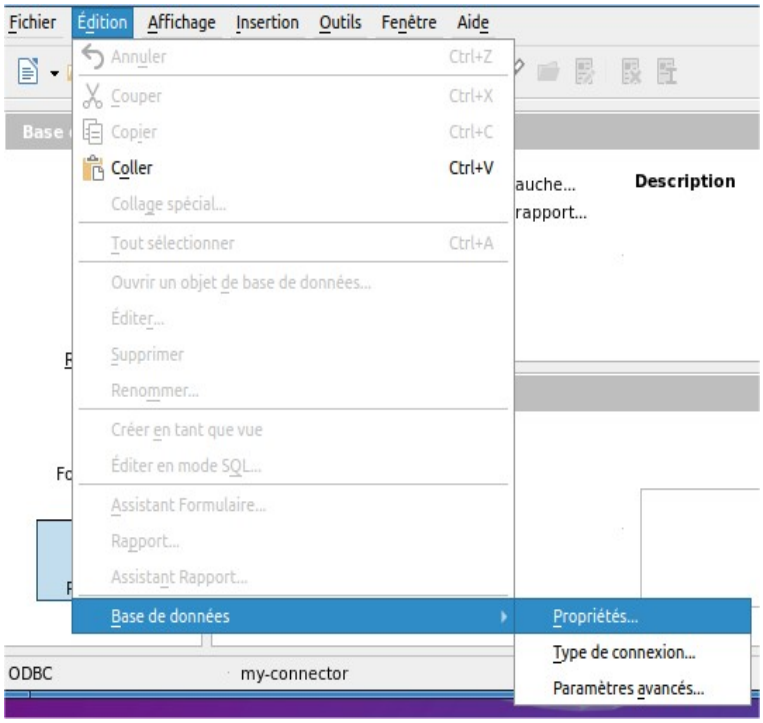
L'ensemble des requêtes, formulaires et rapports seront toujours utilisables, car la base de données du fichier * .odb est accessible de la même manière et les commandes SQL spécifiques peuvent être utilisées pour la base de données externe (réelle) HSQLDB.

Modification ultérieure des propriétés de connexion

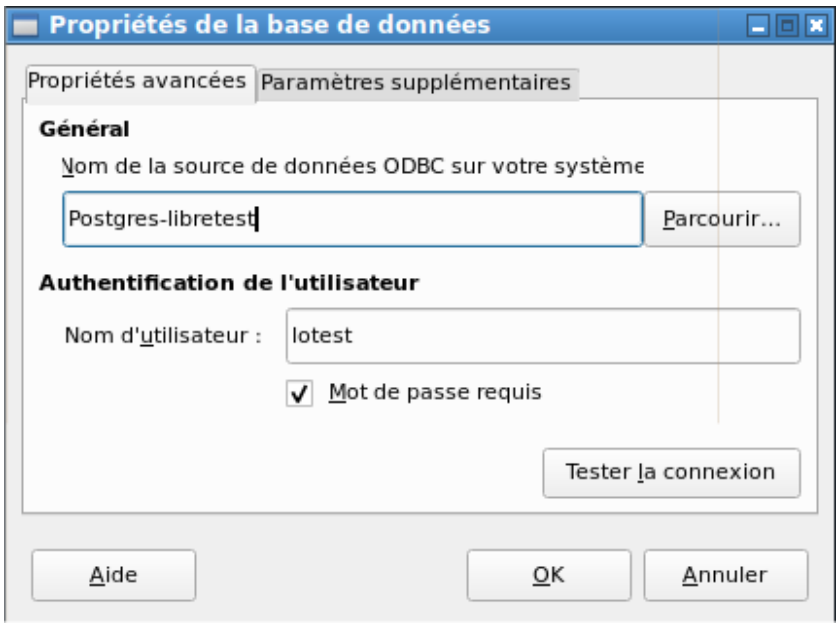
Surtout avec les connexions à des bases de données externes, une connexion peut ne pas fonctionner comme vous le souhaitez. Le jeu de caractères peut ne pas être correct, ou les sous-formulaires peuvent ne pas fonctionner sans erreurs, ou quelque chose dans les paramètres sous-jacents doit être modifié.

Les captures d'écran suivantes illustrent comment vous pouvez modifier les paramètres de connexion d'une base de données PostgreSQL externe.

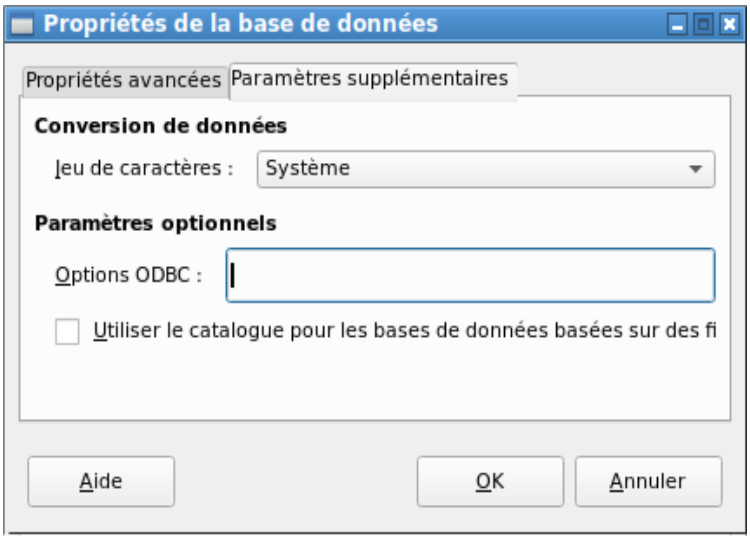
Sous **Édition > Base de données**, vous trouverez les choix Propriétés, Type de connexion et Paramètres avancés. Choisissez **Propriétés**.



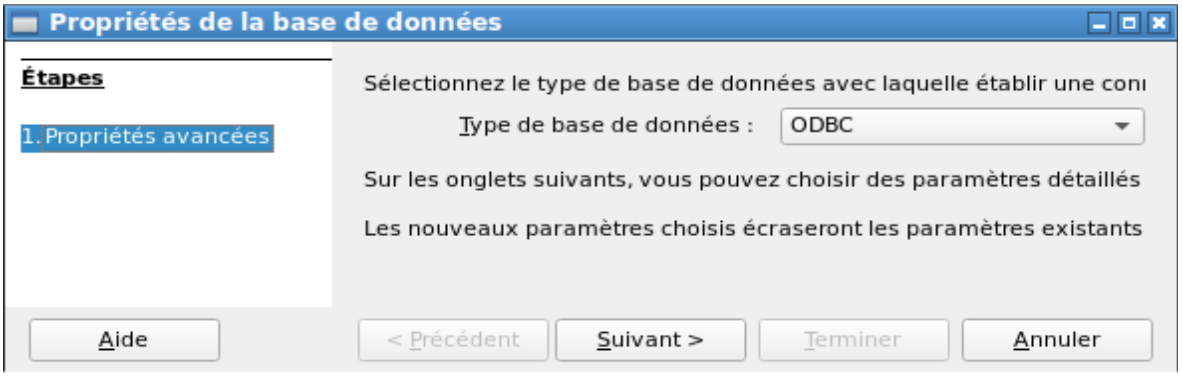
Si le nom de la source de données a changé, il peut être modifié ici. Pour une connexion ODBC, le nom par lequel la base de données est appelée est indiqué dans le fichier odbc.ini. Le nom n'est généralement pas tout à fait le même que le nom réel de la base de données dans PostgreSQL.



Y a-t-il un problème avec le jeu de caractères ? Ces problèmes peuvent être résolus à l'aide du deuxième onglet de la boîte de dialogue.

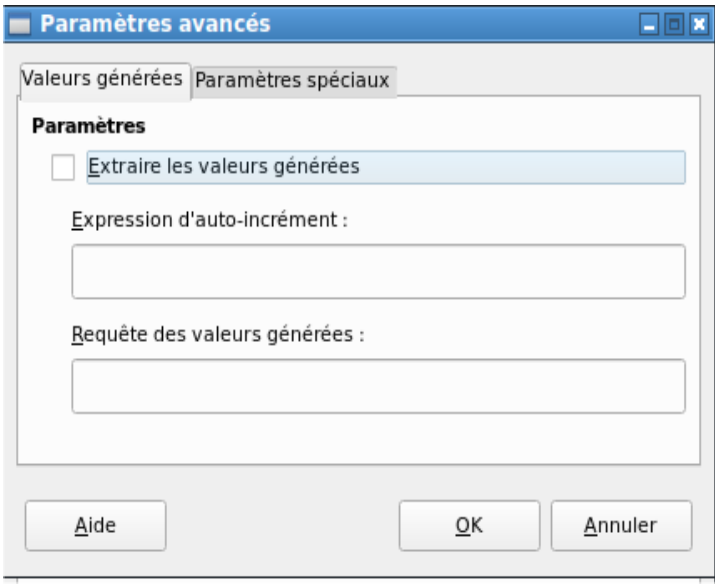


Une configuration spéciale supplémentaire du pilote est possible si un paramètre doit être implémenté qui n'est pas actuellement dans le fichier `odbc.ini`.

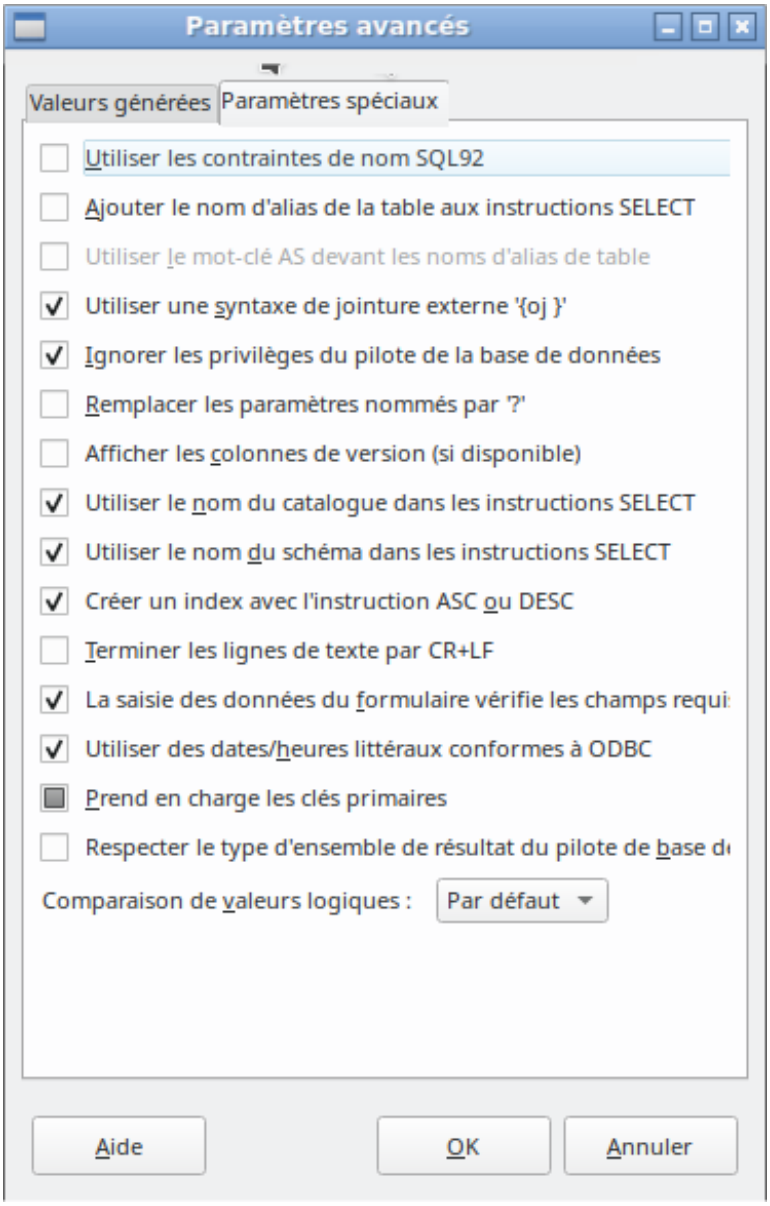


Si le type de connexion est sélectionné, tout le contact avec la source de données peut être modifié. Les étapes suivantes sont similaires à celles de l'Assistant Base de données à partir de

l’étape 2. Par exemple, vous pouvez passer d’ODBC à JDBC ou une connexion directe avec le pilote interne de LibreOffice. Ceci est utile si vous effectuez des tests préliminaires pour déterminer la méthode de connexion la plus appropriée pour un projet.



Selon le système de base de données, il existe différentes commandes pour créer des valeurs incrémentées automatiquement. Si vous avez besoin de faire quelque chose de ce genre qui n’est actuellement pas possible avec ce pilote, vous devrez le faire à la main. Cela nécessitera une commande pour créer un champ AutoChamp et également une pour interroger la valeur la plus récente.



Les paramètres spéciaux accessibles via **Edition > Base de données > Paramètres avancés** affectent l'interaction des bases de données externes avec Base de différentes manières. Certains paramètres sont grisés car ils ne peuvent pas être modifiés dans la base de données sous-jacente. Dans l'exemple ci-dessus *Remplacez les paramètres nommés par ?* a été coché. Il a été montré qu'autrement, la transmission de valeurs d'un formulaire principal à un sous-formulaire dans PostgreSQL ne fonctionne pas. Ce n'est qu'avec ce paramètre que la construction de formulaire du chapitre 4, Formulaires, fonctionne correctement.

LibreOffice

Guide Base

Chapitre 3
Tables

Informations générales sur les tables

Les bases de données stockent les données dans des tables. La principale différence entre les tables d'une base de données et les plages de cellules dans un tableur simple est que les champs dans lesquels les données sont écrites doivent être clairement définis au préalable. Par exemple, une base de données ne permet pas à un champ de texte de contenir des nombres à utiliser dans les calculs. Ces nombres sont affichés, mais uniquement sous forme de chaînes, dont la valeur numérique réelle est zéro. De même, les images ne peuvent pas être incluses dans tous les types de champs.

Les détails sur les types de données disponibles peuvent être obtenus à partir de la fenêtre Ébauche de table dans Base. Ils sont présentés dans l'annexe A de ce manuel.

Les bases de données simples sont basées sur une seule table. Tous les éléments de données sont saisis indépendamment, ce qui peut entraîner une saisie multiple des mêmes données. Un simple carnet d'adresses à usage privé peut être créé de cette manière. Cependant, le carnet d'adresses d'une école ou d'une association sportive pourrait contenir tellement de répétitions de codes postaux et d'emplacements que ces champs sont mieux placés dans une ou même deux tables séparées.

Le stockage des données dans des tables séparées permet de :

- Réduire la saisie répétée du même contenu
- Éviter les erreurs d'orthographe dues à des saisies répétées
- Améliorer le filtrage des données dans les tables affichées

Lors de la création d'une table, vous devez toujours considérer si plusieurs répétitions, en particulier de texte ou d'images (qui consomment beaucoup de stockage), peuvent se produire dans la table. Si tel est le cas, vous devez les exporter dans une autre table. Comment faire cela en principe est décrit dans le chapitre 1, Introduction à Base, dans la section "Une base de données simple – Exemple de test en détail, page 13".

Note

Une base de données relationnelle est un groupe de tables liées les unes aux autres via des attributs communs. Le but d'une base de données relationnelle est d'éviter autant que possible la saisie en double d'éléments de données. Les redondances sont à éviter.

Ceci peut être réalisé en :

Séparant le contenu en autant de champs uniques que possible (par exemple, au lieu d'utiliser un champ pour une adresse complète, utilisez des champs séparés pour le numéro de rue, la rue, la ville et le code postal).

Empêchant les données en double pour un champ dans plusieurs enregistrements (par exemple en important le code postal et la ville d'une autre table).

Ces procédures sont connues sous le nom de **Normalisation des bases de données.**

Relations entre les tables

Ce chapitre explique plusieurs de ces étapes en détail, en utilisant un exemple de base de données pour une bibliothèque : `Media_sans_Macros`. Construire les tables pour cette base de données est un travail considérable, car il couvre non seulement l'ajout d'éléments dans une médiathèque, mais également leur prêt ultérieur.

Relations pour les tables dans les bases de données

Les tables de la base de données interne HSQLDB ont toujours un champ distinctif et unique, la clé primaire. Ce champ doit être défini avant que des données puissent être écrites dans la table. En utilisant ce champ, des enregistrements spécifiques dans une table peuvent être trouvés.

Dans certains cas, une clé primaire est formée de plusieurs champs en combinaison. Ces champs doivent être uniques lorsqu'ils sont considérés ensemble. C'est ce qu'on appelle une *clé primaire composite*.

Note

La combinaison des champs dans une clé primaire composite est unique lorsque chaque enregistrement de la table contient une combinaison unique de valeurs pour ces champs.

La table 2 peut avoir un champ qui pointe un enregistrement dans la table 1. Ici, la clé primaire de la table 1 est écrite sous forme de valeur dans le champ de la table 2. La table 2 comporte désormais un champ qui pointe vers le champ de clé d'une autre table, appelé clé étrangère. Cette clé étrangère existe dans la table 2 en plus de sa clé primaire.

Plus il y a de relations entre les tables, plus la tâche de conception est complexe. La figure 41 montre la structure de table globale de cet exemple de base de données, mise à l'échelle pour s'adapter à la taille de page de ce document. Pour lire le contenu, agrandissez la page à 200 %.

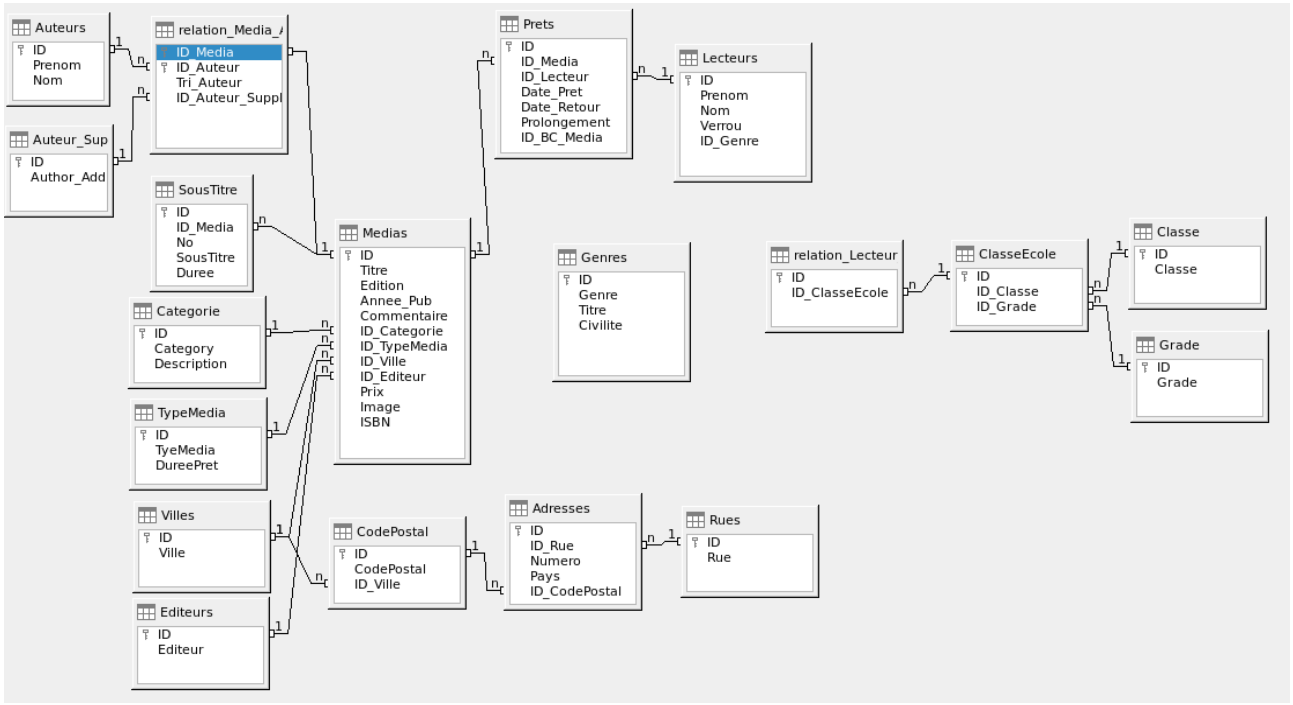


Figure 41: Diagramme de relations pour l'exemple de base de données Media_sans_Macros.

Relations un-à-plusieurs

La base de données `Media_sans_Macros` répertorie les titres des médias dans une table. Étant donné que les titres peuvent avoir plusieurs sous-titres ou parfois aucun, les sous-titres sont stockés dans une table séparée.

Cette relation est connue sous le nom de *un-à-plusieurs* (1: n). De nombreux sous-titres peuvent être attribués à un support, par exemple les nombreux titres de pistes d'un CD musical. La clé primaire de la table Medias est stockée en tant que clé étrangère dans la table SousTitre. La plupart des relations entre les tables d'une base de données sont des relations un-à-plusieurs.

Relations plusieurs-à-plusieurs

Une base de données pour une bibliothèque peut contenir une table pour les noms des auteurs et une table pour les médias. Le lien entre un auteur et, par exemple, des livres qu'il a écrits, est évident. La bibliothèque peut contenir plus d'un livre d'un auteur. Il peut également contenir des livres avec plusieurs auteurs. Cette relation est connue sous le nom de *plusieurs à plusieurs* (n : m). De telles relations nécessitent une table qui joue le rôle d'intermédiaire entre les deux tables concernées. Ceci est représenté sur la figure 42 par la table relation_Media_Auteur.

Ainsi, en pratique, la relation n: m est résolue en la traitant comme deux relations 1: n. Dans la table intermédiaire, le ID_ Media peut apparaître plusieurs fois, tout comme l'ID_Auteur. Mais lorsque vous les utilisez par paire, il n'y a pas de duplication : il n'y a pas deux paires identiques. Donc, cette paire répond aux exigences de la clé primaire pour le tableau intermédiaire.

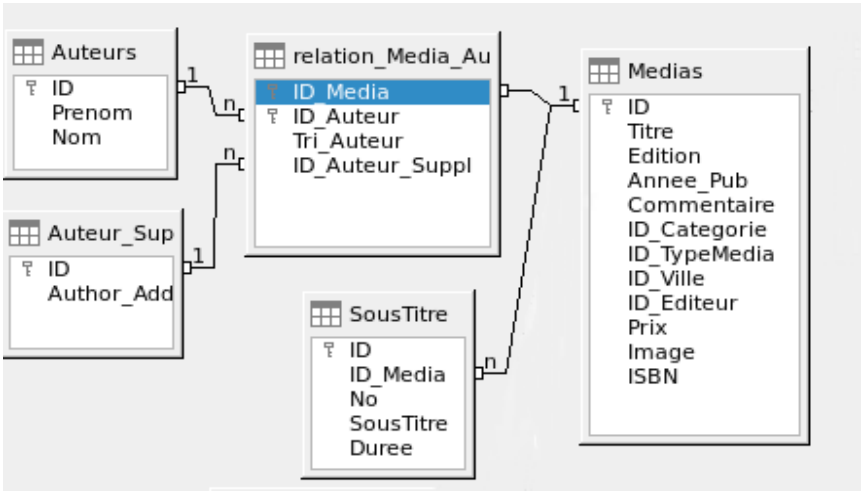


Figure 42: Exemple relation 1: n ; relation n : m

Note

Pour une valeur donnée de l'ID_Media, il n'y a qu'un seul titre du média et un ISBN. Pour une valeur donnée de ID_Auteur, il n'y a qu'un seul prénom et nom d'auteur. Ainsi, pour une paire donnée de ces valeurs, il n'y a qu'un seul ISBN et un seul auteur. Cela rend la paire unique.

Relations individuelles (un-à-un)

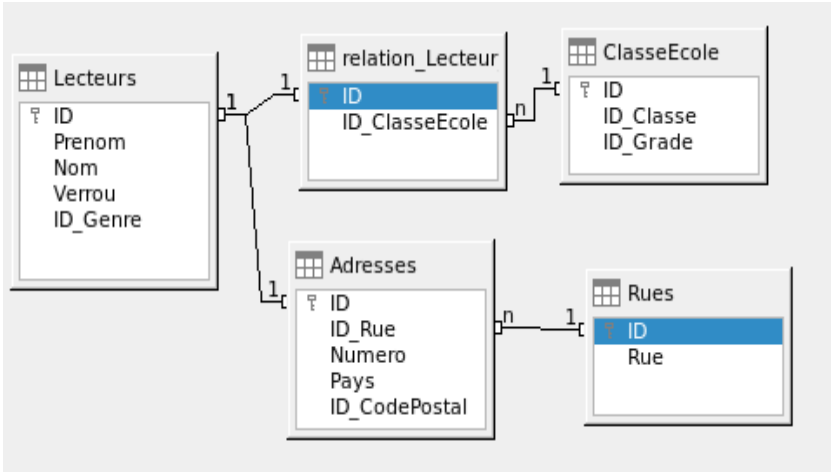


Figure 43: Exemple relation 1:1

La base de données de la bibliothèque décrite ci-dessus nécessite une table pour les lecteurs. Dans ce tableau, seuls les champs directement nécessaires ont été planifiés à l'avance. Mais pour une base de données scolaire (Cf. Figure 43) , la classe de l'école est également requise. À partir des registres de classe de l'école, vous pouvez trouver les adresses des emprunteurs si nécessaire. Par conséquent, il n'est pas nécessaire d'inclure ces adresses dans la base de données. La relation école-classe des élèves est séparée de la table des lecteurs, car la correspondance avec les classes n'est pas appropriée dans tous les domaines. De cela découle une relation 1:1 entre le lecteur et l'affectation de classe individuelle.

Dans une base de données pour une bibliothèque publique, les adresses des lecteurs sont requises. Pour chaque lecteur, il y a une seule adresse. S'il y a plusieurs lecteurs à la même

adresse, cette structure nécessiterait que l'adresse soit à nouveau saisie, car la clé primaire de la table Lecteurs est entrée directement comme clé primaire dans la table Adresses. La clé primaire et la clé étrangère sont identiques dans la table d'adresses. Il s'agit donc d'une relation 1:1.

Une relation 1:1 ne signifie pas que pour chaque enregistrement d'une table, il y aura un enregistrement correspondant dans une autre table. Mais, il n'y aura **au plus** qu'un seul enregistrement correspondant. Une relation 1:1 conduit donc à l'exportation de champs qui ne seront remplis de contenu que pour certains des enregistrements.

Tables et relations pour l'exemple de base de données

L'exemple de base de données (Media_sans_macros) doit répondre à trois exigences : ajouts et suppressions de médias, prêts et administration des utilisateurs.

Tableau d'ajout de médias

Tout d'abord, les médias doivent être ajoutés à la base de données pour qu'une bibliothèque puisse les utiliser. Cependant, pour un simple résumé d'une collection de médias à la maison, vous pouvez créer des bases de données plus faciles avec l'assistant ; cela pourrait être suffisant pour un usage domestique.

La table centrale pour l'ajout de médias est la table des médias (voir Figure 44).

Dans cette table, tous les champs qui sont directement saisis sont supposés ne pas être également utilisés pour d'autres médias avec le même contenu. La duplication doit donc être évitée.

Pour cette raison, les champs prévus dans la table incluent le titre, l'ISBN, une image de la couverture et l'année de publication. La liste des champs peut être étendue si nécessaire. Par exemple, les bibliothécaires peuvent vouloir inclure des champs pour la taille (nombre de pages), le titre de la série, etc.

La table des sous-titres contient le contenu détaillé des CD. Comme un CD peut contenir plusieurs morceaux de musique, un enregistrement des morceaux individuels dans le tableau principal nécessiterait beaucoup de champs supplémentaires (sous-titre 1, sous-titre 2, etc.) ou le même élément devrait être saisi plusieurs fois. La table des sous-titres se trouve donc dans une relation n:1 avec la table des Medias.

Les champs de la table des sous-titres sont (en plus du sous-titre lui-même) le numéro de séquence du sous-titre et la durée de la piste. Le champ Duree doit d'abord être défini comme un champ d'heure. De cette manière, la durée totale du CD peut être calculée et affichée dans un résumé si nécessaire.

Les auteurs ont une relation n:m avec les médias. Un élément peut avoir plusieurs auteurs et un auteur peut avoir créé plusieurs éléments. Cette relation est contrôlée par la table **relation_Media_Auteur**. La clé primaire de cette table de liaison est la clé étrangère, formée à partir des tables Auteur et Media. La table **relation_Media_Auteur** inclut un tri supplémentaire (Tri_Auteur) des auteurs, par exemple selon l'ordre dans lequel ils sont nommés dans le livre. De plus, une étiquette supplémentaire telle que Producteur, Photographe, etc. est ajoutée à l'auteur si nécessaire.

Categorie, TypeMedia, Ville, et Editeur ont une relation a 1:n.

Pour le champ *Categorie*, une petite bibliothèque peut utiliser quelque chose comme Art ou Biologie. Pour les bibliothèques plus importantes, des systèmes généraux pour les bibliothèques sont disponibles. Ces systèmes fournissent à la fois des abréviations et des descriptions complètes. Par conséquent, les deux champs apparaissent sous Categorie.

Le *TypeMedia* est lié à la durée du prêt DureePret. Par exemple, les DVD vidéo peuvent en principe avoir une durée de prêt de 7 jours, mais les livres peuvent être prêtés pour 21 jours. Si la durée du prêt est liée à d'autres critères, il y aura des changements correspondants dans votre méthodologie.

La table *Villes* sert non seulement à stocker les données de localisation des médias, mais également à stocker les emplacements utilisés dans les adresses des utilisateurs.

Étant donné que les *Editeurs* reviennent fréquemment, une table séparée leur est affectée.

La table Media, en sus de sa clé primaire (ID) qui est utilisée comme clé étrangère dans les tables relation_Media_Auteur et SousTitre, possède quatre clés étrangères qui la relie aux tables Categorie, TypeMédia, Ville, et Editeur, comme indiqué dans la Figure 44.

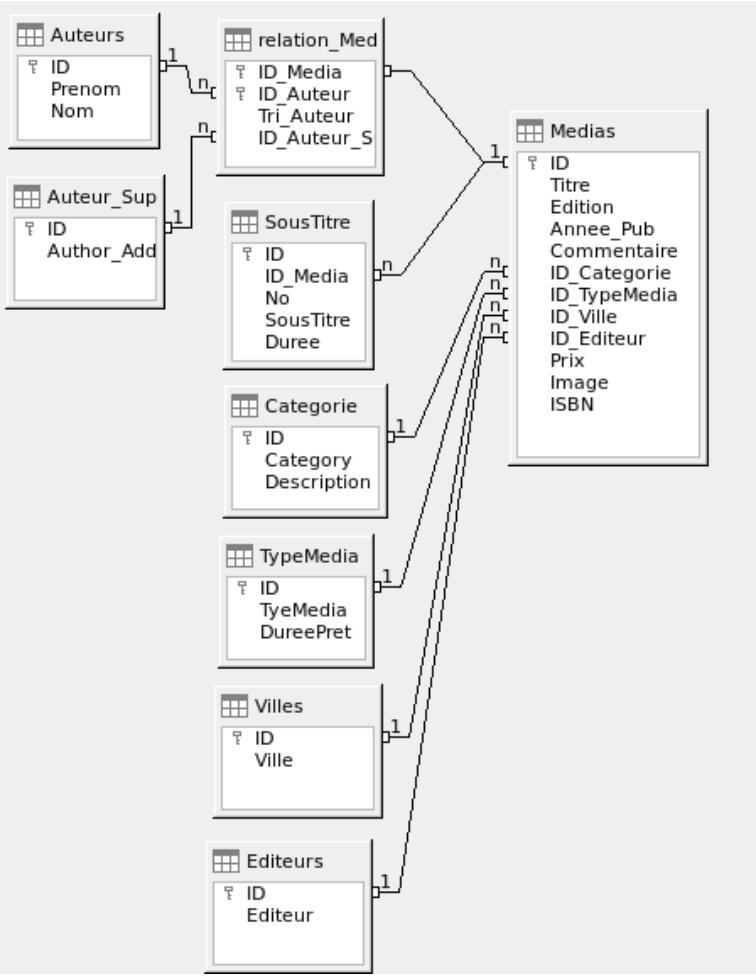


Figure 44: Ajout de médias

Table Prêt

La table centrale est *Pret* (voir Figure 45). C'est le lien entre les tables Medias et Lecteurs.

Lorsqu'un support est retourné, la plupart de ses données peuvent être supprimées, car elles ne sont plus nécessaires. Mais deux des champs ne doivent pas l'être : ID et Date_Pret. Le premier est la clé primaire. Le second est créé lorsque l'article a été prêté. Cela sert à deux fins. Premièrement, il est utile de déterminer les médias les plus populaires. Deuxièmement, si un objet est endommagé lors de son retrait, ce champ permettra de déterminer qui a été la dernière personne à l'emprunter. De plus, la Date_Retour est enregistrée lorsque l'article est retourné.

De même, les rappels sont intégrés dans la procédure de prêt. Chaque rappel est saisi séparément dans le tableau des rappels afin que le nombre total de rappels puisse être déterminé.

En plus d'une période de prolongation en semaines, il y a un champ supplémentaire dans l'enregistrement de prêt qui permet de prêter des supports à l'aide d'un lecteur de codes-barres (ID_BC_Media). Les codes-barres contiennent, en plus du ID_Media individuel, un chiffre de contrôle que le scanner peut utiliser pour déterminer si la valeur numérisée est correcte. Ce champ de code-barres est inclus ici uniquement à des fins de test. Il serait préférable que la clé primaire de la table Media puisse être saisie directement sous forme de code à barres, ou si une macro était utilisée pour supprimer le chiffre de contrôle du numéro de code à barres, entré avant le stockage.

Enfin, nous devons connecter le Lecteur au prêt. Dans la table de lecture réelle, seuls le nom, un verrou facultatif et une clé étrangère reliant la table Genres sont inclus dans le plan.

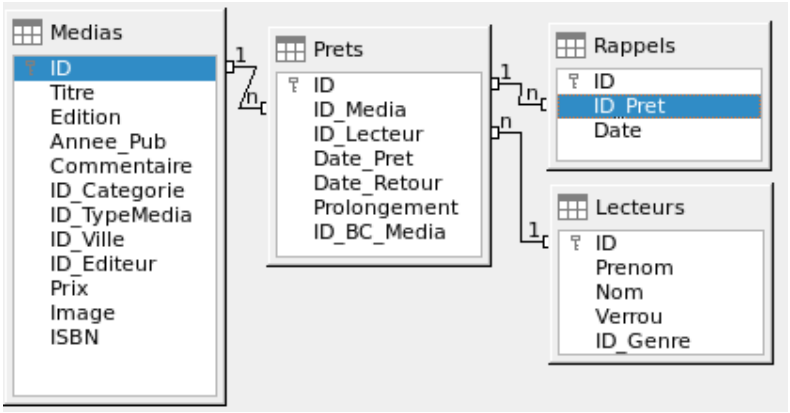


Figure 45: Prêts

Table d'administration des utilisateurs

Pour cette conception de table, deux scénarios sont envisagés. La chaîne de tables illustrée à la figure 46 est conçue pour les bibliothèques scolaires. Ici, il n'y a pas besoin d'adresses, car les élèves peuvent être contactés via l'école. Les rappels n'ont pas besoin d'être envoyés par la poste mais peuvent être distribués en interne.

La chaîne d'adresses est nécessaire dans le cas des bibliothèques publiques. Ici, vous devez saisir les données nécessaires à la création de lettres de rappel. Voir la figure 46.

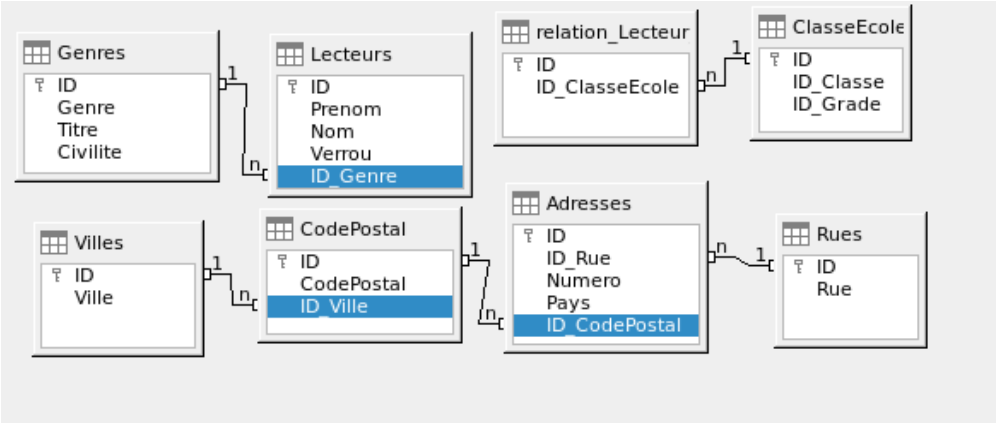


Figure 46: Lecteurs – une chaîne de classe scolaire et une chaîne d'adresses

La table *Genres* garantit que la civilité correcte est utilisée dans les rappels. La rédaction des rappels peut alors être automatisée dans la mesure du possible. De plus, certains prénoms peuvent être aussi bien masculins que féminins. Par conséquent, la liste séparée du sexe est requise même lorsque les rappels sont écrits à la main. La table *relation_Lecteur_ClasseEcole*, comme la table *Adresses*, a une relation 1:1 avec la table *Lecteurs*. Cela a été choisi parce que la classe de l'école ou l'adresse peuvent être requises. Sinon, le *ID_ClasseEcole* pourrait être placé directement dans la table Eleves ; il en serait de même pour le contenu complet de la table d'adresses dans un système de bibliothèque publique.

Une *classe d'école* se compose généralement d'une désignation d'année et d'un suffixe de cohorte. Dans une école à 4 cohortes, ce suffixe peut aller de *a* à *d*. Le suffixe est entré dans le tableau des classes. L'année est dans un tableau de notes séparé. De cette façon, si les lecteurs montent d'une classe à la fin de chaque année scolaire, vous pouvez simplement modifier l'entrée de l'année pour tout le monde.

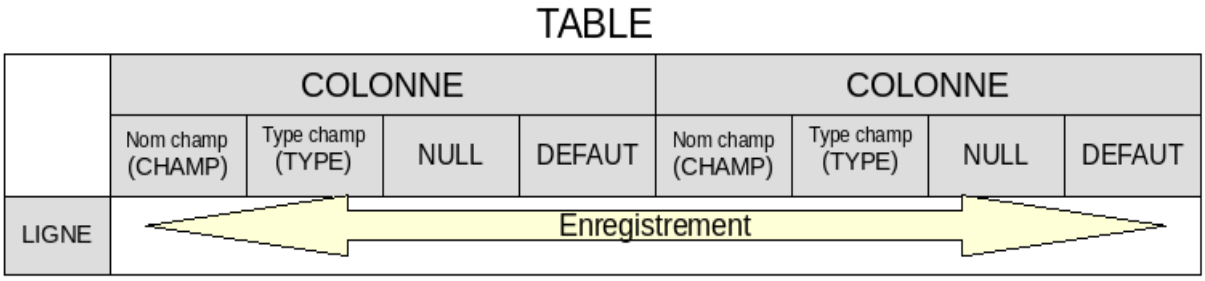
L'adresse est également divisée. La *Rue* est stockée séparément, car les noms de rue dans une zone sont souvent répétés. Le code postal et la ville sont séparés, car il y a souvent plusieurs codes postaux pour une même zone et donc plus de codes postaux que de villes. Ainsi, par rapport à la table Adresses, la table CodePostal contient beaucoup moins d'enregistrements et la table Villes encore moins.

L'utilisation de cette structure de table est expliquée plus en détail dans le chapitre 4, Formulaires, de ce manuel.

Créer des tables

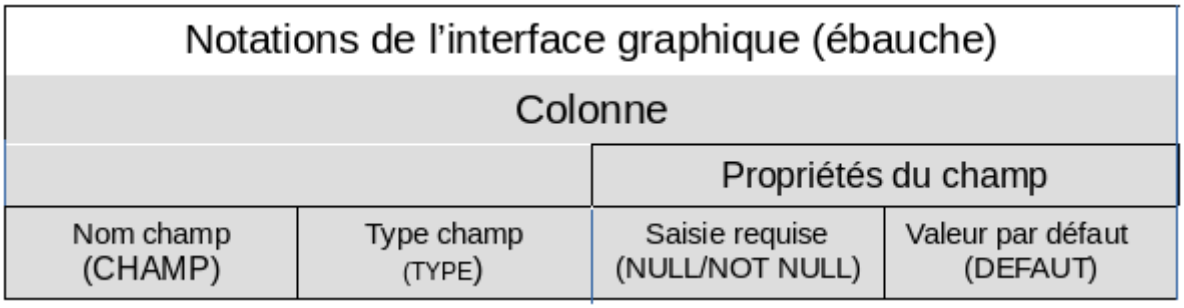
La plupart des utilisateurs de LibreOffice utiliseront généralement l'interface utilisateur graphique (GUI) exclusivement pour créer des tables. La saisie directe de commandes SQL devient nécessaire lorsque, par exemple, un champ doit être inséré ultérieurement à une position particulière, ou une valeur standard doit être définie après l'enregistrement de la table.

Terminologie des tables : L'image ci-dessous montre la division standard des tables en colonnes et en lignes.



Chaque enregistrement de données est stocké dans sa propre ligne du tableau. Les colonnes individuelles sont largement définies par le nom du champ, le type et les règles qui déterminent si le champ peut être vide. Selon le type, la taille du champ en caractères peut également être déterminée. De plus, une valeur par défaut peut être spécifiée pour être utilisée lorsque rien n'a été saisi dans le champ.

Dans l'interface graphique de base, les termes d'une colonne sont décrits quelque peu différemment, comme illustré ci-dessous.



Le champ devient Nom du champ, Type devient Type de champ. Le nom du champ et le type de champ sont entrés dans la zone supérieure de la fenêtre **Ébauche de table**. Dans la zone inférieure, vous avez la possibilité de définir, sous les propriétés du champ, les autres propriétés de la colonne, dans la mesure où elles peuvent être définies à l'aide de l'interface graphique. Les limitations incluent la définition de la valeur par défaut d'un champ de date sur la date d'entrée réelle. Cela n'est possible qu'en utilisant la commande SQL appropriée (voir "Saisie directe de commandes SQL" à la page 125).

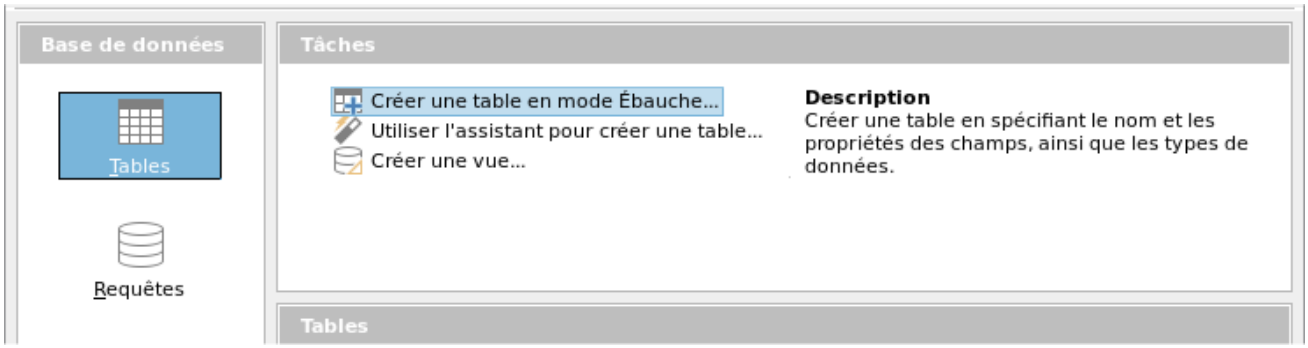
Note

Valeurs par défaut : le terme "Valeur par défaut" dans l'interface graphique ne signifie pas ce que l'utilisateur de la base de données comprend généralement comme valeur par défaut. L'interface graphique affiche visiblement une certaine valeur, qui est enregistrée avec les données.

La valeur par défaut dans une base de données est stockée dans la définition de table. Elle est ensuite écrit dans le champ chaque fois qu'il est laissé vide dans un nouvel enregistrement de données. Les valeurs par défaut SQL n'apparaissent pas lors de la modification des propriétés de la table.

Création à l'aide de l'interface utilisateur graphique

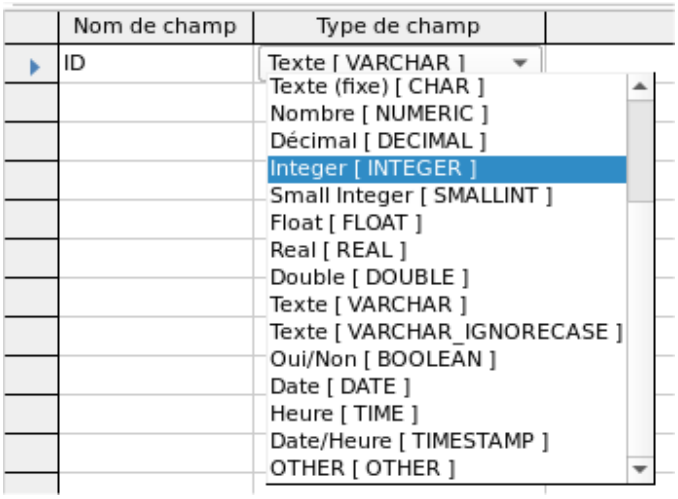
La création de la base de données à l'aide de l'interface utilisateur graphique (GUI) est expliquée étape par étape, en utilisant la table Medias comme exemple.



Lancez l'éditeur de table en cliquant sur **Créer une table en mode Ébauche**.

6. Champ ID :

Dans la première colonne, entrez le nom de champ ID. Tapez ensuite la touche Tab pour passer à la colonne Type de champ. Vous pouvez également cliquer avec la souris sur la colonne suivante pour la sélectionner ou appuyer sur la *touche Entrée*.

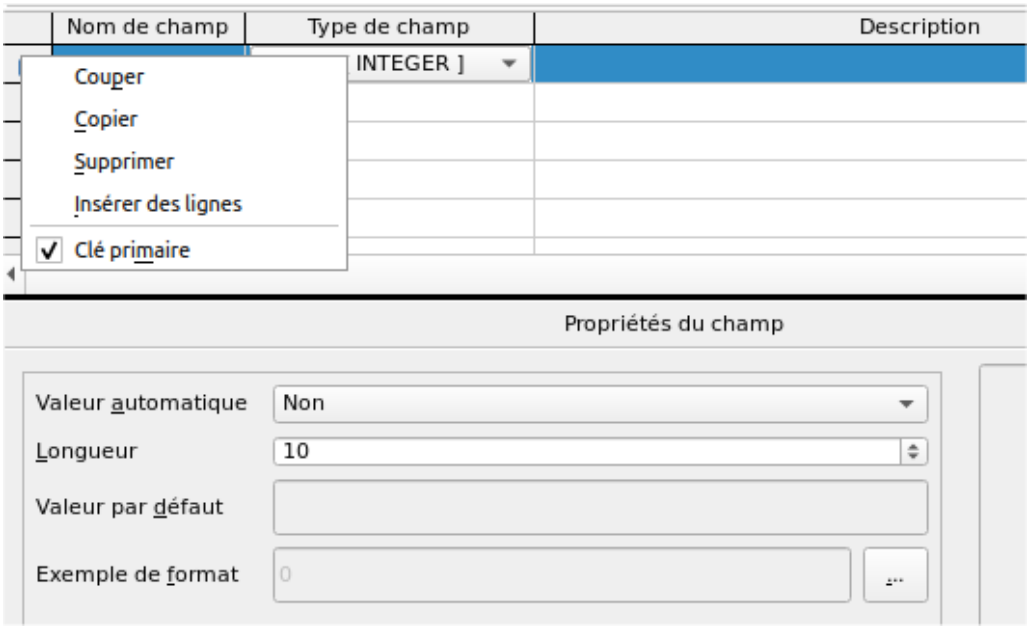


Sélectionnez `Integer [INTEGER]` dans la liste comme type de champ. La valeur par défaut est `Texte [VARCHAR]`. Les nombres entiers peuvent stocker un nombre de 10 chiffres maximum. De plus, Integer est le seul type disponible dans l'interface graphique qui peut recevoir une valeur incrémentée automatiquement.

Conseil

Pour effectuer rapidement une sélection dans la liste Type de champ à l'aide du clavier, tapez la touche correspondant à la première lettre de votre choix. La saisie répétée de cette touche peut être utilisée pour modifier la sélection. Par exemple, taper D peut changer votre sélection de Date à Date / Heure ou à Décimal.

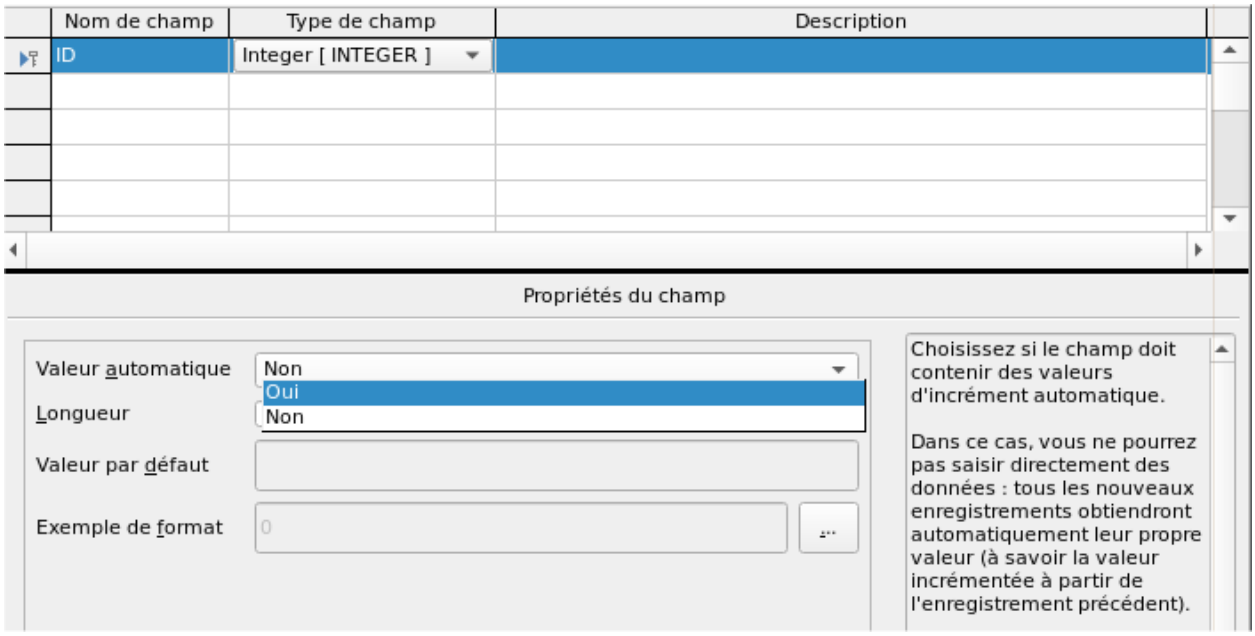
Définissez le champ *ID* comme clé primaire en cliquant sur le rectangle devant son nom de champ avec le bouton droit de la souris et en sélectionnant Clé primaire dans le menu contextuel. Un symbole de clé apparaît avant l'ID.



Note

La clé primaire n'a qu'un seul but : identifier de manière unique l'enregistrement. Par conséquent, vous pouvez utiliser un nom arbitraire pour ce champ. Dans l'exemple, nous avons utilisé le nom ID (identification) couramment utilisé.

a) Sous Propriétés du champ pour le champ ID, modifiez *Valeur automatique* de Non à Oui. Cela provoque l'incrémentation automatique de la clé primaire. Dans la base de données interne, le décompte commence à 0.

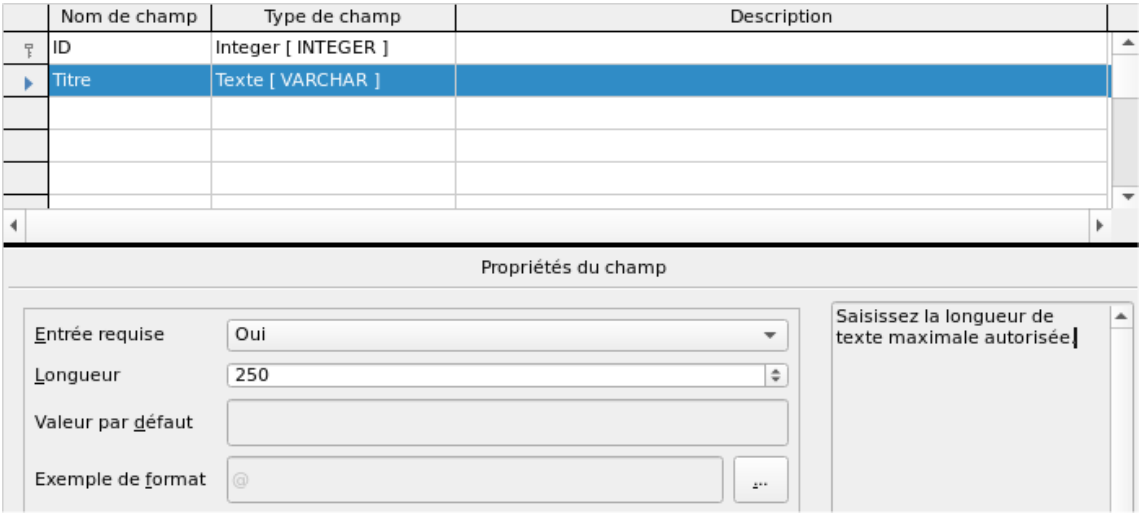


Valeur automatique ne peut être défini que pour un seul champ dans une table. Choisir **Valeur automatique > Oui** définit automatiquement ce champ comme clé primaire si aucune clé primaire n'a déjà été définie.

7. Le champ suivant est *Titre*.

Le nom du champ *Titre* est entré dans la colonne Nom du champ.

Le type de champ n'a pas besoin d'être modifié ici, car il est déjà défini sur `Texte [VARCHAR]`.



Dans les propriétés du champ, la longueur du champ doit être ajustée. La longueur par défaut est de 100 dans les versions récentes de LibreOffice, mais devrait être augmentée à 250 pour les titres multimédias.

Dans Propriétés du champ, modifiez *Entrée requise* de Non à Oui. Un média sans titre n'aurait aucun sens.

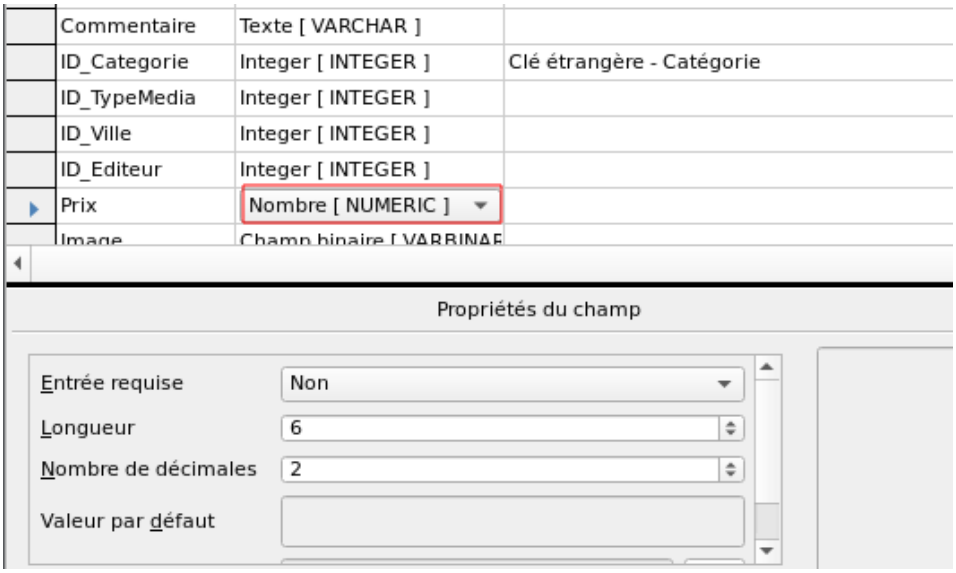
b) *Description* peut être n'importe quoi. Cette colonne peut également être laissée vide. La description sert uniquement à expliquer le contenu du champ aux personnes qui souhaitent afficher la définition de la table ultérieurement.

Nom de champ	Type de champ	Description
ID	Integer [INTEGER]	
Titre	Texte [VARCHAR]	
Edition	Texte [VARCHAR]	N° d'édition, nouvelle édition, etc...
Annee_Pub	Small Integer [SMALLINT]	Année de publication

8) Pour le champ *Annee_Pub*, le type `Small Integer [SMALLINT]` a été choisi. Celui-ci peut contenir un entier d'une taille maximale de 5 chiffres. L'année de publication n'est pas utilisée dans les calculs, mais en faire un entier garantit qu'elle ne contiendra aucun caractère `alphabétique`.

Nom de champ	Type de champ	Description
ID	Integer [INTEGER]	
Titre	Texte [VARCHAR]	
Edition	Texte [VARCHAR]	N° d'édition, nouvelle édition, etc...
Annee_Pub	Small Integer [SMALLINT]	Année de publication
Commentaire	Texte [VARCHAR]	
ID_Categorie	Integer [INTEGER]	Clé étrangère - Catégorie

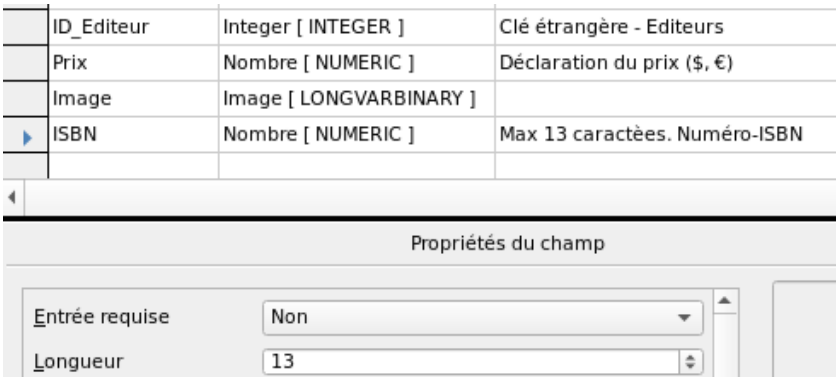
9) Pour le champ ID_Categorie, nous avons choisi le type Integer. Dans la table Categorie, la clé primaire doit avoir ce type de champ, donc ce qui est entré ici en tant que clé étrangère doit avoir le même type. Cela s'applique également aux clés étrangères suivantes ID_TypeMedia, ID_Ville et ID_Editeur.



10. Pour le champ Prix, utilisez le type `[NUMERIC]` ou `[DECIMAL]`. Ces deux types de champs peuvent contenir des valeurs avec un point décimal. Sous Propriétés du champ, définissez une longueur de 6 caractères. Cela devrait être suffisant pour les prix de nos médias.

Le nombre de décimales est fixé à 2. Cela fournit un prix maximum de 9999,99 puisque le point décimal lui-même n'est pas inclus dans le décompte.

Il n'est pas nécessaire de spécifier le caractère € dans le format, car une formule le gérera.



11. Pour le champ ISBN, utilisez le type [NUMERIC]. Cela peut être défini exactement à la longueur de champ correcte pour un ISBN. Les ISBN comportent 10 ou 13 caractères. Ils seront stockés sous forme de nombres sans séparateur. La longueur est fixée à un maximum de 13 caractères. Le nombre de décimales est défini sur zéro.
12. Enregistrez la table sous le nom Medias.

Nous avons maintenant créé la table principale de l'exemple de base de données. Toutes les autres tables peuvent être créées de la même manière. Veillez à ce que les types de champs et les propriétés des champs correspondent à ce qui va être stocké dans ces champs. Ceci est différent d'une feuille de calcul, dans laquelle une colonne peut contenir un mélange de propriétés.

Note

L'ordre des champs dans la table ne peut être modifié que jusqu'à ce que la table soit d'abord enregistrée dans l'interface graphique. Lorsque les données sont ensuite saisies directement dans la table, l'ordre des champs est fixe. Cependant, l'ordre peut toujours être librement modifié dans les requêtes, les formulaires et les rapports.

Clés primaires

Si aucune clé primaire n'est définie lors de la conception de la table, il vous sera demandé lors de l'enregistrement de la table si une clé primaire doit être créée. Cela indique qu'il manque un champ significatif dans le tableau. Sans clé primaire, la base de données HSQLDB ne peut pas accéder à la table dans l'interface graphique. Ce champ est généralement nommé ID et reçoit le type INTEGER avec *Valeur automatique* > **Oui** pour incrémenter automatiquement la valeur. Cliquez sur **Oui** dans la boîte de dialogue dedemande de clé primaire pour créer automatiquement un champ de clé primaire. Cliquer sur **Non** ou **Annuler** dans la boîte de dialogue de la clé primaire vous permet de désigner un champ existant comme clé primaire, en cliquant avec le bouton droit sur la flèche verte à gauche du champ correspondant.

Vous pouvez également utiliser une combinaison de champs comme clé primaire. Les champs doivent être déclarés ensemble comme clé primaire (maintenez la touche Ctrl ou Maj enfoncée). Ensuite, un clic droit fait de la combinaison de tous les champs en surbrillance la clé primaire.

Si des informations sont importées dans cette table à partir d'autres (par exemple une base de données d'adresses avec des codes postaux et des villes stockés en externe), un champ avec le même type de données que la clé primaire de l'autre table doit être inclus. Supposons que la table CodePostal ait comme clé primaire un champ appelé ID avec le type Tiny Integer. La table Adresses doit alors avoir un champ ID_CodePostal avec le type Tiny Integer. Ce qui est entré dans la table d'adresses est toujours le numéro qui sert de clé primaire pour l'emplacement donné dans la table de codes postaux. Cela signifie que la table d'adresses a désormais une clé étrangère en plus de sa propre clé primaire.

Une règle fondamentale pour nommer les champs dans une table est qu'aucun champ ne peut avoir le même nom. Par conséquent, vous ne pouvez pas avoir un deuxième champ appelé ID se produisant comme une clé étrangère dans la table d'adresses.

Le type de champ ne peut être modifié que dans une mesure limitée. L'augmentation d'une propriété (longueur d'un champ de texte, plus grande taille en nombre) est toujours autorisée, car toutes les valeurs déjà saisies correspondent aux nouvelles conditions. La diminution d'une propriété est plus susceptible de causer des problèmes et peut entraîner une perte de données.

Les champs de temps dans les tables ne peuvent pas être créés pour contenir des fractions de seconde. Pour cela, vous avez besoin d'un champ de type TIMESTAMP. Cependant, l'interface graphique vous permet uniquement de créer un champ de type TIMESTAMP avec date, heure, minute et seconde. Vous devrez ensuite modifier ce champ en utilisant **Outils > SQL.**

```
ALTER TABLE "Nom_Table" ALTER COLUMN "Nom_Champ" TIMESTAMP(6)
```

Le paramètre "6" rend le champ de type TIMESTAMP (Date/Heure) capable de stocker des fractions de seconde.

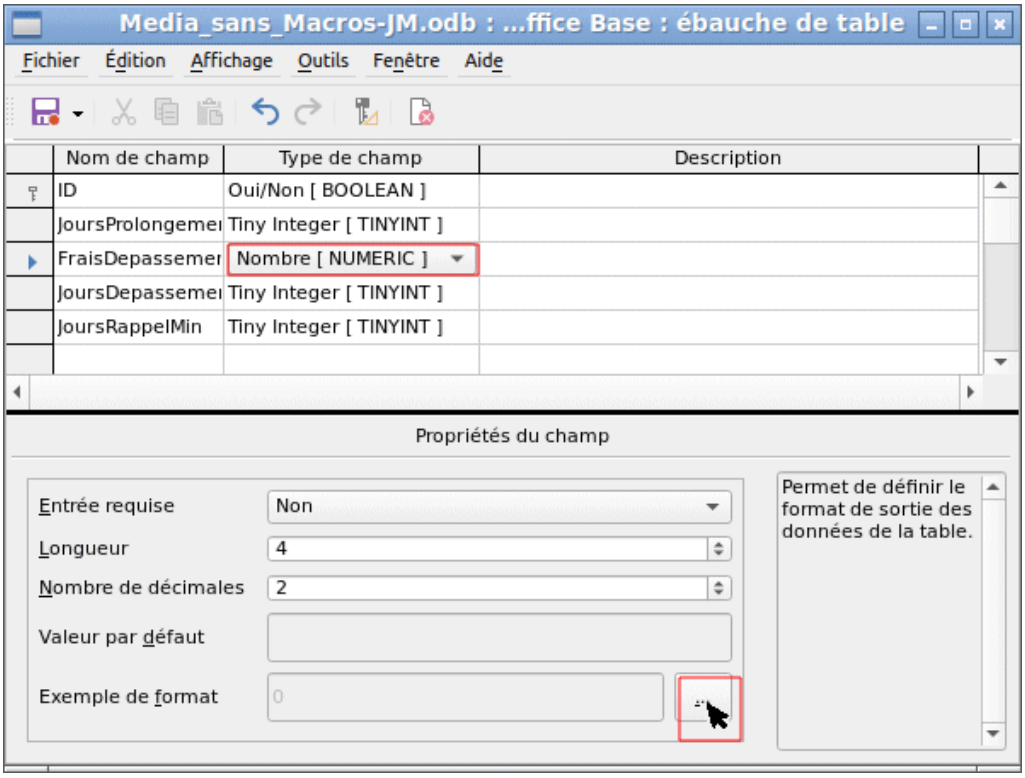
Formater les champs

Le formatage présente les valeurs de la base de données à l'utilisateur et permet la saisie de valeurs en fonction des conventions d'entrée normales dans ce pays. Sans mise en forme, les décimales sont marquées d'un point alors que la plupart des pays européens utilisent une virgule (*4,21* au lieu de *4.21*). Les valeurs de date sont présentées sous la forme *2020-08-23*. Lors de la configuration du formatage, vous devez tenir compte des normes locales.

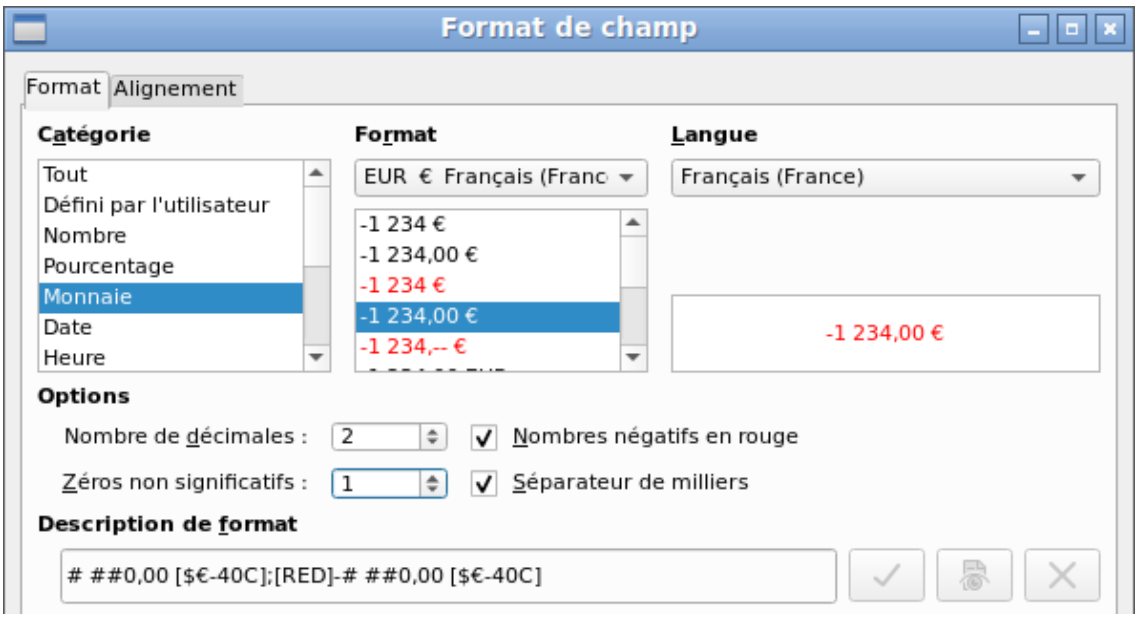
Le formatage ne fournit qu'une représentation du contenu. Une date représentée par un numéro d'année à deux caractères est toujours stockée sous la forme d'une année à quatre caractères. Si un champ est créé pour un nombre avec deux décimales, comme le montant des frais (appelé FraisDepassement) dans l'exemple suivant (table preference), le nombre est stocké avec deux décimales, même si le formatage a été défini par erreur pour ne pas les afficher. Un nombre avec deux décimales peut même être entré dans un champ formaté sans décimales. La partie décimale semble disparaître en entrée mais devient visible si le formatage est contourné.

Pour n'afficher qu'une heure, pas une date, les formulaires peuvent être formatés pour n'afficher que les informations nécessaires, masquant le reste du champ Date/Heure (TIMESTAMP). Dans le cas de l'enregistrement de l'heure à partir d'un chronomètre, par exemple, les minutes, secondes et fractions de seconde dans un horodatage peuvent être affichées en utilisant MM : SS.00 dans le format d'affichage. Un format sans date peut être défini ultérieurement dans des Formulaires à l'aide du champ formaté, mais pas directement dans le champ Date/Heure.

Le formatage des champs lors de la création de la table ou ultérieurement, via les propriétés des champs, utilise une boîte de dialogue distincte :



Le bouton en regard de **Propriétés du champ > Exemple de format** ouvre la boîte de dialogue permettant de modifier le format.



Lors de la création de champs de devise, veillez à ce que le champ numérique ait deux décimales définies. Le formatage peut être effectué lors de la création de la table dans l'interface graphique pour utiliser la devise appropriée lors de la saisie. Cela affecte uniquement l'entrée dans la table et dans les requêtes qui utilisent la valeur d'entrée sans recalcul. Dans les formulaires, la désignation de la devise doit être formatée séparément.

Note

Base enregistre le formatage des tables lors de la création des champs ou lors de la saisie des données si les formats de colonne sont modifiés par un clic droit sur les en-têtes de colonne. Les largeurs de colonne sur l'écran de saisie sont également enregistrées lorsqu'elles sont modifiées pendant la saisie des données.

Dans les requêtes, formulaires ou rapports, la mise en forme de l'affichage peut être modifiée selon les besoins.

Dans le cas des champs qui doivent contenir un pourcentage, notez que 1 % doit être stocké comme 0,01. L'écriture des pourcentages nécessite donc au moins deux décimales. Si des pourcentages fractionnaires tels que 3,45 doivent être stockés, la valeur numérique stockée nécessite quatre décimales.

Créer un index

Parfois, il est utile d'indexer d'autres champs ou une combinaison d'autres champs en plus de la clé primaire. Un index accélère la recherche et peut également être utilisé pour éviter les doublons.

Chaque index a un ordre de tri défini. Si une table est affichée sans tri, l'ordre de tri sera en fonction du contenu des champs spécifiés dans l'index.

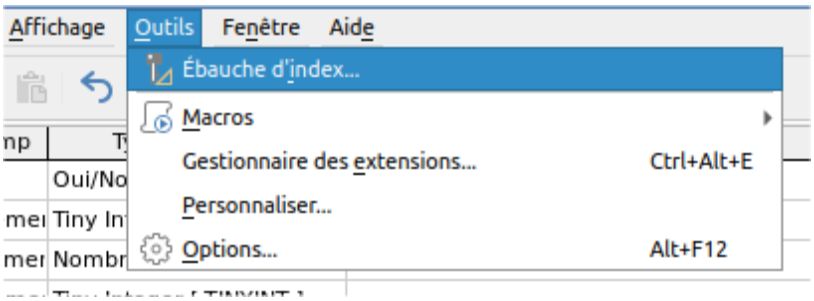


Figure 47: Accès à la conception d'index

Ouvrez la table Lecteurs pour l'édition en cliquant avec le bouton droit de la souris et en utilisant le menu contextuel *Éditer*. Ensuite, vous pouvez accéder à la création d'index avec **Outils > Ébauche d'index** (figure 47).

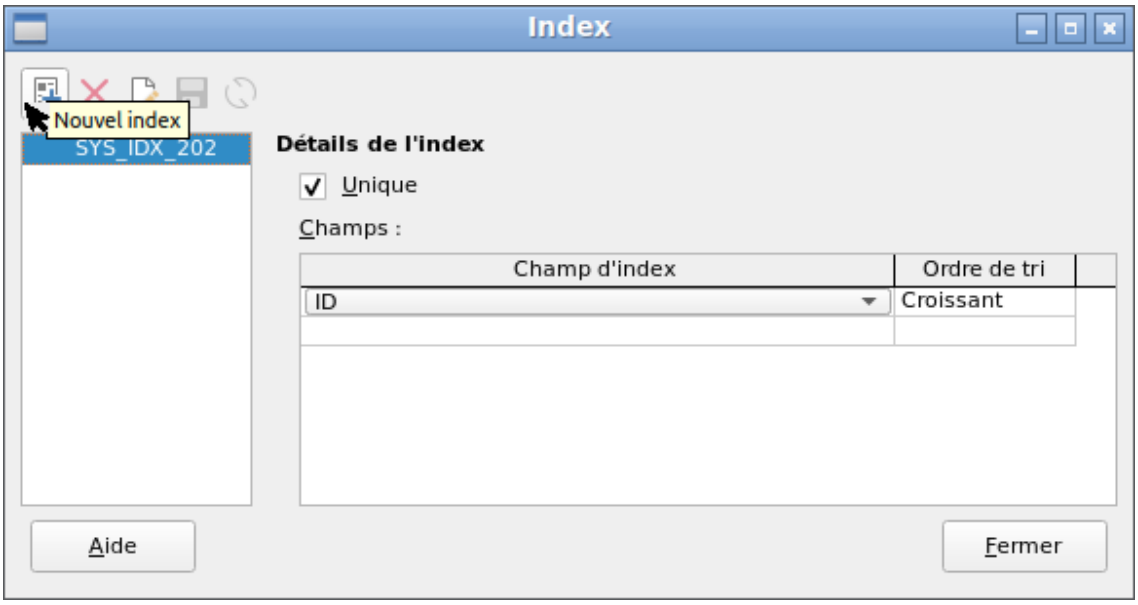


Figure 48: Créer un nouvel index

Dans la boîte de dialogue Index (Figure 48), cliquez sur l'icône **Nouvel index** pour créer un index en plus de la clé primaire.

Le nouvel index reçoit automatiquement le nom index1. Le **Champ d'Index** spécifie le ou les champs à utiliser pour cet index. En même temps, vous pouvez choisir l'ordre de tri.

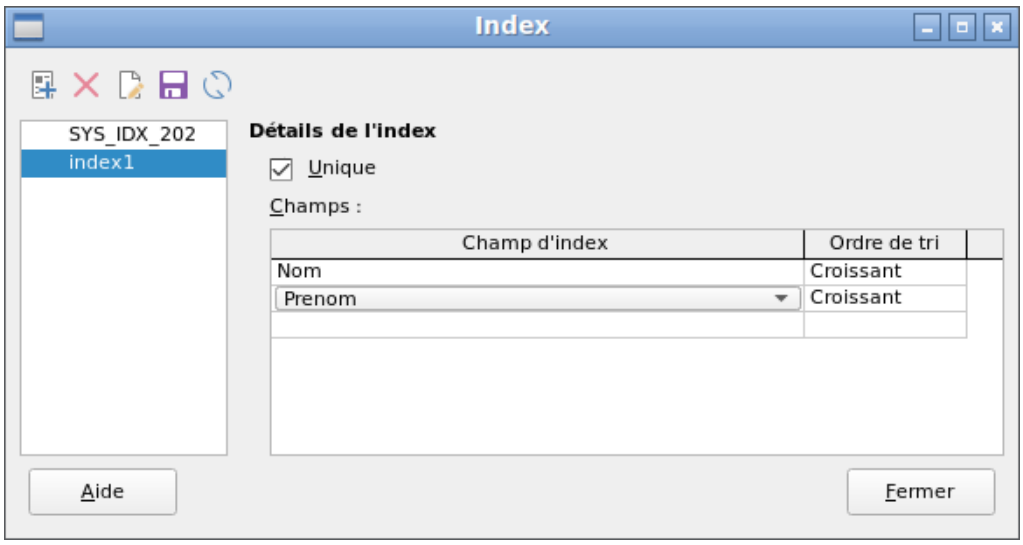


Figure 49: L'index est défini comme Unique

En principe, un index peut également être créé à partir de champs de table qui ne contiennent pas de valeurs uniques. Toutefois, dans la figure 49, le détail d'index **Unique** a été coché, de sorte que le champ Nom combiné avec le champ Prenom ne puisse avoir que des entrées qui ne se produisent pas déjà dans cette combinaison. Ainsi, par exemple, Fernand Sardou et Jackie Sardou sont possibles, de même que Robert Redford et Robert DeNiro.

Si un index est créé pour un seul champ, l'unicité s'applique à ce champ. Un tel index est généralement la clé primaire. Dans ce champ, chaque valeur ne peut apparaître qu'une seule fois. En outre, dans le cas des clés primaires, le champ ne peut en aucun cas être NULL.

Une circonstance exceptionnelle pour un index unique est lorsqu'il n'y a pas d'entrée dans un champ (le champ est NULL). Puisque NULL peut avoir n'importe quelle valeur arbitraire, un index utilisant deux champs est toujours autorisé à avoir la même entrée à plusieurs reprises dans l'un des champs tant qu'il n'y a pas d'entrée dans l'autre.

Note

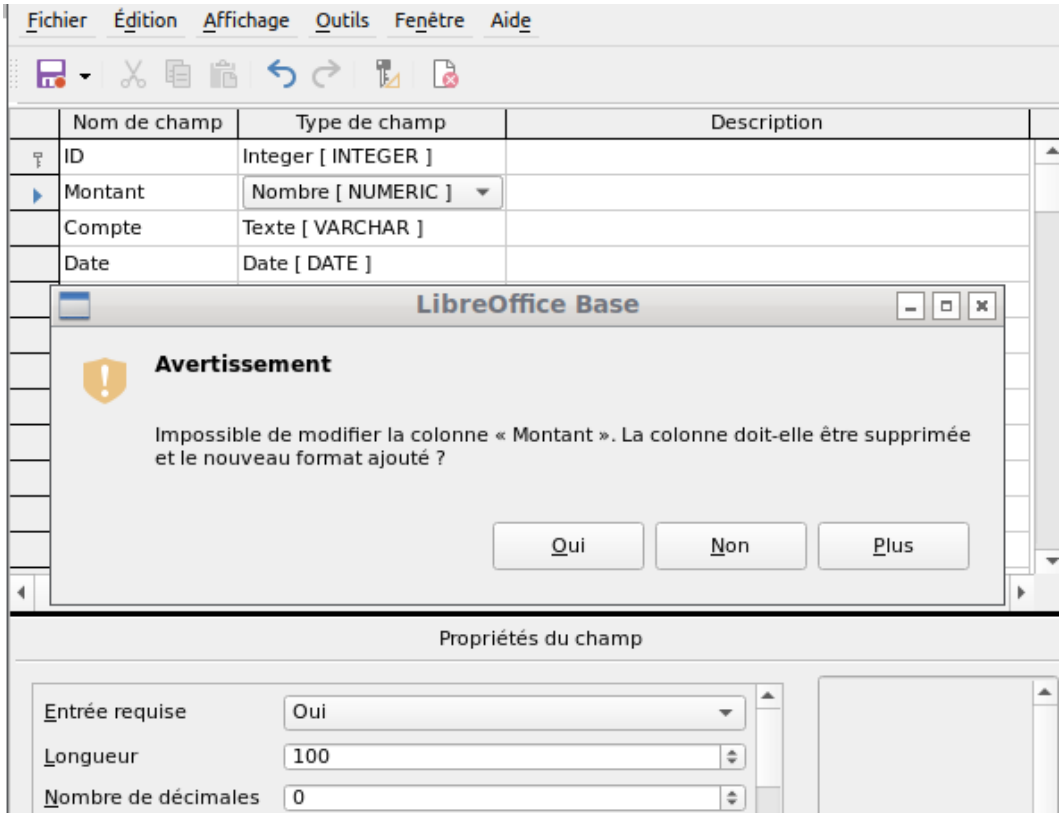
NULL est utilisé dans les bases de données pour désigner une cellule vide, qui ne contient rien. Aucun calcul n'est possible avec un champ NULL. Cela contraste avec les feuilles de calcul, dans lesquelles les champs vides contiennent automatiquement la valeur 0 (zéro).

Exemple : dans une base de données média, le numéro du média et la date de prêt sont saisis lorsque l'article est prêté. Lorsque l'article est retourné, une date de retour est saisie. En théorie, un index utilisant les champs ID_Media et DateRetour pourrait facilement empêcher le même article d'être prêté à plusieurs reprises sans que la date de retour ne soit notée. Malheureusement, cela ne fonctionnera pas, car la date de retour n'a initialement aucune valeur. L'index empêchera un élément d'être marqué comme retourné deux fois avec la même date, mais il ne fera rien d'autre.

Problèmes lors de la modification des tables

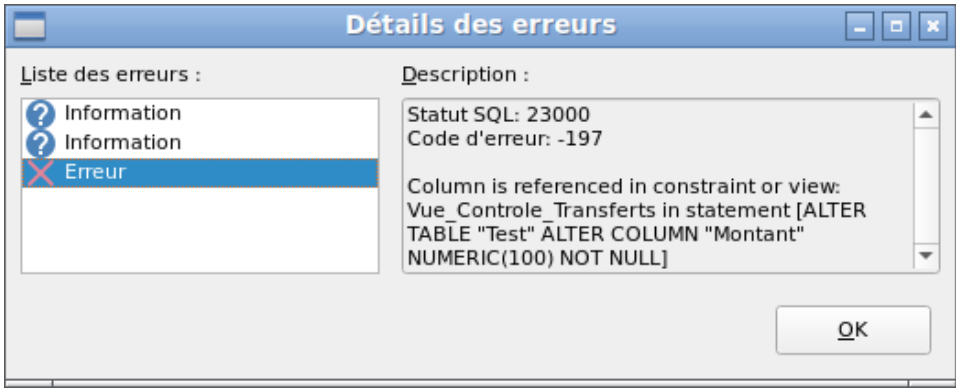
Il est préférable de créer des tables complètes avec tous leurs paramètres requis, de sorte que les modifications de la configuration des tables ne soient pas nécessaires ultérieurement. Lorsque les propriétés des champs (nom de champ, entrée obligatoire, etc.) sont modifiées ultérieurement,

cela peut conduire à des messages d'erreur qui ne sont pas dus à l'interface graphique mais à une tentative de modification de la base de données sous-jacente d'une manière indésirable.



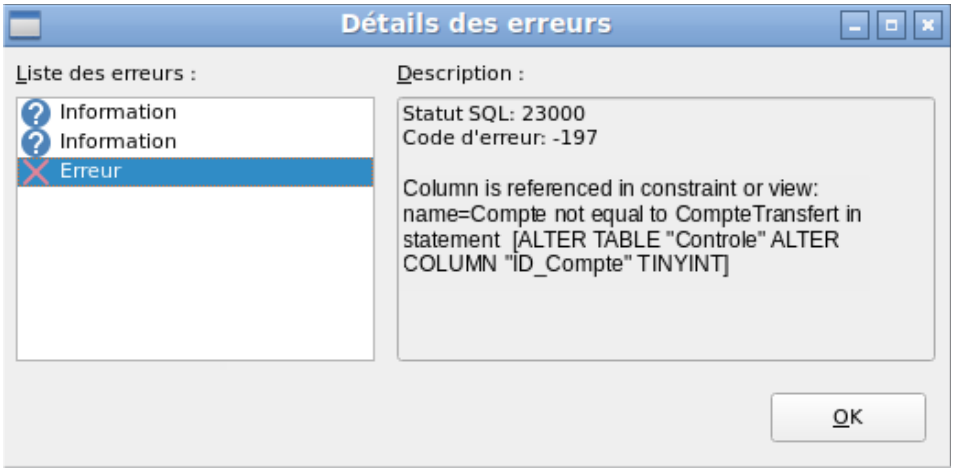
Dans ce cas, le champ Montant doit être réinitialisé sur Entrée requise = oui. Le symbole d'avertissement nous informe que ce changement peut entraîner une perte de données. Une simple modification n'est pas possible, car il se peut qu'il y ait déjà des enregistrements qui n'ont aucune entrée dans ce champ.

Cliquer sur **Oui** entraîne un autre message d'erreur, car la structure de la base de données ne permet pas de supprimer ce champ. Cliquer sur **Non** annule toute l'opération. L'option **Plus** est fournie chaque fois que possible afin de vous donner des informations supplémentaires sur la résolution du problème.



La notification d'erreur `Column is referenced in constraint or view` signifie : La colonne avec le nom de champ "Montant" est référencée dans une autre partie de la base de données. Il peut s'agir d'une définition contraignante ou d'une vue de table créée par un utilisateur après la création de la table elle-même. L'illustration ci-dessus montre que le nom de la contrainte

ou de la vue est Vue_Controle_Transferts. Cela indique clairement à l'utilisateur où des modifications doivent être apportées dans la base de données. Par exemple, le code SQL de la vue pourrait d'abord être enregistré en tant que requête, puis la vue pourrait être détruite et une nouvelle tentative de recréation du champ pourrait être effectuée.



Dans ce cas, le nom de la contrainte Compte non égal à CompteTransfert nous conduit à la définition de cette contrainte. La condition est que la valeur du champ nommé ID_Compte ne soit pas autorisée à être la même que la valeur du champ ID_CompteTransfert. La colonne ne peut être modifiée que si cette condition est supprimée.

Maintenant, si une autre erreur se produit, cela est plus susceptible d'être causé par le champ correspondant étant lié à un champ dans une autre table par une relation définie. Dans ce cas, le lien doit être rompu en utilisant **Outils > Relations** avant que la modification puisse être effectuée.

Limitations de l'ébauche graphique de tables

La séquence des champs dans une table ne peut pas être modifiée une fois la base de données enregistrée. Pour afficher une séquence différente, une requête est nécessaire.

Seule l'entrée de commandes SQL directes peut insérer un champ dans une position spécifique dans la table. Cependant, les champs déjà créés ne peuvent pas être déplacés par cette méthode.

Les propriétés des tables doivent être définies au début : par exemple quels champs ne doivent pas être NULL et lesquels doivent contenir une valeur standard (par défaut). Ces propriétés ne peuvent pas être modifiées ultérieurement à l'aide de l'interface graphique.

Les valeurs par défaut que vous pouvez définir dans l'interface graphique ne sont pas aussi puissantes que les valeurs par défaut possibles dans la base de données elle-même. Par exemple, vous ne pouvez pas définir la valeur par défaut d'un champ de date comme étant la date d'entrée. Cela n'est possible qu'avec des commandes SQL entrées directement.

Saisie directe des commandes SQL

Pour saisir directement des commandes SQL, accédez à **Outils > SQL**.

Les commandes sont entrées dans la zone supérieure de la fenêtre (Figure 50). Dans la zone inférieure (état), le succès ou la raison de l'échec est affiché. Les résultats des requêtes peuvent être affichés dans la zone Sortie si la case est cochée.

Figure 50: Boîte de dialogue pour la saisie directe des commandes SQL

Un résumé des commandes possibles pour le moteur HSQLDB intégré peut être trouvé à http://www.hsqldb.org/doc/1.8/guide/ch09.html. Le contenu est décrit dans les sections suivantes. Certaines commandes n'ont de sens que lorsqu'il s'agit d'une base de données HSQLDB externe (Spécifier l'utilisateur, le mot de passe, etc.). Le cas échéant, ceux-ci sont traités dans la section "Travailler avec une base HSQLDB externe" dans l'annexe de ce manuel.

Note

LibreOffice est basé sur la version 1.8.0 de HSQLDB. La version serveur actuellement disponible est 2.5. Les fonctions de la nouvelle version sont plus étendues. Elles peuvent être trouvées sur http://hsqldb.org/web/hsqlDocsFrame.html. La description de la version 1.8 est maintenant sur http://www.hsqldb.org/doc/1.8/guide/. Une description plus détaillée est donnée dans les packages d'installation de HSQLDB, qui peuvent être téléchargés à partir de http://sourceforge.net/projects/hsqldb/files/hsqldb/.

Création de table

Une commande simple pour créer une table utilisable est :

```
CREATE TABLE "Test" ("ID" INT PRIMARY KEY, "Texte" VARCHAR(50)) ;
```

Décomposition de cette commande :

```
CREATE TABLE "Test" : Crée une tablel avec le nom "Test".
```

`( )` : les noms de champs, types de champs et options spécifiés sont insérés entre parenthèses.

```
"ID" INT PRIMARY KEY, "Text" VARCHAR(50) : Nom de champ ID, entier de type
```

numérique comme clé primaire ; nom de champ Texte de type texte d'une longueur variable et une taille limitée à 50 caractères.

Paramètres de la commande CREATE :

```
CREATE [MEMORY | CACHED | [GLOBAL] TEMPORARY | TEMP | TEXT] TABLE "Table
name" (<Field definition> [,…] [, <Constraint Definition>…]) [ON COMMIT
{DELETE | PRESERVE} ROWS] ;
```

`[MEMORY | CACHED | [GLOBAL] TEMPORARY | TEMP | TEXT] :`

Spécifie l'emplacement de la table nouvellement créée. Le paramètre par défaut est `MEMORY` : HSQLDB crée toutes les tables dans la mémoire centrale. Ce paramètre s'applique également aux tables écrites dans la base de données intégrée par LibreOffice Base. Une autre possibilité serait d'écrire les tables sur le disque dur et d'utiliser la mémoire uniquement pour mettre en mémoire tampon l'accès au disque dur (`CACHED`).

Note

```
CREATE TABLE "Test" ("ID" INT PRIMARY KEY, "Texte" VARCHAR(50))
```

Crée une table de texte dans HSQLDB. Elle doit maintenant être liée à un fichier texte externe (par exemple un fichier *.csv) : `SET TABLE "Text" SOURCE "Text.csv";`

Naturellement, le fichier Text.csv doit avoir les champs correspondants dans le bon ordre. Lors de la création du lien, diverses options supplémentaires peuvent être sélectionnées. Pour plus de détails, voir http://www.hsqldb.org/doc/1.8/guide/guide.html#set_table_source-section

Les tables texte ne sont pas protégées en écriture contre d'autres programmes. Il peut donc arriver qu'un autre programme ou utilisateur modifie la table au moment où Base y accède. Les tables texte sont principalement utilisées pour l'échange de données entre différents programmes.

Les tables au format TEXT (comme CSV) ne sont pas inscriptibles dans les bases de données internes qui sont configurées uniquement dans MEMORY, tandis que Base ne peut pas accéder aux tables TEMPORARY ou TEMP.

Dans ce cas, les commandes SQL sont exécutées mais les tables ne sont pas affichées (et donc ne peuvent pas être supprimées) à l'aide de l'interface graphique. Les données saisies via SQL ne sont pas non plus visibles par le module de requête de l'interface graphique, sauf si la suppression automatique du contenu, après que le commit final, est empêchée (avec ON COMMIT PRESERVE ROWS). Toute demande dans ce cas montre une table sans aucun contenu.

Les tables créées directement avec SQL ne sont pas immédiatement affichées. Vous devez soit utiliser **Affichage > Actualiser les tables** ou simplement fermer la base de données, puis la rouvrir.

`<Field definition>:`

`"Nom du champ" Type de Données [(Nombre de caractères[, Nombre de Décimales])] [{DEFAULT "Valeur par défaut" | GENERATED BY DEFAULT AS IDENTITY (START WITH <n>[, INCREMENT BY <m>])}] | [[NOT] NULL] [IDENTITY] [PRIMARY KEY]`

Permet aux valeurs par défaut d'être incluses dans la définition de champ.

Pour les champs de texte, vous pouvez saisir du texte entre guillemets simples ou `NULL`. La seule fonction SQL autorisée est `CURRENT_USER`. Cela n'a de sens que si HSQLDB est utilisé comme base de données serveur externe avec plusieurs utilisateurs.

Pour les champs de date et d'heure, une date, une heure ou une combinaison des deux peuvent être saisies entre guillemets simples ou bien NULL. Vous devez vous assurer que la date respecte les conventions nationales (jj-mm-aaaa), que l'heure a le format hh:mm:ss et qu'une valeur date/heure combinée a le format jj-mm-aaaa hh :mm:ss.

Fonctions SQL autorisées :

pour la date actuelle	*CURRENT_DATE, TODAY, CURDATE()*
pour l'heure actuelle	*CURRENT_TIME, NOW, CURTIME()*
pour l'horodatage actuel	*CURRENT_TIMESTAMP, NOW*.

Pour les champs booléens (oui/non), les expressions *FALSE, TRUE, NULL* peuvent être saisies. Ceux-ci doivent être saisis sans guillemets simples.

Pour les champs numériques, tout nombre valide dans la plage, ou NULL, est possible. Ici aussi, si vous entrez `NULL`, n'utilisez pas de guillemets. Lorsque vous entrez des décimales, assurez-vous que le point décimal est une virgule. (Certaines personnes francophones utilisent le point décimal).

Pour les champs binaires (images, etc.), toute chaîne hexadécimale valide entre guillemets simples ou NULL est possible. Un exemple de chaîne hexadécimale est : 0004FF, qui représente 3 octets : d'abord0, puis4 et enfin 255 (FF). Comme les champs binaires en pratique ne doivent être saisis que pour les images, vous devez connaître le code binaire de l'image qui doit servir par défaut.

Note

Système hexadécimal : Les nombres sont en base 16. Un système mixte composé des nombres de 0 à 9 et des lettres de A à F fournit 16 valeurs possibles pour chaque colonne. Avec deux colonnes, vous pouvez avoir 16 × 16 = 256 valeurs possibles de 0 à 255. Cela correspond à 1 octet (2^8).

NOT NULL : la valeur du champ ne peut pas être NULL. Cette condition ne peut être donnée que dans la définition du champ.

Exemple :

```
CREATE TABLE "Test" ("ID" INT GENERATED BY DEFAULT AS IDENTITY (START WITH
10), "Nom" VARCHAR(50) NOT NULL, "Date" DATE DEFAULT TODAY) ;
```

Une table nommée Test est créée. L'ID du champ clé est défini comme AutoChamp, avec des valeurs commençant à 10. Le champ de saisie Nom est un champ de texte d'une taille maximale de 50 caractères. Il ne doit pas être vide. Enfin, nous avons le champ de date Date qui stocke par défaut la date actuelle, si aucune autre date n'est saisie. Cette valeur par défaut ne prend effet que

lorsqu'un nouvel enregistrement est créé. La suppression d'une date dans un enregistrement existant laisse le champ vide.

```
<Définition des contraintes>:
    [CONSTRAINT "Nom"]
```

```
UNIQUE ("Nom_du_Champ_1" [," Nom_du_Champ_2"…]) |
PRIMARY KEY ("Nom_du_Champ_1" [," Nom_du_Champ_2"…]) |
FOREIGN KEY ("Nom_du_Champ_1" [," Nom_du_Champ_2"…])
REFERENCES "Nom_Autre_Table" ("Nom_du_Champ_1" [," Nom_du_Champ_2"…])
[ON {DELETE | UPDATE}
{CASCADE | SET DEFAULT | SET NULL}] |
CHECK(<Condition_recherche>)
```

Les contraintes définissent les conditions qui doivent être remplies lors de la saisie des données. Les contraintes peuvent recevoir un nom.

`UNIQUE ("Nom_du_Champ_1")` : la valeur du champ doit être unique dans cette colonne.
PRIMARY KEY ("Nom_du_Champ") : la valeur du champ doit être unique et ne peut pas être NULL (clé primaire)

`FOREIGN KEY ("Nom_du_Champ") REFERENCES <"Nom_Autre_Table"> ("Nom_du_Champ")` : les champs spécifiés de cette table sont liés aux champs d'une autre table. La valeur des champs doit être testée pour l'intégrité référentielle en tant que clés étrangères ; autrement dit, il doit y avoir une clé primaire correspondante dans l'autre table, si une valeur est entrée ici.

`[ON {DELETE | UPDATE} {CASCADE | SET DEFAULT | SET NULL}]` : Dans le cas d'une clé étrangère, cela spécifie ce qui doit se passer si, par exemple, l'enregistrement étranger est supprimé. Cela n'a aucun sens, dans une table de prêt pour une bibliothèque, d'avoir un numéro d'utilisateur pour lequel l'utilisateur n'existe plus. L'enregistrement correspondant doit être modifié pour que la relation entre les tables reste valide. En général, l'enregistrement est simplement supprimé. Cela se produit si vous sélectionnez `ON DELETE CASCADE.`

`CHECK(<Condition_recherche>)` : Formulé comme une condition WHERE, mais uniquement pour l'enregistrement en cours.

```
CREATE TABLE "Mesure_Temps" ("ID" INT PRIMARY KEY, "Heure_debut" TIME,
"Heure_fin" TIME, CHECK ("Heure_debut" <= "Heure_fin")) ;
```

La condition CHECK exclut l'entrée d'une valeur d'heure de fin antérieure à l'heure de début. Une tentative pour faire cela produit un message d'erreur semblable à :

```
Check constraint violation SYS_CT_357 table : Mesure_Temps…
```

La contrainte de recherche se voit attribuer un nom qui n'est pas très informatif(SYS_CT_357). Au lieu de cela, le nom pourrait être défini dans la définition de la table :

```
CREATE TABLE "Mesure_Temps" ("ID" INT PRIMARY KEY, "Heure_debut" TIME,
"Heure_fin" TIME, CONSTRAINT "Heure_debut<=Heure_fin" CHECK
("Heure_debut<=Heure_fin")) ;
```

Cela donne un message d'erreur un peu plus clair puisque le nom de la contrainte impliquée apparaît à la place du code.

Les contraintes doivent être respectées lors de l'établissement de relations entre les tables ou de l'indexation de champs particuliers. Les contraintes sont établies en utilisant la condition "CHECK",

dans l'interface graphique en utilisant **Outils > Relations**, et aussi dans les **index créés** lors de la conception de table sous **Outils > Ébauche d'index**.

`[ON COMMIT {DELETE | PRESERVE} ROWS] :`

Le contenu des tables de type `TEMPORARY` ou `TEMP` est effacé par défaut lorsque vous avez fini de travailler avec un enregistrement particulier (`ON COMMIT DELETE ROWS`). Cela vous permet de créer des enregistrements temporaires, qui contiennent des informations pour d'autres actions à effectuer en même temps.

Si vous souhaitez qu'une table de ce type contienne des données disponibles pour toute une session (de l'ouverture d'une base de données à sa fermeture), choisissez `ON COMMIT PRESERVE ROWS`.

Modification d'une table

Parfois, vous souhaiterez peut-être insérer un champ supplémentaire à une position particulière de la table. Supposons que vous ayez une table appelée Adresses avec les champs ID, Nom, Rue, etc. Vous vous rendez compte qu'il serait peut-être judicieux de distinguer les prénoms et les noms de famille.

```
ALTER TABLE "Adresses" ADD "Prenom" VARCHAR(25) BEFORE "Nom";
```

`ALTER TABLE "Adresses"`: Modifier la table de nom Adresses.
`ADD "Prenom" VARCHAR(25)` : insérer le champ Prenom d'une longueur de 25 caractères.
`BEFORE "Nom"`: avant le champ Nom

La possibilité de spécifier la position de champs supplémentaires après la création de la table n'est pas disponible dans l'interface graphique.

```
ALTER TABLE "Nom_Table" ADD [COLUMN] <Définition_de_champ> [BEFORE
"Nom_de_champ_existant"] ;
```

La désignation supplémentaire COLUMN n'est pas nécessaire dans les cas où aucun choix alternatif n'est disponible.

```
ALTER TABLE "Nom_Table" DROP [COLUMN] "Nom_Champ";
```

Le champ *Nom_Champ* est effacé de la table *Nom_Table*. Cependant, cela n'a pas lieu si le champ est impliqué dans une vue ou en tant que clé étrangère dans une autre table.

```
ALTER TABLE "Nom_Table" ALTER COLUMN "Nom_Champ" RENAME TO
"Nouveau_Nom_Champ"
```

Modifie le nom d'un champ.

```
ALTER TABLE "Nom_Table" ALTER COLUMN "Nom_Champ" SET DEFAULT <Valeur par
defaut>} ;
```

Définit une valeur par défaut spécifique pour le champ. `NULL` supprime une valeur par défaut existante.

```
ALTER TABLE "Nom_Table" ALTER COLUMN "Nom_Champ" SET [NOT] NULL
```

Définit ou supprime une condition `NOT NULL`pour un champ.

```
ALTER TABLE "Nom_Table" ALTER COLUMN <Définition_du_champ>;
```

La définition du champ correspond à celle de Création de table avec les restrictions suivantes :

- Le champ doit déjà être un champ de clé primaire pour accepter la propriété `IDENTITY`. IDENTITY signifie que le champ a la propriété AutoChamp. Cela n'est possible que pour

les champs `INTEGER` ou `BIGINT`. Pour les descriptions de ces types de champs, voir l'annexe de ce manuel.
- Si le champ possède déjà la propriété `IDENTITY` mais qu'elle n'est pas répétée dans la définition de champ, la propriété `IDENTITY` existante est supprimée.
- La valeur par défaut deviendra celle spécifiée dans la nouvelle définition de champ. Si la définition de la valeur par défaut est laissée vide, toute valeur par défaut déjà définie est supprimée.
- La propriété `NOT NULL` continue dans la nouvelle définition, si elle n'est pas définie autrement. Ceci est en contraste avec la valeur par défaut.
- Dans certains cas, selon le type de modification, la table doit être vide pour que la modification se produise. Dans tous les cas, le changement n'aura d'effet que s'il est en principe possible (par exemple un changement de `NOT NULL` à `NULL`) et les valeurs existantes peuvent toutes être traduites (par exemple un changement de `TINYINT` à `INTEGER`).

```
ALTER TABLE "Nom_Table" ALTER COLUMN "Nom_Champ" RESTART WITH
<Nouvelle_valeur>
```

Cette commande est utilisée exclusivement pour un champ `IDENTITY`. Il détermine la valeur suivante pour un champ avec l'ensemble de fonctions AutoChamp. Il peut être utilisé, par exemple, lorsqu'une base de données est initialement utilisée avec des données de test, puis alimentée avec des données réelles. Cela nécessite que le contenu des tables soit supprimé et qu'une nouvelle valeur telle que "1" soit définie comme valeur de départ pour le champ.

```
ALTER TABLE "Nom_Table"
ADD [CONSTRAINT "Nom_condition"] CHECK (<Condition_recherche>) ;
```

Cela ajoute une condition de recherche introduite par le mot CHECK. Une telle condition ne s'appliquera pas rétrospectivement aux enregistrements existants, mais elle s'appliquera à toutes les modifications ultérieures et aux enregistrements nouvellement saisis. Si un nom de contrainte n'est pas défini, il sera attribué automatiquement.

Exemple :

```
ALTER TABLE "Pret" ADD CHECK
(IFNULL("Date_Retour","Date_Pret")>="Date_Pret")
```

La table *Pret* doit être protégée contre les erreurs de saisie. Par exemple, vous devez éviter qu'une date de retour ne soit donnée qui est antérieure à la date du prêt. Maintenant, si cette erreur se produit pendant le processus de retour, vous obtiendrez un message d'erreur `Check constraint violation…` (vérifier la violation de contrainte)

```
ALTER TABLE "Nom_Table"
ADD [CONSTRAINT "Nom_contrainte"] UNIQUE ("Nom_Champ1", "Nom_Champ2"…) ;
```

Ici, une condition est ajoutée qui force les champs nommés à avoir des valeurs différentes dans chaque enregistrement. Si plusieurs champs sont nommés, cette condition s'applique à la combinaison plutôt qu'aux champs individuels. NULL ne compte pas ici. Un champ peut donc avoir la même valeur à plusieurs reprises sans poser de problème, si l'autre champ de chacun des enregistrements est `NULL`.

Cette commande ne fonctionnera pas s'il existe déjà une condition `UNIQUE` pour la même combinaison de champs.

```
ALTER TABLE "Nom_Table"
ADD [CONSTRAINT "Nom_contrainte"] PRIMARY KEY ("Nom_Champ1",
"Nom_Champ2"…) ;
```

Ajoute une clé primaire, éventuellement avec une contrainte, à une table. La syntaxe de la contrainte est la même que lors de la création d'une table.

```
ALTER TABLE "Nom_Table" ADD [CONSTRAINT "Nom_contrainte"] FOREIGN KEY
("Nom_Champ1", "Nom_Champ2"…)
REFERENCES "Nom_autre_Table" ("Nom_Champ1_autre_table",
"Nom_Champ2_autre_table"…)
[ON {DELETE | UPDATE} {CASCADE | SET DEFAULT | SET NULL}] ;
```

Cela ajoute une clé étrangère (`FOREIGN KEY`) à la table. La syntaxe est la même que lors de la création d'une table.

L'opération se terminera par un message d'erreur, si une valeur de la table n'a pas de valeur correspondante dans la table contenant cette clé primaire.

Exemple : les tables Noms et Adresses doivent être liées. La table Noms contient un champ avec le nom ID_Adresse. La valeur de ceci doit être liée au champ ID dans la table Adresses. Si la valeur 1 se trouve dans ID_Adresse mais pas dans le champ ID de la table Adresses, le lien ne fonctionnera pas. Cela ne fonctionnera pas non plus si les deux champs sont de types différents.

```
ALTER TABLE "Nom_Table" DROP CONSTRAINT "Nom_contrainte";
```

Cette commande supprime la contrainte nommée *Nom_contrainte* (`UNIQUE, CHECK, FOREIGN KEY`) d'une table.

```
ALTER TABLE "Nom_Table" RENAME TO "Nouveau_Nom_Table";
```

Pour terminer, cette commande ne change que le nom d'une table.

Note

Lorsque vous modifiez une table à l'aide de SQL, la modification affecte la base de données mais n'est pas nécessairement apparente ou effective dans l'interface graphique. Lorsque la base de données est fermée et rouverte, les modifications apparaissent également dans l'interface graphique.

Les modifications sont également affichées si vous choisissez **Affichage > Actualiser les tables** dans le conteneur de table.

Supprimer des tables

```
DROP TABLE "Nom_Table" [IF EXISTS] [RESTRICT | CASCADE] ;
```

Supprime la table *Nom_Table*.

`IF EXISTS` empêche qu'une erreur se produise si cette table n'existe pas.

`RESTRICT` est l'arrangement par défaut et n'a pas besoin d'être choisi explicitement ; cela signifie que la suppression ne se produit pas si la table est liée à une autre table par l'utilisation d'une clé étrangère ou s'il existe une vue active de cette table. Les requêtes ne sont pas affectées, car elles ne sont pas stockées dans HSQLDB, mais dans le conteneur de la base.

Si à la place vous choisissez `CASCADE`, tous les liens vers la table *Nom_Table* sont supprimés. Dans les tables liées, toutes les clés étrangères sont définies sur `NULL`. Toutes les vues faisant référence à la table nommée sont également complètement supprimées.

Lier des tables

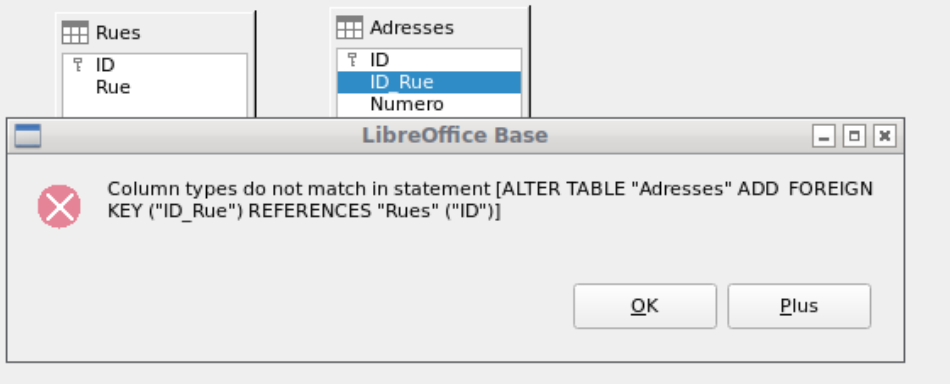
En principe, vous pouvez avoir une base de données sans liens entre les tables. L'utilisateur doit alors s'assurer lors de la saisie des données, que les relations entre les tables restent correctes.

Cela se produit généralement grâce à l'utilisation de formulaires de saisie appropriés qui gèrent cela.

La suppression d'enregistrements dans des tables liées n'est pas une mince affaire. Supposons que vous souhaitiez supprimer une rue particulière de la table Rues de la figure ci-dessous, où ce champ est lié à la table Adresses en tant que clé étrangère dans cette table. Les références dans la table Adresses disparaîtraient. La base de données ne permet pas cela, une fois la relation créée. Pour supprimer la Rue, la condition préalable doit être remplie, qu'elle ne soit plus référencée dans la table Adresses.

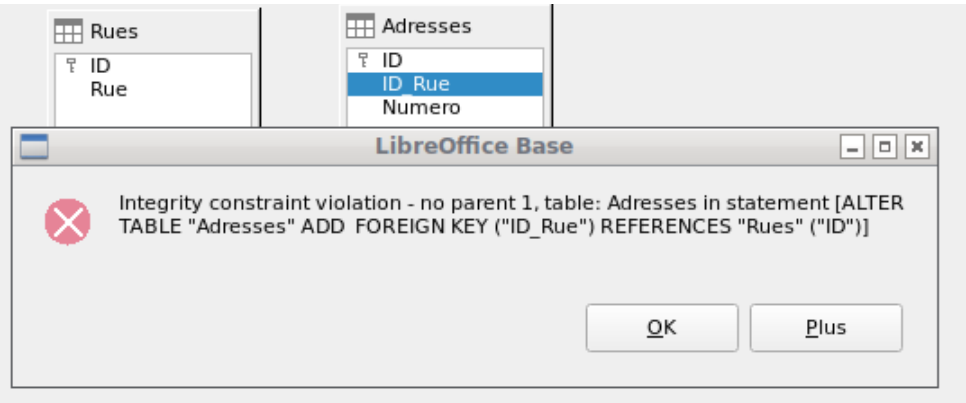
Les liens de base sont créés en utilisant **Outils > Relations**. Cela crée une ligne de connexion de la clé primaire dans une table à la clé étrangère définie dans l'autre.

Vous pouvez recevoir le message d'erreur suivant lors de la création d'un tel lien :



Ce message affiche l'erreur qui s'est produite et la commande SQL interne qui a provoqué l'erreur.

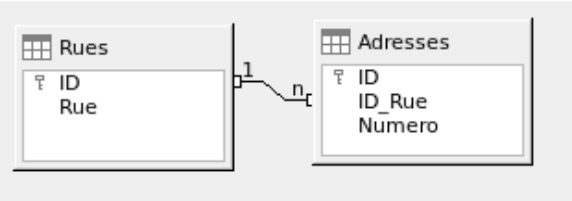
`Column types do not match in statement`—Comme la commande SQL est également affichée, la référence est clairement aux colonnes Adresses. ID_Rue et Rues. ID. À des fins de test, l'un de ces champs a été défini comme un entier, l'autre comme un petit entier. Par conséquent, aucun lien n'a pu être créé car un champ ne peut pas avoir la même valeur que l'autre.



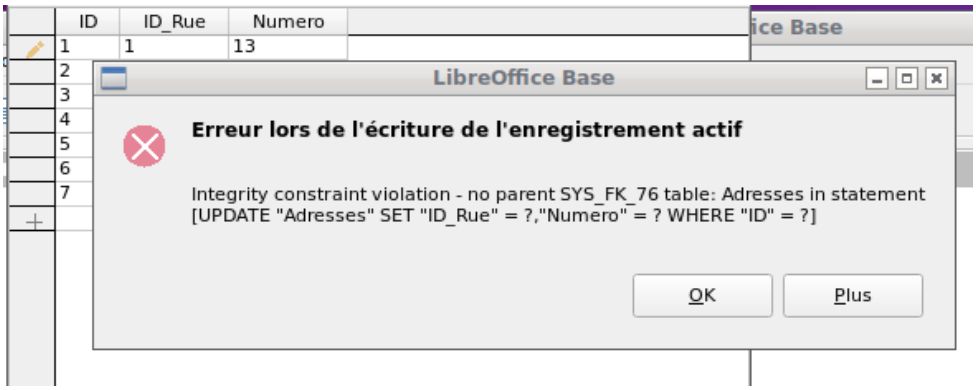
Dans ce cas, les types de colonnes correspondent. L'instruction SQL est la même que dans le premier exemple. Mais encore une fois, il y a une erreur :

`Integrity constraint violation – no parent 1, table : Adresses… –`
L'intégrité de la relation n'a pas pu être établie. Dans le champ ID_Rue de la table Adresses, il y a un numéro 1, qui n'est pas présent dans le champ ID de la table Rues. La table parent ici est Rues, car sa clé primaire est celle qui doit exister. Cette erreur est très courante lorsque deux

tables doivent être liées et que certains champs de la table avec la clé étrangère potentielle contiennent déjà des données. Si le champ de clé étrangère contient une entrée qui n'est pas présente dans la table parent (la table contenant la clé primaire), il s'agit d'une entrée non valide.



Si la liaison est effectuée avec succès et que par la suite, il y a une tentative d'entrer un enregistrement non valide similaire dans la table, vous obtenez le message d'erreur suivant :



Encore une fois, c'est une violation de l'intégrité. Base refuse d'accepter la valeur 1 pour le champ ID_Rue après que le lien a été établi, car la table Rues ne contient pas une telle valeur dans le champ ID.

Figure 51: Lien peut être édité avec clic-droit

Les propriétés d'un lien peuvent être modifiées (Cf. Figure 51)de sorte que la suppression d'un enregistrement de la table Rues mettra simultanément à NULL les entrées correspondantes dans la table Adresses.

Les propriétés présentées dans la figure 52 concernent toujours une action liée à la modification d'un enregistrement de la table contenant la clé primaire correspondante. Dans notre cas, il s'agit de la table Rues. Si la *clé primaire d'un enregistrement* de cette table est *modifiée* (*mise à jour*), les actions suivantes peuvent avoir lieu.

Aucune action

La modification de la clé primaire Rue. ID n'est pas autorisée dans ce cas, car elle romprait la relation entre les tables.

Mise à jour en cascade

Si la clé primaire Rues.ID est modifiée, la clé étrangère est automatiquement remplacée par sa nouvelle valeur. Cela garantit que la liaison n'est pas endommagée. Par exemple, si une

valeur est modifiée de 3 à 4, tous les enregistrements de la table Adresses qui contiennent la clé étrangère Adresses. ID_Rue avec la valeur 3, l'auront changée en 4.

Définir NULL

Tous les enregistrements qui contiennent cette clé primaire particulière n'auront désormais aucune entrée dans le champ de clé étrangère Adresses. ID_Rue ; le champ sera NULL.

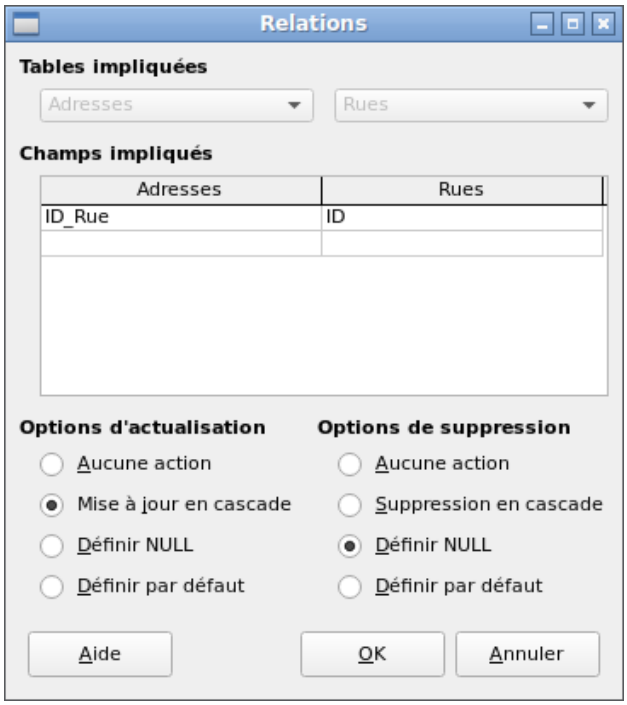


Figure 52: Edition des propriétés d'une relation

Définir par défaut

Si la clé primaire Rues.ID est modifiée, la valeur de Adresses.ID_Rue qui lui était liée à l'origine est définie sur la valeur par défaut précédemment définie. Pour cela, nous avons besoin d'une définition sans ambiguïté d'une valeur par défaut. Si la valeur par défaut est définie à l'aide de l'instruction SQL :

```
ALTER TABLE "Adresses" ALTER COLUMN "ID_Rue" SET DEFAULT 1 ;
```

la définition du lien garantit que le champ reviendra à cette valeur dans le cas d'une mise à jour. Ainsi, si la clé primaire dans la table Rues est modifiée, la clé étrangère correspondante dans la table Adresses sera définie sur 1. Cela est utile lorsqu'un enregistrement doit avoir un champ Rue, en d'autres termes ce champ ne peut pas être NULL. Mais attention : si 1 n'est pas utilisé, vous aurez créé un lien vers une valeur inexistante. Il est donc possible de détruire l'intégrité de la relation.

Avertissement

Si la valeur par défaut d'un champ de clé étrangère n'est pas liée à une valeur de clé primaire de la table étrangère, un lien vers une valeur devrait être créé, ce qui n'est pas possible. L'intégrité référentielle de la base de données serait détruite.

Il serait préférable de **ne pas** utiliser la possibilité de définir la valeur par défaut.

Si un enregistrement est *supprimé* de la table Rues, les options suivantes sont disponibles.

Aucune action

Aucune action n'a lieu. Si la suppression demandée affecte un enregistrement dans la table d'adresses, la demande sera refusée.

Suppression en cascade

Si un enregistrement est supprimé de la table Rues et que cela affecte un enregistrement de la table Adresses, cet enregistrement sera également supprimé.

Cela peut sembler étrange dans ce contexte, mais il existe d'autres structures de table dans lesquelles cela a du sens. Supposons que vous ayez une table de CD et une table qui stocke les titres sur ces CD. Désormais, si un enregistrement de la table CD est supprimé, de nombreux titres de l'autre table n'ont plus de sens, car ils ne sont plus disponibles pour vous. Dans de tels cas, une suppression en cascade a du sens. Cela signifie que vous n'avez pas besoin de supprimer tous les titres avant de supprimer le CD de la base de données.

Définir NULL

C'est la même chose que pour l'option de mise à jour.

Définir par défaut

Ceci est la même chose que pour l'option de mise à jour et nécessite les mêmes précautions.

Conseil

L'option Aucune action doit être évitée dans la plupart des cas afin d'éviter d'afficher des messages d'erreur de la base de données à l'utilisateur, car ceux-ci peuvent ne pas toujours être compréhensibles pour l'utilisateur.

Dans **Outils > Relations**, faire glisser avec la souris crée des clés étrangères qui font référence à un seul champ dans une autre table. Pour créer un lien vers une table qui a une clé primaire composite, allez dans **Outils > Relations**, puis **Insertion > Nouvelle relation**, ou utilisez le bouton correspondant. Une boîte de dialogue apparaît pour éditer les propriétés d'une relation avec un libre choix des tables disponibles.

Saisie de données dans des tables

Les bases de données constituées d'une seule table ne nécessitent généralement pas de formulaire de saisie, sauf si elles contiennent un champ pour les images. Cependant, dès qu'une table contient des clés étrangères provenant d'autres tables, les utilisateurs doivent soit se souvenir des numéros de clé à saisir, soit pouvoir consulter les autres tables simultanément. Dans de tels cas, un formulaire est utile.

Entrée à l'aide de l'interface graphique de Base

Les tables du conteneur de tables sont ouvertes en double-cliquant dessus, la saisie s'effectue en mode Tableau. Si la clé primaire est un champ à incrémentation automatique, l'un des champs visibles contiendra le texte AutoChamp. Aucune entrée n'est possible dans le champ AutoChamp.

Sa valeur assignée peut être modifiée si nécessaire, mais seulement après la validation de l'enregistrement.

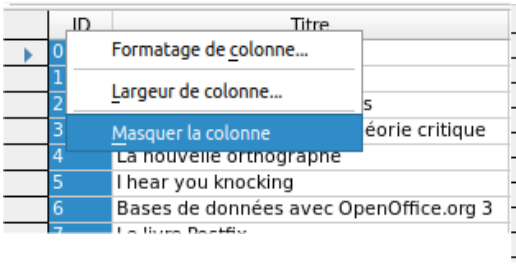


Figure 53: Saisie de table – Masquer colonne

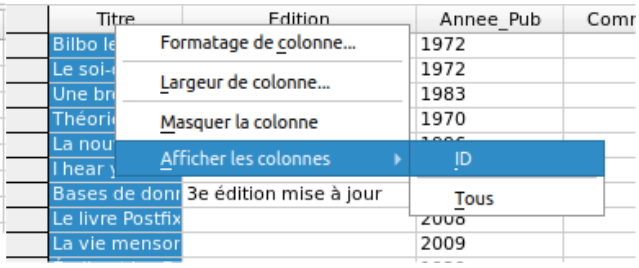


Figure 54: Saisie de table – Afficher colonne

Les colonnes individuelles de la vue des données de table peuvent être masquées (Cf.figure 53). Par exemple, si le champ de clé primaire n'a pas besoin d'être visible, cela peut être spécifié dans le tableau en vue de saisie de données en cliquant avec le bouton droit sur l'en-tête de colonne. Ce paramètre est stocké avec l'interface graphique. La colonne continue d'exister dans le tableau et peut toujours être rendue visible à nouveau (Cf. Figure 54). Il est également possible, comme avec Calc, de modifier la largeur d'une colonne avec les séparateurs de colonne.

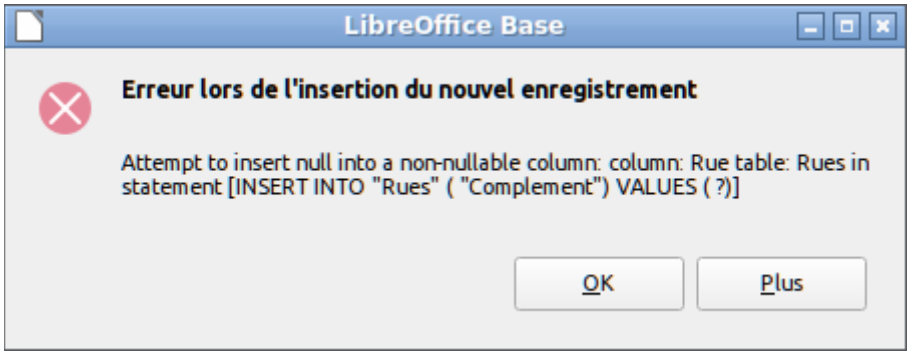
L'entrée dans le tableau s'effectue généralement de gauche à droite à l'aide du clavier avec les touches Tab ou Entrée. Vous pouvez également utiliser la souris.

Lorsque vous atteignez le dernier champ d'un enregistrement, le curseur passe automatiquement à l'enregistrement suivant. L'entrée précédente est réservée au stockage. Un stockage supplémentaire en utilisant **Fichier > Enregistrer** n'est pas nécessaire et même pas possible. Les données sont déjà dans la base de données.

Attention

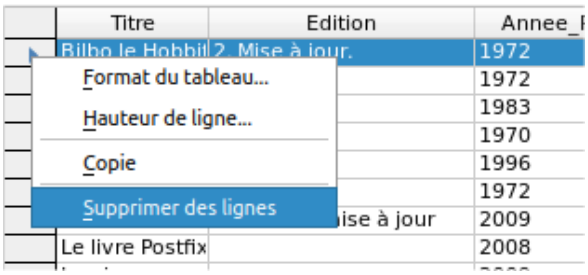
Pour HSQLDB, les données sont en mémoire de travail. elles ne seront transférée sur le disque dur qu'à la fermeture de Base, ou à l'enregistrement du fichier sur disque (malheureusement du point de vue de la sécurité des données). Si Base, pour une raison quelconque, ne ferme pas de la bonne manière, cela peut entraîner une perte de données.

Si aucune donnée n'est entrée dans un champ qui a été précédemment défini lors de la conception de la table comme obligatoire (`NOT NULL`), le message d'erreur approprié s'affiche: `Attempt to insert null into a non-nullable column…`



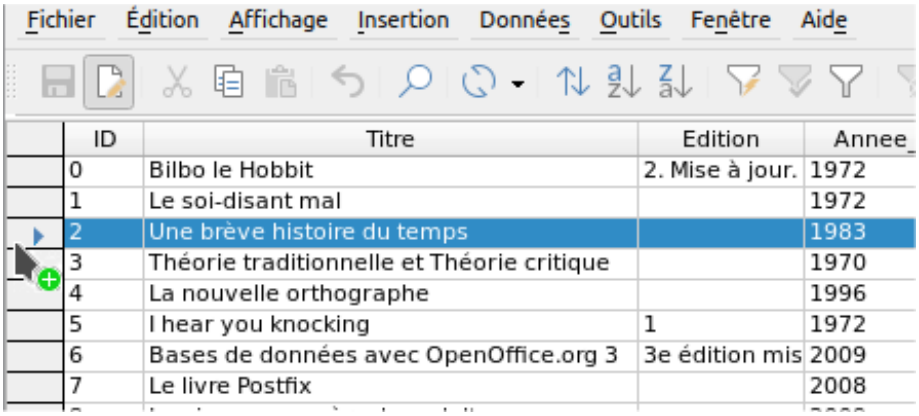
La colonne correspondante, la table et la commande SQL (telle que traduite par l'interface graphique) sont également affichées.

La modification d'un enregistrement est facile : recherchez le champ, entrez une valeur différente et quittez à nouveau la ligne.

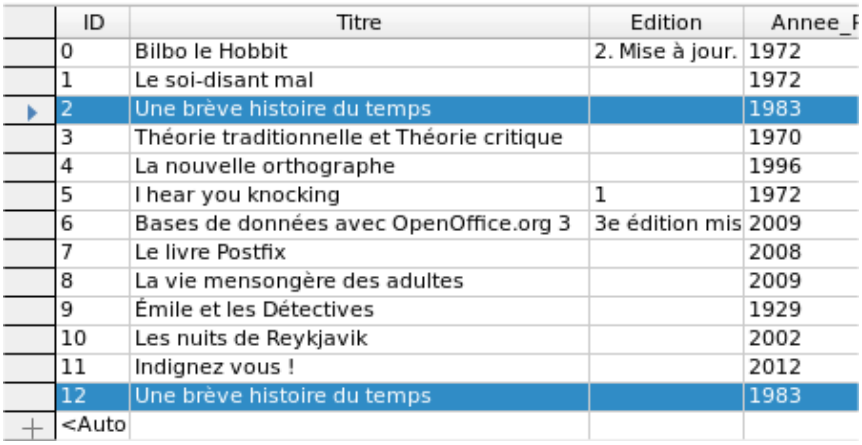


Pour supprimer un enregistrement, sélectionnez la ligne en cliquant sur son en-tête (la zone grise à gauche), cliquez avec le bouton droit et choisissez **Supprimer des lignes**.

Il existe une méthode, plutôt bien cachée, pour copier des lignes complètes. Pour que cela fonctionne, la clé primaire de la table doit être définie comme AutoChamp.



Tout d'abord, l'en-tête de ligne est cliqué avec le bouton gauche de la souris. Ensuite, maintenez le bouton enfoncé et faites glisser la souris. Le curseur se transforme en symbole avec un signe +. Cela signifie que l'enregistrement est copié dans la dernière entrée de la table.



L'enregistrement avec la clé primaire **2** est inséré en tant que nouvel enregistrement avec la nouvelle clé primaire **12**.

Si la touche Ctrl ou Maj est utilisée pour mettre en surbrillance un groupe d'enregistrements, ceux-ci seront copiés en tant que groupe.

Conseil

Les en-têtes de colonne peuvent être glissés à une largeur appropriée pour la saisie. Si cela est fait dans un tableau, Base enregistre automatiquement la nouvelle largeur de colonne dans le tableau.

Les largeurs de colonne dans les tableaux affectent celles des requêtes. Si les colonnes d'une requête sont trop étroites, les élargir n'aura qu'un effet temporaire. La nouvelle largeur ne sera pas enregistrée. Il est préférable d'élargir la colonne du tableau afin qu'elle apparaisse correctement dans les requêtes sans avoir besoin de la redimensionner.

Les fonctions Trier, Rechercher et Filtrer sont très utiles pour récupérer des enregistrements particuliers.

Tri de tables

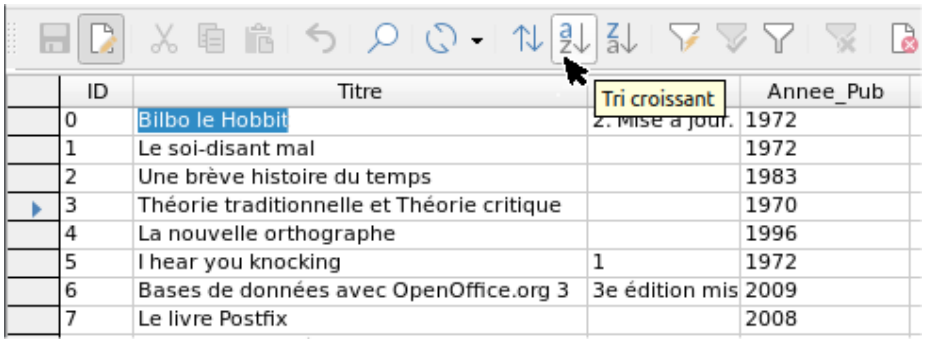


Figure 55: Tri rapide

Les boutons A>Z et Z>A permettent un tri rapide. Tout d'abord, sélectionnez un champ. Ensuite, cliquez sur le bouton correspondant au tri croissant ou décroissant, et les données sont triées par cette colonne. La figure 55 montre un tri croissant sur le champ Titre.

Le tri rapide ne triera que par une colonne. Pour trier sur plusieurs colonnes simultanément, une fonction de tri plus avancée est fournie par le bouton à gauche des boutons de tri rapide (C. Figure 56):

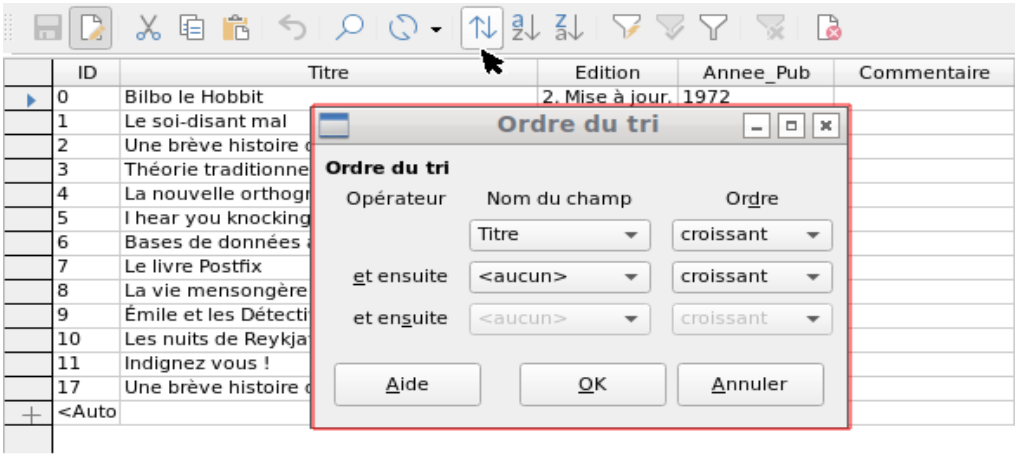
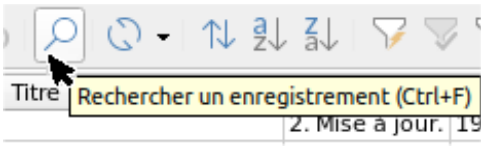


Figure 56: Tri sur plus d'une colonne

Le nom de champ de la colonne et l'ordre de tri actuel sont sélectionnés. Si un tri rapide précédent a été effectué, la première ligne contiendra déjà le nom de champ et l'ordre de tri correspondants.

Recherche dans les tables



Le bouton **Rechercher un enregistrement** est une méthode simple pour rechercher des enregistrements dans une grande table. Cependant, la fonction de recherche est très lente pour les grandes bases de données, car la recherche n'utilise pas de commande SQL dans la base de données. Pour une recherche plus rapide, au lieu d'utiliser Rechercher un enregistrement, utilisez une requête. Pour éliminer les modifications fréquentes de la requête, elle peut être conçue pour s'exécuter à l'aide de paramètres. Reportez-vous au chapitre 5, Requêtes, dans la section "Utilisation des paramètres dans les requêtes".

Conseil

Avant de rechercher, assurez-vous que les colonnes que vous rechercherez sont suffisamment larges pour afficher correctement les enregistrements que vous trouverez. La fenêtre de recherche reste au premier plan et vous ne pourrez pas corriger les paramètres de largeur de colonne dans le tableau sous-jacent. Pour accéder à la table, vous devez interrompre la recherche.

Le bouton **Rechercher** un enregistrement remplit automatiquement le terme de recherche avec le contenu du champ à partir duquel il a été appelé.

Pour rendre la recherche efficace, la zone de recherche doit être limitée autant que possible. Il serait inutile de rechercher le texte ci-dessous figure 57 dans le champ Titre du champ Auteur. Au lieu de cela, le nom de champ Titre est déjà suggéré comme nom de champ unique.

D'autres paramètres de recherche peuvent faciliter les choses grâce à des combinaisons spécifiques. Vous pouvez utiliser les espaces réservés SQL normaux ("_" pour un caractère

variable, "%" pour un nombre arbitraire de caractères variables, ou "\" comme caractère d'échappement pour permettre la recherche de ces caractères spéciaux eux-mêmes).

Les expressions régulières sont décrites en détail dans l'aide de LibreOffice. En dehors de cela, l'aide disponible pour ce module est plutôt rare.

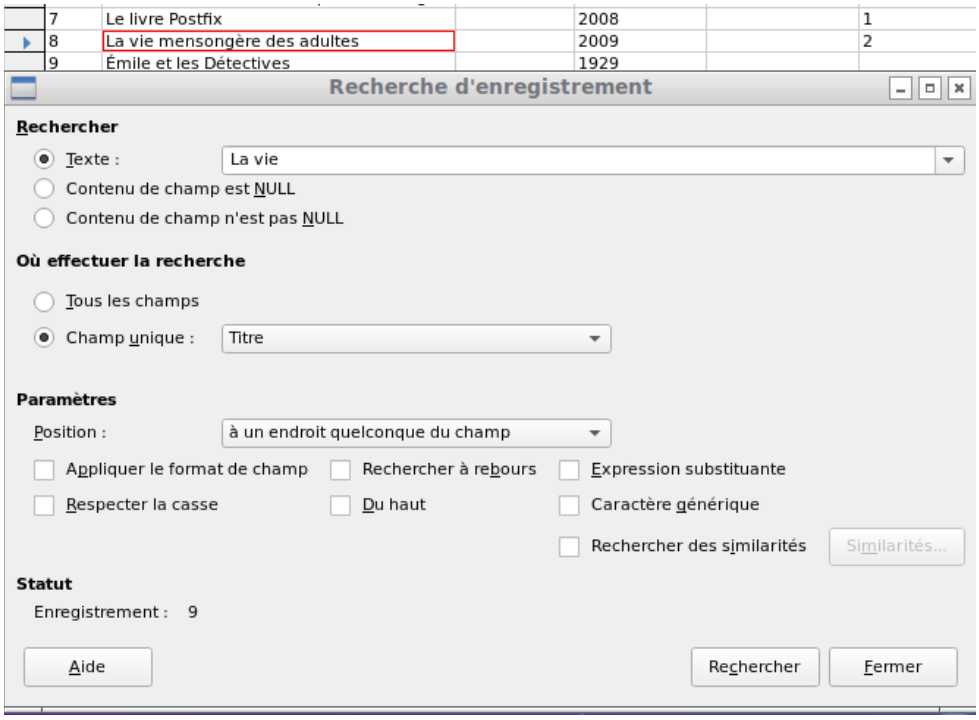


Figure 57: Masque d'entrée pour une recherche d'enregistrement

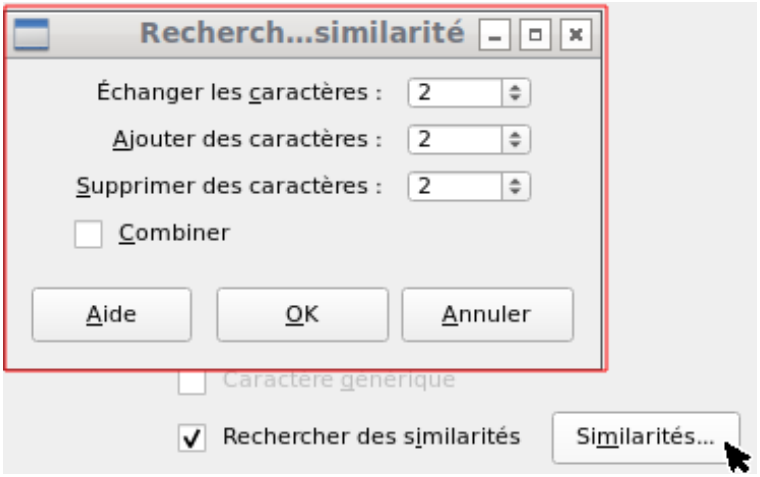


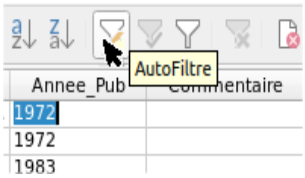
Figure 58: Limiter la recherche de similarité

La fonction de recherche de similarité figure 58est utile lorsque vous devez exclure des fautes d'orthographe. Plus les valeurs que vous définissez sont élevées, plus d'enregistrements seront affichés dans la liste finale.

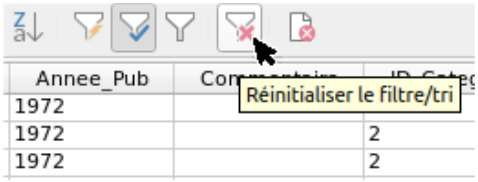
Ce module de recherche est le plus adapté aux personnes qui savent, grâce à une utilisation régulière, exactement comment obtenir un résultat donné. La plupart des utilisateurs sont plus susceptibles de réussir à trouver des enregistrements à l'aide d'un filtre.

Le chapitre 4 de ce manuel décrit l'utilisation des formulaires pour la recherche et comment l'utilisation de SQL et des macros peut effectuer une recherche par mot-clé.

Filtrer les tables



Vous pouvez filtrer rapidement un tableau à l'aide du filtre automatique **AutoFiltre**. Placez le curseur dans la cellule d'un champ, et un clic sur l'icône amène le filtre à reprendre le contenu de ce champ. Seuls les enregistrements pour lesquels le champ choisi a le même contenu que la cellule sélectionnée sont affichés. La figure ci-dessous montre le filtrage en fonction d'une entrée dans la colonne **Annee_Pub**.

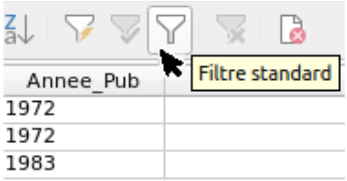


Le filtre est actif, comme indiqué par l'icône de filtre avec une coche verte. Le symbole du filtre est affiché enfoncé. Ce bouton est une bascule, donc si vous cliquez à nouveau, le filtre continue d'exister, mais tous les enregistrements sont maintenant affichés. Donc, si vous le souhaitez, vous pouvez toujours cliquer dessus pour revenir à l'état filtré.

Cliquez sur l'icône **Réinitialiser le Filtre/tri** à l'extrême droite pour supprimer tous les filtres et tris existants. Les filtres deviennent inactifs et ne peuvent plus être récupérés avec leurs anciennes valeurs.

Conseil

Vous pouvez toujours saisir des enregistrements normalement dans une table filtrée ou dans une table restreinte par une recherche. Ils restent visibles dans la vue du tableau jusqu'à ce que le tableau soit mis à jour en cliquant sur le bouton **Appliquer le filtre** (coche verte).



L'icône **Filtre standard** ouvre une boîte de dialogue dans laquelle vous pouvez filtrer en utilisant plusieurs critères simultanés, similaires au tri. Si le filtre automatique est utilisé, la première ligne du filtre standard affichera déjà ce critère de filtre existant.

Figure 59: Filtrage par données multiples à l'aide du filtre standard

Le filtre standard fournit de nombreuses fonctions de filtrage des données SQL (Cf.figure 59). Les commandes SQL suivantes sont disponibles.

Condition	***Description***
=	Égalité exacte, correspond à like, mais sans espaces réservés supplémentaires
<>	Inégal, différent
<	Inférieur à
<=	Inférieur ou égal
>	Supérieur à
>=	Supérieur ou égal
comme	Pour le texte, écrit entre guillemets (") ; "_" pour un caractère variable, "%" pour un nombre arbitraire de caractères variables, en SQL LIKE.
différent de	Opposé de comme, en SQL `NOT LIKE`
nulle	Aucune entrée, pas même un caractère d'espace. En SQL, cela est exprimé par le terme `NULL`
Non nulle	Opposé de nulle, en SQL `NOT NULL`

Avant qu'un critère de filtre puisse être combiné avec un autre, la ligne suivante doit avoir au moins un nom de champ sélectionné. Dans la figure 59, le mot – *aucun(e)* – est affiché à la place d'un nom de champ, la combinaison n'est donc pas active. Les opérateurs de combinaison disponibles sont `AND` et `OR`.

Le nom de champ peut être un nouveau nom de champ ou un nom précédemment sélectionné.

Même pour de grandes collections de données, le nombre d'enregistrements récupérés peut être réduit à un ensemble gérable avec un filtrage habile utilisant ces trois conditions possibles.

Dans le cas des formulaires de filtrage également, il existe d'autres possibilités (décrites au chapitre 4) qui ne sont pas fournies par l'interface graphique.

Saisie directe à l'aide de SQL

La saisie directe de données à l'aide de SQL est utile pour saisir, modifier ou supprimer plusieurs enregistrements avec une seule commande.

Saisie de nouveaux enregistrements

```
INSERT INTO "Nom_Table" [("Nom_Champ" [,…])]
{ VALUES("Valeur champ" [,…]) | <Formule-Select>} ;
```

Si aucun "Nom_Champ" n'est spécifié, tous les champs doivent être remplis et dans le bon ordre (comme indiqué dans la définition de la table). Cela inclut le champ de clé primaire automatiquement incrémenté, le cas échéant. Les valeurs saisies peuvent également être le résultat d'une requête (<Formule-Select>). Des informations plus précises sont données ci-dessous.

```
INSERT INTO "Nom_Table" ("Nom_Champ") VALUES ('Test') ;
CALL IDENTITY() ;
```

Dans le tableau, dans la colonne Nom_Champ, la valeur Test est insérée. L'ID de champ de clé primaire incrémenté automatiquement n'est pas touché. La valeur correspondante pour ID doit être créée séparément à l'aide de `CALL IDENTITY()`. Ceci est important lorsque vous utilisez des macros, afin que la valeur de ce champ clé puisse être utilisée ultérieurement.

```
INSERT INTO "Nom_Table" ("Nom_Champ") SELECT "Nom_Autre_Champ" FROM
"Nom_Autre_Table";
```

Dans la première table, autant de nouveaux enregistrements sont insérés dans Nom_Champ que dans la colonne Nom_Autre_Champ de la seconde table. Naturellement, une formule de sélection peut être utilisée ici pour limiter le nombre d'entrées.

Édition d'enregistrements existants

```
UPDATE "Nom_Table" SET "Nom_Champ" = <Expression> [,…] [WHERE <Expression>] ;
```

Lorsque vous modifiez plusieurs enregistrements à la fois, il est très important de vérifier attentivement la commande SQL que vous entrez. Supposons que tous les élèves d'une classe doivent être remontés d'un an :

```
UPDATE "Nom_Table" SET "Annee" = "Annee"+1
```

Rien de plus rapide : tous les enregistrements de données sont modifiés avec une seule commande. Mais imaginez que vous devez maintenant déterminer quels élèves n'auraient pas dû être affectés par ce changement. Il aurait été plus simple de cocher un champ Oui/Non pour la répétition d'une année et ensuite de ne remonter que les étudiants pour lesquels ce champ n'a pas été coché :

```
UPDATE "Nom_Table" SET "Annee" = "Annee"+1 WHERE "Repetition" = FALSE
```

Ces conditions ne fonctionnent que lorsque le champ en question ne peut prendre que les valeurs `FALSE` et `TRUE` ; il ne peut pas être `NULL`. Ce serait plus sûr si la condition était formulée comme `WHERE "Repetition" <> TRUE`.

Si vous souhaitez par la suite qu'une valeur par défaut soit entrée dans un champ particulier là où il est vide, vous pouvez le faire avec la commande :

```
UPDATE "Nom_Table" SET "Nom_Champ" = 1 WHERE "Nom_Champ" IS NULL
```

Vous pouvez modifier plusieurs champs à la fois en leur attribuant directement des valeurs. Supposons qu'une table pour les livres inclue les noms de leurs auteurs. On découvre qu'Erich Kästner a souvent été inscrit comme Eric Käschtner.

```
UPDATE "Livres" SET "Prenom_Auteur" = 'Erich', "Nom_Auteur" = 'Kästner'
WHERE "Prenom_Auteur" = 'Eric' AND "Nom_Auteur" = 'Käschtner'
```

D'autres étapes de calcul sont également possibles avec Update. Si, par exemple, des marchandises coûtant plus de 150,00 € doivent être incluses dans une offre spéciale avec une réduction de 10 %, cela peut être effectué comme suit :

```
UPDATE "Nom_Table" SET "Prix" = "Prix"*0.9 WHERE "Prix" >= 150
```

Lorsque vous choisissez le type de données CHAR, le champ a une largeur fixe. Si nécessaire, le texte est complété par des caractères nuls. Si vous le convertissez en VARCHAR, ces caractères nuls restent. Pour les supprimer, utilisez la fonction de découpe à droite :

```
UPDATE "Nom_Table" SET "Nom_Champ" = RTRIM("Nom_Champ")
```

Supprimer des enregistrements existants

```
DELETE FROM "Nom_Table" [WHERE <Expression>] ;
```

Sans l'expression conditionnelle, la commande

```
DELETE FROM "Nom_Table"
```

supprime tout le contenu de la table.

Pour cette raison, il est préférable que la commande soit plus spécifique. Par exemple, si la valeur de la clé primaire est donnée, seul cet enregistrement précis sera supprimé.

```
DELETE FROM "Nom_Table" WHERE "ID" = 5 ;
```

Si, dans le cas d'un prêt, l'enregistrement média doit être supprimé lors du retour de l'article, cela peut être fait en utilisant

```
DELETE FROM "Nom_Table" WHERE NOT "Date_Retour" IS NULL ;
```

ou alternativement avec

```
DELETE FROM "Nom_Table" WHERE "Date_Retour" IS NOT NULL ;
```

Importer des données à partir d'autres sources

Parfois, il y a des ensembles de données complets dans un autre programme qui doivent être importés dans Base via le presse-papiers. Cela peut impliquer la création d'une nouvelle table ou l'ajout d'enregistrements à une table existante.

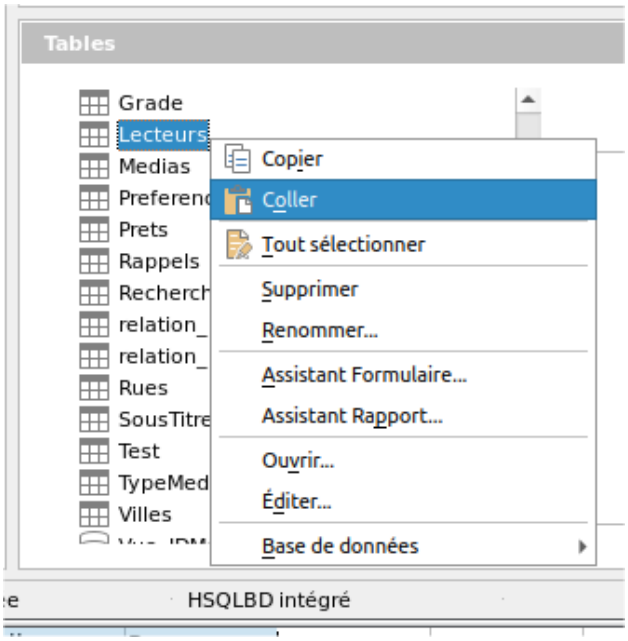
Note

Pour importer des données à l'aide du presse-papiers, le format de données doit être lisible par Base. Ce sera toujours le cas pour les fichiers de données ouverts dans LibreOffice.

Par exemple, si les tables d'une base de données externe doivent être lues dans un fichier *.odb, cette base de données doit d'abord être ouverte dans LibreOffice ou enregistrée auprès de LibreOffice en tant que source de données. Reportez-vous à la section "Accès aux bases de données externes" au chapitre 2, Création d'une base de données.

	A	B	C
16	14	Adele	Scott
17	15	Ivan	Tilelapièce
18	16	Vladimir	Pourlavesse
19	17	Teddy	Ferrant
20	18	Cécile	Cligne
21	19	Bruno	Zieuvair
22	20	Dolly	Prane
23			

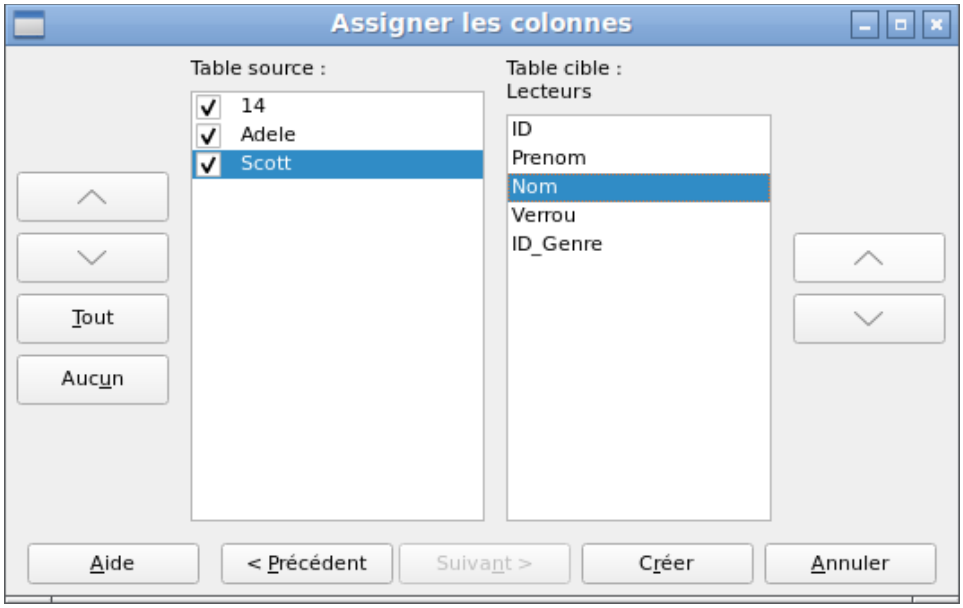
Ici, un petit exemple de tableau a été copié du tableur Calc dans le presse-papiers. Ensuite, il est collé dans le conteneur Table de la base. Bien sûr, cela aurait également pu être fait en le sélectionnant avec le bouton gauche de la souris, puis en le faisant glisser.



Dans le conteneur Table, cliquez avec le bouton droit pour ouvrir le menu contextuel de la table à laquelle les enregistrements doivent être ajoutés.

Ajout d'enregistrements importés à une table existante

Le nom de la table apparaît dans l'assistant d'importation. En même temps, *Ajouter des données* est sélectionné. *Utiliser la première ligne comme noms de colonnes* peut être requis ou non, selon votre version de LibreOffice. Si les enregistrements doivent être ajoutés, aucune définition de données n'est requise. Une clé primaire doit également être disponible.



Les colonnes de la table source Calc et de la table de destination dans Base ne doivent pas nécessairement concorder dans leur séquence, leurs noms ou leur nombre global. Seuls les éléments sélectionnés sur le côté gauche sont transférés. La correspondance entre les tables source et destination doit être ajustée à l'aide des flèches de chaque côté.

Ceci termine l'importation.

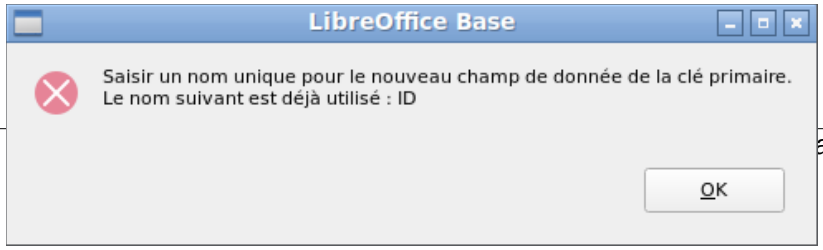
L'importation peut entraîner des problèmes si :

- Les champs de la table de destination nécessitent une entrée obligatoire, mais que la table source ne fournit aucune donnée pour eux.
- Les définitions de champ dans la table de destination ne sont pas cohérentes avec celles de la table source (par exemple, un nom doit être entré dans un champ numérique, ou le champ de destination contient trop peu de caractères pour les données).
- La table source fournit des données incohérentes avec celles de la table de destination, par exemple des valeurs non uniques pour les clés primaires ou d'autres champs définis comme uniques.

Créer une nouvelle table pour les données importées

Lorsque l'assistant d'importation est lancé, le nom de la table précédemment sélectionnée apparaît automatiquement. Ce nom de table doit être changé si vous créez une nouvelle table, car il est interdit d'avoir une table avec le même nom qu'une table existante. Le nom de cette table est Noms. La définition et les données doivent être transférées. La première ligne doit être utilisée comme en-tête de colonne.

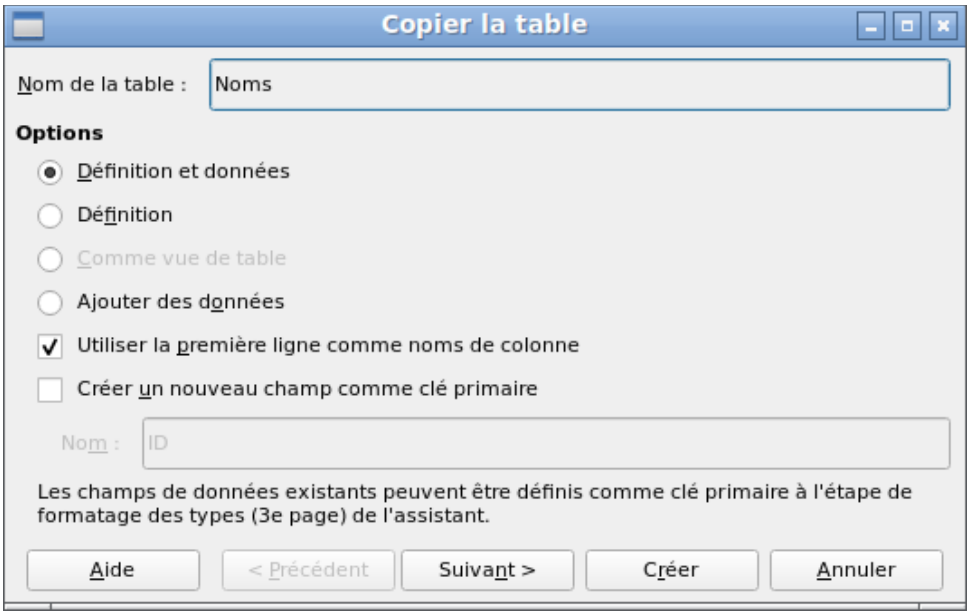
À ce stade, vous pouvez créer un nouveau champ supplémentaire pour une clé primaire. Le nom de ce champ ne doit pas exister en tant qu'en-tête de colonne dans la table Calc. Sinon, vous obtenez le message d'erreur :



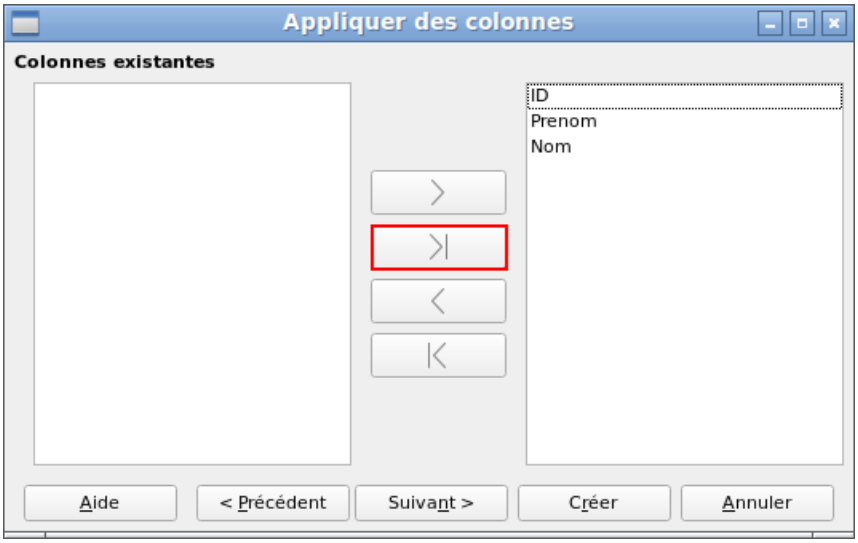
Malheureusement, ce message n'explique pas assez correctement la situation.

Si vous souhaitez qu'un champ existant serve de clé primaire, ne sélectionnez pas Créer une clé primaire. Dans ce cas, vous établissez votre champ de clé primaire sur la troisième page de la boîte de dialogue Assistant.

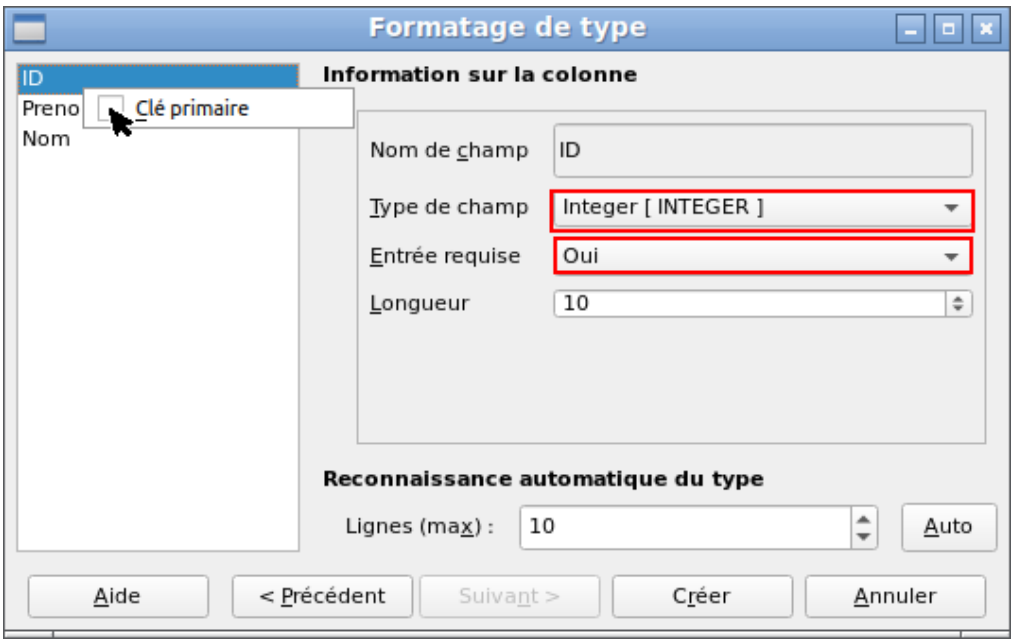
Lors de l'importation, la définition de la table et les données sont transférées.



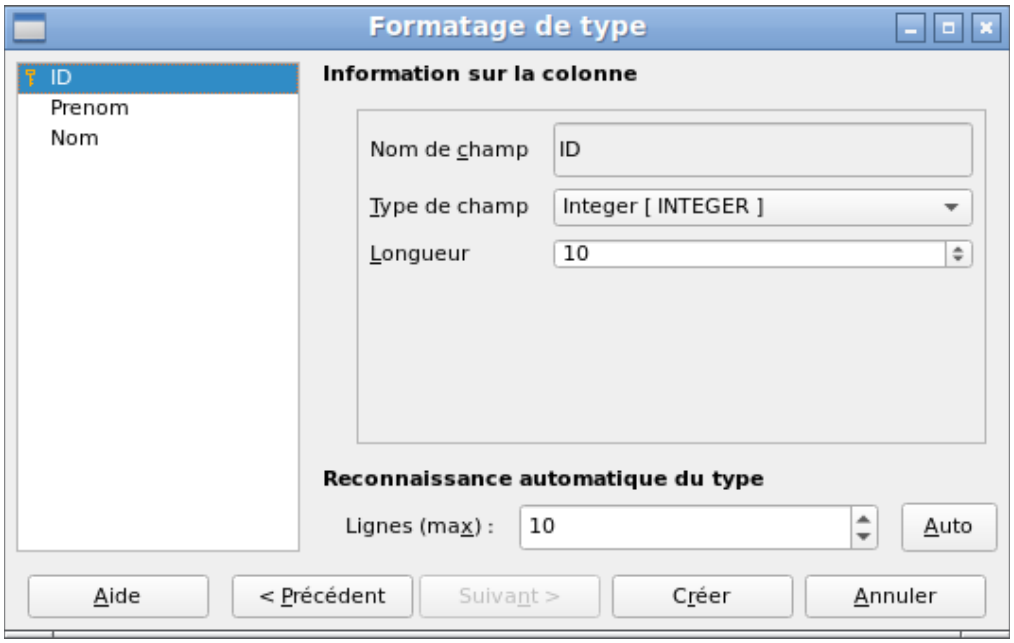
Toutes les colonnes disponibles sont transférées.



La mise en forme des types de tables nécessite souvent des ajustements. Habituellement, les champs ont été prédéfinis comme des champs de texte de très grande taille. Les champs numériques et de date doivent donc être réinitialisés en utilisant **Formatage de type > Informations sur la colonne > Type de champ**. Dans le cas des nombres décimaux, vous devrez vérifier le nombre de décimales.



L'option de choisir une clé primaire est présente, quelque peu obscurément, dans le menu contextuel du champ qui doit la contenir. Dans cet exemple, le champ ID a été formaté de manière à permettre son utilisation comme clé primaire. Cela doit maintenant être défini explicitement à l'aide du menu contextuel du nom de champ, si une clé primaire n'a pas été créée en tant que champ supplémentaire dans la fenêtre Copier la table de l'assistant.



Lorsque vous cliquez sur le bouton **Créer**, la table est créée et remplie avec les données copiées. La nouvelle clé primaire n'est pas une clé AutoChamp. Pour en créer une, la table doit être ouverte pour modification. Vous pouvez ensuite effectuer d'autres opérations de formatage.

Fractionnement des données lors de l'importation

Parfois, les données sources ne sont pas disponibles sous la forme souhaitée. Les adresses, par exemple, sont souvent saisies dans les feuilles de calcul comme un seul champ, y compris la ville

et le code postal. Lors de leur importation, vous souhaiterez peut-être placer ces éléments dans une table séparée, qui pourra ensuite être liée à la table principale.

Voici un moyen possible de créer directement cette relation :

1. La table complète avec toutes les informations d'adresse est importée dans Base sous forme de table appelée Adresses. Voir les chapitres précédents pour plus de détails.
2) Les champs CodePostal et Ville sont lus avec une requête (en découpant le champ d'adresse), copiés et stockés dans une table Ville_CodePostal distincte. Pour cela, un champ ID est ajouté et spécifié comme clé primaire avec AutoChamp.

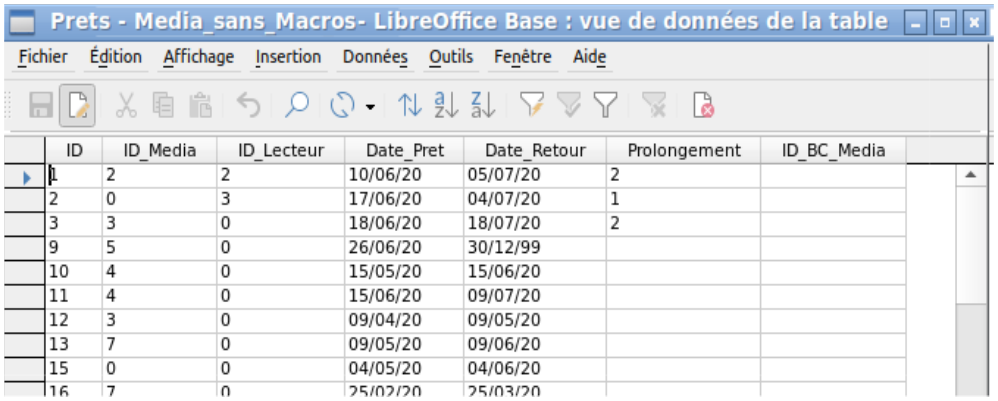
   Voici la requête:
   ```
   SELECT DISTINCT "CodePostal", "Ville" FROM "Adresses"
   ```
3. Un nouveau champ appelé ID_CodePostal est ajouté à la table Adresses.
4. À l'aide du menu **Outils > SQL**, une mise à jour est effectuée pour cette table:
   ```
   UPDATE "Adresses" AS "a" SET "a"."ID_CodePostal" = (SELECT "ID" FROM "Ville_CodePostal" WHERE "CodePostal"||"Ville" = "a"."CodePostal"||"a"."Ville")
   ```
5. La table Adresses est ouverte pour modification et les champs Code postal et Ville sont supprimés. Cette modification est enregistrée et la table est à nouveau fermée.

Cela sépare les tables de sorte qu'une relation 1:n peut être créée entre la table Ville_CodePostal et la table Adresses. Cette relation est définie à l'aide du menu **Outils > Relations.**

Pour plus d'informations sur le code SQL, reportez-vous également au Chapitre 5, Requêtes.

Problèmes avec ces méthodes de saisie de données

La saisie utilisant une table seule ne tient pas compte des liens vers d'autres tables. Cela ressort clairement d'un exemple de prêt média.



ID	ID_Media	ID_Lecteur	Date_Pret	Date_Retour	Prolongement	ID_BC_Media
1	2	2	10/06/20	05/07/20	2	
2	0	3	17/06/20	04/07/20	1	
3	3	0	18/06/20	18/07/20	2	
9	5	0	26/06/20	30/12/99		
10	4	0	15/05/20	15/06/20		
11	4	0	15/06/20	09/07/20		
12	3	0	09/04/20	09/05/20		
13	7	0	09/05/20	09/06/20		
15	0	0	04/05/20	04/06/20		
16	7	0	25/02/20	25/03/20		

La table Prets se compose de clés étrangères pour l'article prêté (ID_Media) et du lecteur correspondant (ID_Lecteur) ainsi que d'une date de prêt (Date_Pret). Dans la table, nous devons donc saisir au moment du prêt deux valeurs numériques (numéro de support et numéro de lecteur) et une date. La clé primaire est automatiquement saisie dans le champ ID. Que le lecteur corresponde réellement au nombre n'apparaît pas à moins qu'une deuxième table pour les lecteurs ne soit ouverte en même temps. Il n'est pas non plus évident de savoir si l'article a été prêté avec le bon numéro. Ici, le prêt doit reposer sur l'étiquette de l'article ou sur une autre table ouverte.

Tout cela est beaucoup plus facile à réaliser en utilisant des formulaires. Ici, les utilisateurs et les médias peuvent être recherchés à l'aide des contrôles de zone de liste. Dans les formulaires, les

noms d’utilisateur et d’élément sont visibles et leurs identifiants numériques sont masqués. De plus, un formulaire peut être conçu de telle sorte qu’un utilisateur puisse être sélectionné en premier, puis une date de prêt, et chaque ensemble de supports se voit attribuer cette date par numéro. Ailleurs, ces numéros peuvent être rendus visibles avec les descriptions de médias exactement correspondantes.

L’entrée directe dans les tables n’est utile que pour les bases de données avec des tables simples. Dès que vous avez des relations entre les tables, un formulaire spécialement conçu est préférable. Dans les formulaires, ces relations peuvent être mieux gérées en utilisant des sous-formulaires ou des champs de liste.

Guide Base

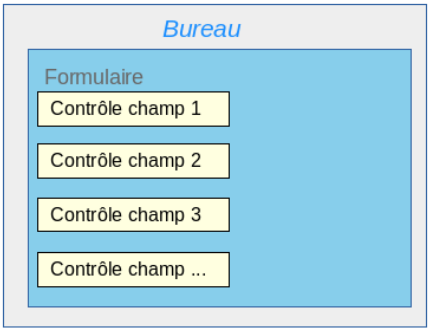
Chapitre 4
Formulaires

Les formulaires facilitent la saisie des données

Les formulaires sont utilisés lorsque la saisie directe dans une table n'est pas pratique, pour détecter rapidement les erreurs de saisie de données, ou lorsque trop de tables rendent la gestion directe des données impossible.

Note

Un formulaire dans Base est une structure invisible pour l'utilisateur. Il sert à permettre l'accès à la base de données. Ce qui est visible pour l'utilisateur est l'ensemble des contrôles, qui servent à la saisie ou à l'affichage de texte, de nombres, etc. Ces contrôles sont divisés par l'interface graphique en différents types.



Le terme *Formulaire* a deux significations. Il peut représenter tout le contenu de la fenêtre de saisie qui est utilisée pour gérer les données d'une ou plusieurs tables. Une telle fenêtre peut contenir un ou plusieurs formulaires principaux, et chacun de ceux-ci peut contenir des sous-formulaires. Le mot formulaire est également utilisé pour ces zones partielles. Il devrait être clair à partir du contexte quel sens est voulu afin d'éviter les malentendus.

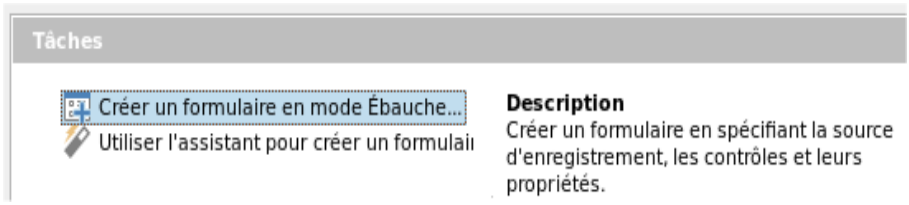
Créer des formulaires

Le moyen le plus simple de créer un formulaire consiste à utiliser l'Assistant Formulaire. Son utilisation pour créer un formulaire est décrite au chapitre 8, *Débuter avec Base*, dans le Guide de mise en route. Ce chapitre explique également comment vous pouvez modifier davantage le formulaire après avoir utilisé l'assistant.

Ce guide présente la création d'un formulaire sans utiliser l'assistant. Il décrit également les propriétés des différents types de contrôles dans un formulaire.

Un formulaire simple

Nous commençons par utiliser la tâche **Créer un formulaire en mode Ébauche** dans la zone Formulaires de la fenêtre principale de base.


Tâches
Créer un formulaire en mode Ébauche...
Utiliser l'assistant pour créer un formulair
Description
Créer un formulaire en spécifiant la source d'enregistrement, les contrôles et leurs propriétés.

Cela appelle l'éditeur de formulaire et le formulaire est affiché dans la fenêtre Vue de création (Figure 60).

La barre d'outils Contrôles de formulaire est ancrée sur le côté gauche. La barre d'outils Ébauche de formulaire (Figure 61) est ancrée en bas. Si ces barres d'outils n'apparaissent pas automatiquement, utilisez **Affichage >Barres d'outils** pour les afficher. Sans ces barres d'outils, il n'est pas possible de créer un formulaire.

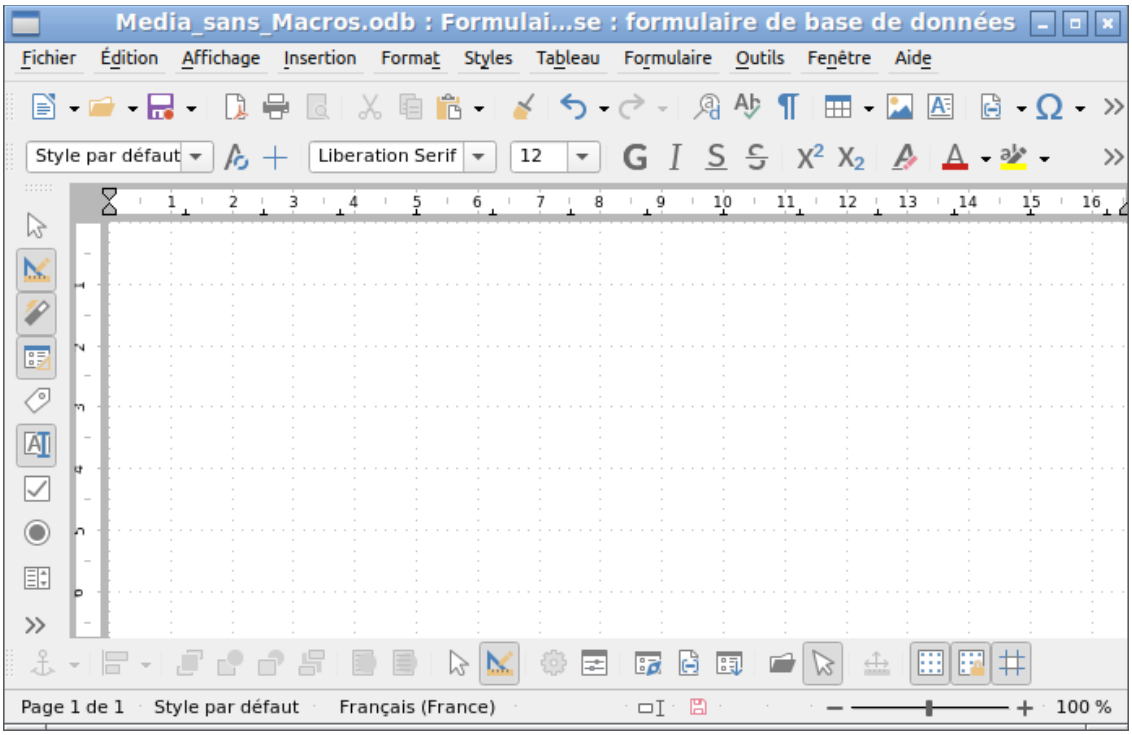


Figure 60: Formulaire affiché en mode Création

La zone vide montre une grille de points. Cette grille vous aide à positionner les contrôles avec précision, en particulier les uns par rapport aux autres. Les symboles à l'extrémité droite de la barre d'outils Création de formulaire indiquent que la grille est visible et active. Les trois derniers symboles (Afficher la grille, Aligner…), doivent être visibles et actifs. Si tous ne le sont pas, cliquez sur ceux qui ne le sont pas.

Barres d'outils pour la conception (Ébauche) de formulaires

Un formulaire est créé sur la page vide. Ceci peut être fait de deux façons :

- Utilisez le Navigateur de formulaires pour configurer un formulaire, ou
- Concevez les contrôles de formulaire et configurez le formulaire à l'aide du menu contextuel.

Configurer un formulaire avec le Navigateur de formulaires

Pour afficher le navigateur de formulaires, cliquez sur le bouton Navigateur de formulaires (illustré à la Figure 61). Une fenêtre apparaît (Figure 62), elle, n'affiche qu'un seul dossier, intitulé Formulaires. Il s'agit du niveau le plus élevé de la zone que nous éditons. Plusieurs formulaires peuvent être accueillis ici.

Figure 61: Barre d'outils Ébauche (conception) de formulaires.

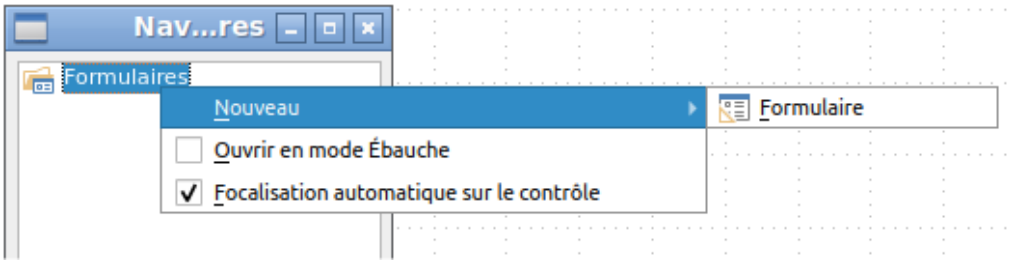


Figure 62: Utilisation du navigateur de formulaires pour créer un nouveau formulaire

Dans le navigateur de formulaires (Figure 62), cliquez avec le bouton droit sur Formulaires pour ouvrir un menu contextuel. Choisissez **Nouveau >Formulaire** pour créer un nouveau formulaire. Les autres choix du menu contextuel (Ouvrir en mode Ébauche et Focalisation automatique sur le contrôle) correspondent aux boutons de la Figure 63, nous en discuterons plus tard.

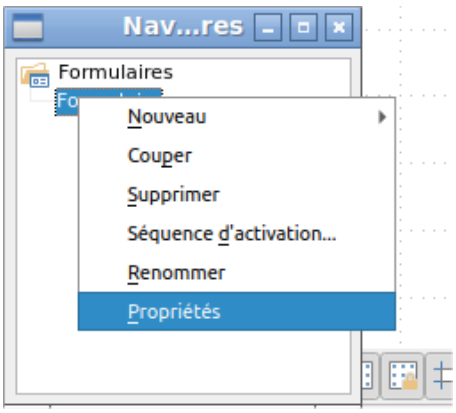
Note

Pour lancer un formulaire avec le curseur dans le premier champ, utilisez l'option Focalisation automatique sur le contrôle. Le premier élément est déterminé par la séquence d'activation du formulaire.

Malheureusement, il y a actuellement un bogue (bogue 87290) dans cette fonction. Si un formulaire contient un contrôle de table, le focus est défini automatiquement sur le premier champ de ce contrôle. Curieusement, ce bogue est corrigé si, après avoir choisi le Focalisation automatique sur le contrôle, vous changez la langue de l'interface utilisateur.

Le formulaire porte le nom par défaut Formulaire. Vous pouvez modifier ce nom immédiatement ou ultérieurement. Cela n'a aucune signification sauf si vous devez accéder à une partie du formulaire à l'aide de macros. La seule chose dont vous devez vous assurer est que deux éléments portant le même nom n'apparaissent pas au même niveau dans l'arborescence des dossiers.

Le menu contextuel du formulaire (illustré ci-dessous) permet de créer des propriétés de formulaire.



Créer un formulaire à l'aide d'un champ de formulaire

La barre d'outils Contrôles de formulaire (Figure 63) contient les boutons avec lesquels les contrôles (étiquette plus champ) peuvent être créés. Tous les boutons ne sont pas directement visibles sur le côté gauche du formulaire. Cliquez sur le bouton du bas (**»**) pour voir le reste des boutons visibles. Pour voir tous les boutons, visibles ou non, cliquez avec le bouton droit de la souris sur n'importe quelle partie de la barre d'outils et sélectionnez **Boutons visibles**. Les boutons visibles sont cochés (icône grisée), les non affichés ne le sont pas (icône sur fond blanc) (Figure 64).

Cela permet de sélectionner les boutons que l'utilisateur souhaite utiliser régulièrement tout en supprimant les indésirables. Par exemple, si le bouton Contrôle de table n'est pas visible, mais peut être activé en le localisant d'abord dans la liste de tous les boutons, puis en le cochant.

Figure 63: Contrôles de formulaire

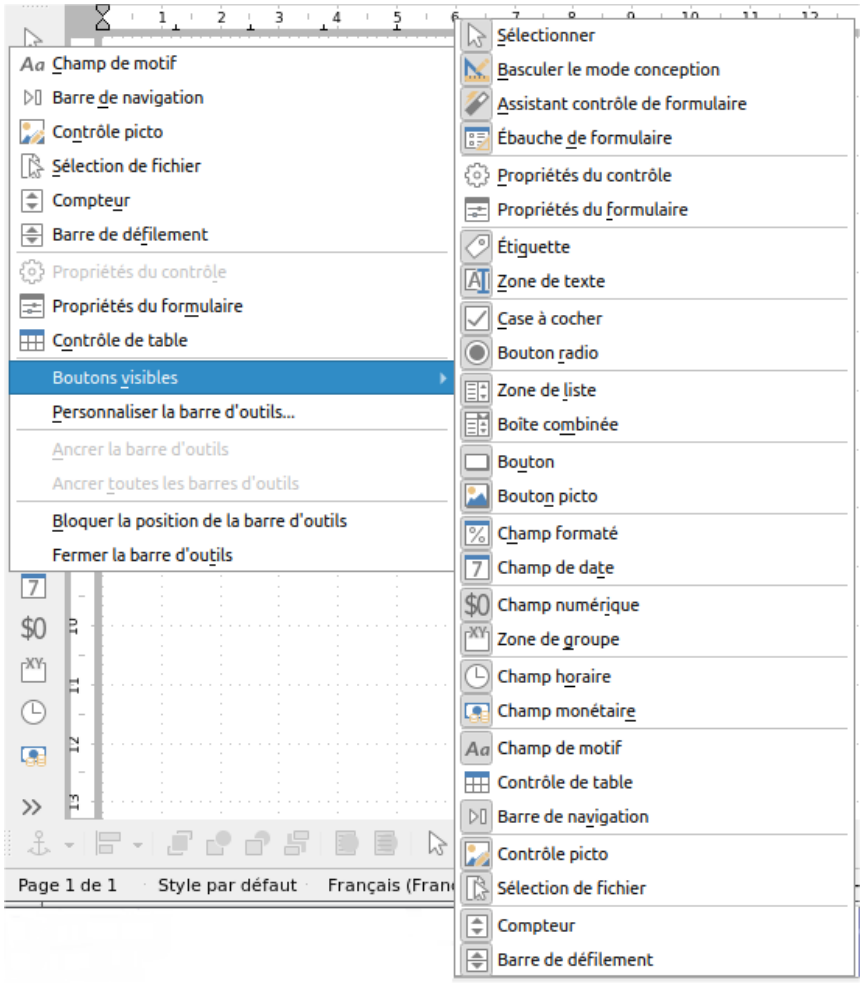


Figure 64: Choix des boutons visibles

Lorsque vous sélectionnez un contrôle de formulaire, vous créez automatiquement un formulaire. Par exemple, supposons que vous choisissiez un champ de texte : le curseur change de forme et une forme rectangulaire peut être dessinée sur la surface blanche du formulaire. Ensuite, sur la surface pointillée du formulaire, un champ de texte apparaît.

Vous pouvez maintenant créer le formulaire en cliquant avec le bouton droit de la souris et en utilisant le menu contextuel du contrôle (Figure 65). Sur cette figure, on a choisi de dessiner un champ d'étiquette.

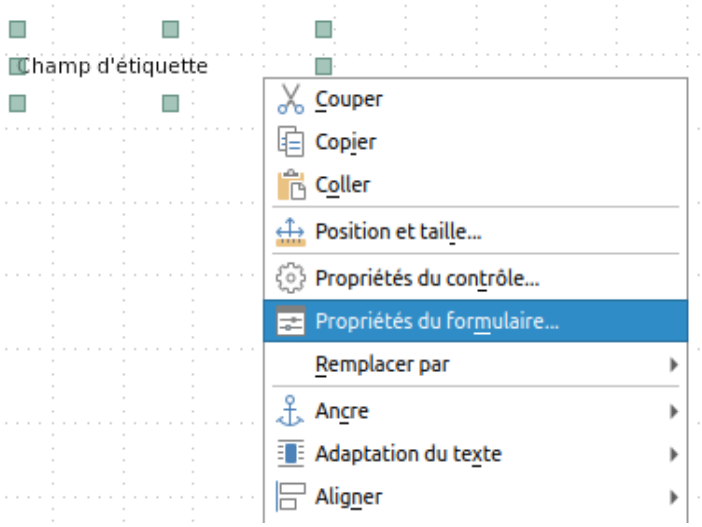


Figure 65: Menu contextuel de contrôle

Sélectionnez l'option **Propriétés de formulaire** (mise en évidence dans l'illustration) pour définir les propriétés du formulaire que vous venez de créer. Le formulaire a le nom par défaut Formulaire.

Il est également possible d'accéder aux propriétés du formulaire avec le bouton prévu dans la barre d'outils de conception (s'il est visible).

Formulaires externes

Outre les formulaires créés dans Base, il est également possible de créer des formulaires dans Writer ou Calc. Ceux-ci sont décrits au Chapitre 7, Connexions aux bases de données.

Propriétés du formulaire

Lorsque les propriétés du formulaire sont appelées à l'aide du menu contextuel du navigateur de formulaires ou du menu contextuel d'un contrôle de formulaire ou encore de l'icône de la barre d'outils, une fenêtre Propriétés du formulaire apparaît. Il comporte trois onglets : Général, Données et Événements.

Onglet Général

Ici, vous pouvez modifier le nom du formulaire. En outre, il existe des possibilités de conception qui n'ont aucune importance à l'intérieur de Base. Elles ne montrent que les possibilités plus générales de conception à l'aide d'un éditeur de formulaires : lorsque vous créez un formulaire Web, vous devrez les utiliser.

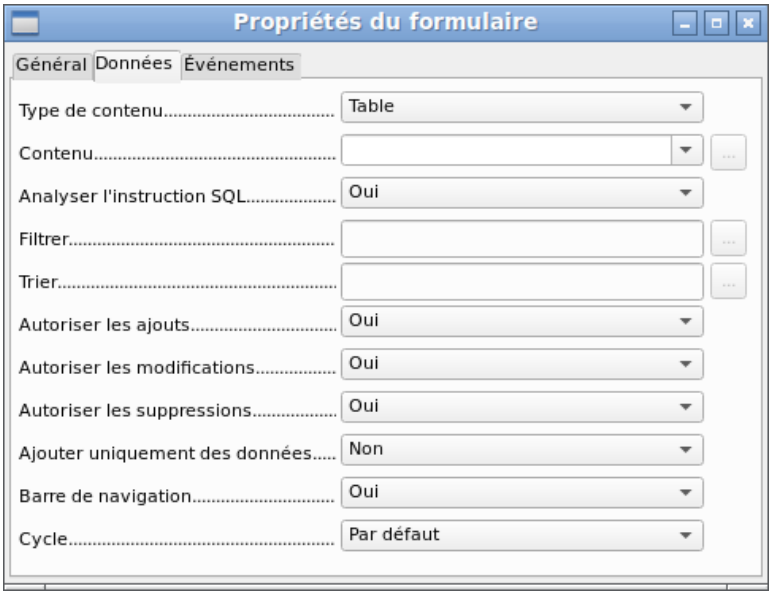
URL : Destination pour les données.

Cadre : Section du site Web de destination à adresser si nécessaire.

Encodage de l'envoi : en plus du codage de caractères normal pour la transmission à l'URL, vous pouvez spécifier ici le codage de texte et le codage en Multipart (par exemple, pour le transfert de données).

Type d'envoi : GET (*Obtenir* visible via l'URL attachée au nom de fichier, vous pouvez le voir souvent sur le Web si vous utilisez un moteur de recherche) ou POST (non visible, convient aux gros volumes de données). GET et POST sont des méthodes de transmission de données pour les formulaires Web.

Onglet Données



Pour créer des formulaires internes dans Base, il s'agit de l'onglet le plus important. Ici, vous pouvez définir les propriétés initiales suivantes pour le formulaire :

Type de contenu : Choisissez entre les contenus Table, Requête et SQL. Bien que Table puisse toujours être utilisé pour la saisie de données dans un formulaire, ce n'est pas toujours le cas pour Requête (pour plus d'informations, voir le chapitre 5, Requêtes) ou la saisie directe d'une commande SQL. Il s'agit ici d'une requête qui n'est pas visible dans le conteneur de requêtes de Base mais qui a en principe la même structure.

Contenu : Selon que Table ou Requête a été choisie ci-dessus, toutes les tables et requêtes disponibles sont répertoriées. Si une commande SQL doit être créée, vous pouvez appeler l'éditeur de requête en utilisant les points de suspension (…) à droite du champ Contenu.

Analyser la commande SQL : Si l'analyse des commandes SQL ne doit pas être autorisée (parce que, par exemple, vous utilisez du code que l'interface graphique ne peut pas afficher correctement), vous devez choisir **Non** ici. Cependant, cela empêchera le formulaire d'accéder aux données sous-jacentes à l'aide d'un filtre ou d'un tri.

Filtrer : Ici, vous pouvez définir un filtre. Pour obtenir de l'aide, cliquez sur le bouton à droite du champ. Voir également le Chapitre 3, Tables.

Tri : Ici, vous pouvez configurer un tri pour vos données. Pour obtenir de l'aide, cliquez sur le bouton à droite du champ. Voir également le Chapitre 3, Tables.

Autoriser les ajouts : L'entrée de nouvelles données est-elle autorisée ? Par défaut, il est défini sur **Oui**.

Autoriser les modifications : L'édition des données est-elle autorisée ? Par défaut également **Oui**.

Autoriser les suppressions : La suppression des données est également autorisée par défaut.

Ajouter uniquement des données : Si vous choisissez **OUI** pour cette option et **Non** pour les autres sélections, un formulaire vide sera toujours affiché. Il n'y aura aucun accès aux données existantes, qui ne peuvent être ni modifiées ni visualisées. Cependant, si **Autoriser les ajouts** et **Ajouter uniquement des données** est sélectionné, les données peuvent être ajoutées à un champ. Mais une fois que les données ont été enregistrées, elles ne sont plus visibles et les données sont écrites dans les tables.

Conseil

Cette propriété peut être utile lorsque des données doivent être ajoutées à la base de données mais que la personne qui le fait n'est pas autorisée à modifier ou supprimer des données existantes. Ces actions étant laissées à la charge d'une autre personne qui utilise pour cela un formulaire différent offrant ces possibilités.

Barre de navigation : La visibilité de la barre de navigation en bas de l'écran peut être activée ou désactivée. Il est également possible, lorsque vous avez un sous-formulaire, de toujours afficher la barre de navigation pour le formulaire principal, de sorte que l'activation de cette barre d'outils n'affecte que le formulaire principal. Ce paramètre de la barre de navigation n'est pas pertinent pour la barre d'outils de navigation interne qui peut être ajoutée en tant que contrôle de formulaire si nécessaire. Celle ci ne servira que dans le formulaire (ou sous formulaire) dans lequel elle se trouve.

Cycle : L'option par défaut pour les bases de données Base est qu'après l'entrée dans le dernier champ d'un formulaire, la touche Tabulation vous amène au premier champ de l'enregistrement suivant – c'est-à-dire qu'un nouvel enregistrement sera créé. Pour les bases de données, cela a le même effet que ***Tous les enregistrements*.** En revanche, si vous choisissez **Enregistrement actif**, le curseur se déplacera uniquement dans l'enregistrement ; lorsqu'il atteint le dernier champ, il reviendra au premier champ de cet

enregistrement. **La page actuelle** fait référence en particulier aux formulaires HTML. Le curseur saute de la fin d'un formulaire au formulaire suivant sur cette page plus bas

Onglet Événements

Événement peut déclencher des macros. Un clic sur le bouton à droite (…) permet d'associer des macros à l'événement

Réinitialisation/Rétablissement : Le formulaire est vidé de toutes les nouvelles entrées qui n'ont pas encore été enregistrées.

Avant l'envoi : Avant l'envoi des données du formulaire. Cela n'a de sens que pour les formulaires Web.

Avant le rechargement : Uniquement lors de l'ouverture du formulaire, pas lors du chargement d'un nouvel enregistrement dans le formulaire.

Rechargement : Cela se produit lorsque le contenu du formulaire est actualisé, par exemple à l'aide d'un bouton dans la barre de navigation.

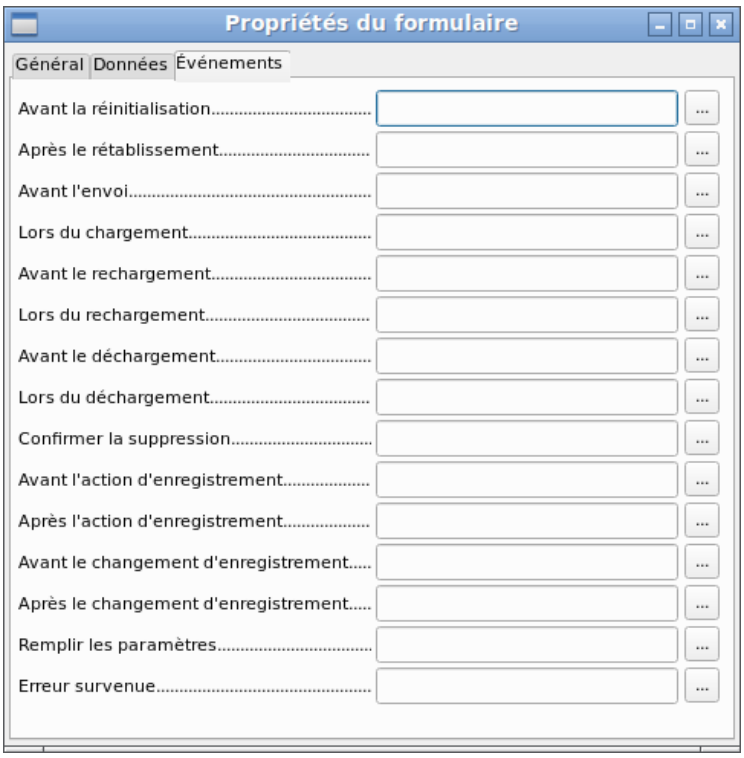
Déchargement : Cette option ne semble pas fonctionner. Il devrait se référer à la fermeture du formulaire.

Confirmer la suppression : L'événement **Confirmation de suppression** se produit dès que les données ont été supprimées du formulaire. Par exemple, la macro liée peut demander une confirmation dans une boîte de dialogue.

Action d'enregistrement : Cela inclut, par exemple, le stockage à l'aide d'un bouton. Dans les tests, cette action se duplique régulièrement ; les macros s'exécutent deux fois de suite.

En effet, ici différentes fonctions ("implémentations") sont effectuées. Les deux sont nommées : `org.openoffice.comp.svx.FormController` et `com.sun.star.comp.forms.ODatabaseForm`. Si dans la macro qui utilise `oForm.ImplementationName`, le nom correspondant est interrogé, la macro peut être limitée à une exécution.

Changement d'enregistrement : l'ouverture d'un formulaire compte comme un changement d'enregistrement. Chaque fois qu'on passe à d'un enregistrement à un autre dans le formulaire, cette action se produit également deux fois. Les macros sont donc exécutées deux fois de suite. Ici aussi, nous pouvons distinguer les causes de ce résultat.



Remplir les paramètres : Cette macro sera exécutée si une requête de paramètre doit être appelée dans un sous-formulaire, mais pour une raison quelconque, le paramètre n'est pas correctement transmis à partir du formulaire principal. Si cet événement n'est pas intercepté, une requête de paramètre suivra le chargement du formulaire.

Erreur survenue : L'événement **Erreur survenue** est activé si une erreur se produit lors de l'accès à la source de données. Cela s'applique aux formulaires, aux zones de liste et aux boîtes combinées

Propriétés des contrôles

Une fois qu'un formulaire a été créé, il peut être rempli de contrôles visibles. Certains contrôles permettent d'afficher le contenu de la base de données ou de saisir des données dans la base. D'autres contrôles sont utilisés exclusivement pour la navigation, pour la recherche et pour l'exécution de commandes (interaction). Certains contrôles servent à une retouche graphique supplémentaire du formulaire.

Saisie et affichage des données	
Contrôle	***Usage***
Zone de texte	Saisie de texte
Champ numérique	Saisie de nombres
Champ de date	Saisie de dates
Champ horaire	Saisie des heures
Champ monétaire	Saisie numérique, préformée pour la devise locale
Champ formaté	Affichage et saisie avec mise en forme supplémentaire, par exemple à l'aide d'unités de mesure

Saisie et affichage des données	
Contrôle	***Usage***
Zone de liste	Choix entre plusieurs possibilités différentes, également pour le transfert dans la base de données de valeurs autres que celles affichées
Zone combinée	Similaire à un champ de liste, mais avec seulement la valeur affichée transférée, où vous pouvez entrer de nouvelles valeurs à la main
Case à cocher	Champ Oui/Non
Bouton radio	Permet de choisir parmi un petit nombre de possibilités
Contrôle picto	Affichage d'images d'une base de données et entrée d'images dans une base de données via une sélection de chemin
Champ de motif	Entrée dans un masque prédéfini ; limite les possibilités de saisie à des combinaisons de caractères spécifiques
Contrôle de table	Module d'entrée universel, qui peut afficher une table entière. Plusieurs des commandes ci-dessus sont intégrées à ce contrôle
Conception	
Contrôle	***Usage***
Étiquette	En-tête du formulaire, description des autres contrôles
Zone de groupe	Un cadre autour, par exemple, d'un ensemble de boutons d'option
Interaction	
Contrôle	***Usage***
Bouton	Bouton avec étiquette
Bouton picto	Comme un bouton, mais avec une image supplémentaire (graphique) affichée dessus
Barre de navigation	Barre d'outils très similaire à celle en bas de l'écran
Sélection de fichier	Pour sélectionner des fichiers, par exemple pour les télécharger dans un formulaire HTML – pas décrit plus en détail
Compteur	Ne peut être utilisé qu'avec une macro – pas de description plus détaillée
Barre de défilement	Ne peut être utilisé qu'avec une macro – pas de description plus détaillée
Contrôle caché	Ici, une valeur peut être stockée à l'aide de macros puis lue à nouveau,

Paramètres par défaut pour de nombreux contrôles

Comme pour les formulaires, les propriétés de contrôle (Cf. Fig 66) sont regroupées en trois catégories : Général, Données et Événements.

L'onglet Général comprend tout ce qui est visible pour l'utilisateur.

L'onglet Données spécifie la liaison à un champ de la base de données ou une requête SQL.

L'onglet Evenements contrôle les actions, qui peuvent être liées à une macro. Dans une base de

données sans macros, cette catégorie ne joue aucun rôle.

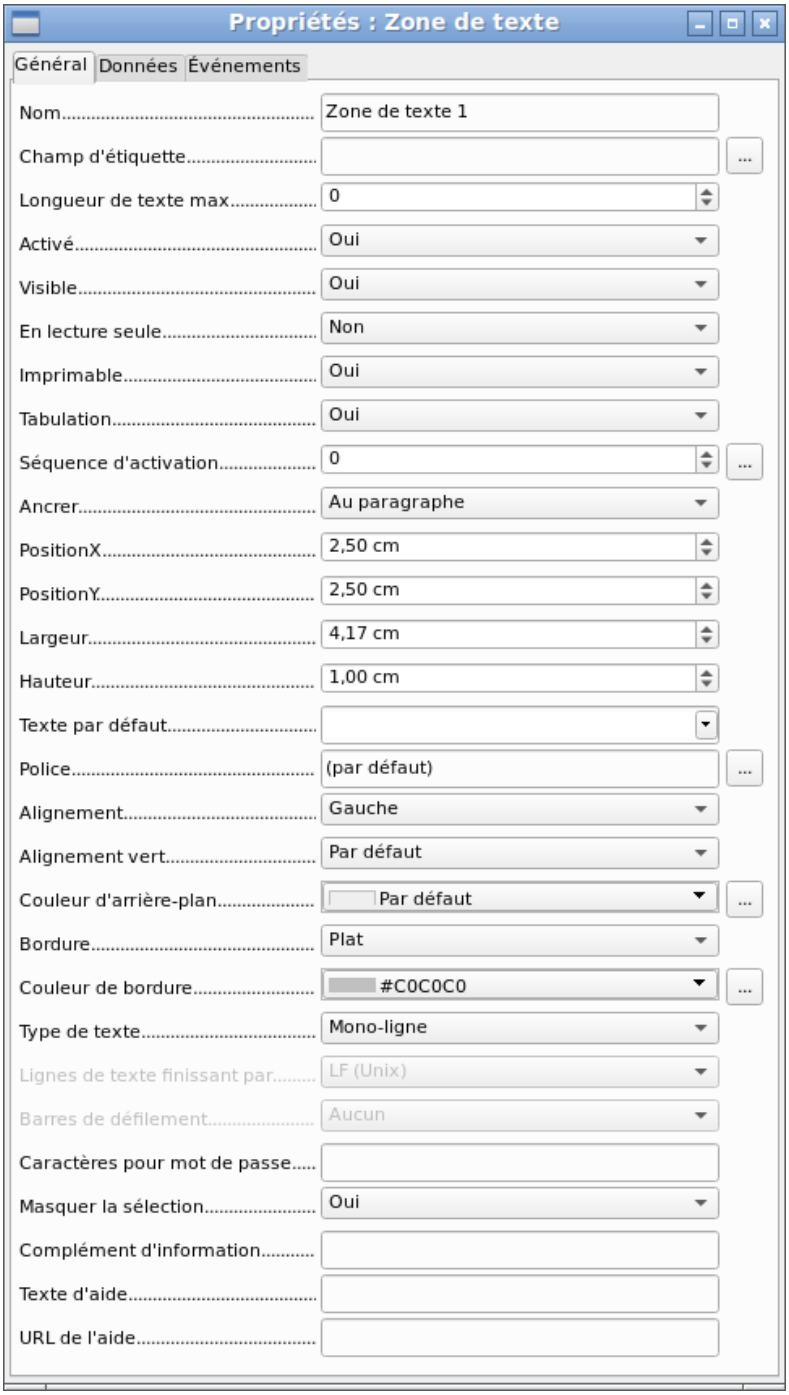


Figure 66: Propriétés de contrôle Texte

Onglet Général

Le nom d'un contrôle doit être unique dans le formulaire – utilisé pour accéder à l'aide de macros. [Name]

Le champ a-t-il une étiquette ? Cela regroupe le champ et l'étiquette.
Une étiquette permet d'accéder directement au champ du formulaire à l'aide d'un raccourci clavier.
[LabelControl]

Les champs non activés ne peuvent pas être utilisés et sont grisés. Utile pour contrôler à l'aide de macros. (Exemple : si le champ 1 contient une valeur, le champ 2 ne doit pas en contenir ; le champ 2 est désactivé.)
[Enabled]

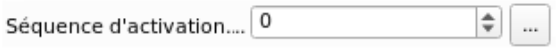
Généralement Oui ; Les champs invisibles peuvent être utilisés comme stockage intermédiaire, par exemple pour créer des champs de combinaison avec des macros. Voir le chapitre 9, Macros.
[EnableVisible]

Oui exclura toute modification de la valeur. Ceci est utile, par exemple, pour une clé primaire générée automatiquement.
[ReadOnly]

Parfois, il est utile d'imprimer une page à partir d'un formulaire plutôt que d'un rapport séparé. Dans ce cas, tous les champs ne doivent pas nécessairement apparaître.
[Printable]

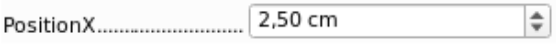
Dans un formulaire, la touche Tab est normalement utilisée pour la navigation.

Un champ en lecture seule n'a pas besoin d'un taquet de tabulation ; il peut être ignoré.
[Tabstop]



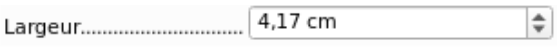
Le champ a-t-il un taquet de tabulation ? Ici, la séquence d'activation dans le formulaire est spécifiée.
[Tabindex]

Ancrage du contrôle par rapport au texte.

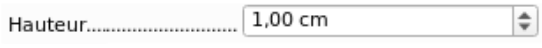


Position à partir du coin supérieur gauche relative au côté gauche du formulaire.
[PosSize.X]

Position à partir du coin supérieur gauche relative bord supérieur du formulaire.
[PosSize.Y]



Largeur du champ
[PosSize.Width]



Hauteur du champ
[PosSize.Height]

La police, la taille de la police et les effets de caractères peuvent être définis ici.
[Fontxxx]

Alignement. Ici, le de texte est justifié à gauche.
[Align]

Alignment Vertical : Par défaut | Haut | Milieu | Bas.
[VerticalAlign]

Couleur d’arrière-plan du champ de texte.
[BackgroundColor]

Encadrement : Sans cadre | Apparence3D-| plat.
[Border]

S’il y a un cadre, sa couleur peut être définie ici uniquement si Plat est sélectionné comme bordure.
[BorderColor]

Le texte en surbrillance perd la surbrillance lorsque le champ de texte perd le focus.
[HideInactiveSelection]

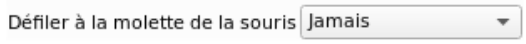
Utilisé pour les informations à lire par les macros. Voir le chapitre 9, Macros.
[Tag]

Apparaît comme une info-bulle lorsque la souris survole le champ de texte.
[HelpText]

Pointe vers un fichier d’aide, utile principalement pour HTML. Peut être appelé à l’aide de F1 lorsque le focus est sur le champ.
[HelpURL]

De plus, les champs numériques, de date, etc. ont les propriétés suivantes.

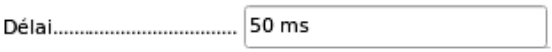
Lorsque la vérification est activée, seuls les nombres et les points décimaux peuvent être saisis. [EnforceFormat]



N'autorise jamais les modifications à l'aide de la molette de la souris. Lorsqu'elle est sélectionnée, permet de tels changements quand le champ est sélectionné et que la souris est au-dessus du champ. Toujours signifie chaque fois que la souris survole le champ.
[MouseWheelBehavior]

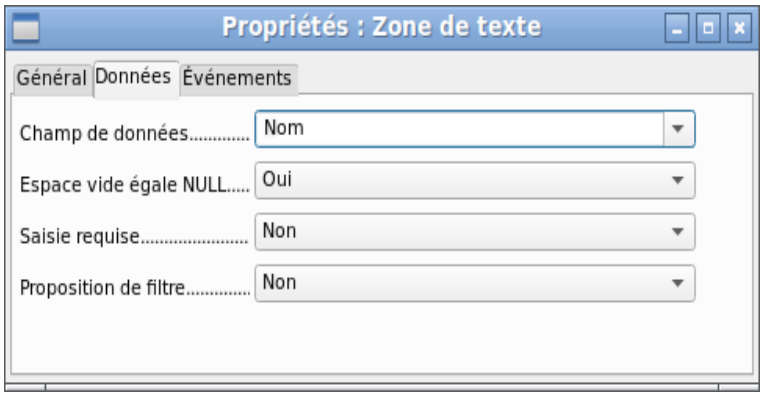
Un symbole de compteur est incorporé dans le côté droit du champ.
[Spin]

Si une flèche de rotation est enfoncée et maintenue, cela détermine si l'entrée dans la case doit être incrémentée continument au-delà de la valeur suivante.
[Repeat]



Détermine le délai minimum après une pression sur le bouton de la souris qui déclenche la répétition.
[RepeatDelay]

Onglet Données



Champ de données : Ici, vous créez la liaison avec la table, la requête ou la vue sur laquelle le formulaire est basé. [Model. DataField]

Espace vide égale NULL : Si une chaîne vide doit être traitée comme (NULL) ou le contenu simplement supprimé.

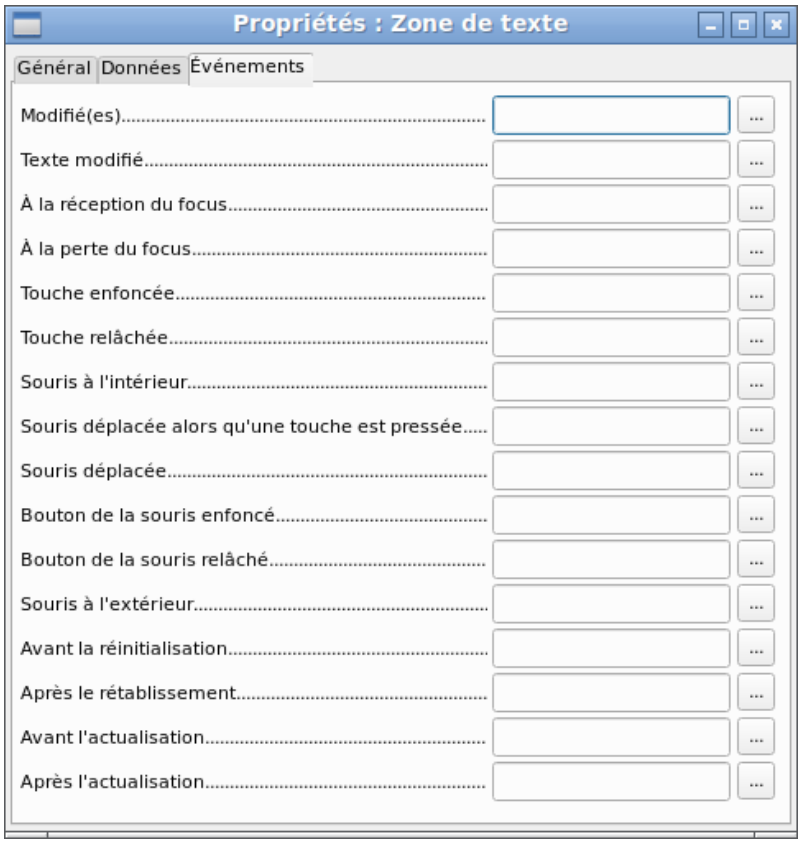
Saisie requise : Cette condition doit correspondre à celle de la table. L'interface graphique demandera une entrée si l'utilisateur n'a pas entré de valeur. [Model. InputRequired]

Proposition de filtre : Lorsque les données doivent être filtrées, le contenu de ce champ est temporairement stocké sous forme de suggestion. [Model. UserValueFilterProposal]

Attention

Attention – avec un contenu volumineux, ce choix peut nécessiter beaucoup de stockage.

Onglet Événements



Modifié(es) : Cet événement a lieu lorsqu'un contrôle est modifié et perd ensuite le focus. L'événement est perdu si vous passez directement à un autre enregistrement. Dans ces circonstances, une modification est enregistrée sans être détectée au préalable. [com.sun.star.lang. EventObject]

Texte modifié : Fait référence au contenu, qui peut en fait être du texte, des chiffres ou autre. Se produit après la saisie de chaque caractère supplémentaire. [com.sun.star.awt. TextEvent]

A la réception du focus : Le curseur entre dans le champ.

Attention

En aucun cas, la macro ne doit créer une boîte de dialogue de message à l'écran ; en cliquant dans une telle boîte de dialogue, le champ de formulaire perd le focus, puis le récupère, déclenchant à nouveau la macro. Une boucle est créée qui ne peut être interrompue qu'en utilisant le clavier.

À la perte du focus : Le curseur quitte le champ. Cela peut conduire au même type d'interaction lorsque la gestion de l'événement le fait se reproduire.

Touche : Fait référence au clavier. Par exemple, une touche est tapée lorsque vous vous déplacez dans le formulaire à l'aide de la touche Tab. Un champ reçoit le focus. Ensuite, la touche est relâchée.

L'événement est passé à l'aide du keyCode ou KeyChar de la touche libérée (lettre, chiffre, clé spéciale). [com.sun.star.awt. KeyEvent]

Souris : Explicite ; ces événements n'ont lieu que si la souris est ou était déjà dans le champ ("extérieur" correspond au javascript onMouseOut). [com.sun.star.awt. MouseEvent]

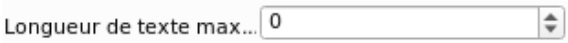
Réinitialisation/rétablissement : Le formulaire est vidé de toutes les données (lors de la création d'un nouvel enregistrement) ou remis à son état d'origine (lors de l'édition d'un enregistrement existant). Pour un champ de formulaire, cet événement est déclenché uniquement lorsque la saisie de données est annulée à l'aide du bouton dans la barre de navigation. [com.sun.star.lang. EventObject] Lorsqu'un formulaire est chargé pour la première fois, les deux événements *Avant la réinitialisation* et *Après le rétablissement* se produisent successivement, avant que le formulaire ne soit disponible pour la saisie.

Actualisation : Si l'événement est lié à un contrôle de formulaire, la mise à jour a lieu lorsque le focus est perdu et passe à un autre contrôle de formulaire, après avoir modifié le contenu du champ. Les modifications du formulaire sont acceptées et affichées. Lorsqu'un formulaire est fermé, les deux événements Avant l'actualisation et Après l'actualisation se succèdent. [com.sun.star.lang. EventObject]

Contrôle de texte

Outre les propriétés décrites à la page 164, les champs de texte peuvent avoir les propriétés supplémentaires suivantes :

Onglet Général

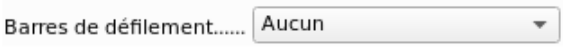


Lorsque cette valeur est 0, la saisie n'est pas autorisée. Habituellement, la longueur du champ de base de données auquel le champ de texte correspond est utilisée ici.
[MaxTextLen]

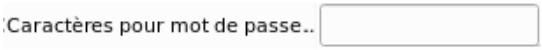
Le texte par défaut doit-il être placé dans un champ vide ? Ce texte sera supprimé si une autre saisie est effectuée avec succès.
[DefaultText]

Types possibles : Mono-ligne | Multi-ligne | Multi-ligne avec formatage (les deux derniers diffèrent par leur comportement de tabulation et, de plus, un champ de modèle ne peut pas être lié à une base de données). L'alignement vertical n'est pas actif pour les champs Multi-ligne.
[MultiLine]

Unix ou Windows ? Cela affecte principalement les fins de ligne. En interne, les lignes Windows se terminent par deux caractères de contrôle (CR et LF).
[LineEndFormat]



Uniquement pour les champs Multi-ligne : Horizontal | Vertical | Les deux.
[HScroll], [VScroll]



Actif uniquement pour les champs Mono-ligne. Modifie les caractères pour ne voir que des points.
[EchoChar]

Onglet Données

Rien de particulier.

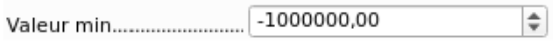
Onglet Événements

Rien de particulier.

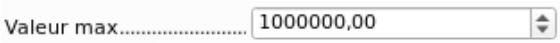
Contrôle numérique

En plus des propriétés déjà décrites, on trouve les propriétés suivantes :

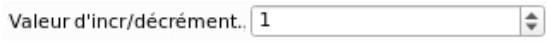
Onglet Général



Valeur minimale du champ. Doit être conforme à la valeur minimale définie dans la table.
[ValueMin]

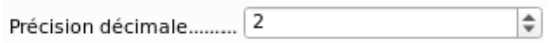


Valeur maximum.
[ValueMax]



Incrément de défilement lors de l'utilisation de la molette de la souris ou dans une boîte de sélection numérique.
[ValueStep]

Valeur affichée lors de la création d'un nouvel enregistrement.
[DefaultValue]



Nombre de décimales, 0 pour les entiers.
[DecimalAccuracy]

Séparateur pour des milliers, généralement un point.
[ShowThousandsSeparator]

Onglet Données

Il n'y a pas de contrôle pour savoir si un champ peut être NULL. S'il n'y a pas de saisie, le champ sera NULL et non 0. Aucune proposition de filtre n'est proposée.

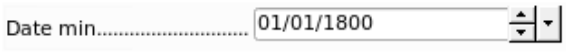
Onglet Événements

L'événement *Modifié* est absent. Les modifications doivent être gérées à l'aide de l'événement *Texte modifié* (le mot *texte* ne doit pas être pris littéralement ici).

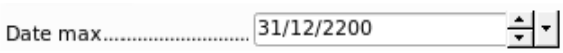
Contrôle de date

Outre les propriétés décrites à la page 164, il convient de noter les éléments suivants.

Onglet Général



Valeur minimale du champ, sélectionnable à l'aide d'un calendrier déroulant.
[DateMin]



Valeur maximum.
[DateMax]

Forme abrégée comme 27.02.20 ou formes diverses utilisant `'/'` au lieu de `'.'` ou `'-'` dans le style national/européen.
[DateFormat]

Ici, vous pouvez entrer une date littérale mais malheureusement pas (encore) la date actuelle (Aujourd'hui) au moment où le formulaire est ouvert.
[DefaultDate]

Un calendrier mensuel pour la sélection des dates peut être déroulé (Dropdown)
[DropDown]

Onglet Données

Il n'y a pas de contrôle pour savoir si un champ peut être NULL. S'il n'y a pas de saisie, le champ sera NULL et non 0.

Aucune proposition de filtre n'est offerte.

Onglet Événements

L'événement *Modifié* est absent. Les modifications doivent être gérées à l'aide de l'événement *Texte modifié* (le mot *texte* ne doit pas être pris littéralement ici).

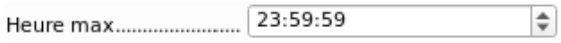
Contrôle Horaire

En plus des propriétés listées à la page 164, les fonctionnalités suivantes sont disponibles :

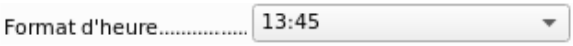
Onglet Général

Valeur minimale du champ, définie par défaut sur 0.

[TimeMin]



Valeur maximale, définie par défaut sur 1 seconde avant 24h00.
[TimeMax]



Forme courte sans secondes, forme longue avec secondes, ainsi que formats 12 heures avec AM et PM.
[TimeFormat]

Vous pouvez définir une heure fixe mais malheureusement pas (encore) l'heure réelle d'enregistrement du formulaire.
[DefaultTime]

Onglet Données

Il n'y a pas de contrôle pour savoir si un champ peut être NULL. S'il n'y a pas de saisie, le champ sera NULL et non 0.

Aucune proposition de filtre n'est offerte.

Onglet Événements

L'événement *Modifié* est absent. Les modifications doivent être gérées à l'aide de l'événement *Texte modifié* (le mot *texte* ne doit pas être pris littéralement ici).

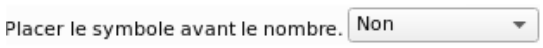
Contrôle monétaire

En plus des propriétés déjà répertoriées à la page 164, les fonctionnalités suivantes sont disponibles :

Onglet Général

Valeur min, Valeur max, Valeur d'incrément/décrément, Valeur par défaut, Précision décimale et Séparateur de milliers correspondent aux propriétés générales répertoriées à la page 171. En plus de celles-ci, il n'y a que :

Le symbole est affiché mais pas stocké dans la table qui sous-tend le formulaire.
[CurrencySymbol]



Le symbole doit-il être placé avant ou après le nombre?
[PrependCurrencySymbol]

Onglet Données

Il n'y a pas de contrôle pour savoir si un champ peut être NULL. S'il n'y a pas de saisie, le champ sera NULL et non 0.

Aucune proposition de filtre n'est offerte.

Onglet Événements

L'événement *Modifié* est absent. Les modifications doivent être gérées à l'aide de l'événement *Texte modifié* (le mot *texte* ne doit pas être pris littéralement ici).

Contrôle formaté

En plus des propriétés listées à la page 164, les fonctionnalités suivantes sont disponibles :

Onglet Général

Les valeurs minimum et maximum, ainsi que la valeur par défaut, dépendent du formatage. Derrière le bouton de mise en forme se trouve un champ flexible qui rend la plupart des champs monétaires et numériques inutiles. Contrairement à un champ monétaire simple, un modèle de champ peut afficher des sommes négatives en rouge.

Le bouton à droite avec les trois points offre un choix de formats numériques, comme vous le faites habituellement dans Calc.
[FormatKey]

Parmi les formats numériques, on peut voir, à côté de la date, de l'heure, de la devise ou du format numérique normal, des possibilités d'utilisation de champs avec une unité de mesure telle que le kg (voir Figure 67). Consultez également l'aide générale sur les codes de format numérique.

Un contrôle formaté permet de créer et d'écrire dans des champs d'horodatage de tables en utilisant un seul champ. L'Assistant Formulaire utilise une combinaison d'une date et d'un champ horaire pour cela.

Si vous souhaitez saisir des données sous la forme minutes : secondes : centièmes de secondes dans un champ d'horodatage, vous devrez utiliser des macros (voir au Chapitre 9.

Onglet Données

Rien de spécial à signaler.

Onglet Événements

L'événement *Modifié* est absent. Les modifications doivent être gérées à l'aide de l'événement *Texte modifié* (le mot *texte* ne doit pas être pris littéralement ici).

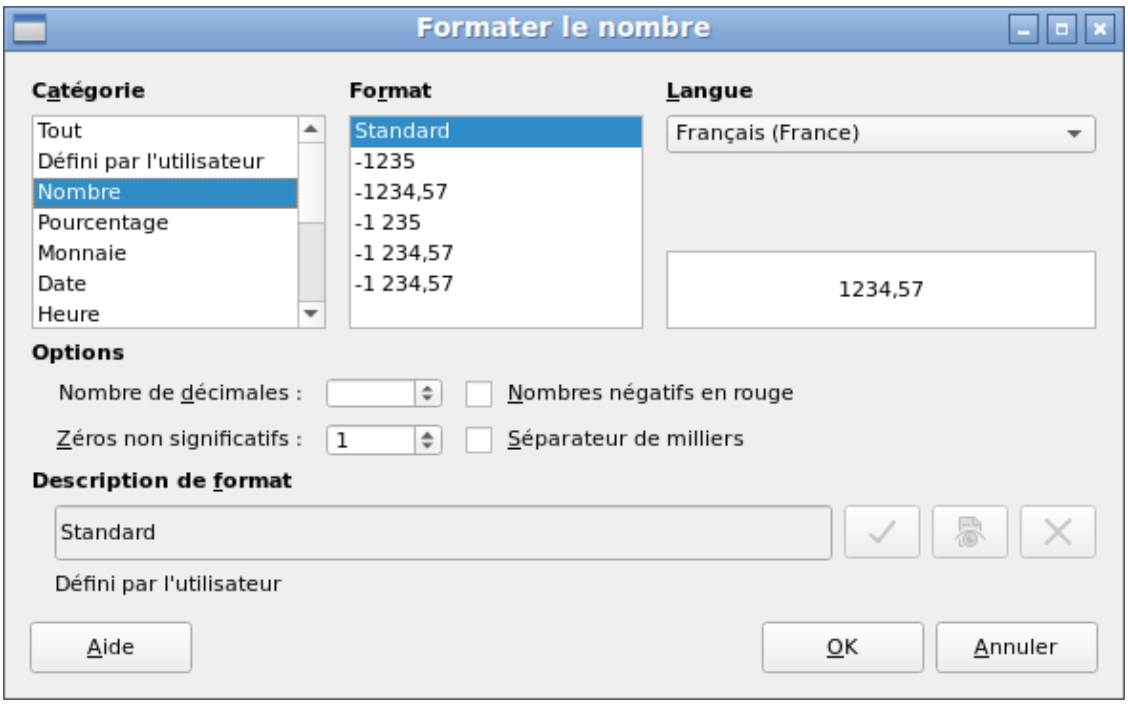
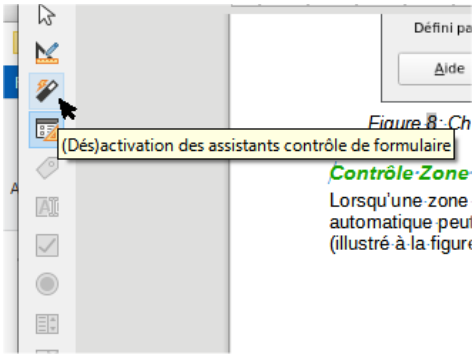
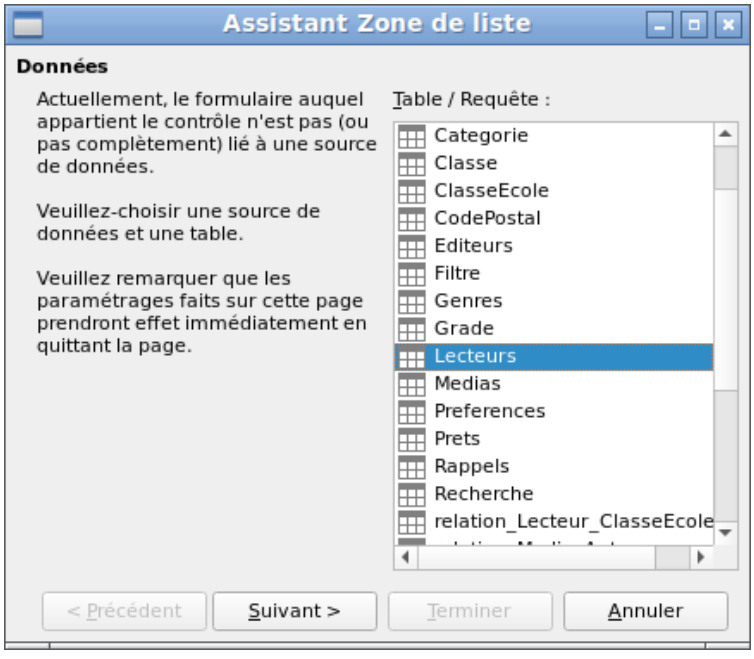


Figure 67: Champ formaté avec options numériques générales

Contrôle Zone de Liste

Lorsqu'une zone de liste est créée, l'Assistant Zone de liste apparaît par défaut. Cette apparence automatique peut être désactivée si nécessaire à l'aide du bouton **Marche / Arrêt** des assistants (illustré à la figure 60 sous l'équerre).



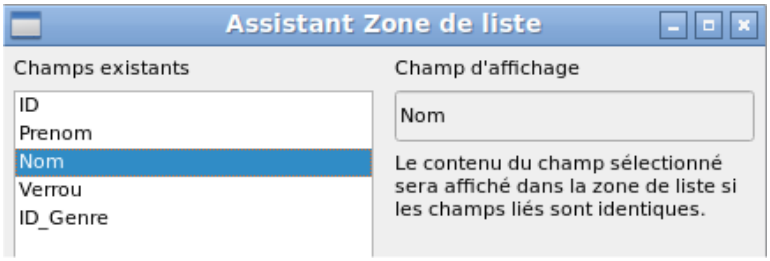


Assistant

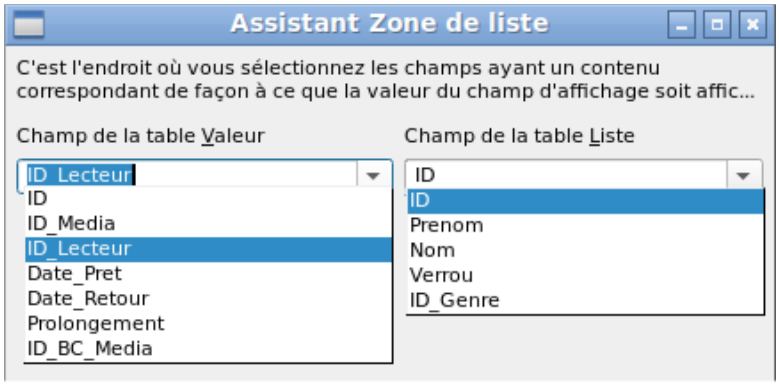
Le formulaire est déjà défini. Il est lié à une table nommée Prets. Une zone de liste montre à l'utilisateur des données différentes de ce qui est réellement transmis dans la table. Ces données proviennent généralement d'une autre table de la base de données et non de la table à laquelle le formulaire est lié.

La table des prêts est censée indiquer quel lecteur a emprunté quel média. Cependant, cette table ne stocke pas le nom du lecteur mais la clé primaire correspondante de la table Lecteurs. C'est donc la table Lecteurs qui forme la base de la zone de liste.

Le champ Nom de la table Lecteurs doit être visible dans la zone de liste. Cela sert de champ d'affichage.



Le champ ID_Lecteur apparaît dans la table Prets qui sous-tend le formulaire. Cette table est décrite ici comme la table Valeur. L'ID de clé primaire de la table Lecteurs doit être lié à ce champ. La table Lecteurs est décrite ici comme la table Liste.



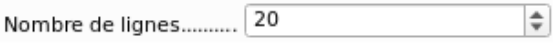
La zone de liste a maintenant été créée avec les données et la configuration par défaut. Elle est entièrement fonctionnelle.

En plus des propriétés listées à la page 164, les fonctionnalités suivantes sont disponibles.

Onglet Général

Les entrées de la liste ont déjà été définies à l'aide de l'assistant. Sinon, vous pouvez ajouter d'autres entrées qui ne proviennent d'aucune table de la base de données. Les entrées de liste ici désignent les entrées visibles, pas celles que le formulaire transmettra à la table.
[StringItemList]

Si le champ n’est pas spécifié comme liste déroulante, des flèches de défilement apparaîtront sur le côté droit de la zone de liste lorsque le formulaire est chargé. Le champ de liste devient alors automatiquement un champ multi-ligne, dans lequel la valeur réelle sélectionnée est mise en évidence.
[Dropdown]



Si le champ est déroulant, cette propriété donne le nombre maximum de lignes visibles. Si le contenu s’étend sur plus de lignes, une barre de défilement apparaît lorsque la liste se déroule.
[LineCount]

Peut-on sélectionner plus d’une valeur ? Dans l’exemple ci-dessus, cela n’est pas possible car une clé étrangère est en cours de stockage. En général, cette fonction n’est pas utilisée pour les bases de données, car chaque champ ne doit contenir qu’une seule valeur. Si nécessaire, les macros peuvent aider à interpréter plusieurs entrées sélectionnées dans le champ de liste.
[MultiSelection] [MultiSelectionSimpleMode]

Comme le bouton désactivé l’indique clairement, une sélection par défaut n’a pas de sens dans le contexte d’une liaison avec une table de base de données, telle que créée par l’Assistant Zone de liste. Il se peut fort bien que l’enregistrement correspondant à la sélection par défaut dans l’exemple de table Lecteurs ne soit plus présent.
[DefaultSelection]

Onglet Données

En plus des propriétés de données habituelles, Champ de données et Saisie requise, il existe des propriétés importantes qui affectent la liaison entre les données affichées et les données à saisir dans la table sous-jacente au formulaire (Cf. Fig 68).

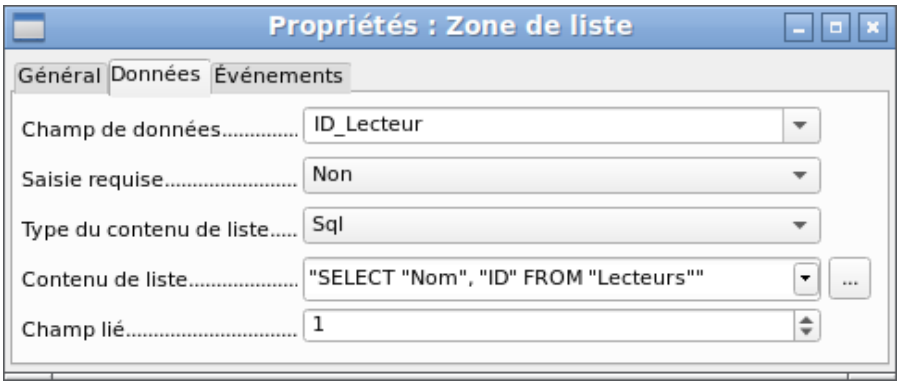


Figure 68: Onglet données d'une zone de liste

Type du contenu de liste : Liste de valeurs | Table | Requête | SQL | SQL [Natif] | Champs de table [ListSourceType]

Contenu de liste Valeurs : Si des entrées de liste ont été créées sous Général, les valeurs correspondantes à stocker sont saisies ici. Le contenu de la liste est chargé avec des éléments individuels séparés par *Maj + Entrée*. Le champ Contenu de liste les affiche alors sous la forme "Valeur1", "Valeur2", "Valeur3"… La propriété Champ lié est alors inactive.

Contenu de liste Table : Ici, l'une des tables de la base de données peut être sélectionnée. Cependant, cela est rarement possible car cela nécessite que le contenu de la table soit structuré de telle sorte que le premier champ de table contienne les valeurs à afficher dans le champ de liste, et l'un des champs suivants contient la clé primaire que la table sous-jacente au formulaire utilise comme clé étrangère. La position de ce champ dans la table est spécifiée dans Champ lié, où *la numérotation commence par 0 pour le premier champ de la table de base de données*. Mais ce 0 est réservé à la valeur affichée, dans l'exemple ci-dessus le Nom, tandis que le 1 fait référence au champ ID.

Contenu de liste Requête : Ici, une requête est d'abord créée séparément et stockée. La création de telles requêtes est décrite au chapitre 5, Requêtes.

En utilisant la requête, il est possible de déplacer le champ ID de la première position dans la table sous-jacente à la deuxième position, ici représentée par le champ lié 1.

Contenu de liste SQL : L'Assistant Zone de liste remplit ce champ. La requête construite par l'assistant ressemble au contenu de l'option de la figure 68

La requête est la plus simple possible. Le champ Nom apparaît à la position 0, le champ ID à la position 1. Les deux sont lus à partir de la table Lecteurs. Comme le champ lié est le champ 1, cette formule SQL fonctionne. Ici devrait être ajouté *ORDER BY "Nom" ASC*. Vous n'aurez ainsi pas besoin de faire défiler trop longtemps la liste pour trouver quelqu'un. Un problème supplémentaire peut être que Nom peut être le même pour plusieurs lecteurs. Donc Prenom doit être ajouté dans la vue de la zone de liste. Lorsqu'il y a des lecteurs avec le même Nom et le même Prenom, l'ID de clé primaire doit également être affiché. Voir le Chapitre 5, Requêtes, pour plus d'informations sur son fonctionnement.

Contenu de liste SQL [Natif] : La formule SQL est saisie directement, sans utiliser l'assistant. Base n'évalue pas la requête. Cela convient lorsque la requête contient des fonctions qui peuvent ne pas être comprises par l'interface graphique de Base. Dans ce cas, la requête n'est pas vérifiée pour les erreurs éventuelles. Pour en savoir plus sur le mode SQL direct, reportez-vous au Chapitre 5, Requêtes.

Contenu de liste Champs de table : Ici, les noms de champs d'une table sont répertoriés, pas leur contenu. Pour la table Lecteurs, le contenu de la liste serait ID, Prenom, Nom, Verrou, ID_Genre.

Note

Si vous voulez un champ de temps qui peut gérer le temps en millisecondes, vous aurez besoin d'un champ d'horodatage, comme décrit dans la section sur "Contrôle Horaire". La représentation des millisecondes est incompatible avec la façon dont les caractères sont assemblés dans les zones de liste. Pour contourner ce problème, vous devez convertir l'horodatage en texte :

```
SELECT REPLACE(LEFT(RIGHT(CONVERT("Service_Requis(??)".
"Heure", VARCHAR),15),8),'.','','') AS "ContenuListe", "ID" FROM
"Service_Requis"
```

Cela donnera un affichage en minutes : secondes : centièmes.

Champ lié : Les zones de liste affichent un contenu qui n'est pas nécessairement identique à ce qui sera stocké dans le formulaire. Habituellement, un nom ou quelque chose de similaire est affiché et la clé primaire correspondante devient la valeur stockée de ce champ.

```
SELECT "Nom", "ID" FROM "Table" ORDER BY "Nom"
```

Le champ ID est stocké dans la table sous-jacente en tant que clé étrangère. Le numéro de champ dans les bases de données commence à zéro de sorte que le champ qui y est lié dans le formulaire a le numéro 1.

Si à la place vous sélectionnez "0", le contenu du champ Nom est enregistré dans le formulaire. Dans ce cas, vous pouvez supprimer le champ ID de la requête.

Il est même possible de choisir la position "–1". Alors ce n'est pas le contenu de la requête mais la position de l'entrée qui est stockée dans la liste. Le premier enregistrement a alors la position 1.

La requête ci-dessus donne le résultat suivant :

Nom	***ID***
Anne-Lise	2
Dorothée	3
Ségolène	1

Dans les champs de liste, seul le nom peut être sélectionné. Si le champ de liste est défini sur le nom "Dorothée", les données enregistrées suivantes sont possibles pour ce champ :

Champ lié = 1 signifie que "3" est enregistré, le contenu du champ ID.

Champ lié = 0 signifie que "Dorothée" est enregistré, le contenu du champ Nom.

Champ lié = –1 signifie que "2" est enregistré, car "Dorothée" vient en deuxième position dans la liste.

Note

Le changement du champ lié à “0” ou “–1” a été introduit avec la version 4.1 de LO. Auparavant, seule la sélection de valeurs> = 1 était possible.

Onglet Événements

En plus des événements standard, les événements suivants sont disponibles :

Exécuter l’action : Si une valeur est choisie par le clavier ou la souris, la zone de liste exécute cette action.

Statut de l’élément modifié : Cela peut être le changement du contenu affiché d’une zone de liste via l’utilisation du bouton déroulant. Cela peut également être un clic sur le bouton déroulant du champ.

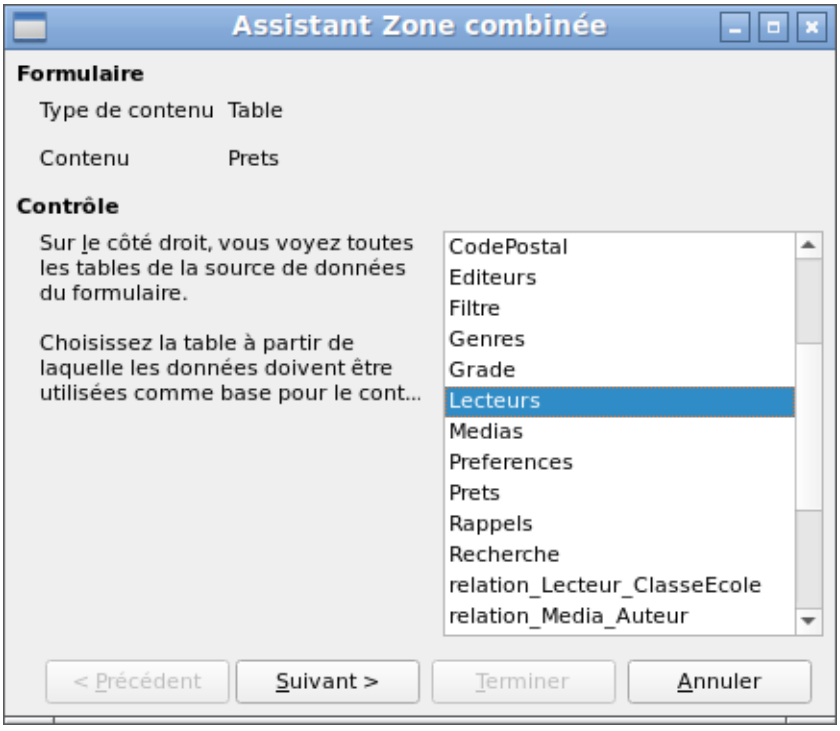
Erreur survenue : Malheureusement, cet événement ne peut pas être reconstruit pour les zones de liste.

Contrôle Zone combinée

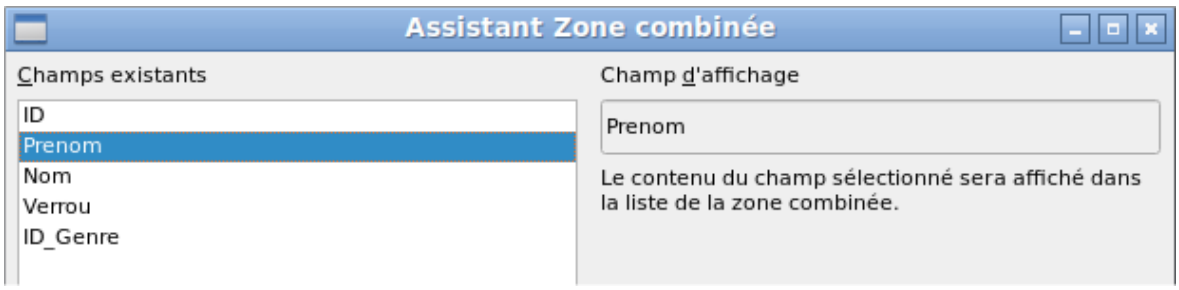
Dès qu’une zone de liste combinée est créée, un assistant apparaît par défaut, tout comme avec une zone de liste. Ce comportement automatique peut être désactivé si nécessaire à l’aide du bouton **Marche / Arrêt** des assistants.

Les zones de liste déroulante écrivent le texte sélectionné directement dans le tableau sous-jacent du formulaire. Par conséquent, l’exemple suivant montre à la fois la table liée au formulaire et celle sélectionnée pour le contrôle est la table Lecteurs.

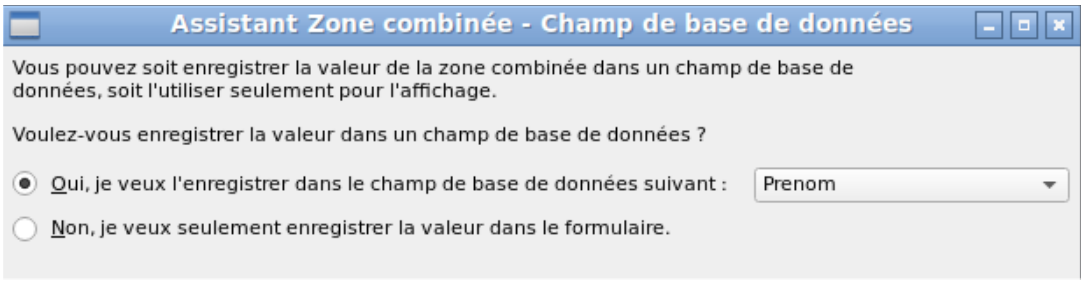
Assistant



Là encore, le formulaire est prédéfini, cette fois avec la table Lecteurs. Comme les données à afficher dans la liste déroulante doivent également être stockées dans cette table, la source sélectionnée pour les données de la liste est également la table Lecteurs.



Dans la table Lecteurs, le champ Prenom apparaît. Cela devrait être affiché dans la zone de liste déroulante.



Dans une base de données, il semble y avoir peu d'intérêt à ne pas stocker la valeur d'une zone de liste déroulante dans un champ. Nous voulons lire les prénoms de la table Lecteurs, et aussi les rendre disponibles pour les nouveaux lecteurs, afin que de nouveaux enregistrements n'aient pas besoin d'être créés pour un nom donné qui existe déjà dans la base de données. La zone de liste déroulante affiche le prénom et la saisie de texte n'est pas nécessaire.

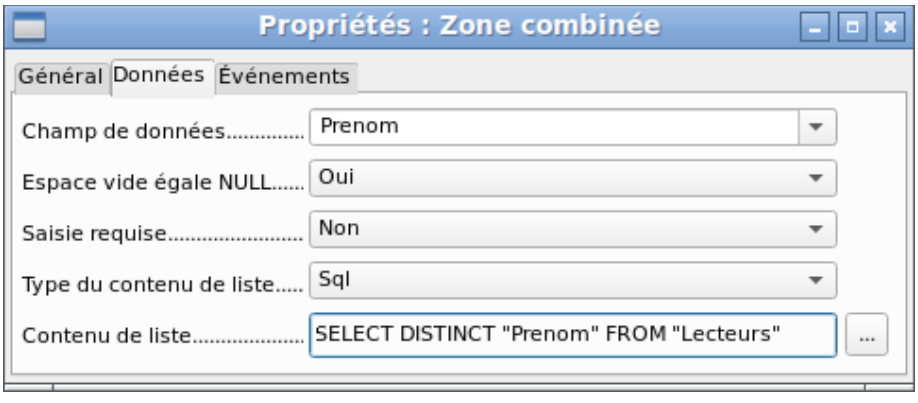
Si une nouvelle valeur doit être saisie, cela peut être fait facilement dans une zone de liste combinée, car la zone montre exactement ce qui se passe dans la table sous-jacente du formulaire.

En plus des propriétés présentées à la page 164 et décrites pour les zones de liste, les fonctionnalités suivantes sont disponibles.

Onglet Général

Lors de la saisie de nouvelles valeurs, une liste de valeurs correspondantes (le cas échéant) s'affiche pour une sélection possible.
[AutoComplete]

Onglet Données



Les champs de données sont conformes aux paramètres par défaut existants et aux paramètres d'une zone de liste. La commande SQL présente cependant une particularité :

```
SELECT DISTINCT "Prenom" FROM "Lecteurs"
```

L'ajout du mot-clé `DISTINCT` garantit que les prénoms en double ne sont affichés qu'une seule fois. Cependant, la création à l'aide de l'assistant une fois de plus rend impossible le tri du contenu.

Onglet Événements

Les événements correspondent à ceux d'une zone de liste.

Case à cocher

La case à cocher apparaît immédiatement sous la forme d'une combinaison d'un champ de case à cocher et d'une étiquette pour la case.

En plus des propriétés listées à la page 164, les fonctionnalités suivantes sont disponibles.

Onglet Général

Le libellé de cette boîte apparaît par défaut à droite de la boîte. De plus, vous pouvez le lier à un champ d'étiquette distinct., à placer au dessus ou devant. [Label]

Ici, en fonction du choix dans le champ *Statut triple*, jusqu'à trois possibilités sont disponibles : Non sélectionné | Sélectionné | Non défini. Non défini correspond à une entrée NULL dans la table sous-jacente au formulaire.
[État]

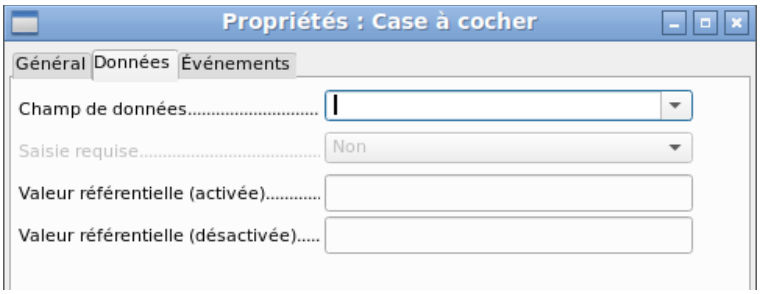
Par défaut, l'étiquette n'est pas fractionnée. L'étiquette est tronquée si le champ n'est pas assez grand.
[MultiLine]

Ici, vous pouvez spécifier une image à la place ou en plus de l'étiquette. Vous devez noter que ces images ne sont pas liées par défaut dans le document *.odb. Pour les petites images, il est utile d'incorporer l' image et non de la lier.
[Graphic]

Si vous avez choisi d'utiliser une image, son alignement avec l'étiquette peut être défini ici.
[ImagePosition] (0=gauche| 1=centré| 2=droite)

Par défaut, les cases à cocher n'ont que deux états : Sélectionné (Valeur : 1) et Non sélectionné (Valeur : 0). Avec Statut triple, une définition de champ vide (NULL) est ajoutée.
[TriState]

Onglet Données



La case à cocher peut recevoir une valeur de référence. Cependant, seules les valeurs de 1 (pour On) ou 0 (pour Off) peuvent être transférées dans le champ de données sous-jacent (les cases à cocher agissent comme des champs pour le choix de Oui et Non).

Onglet Événements

Les valeurs *Modifié, Texte modifié, Avant l'actualisation et Après l'actualisation* sont tous absents. Les champs supplémentaires pour une case à cocher sont *Exécuter l'action* (voir Zone de liste) et *Statut de l'élément modifié* (correspond à Changé).

Bouton radio

Le bouton d'option est similaire à la case à cocher décrite ci-dessus, à l'exception de ses propriétés générales et de sa forme externe (ronde).

Lorsque plusieurs Boutons d'option du formulaire sont liés au même champ de table, une seule des options peut être sélectionnée.

Onglet Général

Le bouton d'option est conçu pour être utilisé en groupe. Une seule des diverses options d'un groupe peut alors être sélectionnée. C'est pourquoi un nom de groupe apparaît ici, sous lequel les options peuvent être adressées.
[GroupName]

Onglet Données

Voir Case à cocher. Ici, cependant, les valeurs de référence saisies sont effectivement transférées dans le champ de données.

Onglet Événements

Voir Case à cocher

Contrôle picto

Un contrôle graphique (image) gère l'entrée et l'affichage des images pour la base de données. Le champ de données sous-jacent doit être un champ binaire s'il doit stocker l'image directement. Il peut également s'agir d'un champ de texte stockant le chemin relatif vers l'image. Dans ce cas, vous devez veiller à ce que le chemin d'accès aux images reste valide si la base de données est copiée.

Attention

Les images doivent dans tous les cas être réduites en taille lorsqu'elles sont stockées dans la base de données. Avec 3 Mo de photographies dans une base de données interne HSQLDB, vous obtiendrez très rapidement des erreurs Java (NullpointerException) qui rendront impossible le stockage des enregistrements. Cela peut entraîner la perte de tous les enregistrements qui ont été saisis dans la session en cours.

La saisie dans un champ image se fait soit par un double-clic avec la souris pour ouvrir une boîte de dialogue de sélection de fichier, soit par un clic droit pour choisir si un graphique existant doit être supprimé ou remplacé.

Un contrôle graphique par défaut n'a pas de taquet de tabulation.

En plus des propriétés listées à la page 164, les fonctionnalités suivantes sont disponibles.

Onglet Général

Le graphique sélectionné ici n'est affiché qu'à l'intérieur du contrôle, pendant que le formulaire est en cours de modification. Cela n'a aucune signification pour une saisie ultérieure.
[Graphic]

Non : l'image ne sera pas adaptée au champ. S'il est trop grand, le champ s'affichera uniquement en dehors de l'image. L'image n'est pas déformée.

Conserver le rapport : l'image est adaptée au contrôle mais n'est pas déformée (rapport d'aspect conservé).

Taille Autom.: l'image est ajustée au contrôle et peut être affichée sous une forme déformée.
[ScaleImage] [ScaleMode]

Onglet Données

Rien de plus à signaler.

Onglet Événements

Les champs *Modifié, Texte modifié, Avant l'actualisation et Après l'actualisation* sont tous absents.

Contrôle de motif

Un masque de saisie est utilisé pour contrôler l'entrée dans le champ. Les caractères sont pré-spécifiés pour des positions particulières, déterminant les propriétés des caractères saisis. Les caractères prédéfinis sont enregistrés avec les caractères saisis.

En plus des propriétés listées à la page 164, les fonctionnalités suivantes sont disponibles.

Onglet Général

Cela détermine quels caractères peuvent être saisis. [EditMask]

C'est ce que voit l'utilisateur du formulaire. [LiteralMask]

La longueur du masque d'édition détermine le nombre de caractères pouvant être saisis. Si l'entrée de l'utilisateur ne correspond pas au masque, l'entrée est rejetée en quittant le contrôle. Les caractères suivants sont disponibles pour définir le masque d'édition.

Caractère	*Signification*
L	Une constante de texte. Cette position ne peut pas être modifiée. Le caractère réel est affiché à la position correspondante dans le masque littéral.
a	Représente l'une des lettres a – z / A – Z. Les majuscules ne sont pas converties en minuscules.
A	Représente l'une des lettres A – Z. Si des lettres minuscules sont saisies, elles seront automatiquement converties en majuscules.
c	Représente l'un des caractères a – z / A – Z plus les chiffres 0–9. Les majuscules ne sont pas converties en minuscules.
C	Représente l'une des lettres A – Z plus les chiffres 0–9. Si des minuscules sont saisies, elles seront automatiquement converties en majuscules.
N	Seuls les chiffres de 0 à 9 peuvent être saisis.
x	Tous les caractères imprimables sont autorisés.
X	Tous les caractères imprimables sont autorisés. Si des lettres minuscules sont saisies, elles seront automatiquement converties en majuscules.

Ainsi, par exemple, vous pouvez définir le masque littéral comme "__/__/2020" et le masque d'édition comme "NNLNNLLLLL", pour permettre à l'utilisateur de saisir quatre caractères uniquement pour une date.

Onglet Données

Rien de plus à signaler.

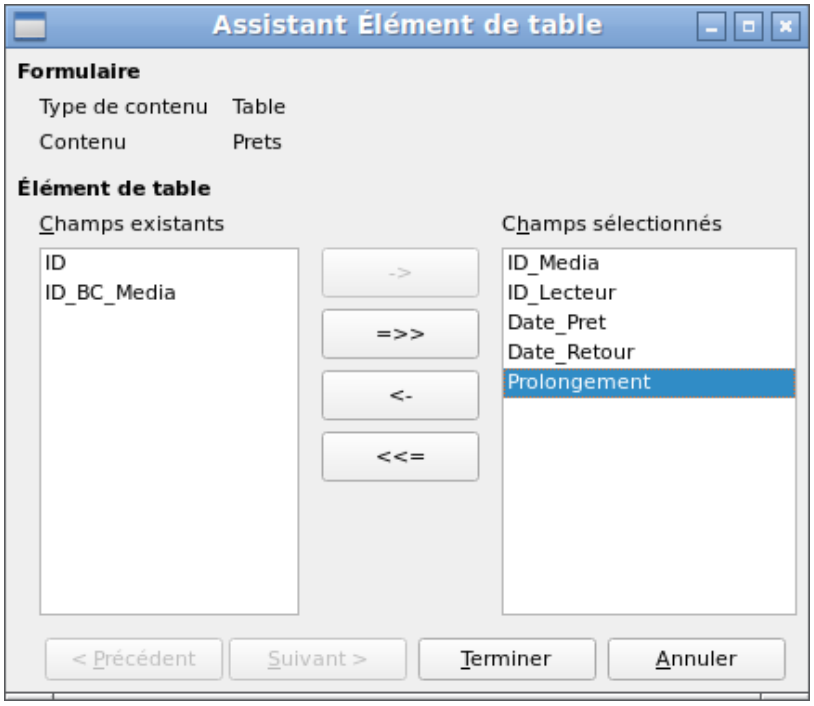
Onglet Événements

L'événement *Modifié* est absent.

Contrôle de table

Il s'agit du contrôle le plus complet. Il fournit un tableau, qui peut ensuite être doté de commandes pour les colonnes individuelles. Cela permet non seulement de visualiser les données réelles lors

de la saisie, mais également les données précédemment saisies, sans qu'il soit nécessaire d'utiliser la barre de navigation pour faire défiler les enregistrements.

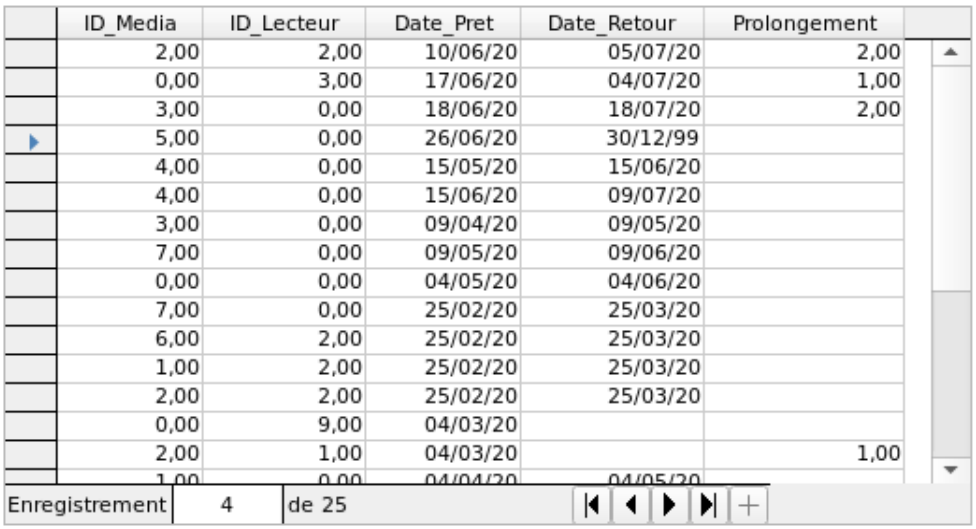


Tous les champs possibles dans un formulaire ne peuvent pas être sélectionnés pour un champ de contrôle de table. Les boutons poussoirs, les boutons d'image et les boutons d'options ne sont pas disponibles.

L'Assistant Contrôle de table assemble dans une fenêtre les champs qui apparaîtront ensuite dans le tableau.

La table Prets est disponible pour modification dans le contrôle. A part le champ ID (clé primaire) et le champ ID_BC_Media (saisie de média à l'aide d'un lecteur de code-barres), tous les champs doivent être utilisés dans le contrôle.

Le contrôle de table créé précédemment doit maintenant être développé plus avant, pour permettre l'entrée dans la table de prêts. Pour des champs tels que ID_Lecteur ou ID_Media, il serait plus utile de pouvoir choisir directement le lecteur ou le média, plutôt qu'un nombre représentant le lecteur ou le média. Pour cela, des contrôles tels que des zones de liste peuvent être placés dans le contrôle de table. Ceci est déclaré plus tard. Le formatage du champ Prolongement avec deux décimales n'était certainement pas prévu.



	ID_Media	ID_Lecteur	Date_Pret	Date_Retour	Prolongement
	2,00	2,00	10/06/20	05/07/20	2,00
	0,00	3,00	17/06/20	04/07/20	1,00
	3,00	0,00	18/06/20	18/07/20	2,00
▶	5,00	0,00	26/06/20	30/12/99	
	4,00	0,00	15/05/20	15/06/20	
	4,00	0,00	15/06/20	09/07/20	
	3,00	0,00	09/04/20	09/05/20	
	7,00	0,00	09/05/20	09/06/20	
	0,00	0,00	04/05/20	04/06/20	
	7,00	0,00	25/02/20	25/03/20	
	6,00	2,00	25/02/20	25/03/20	
	1,00	2,00	25/02/20	25/03/20	
	2,00	2,00	25/02/20	25/03/20	
	0,00	9,00	04/03/20		
	2,00	1,00	04/03/20		1,00
	1,00	0,00	04/04/20	04/05/20	

Enregistrement 4 de 25

Figure 69: Affichage du contrôle de table

En plus des propriétés listées à la page 164, les fonctionnalités suivantes sont disponibles.

Onglet Général

Hauteur des lignes individuelles. Sans valeur ici, la hauteur est automatiquement ajustée à la taille de la police. Les champs de texte multilignes sont alors affichés sous forme de lignes uniques à faire défiler. [RowHeight]

Comme pour les tables, le bord inférieur du contrôle affiche le numéro d'enregistrement et les aides à la navigation.[HasNavigationBar] [HasNavigationBar]

Par défaut, il y a un marqueur d'enregistrement sur le bord gauche du contrôle. Il indique l'enregistrement actuel. Vous pouvez utiliser le marqueur d'enregistrement pour accéder à la fonction de suppression de l'ensemble de l'enregistrement. [HasRecordMarker]

Onglet Données

Puisqu'il s'agit d'un champ qui ne contient pas de données mais gère d'autres champs, il n'y a pas de propriétés de données. Les données sont définies dans les colonnes.

Onglet Événements

Les événements *Modifié* et *Texte modifié* sont absents. L'événement *Erreur survenue* est ajouté.

Contrôle Étiquette

En plus des propriétés listées à la page 164, les fonctionnalités suivantes sont disponibles.

Onglet Général

Le titre agit comme une description d'un autre contrôle. Si le titre est lié à un contrôle, il peut être utilisé comme raccourci vers celui-ci. Un «~» inséré indique une lettre particulière qui devient le raccourci "No ~ m" définit "m" comme cette lettre. Lorsque le curseur se trouve n'importe où dans le formulaire, Alt + m va directement dans le champ.

Par défaut, une étiquette n'est pas scindée. S'il est trop long pour le champ, il est tronqué. Attention : le retour à la ligne ne reconnaît pas les espaces, donc si le champ est trop petit, une rupture peut se produire dans un mot. [MultiLine]

Onglet Données

Absent.

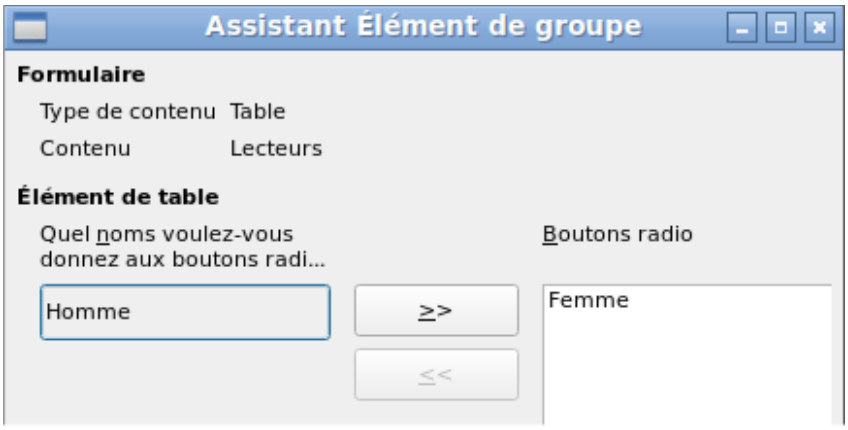
Onglet Événements

Le champ d'étiquette réagit uniquement aux événements liés à la souris, à une touche ou au focus.

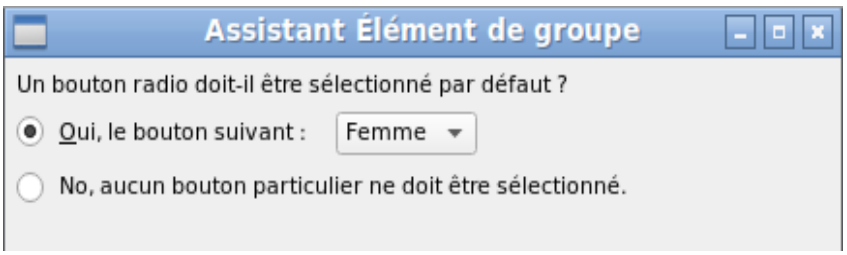
Zone de groupe

Une zone de groupe regroupe graphiquement plusieurs champs et leur fournit une étiquette collective.

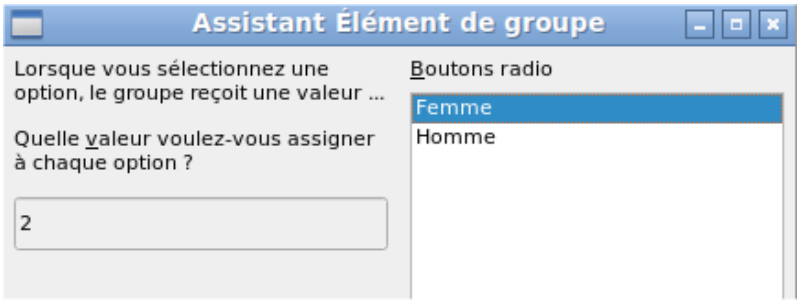
Si une zone de groupe est créée avec des Assistants actifs, l'Assistant part de l'hypothèse que plusieurs boutons d'option apparaîtront ensemble dans ce cadre.



Ce formulaire est basé sur la table Lecteurs. Nous avons affaire au choix du sexe. Les entrées sont les étiquettes des boutons d'option.

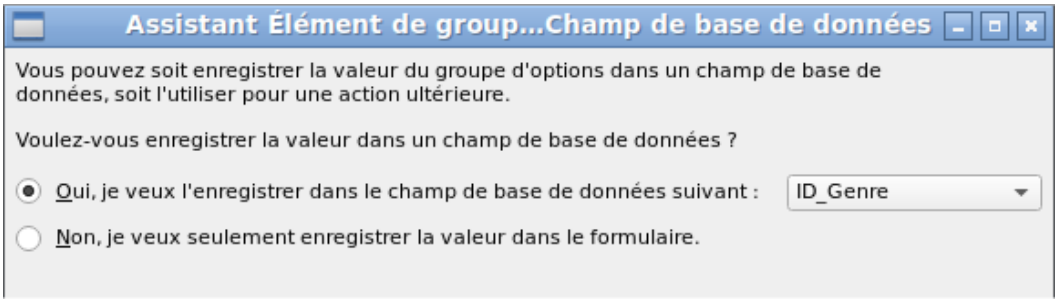


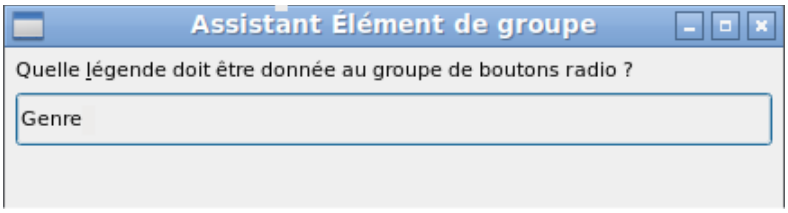
Ici, l'option par défaut est "femme". S'il n'y a pas de champ par défaut, l'entrée par défaut dans la table sous-jacente est NULL.



L'assistant donne aux boutons d'option des valeurs séparées par défaut, ici 1 pour homme et 2 pour femme. Ces valeurs correspondent aux exemples de champs de clé primaire dans la table Genres.

La valeur sélectionnée en cliquant sur un bouton d'option est transférée dans le champ ID_Genre de la table Lecteurs sous-jacente du formulaire. De cette manière, la table Lecteurs est fournie avec la clé étrangère correspondante de la table Genres en utilisant le bouton d'option.





Le groupe de boutons d'option reçoit une zone de groupe (cadre) avec l'étiquette Genre.

Si femme est sélectionnée dans le formulaire actif, l'homme est désélectionné. Il s'agit d'une caractéristique des boutons d'option qui sont liés au même champ dans la table sous-jacente. Dans l'exemple ci-dessus, les boutons d'option remplacent une zone de liste à deux éléments.

En plus des propriétés listées à la page 164, les fonctionnalités suivantes sont disponibles.

Onglet Général

L'étiquette peut être modifiée par rapport à sa valeur par défaut. À l'heure actuelle, les propriétés du cadre (épaisseur de ligne, couleur de ligne) ne peuvent pas être modifiées, mais vous pouvez modifier la mise en forme de la police.

Onglet Données

Aucun, car ce contrôle sert uniquement au regroupement visuel des champs.

Onglet Événements

La zone de groupe réagit aux événements impliquant la souris, une touche ou le focus.

Bouton poussoir

En plus des propriétés listées à la page 164, les fonctionnalités suivantes sont disponibles.

Onglet Général

Étiquette sur le bouton.

En insérant «~», une lettre spécifique de l'étiquette peut être transformée en raccourci. "Bou~ton" définit "t" comme raccourci. Lorsque le curseur est n'importe où dans le formulaire, Alt + t vous amène au bouton.
[Label]

Le bouton reçoit le focus lorsque l'utilisateur clique dessus.

Cela n'est peut-être pas toujours souhaitable. Par exemple, vous pouvez vouloir cliquer sur le bouton pour placer du contenu dans un champ de formulaire différent, auquel cas le curseur doit rester dans ce champ lorsque vous cliquez sur le bouton.
[FocusOnClick]

Si Oui, le bouton peut être affiché enfoncé. L'état du bouton est indiqué comme pour un interrupteur. Lorsque vous appuyez dessus une deuxième fois, il affiche un bouton non enfoncé.
[Toggle]

Actif, lorsque Basculer est défini sur Oui. Sélectionné correspond au bouton enfoncé.
[DefaultState]

Habillage de mots si le bouton est trop étroit. [MultiLine]

Une variété d'actions similaires à celles de la barre de navigation sont disponibles. Agissent sur le formulaire et la base sous jacente.

Si l'action consiste à ouvrir un document ou un site Web, placez l'URL dans le champ suivant.

HTML : Fichier à appeler avec ce bouton. Ici, vous devez choisir la ressource qui doit être ouverte par "Ouvrir le document / site Web" sous "Action".

En plus du HTML, "Action" peut être utilisé pour ouvrir d'autres modules LO. Donc si vous entrez ici.uno : RecSearch, la fonction de recherche du navigateur sera assignée au bouton.

Vous pouvez également ouvrir les fichiers Writer afin que le fait d'appuyer sur le bouton effectue un publipostage à l'aide des enregistrements de la base de données.

Les commandes disponibles ici peuvent être déterminées à l'aide de l'enregistreur de macros. [TargetURL]

Uniquement pour les formulaires HTML : cadre cible (disposition des cadres pour différentes pages HTML) dans lequel le fichier doit être ouvert. [TargetFrame]

Le bouton par défaut est encadré lorsqu'il est défini sur Oui. Lorsqu'il existe plusieurs boutons alternatifs sur un formulaire, celui qui est le plus souvent utilisé doit avoir cette caractéristique. Il est activé en appuyant sur la touche Entrée, lors qu'aucune autre action ne doit dépendre de cette touche. Un seul bouton du formulaire peut être le bouton par défaut.
[DefaultButton]

Une image doit-elle apparaître sur le bouton? [Graphic] [ImageURL]

Actif uniquement lorsqu'une image a été définie. Spécifie l'alignement du graphique sur le texte. [ImagePosition] [ImageAlign]

Onglet Données

Aucun. Un bouton effectue uniquement des actions.

Onglet Événements

Accepter l'action, *Exécuter l'action*, et *Item Statut de l'élément midifié*.

Bouton picto

En plus des propriétés listées à la page 164, les fonctionnalités suivantes sont disponibles.

Onglet Général

Similaire à un bouton normal. Cependant, ce bouton n'a pas de texte et le bouton lui-même n'est pas visible. Vous ne voyez qu'un cadre autour de l'image.

Par défaut, un bouton d'image n'a pas de tabulation.

Attention : au moment de la rédaction de cet article, pratiquement aucune action ne fonctionne avec ce bouton. Il n'est pratiquement utilisable qu'avec des macros.

Onglet Données

Aucun ; ce contrôle ne réalise que des actions.

Onglet Événements

Accepter l'action et tous les événements impliquant la souris, une touche ou le focus.

Barre de navigation

Figure 70: Contrôle barre de navigation

La barre de navigation de formulaire standard est insérée dans les formulaires sur le bord inférieur de l'écran. L'insertion de cette barre d'outils peut provoquer un bref décalage vers la droite du formulaire lors de sa création à l'écran. Cela peut être gênant dans les cas où la barre de navigation est à nouveau désactivée pour certaines parties du formulaire visible, par exemple lorsqu'il existe des sous-formulaires ou plusieurs formulaires dans le formulaire visible.

En revanche, un contrôle de barre de navigation qui fait partie du formulaire, distinct des éléments correspondants, indique clairement les éléments dans lesquels vous naviguez avec la barre d'outils. Le formulaire pour les prêts, par exemple, doit d'abord rechercher parmi les lecteurs, puis montrer les médias prêtés au lecteur. Le contrôle de la barre de navigation est positionné à proximité du lecteur, de sorte que l'utilisateur remarque que la barre de navigation est utilisée pour le lecteur et non pour le média prêté au lecteur.

La barre de navigation de formulaire standard met à disposition les boutons illustrés dans la figurer 71. Le contrôle de la barre de navigation affiche les mêmes boutons à l'exception de ceux de Rechercher un enregistrement, des filtres basés sur un formulaire et une source de données sous forme de tableau.

En plus des propriétés listées à la page 164, les fonctionnalités suivantes sont disponibles.

Onglet Général

La taille de l'icône est réglable. De plus, vous pouvez choisir les groupes à afficher. Ceux-ci sont illustrées dans la figure 70 de gauche à droite en utilisant une ligne verticale comme séparateur de groupe : positionnement, navigation, action sur un enregistrement et groupes de commandes pour le filtrage et le tri.
[IconSize]
[ShowPosition]
[ShowNavigation]
[ShowRecordActions]
[ShowFilterSort]

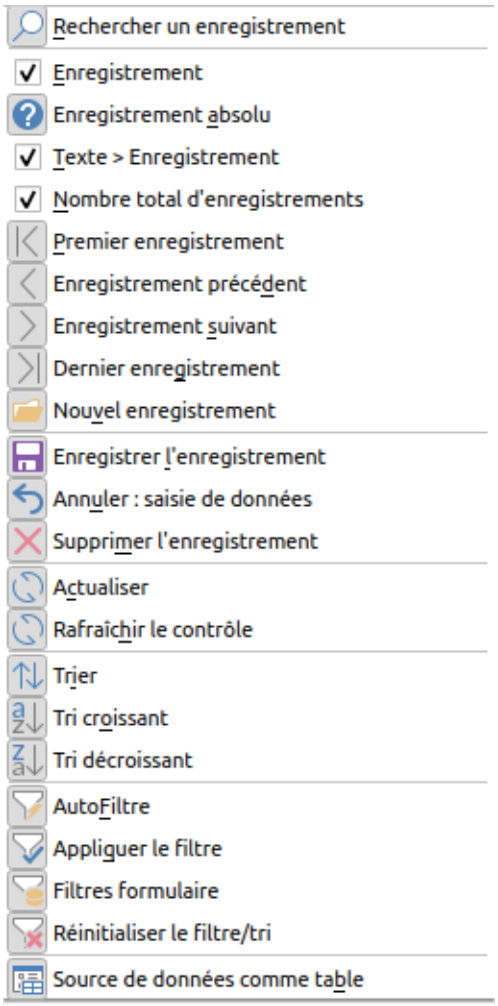


Figure 71: Boutons de navigation

Onglet Données

Aucun, car ce contrôle n'effectue que des actions.

Onglet Événements

Tous les événements qui impliquent la souris, une touche ou le focus.

Indépendamment de ce contrôle de formulaire, la barre de navigation insérable continue naturellement d'exister avec les mêmes éléments que la figure ci-dessus.

Cette barre de navigation insérable fournit en outre la recherche générale d'enregistrements, le filtre basé sur le formulaire et l'affichage de la source de données sous-jacente du formulaire en vue tableau au-dessus du formulaire.

Si vous travaillez non seulement avec un formulaire, mais avec des sous-formulaires et des formulaires auxiliaires, vous devez faire attention à ce que cette barre de navigation insérable ne disparaisse pas lorsque vous changez de formulaire. Cela crée un effet dérangeant sur l'écran.

Compteurs et barres de défilement

Aucun de ces contrôles n'a de connexion directe aux données de la base de données. Ils ne peuvent être utilisés qu'au moyen de macros, où le bouton compteur est intégré, par exemple, dans un champ numérique.

Une barre de défilement diffère d'un compteur en ayant un curseur en plus des boutons pour incrémenter et décrémenter la valeur dans une plage fixe.

Lorsque le formulaire est ouvert, la barre de défilement doit être informée de la valeur actuelle du champ auquel elle est liée. Cette macro est liée à **Propriétés du formulaire > Événements > Après le changement d'enregistrement.**

```
SUB Defilement_Formulaire(oEvent AS OBJECT)
	DIM oFormulaire AS OBJECT
	DIM oChamp AS OBJECT
	oFormulaire = oEvent.Source
	oChamp = oFormulaire.getByName("BarreDefilement")
	oChamp.ScrollValue = oFormulaire.getInt(oFormulaire.findColumn("Champ_Numerique"))
END SUB
```

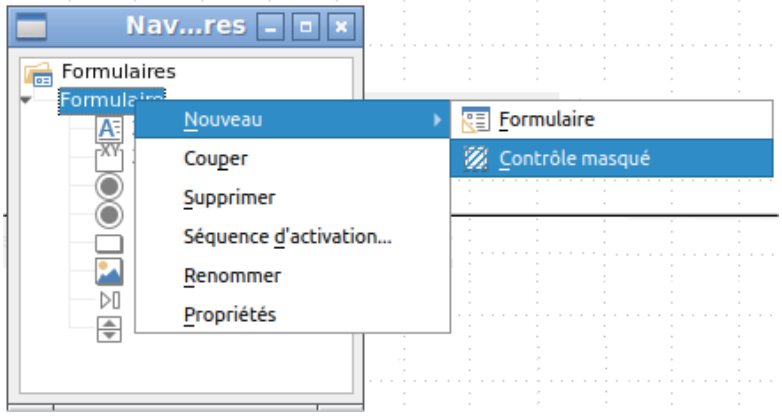
La variable est d'abord déclarée. L'événement qu'il gérera est spécifié sur le formulaire auquel la macro sera éventuellement liée. La barre de défilement sera trouvée par son nom sur le formulaire. Sa valeur actuelle est définie sur le nombre stocké dans Champ_Numerique dans la table sous-jacente. Cela peut être une clé primaire.
Si la valeur est modifiée à l'aide de la barre de défilement, cette modification doit être transmise au champ sous-jacent. Cette macro est liée à **Propriétés : Barre de défilement > Événements > A l'ajustement**.

```
SUB Defilement_Change(oEvent AS OBJECT)
	DIM oFormulaire AS OBJECT
	DIM oChamp AS OBJECT
	DIM inValeur AS INTEGER
	oChamp = oEvent.Source.Model
	inValeur = oEvent.Value
	oFormulaire = oChamp.Parent
	oFormulaire.updateInt(oFormulaire.findColumn("Champ_Numerique"), inValeur)
END SUB
```

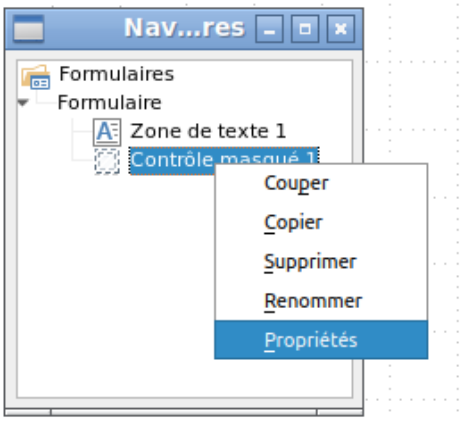
Les variables sont d'abord déclarées. La valeur sélectionnée est lue comme un entier dans le champ. Le champ correspondant dans la table sous-jacente Champ_Numerique est ensuite mis à jour à cette valeur.

Une explication détaillée de ce type de code est donnée au Chapitre 9, Macros.

Contrôle masqué



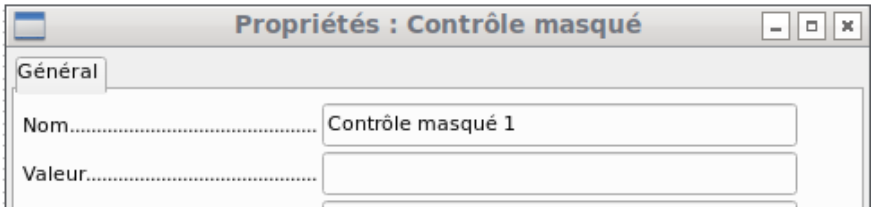
Un type de contrôle qui ne peut pas être inséré à partir de la barre d'outils contrôles est le contrôle masqué. Il doit être créé dans le navigateur de formulaire, à l'aide du menu contextuel du formulaire.



Comme tout autre contrôle, un contrôle masqué fait partie du formulaire. Cependant, il n'est pas visible dans l'interface utilisateur.

Un contrôle masqué n'a que quelques propriétés. Le nom et les informations supplémentaires ont la même signification que pour les autres contrôles. De plus, le contrôle peut se voir attribuer une valeur [Hidden Value].

Un contrôle masqué ne sert à rien si vous n'utilisez pas de macros. Cependant, avec les macros, il est souvent utile de pouvoir stocker des valeurs intermédiaires quelque part sur le formulaire, pour y accéder ultérieurement. Un exemple de cette façon d'utiliser les macros est décrit dans le chapitre 9, Macros, dans la section "Zones de liste hiérarchiques".



Sélection multiple

Si vous utilisez le bouton **Sélectionner** (flèche) pour sélectionner une grande région ou plusieurs éléments d'un formulaire, les modifications suivantes peuvent être effectuées.

Vous ne devez pas modifier le nom. Cela amènerait soudainement tous les éléments sélectionnés à acquérir le même nom. Cela rendrait difficile la recherche d'éléments individuels à l'aide du Navigateur de formulaires et la gestion du formulaire à l'aide de contrôles nommés dans les macros serait impossible.

La sélection multiple est plus utile pour changer la police, la hauteur ou la couleur d'arrière-plan des contrôles. Notez que la modification de la couleur d'arrière-plan affecte également les étiquettes.

Si vous souhaitez modifier uniquement les libellés, maintenez la touche *Contrôle* enfoncée et cliquez sur ces contrôles directement ou dans le Navigateur, ou cliquez avec le bouton droit sur un champ pour appeler les propriétés du contrôle. Désormais, le choix des propriétés que vous pouvez modifier est plus large, car vous ne traitez que des champs similaires. Vous pouvez modifier ici tout ce qui est disponible dans un champ d'étiquette.

Les possibilités de sélection multiple dépendent donc du choix des champs. Vous pouvez modifier simultanément les contrôles du même type qui ont toutes les propriétés qui existent pour une seule instance.

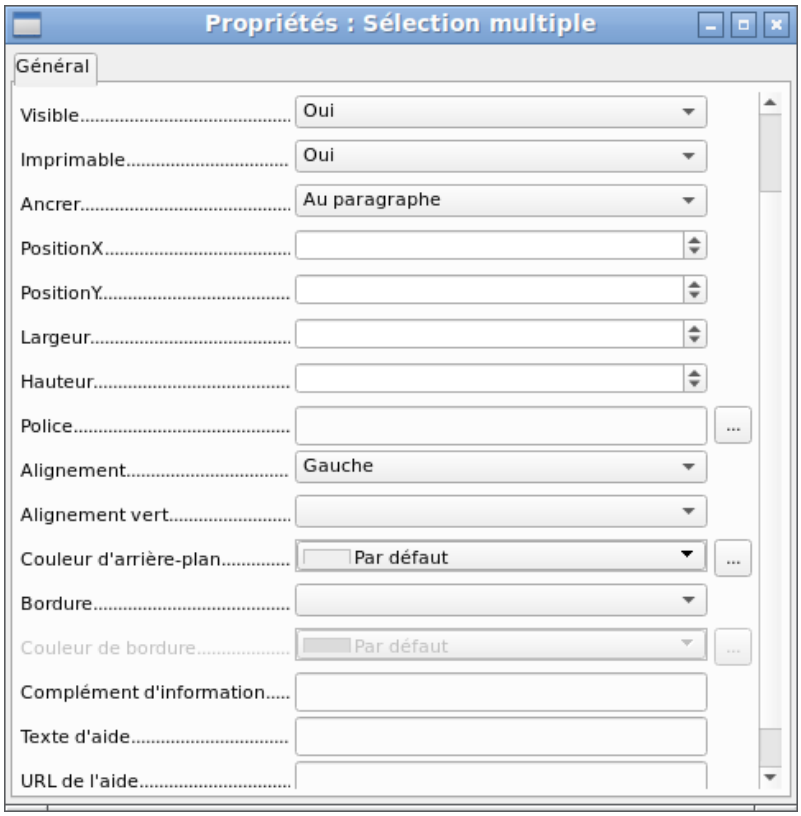
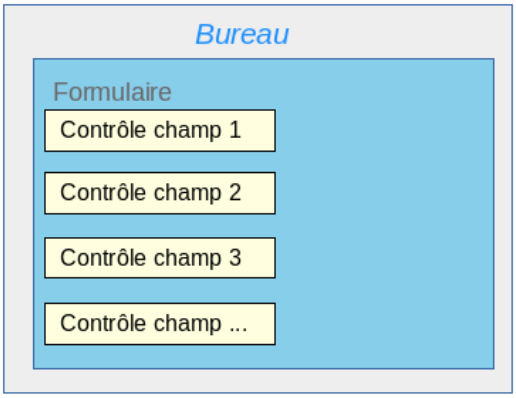


Figure 72: Propriétés générales des champs de formulaire dans une sélection multiple

Un formulaire simple rempli

Un formulaire simple a des contrôles de formulaire pour écrire ou lire des enregistrements à partir d'une seule table ou requête. Sa construction est illustrée par l'exemple suivant.

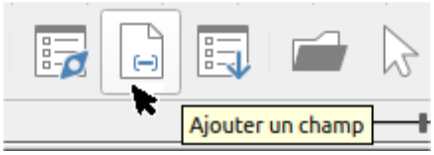


L'exemple d'un formulaire simple pour les prêts de bibliothèque est présenté ici en utilisant plusieurs variantes. La manière rapide d'utiliser l'assistant de création de formulaires est décrite au Chapitre 8, Prise en main de Base, dans le Guide de mise en route. Nous décrivons ici la création en mode Ébauche.

L'en-tête du formulaire a été créé à l'aide d'un champ d'étiquette. La police a été modifiée. Le champ d'étiquette est ancré à un paragraphe dans le coin supérieur gauche du document. À l'aide du menu contextuel du champ d'étiquette, un formulaire a été créé, lié à la table Prêts (voir“Propriétés du formulaire” en page 159). La page a également reçu un fond de couleur uniforme.

Ajout de groupes de champs

Une variante rapide pour la saisie directe de champs avec des étiquettes consiste à utiliser la fonction *Ajouter un champ*.



Cette fonction, disponible dans la barre d'outils de conception de formulaire (voir Figure 61), permet de sélectionner tous les champs de la table sous-jacente.

Double-cliquez sur les champs pour les insérer dans le formulaire en groupe avec des étiquettes (malheureusement tous au même endroit). Le groupe doit être séparé pour que le formulaire ressemble finalement à l'illustration suivante(Cf. Figure 73). Pour une meilleure vue, toutes les barres d'outils inutiles ont été supprimées de la fenêtre, qui a également été compressée afin que tous les éléments de la barre de navigation ne soient pas visibles.

Tous les champs ont été sélectionnés à l'exception de ID_BC_Media, qui est conçu pour être utilisé uniquement avec un lecteur de codes-barres.

Conseil

Juste après avoir ajouté un champ, le déplacer sur la feuille (vers sa position définitive, si possible), pour éviter la superposition avec ceux créés ultérieurement ou le sélectionner et le glisser dans le formulaire.

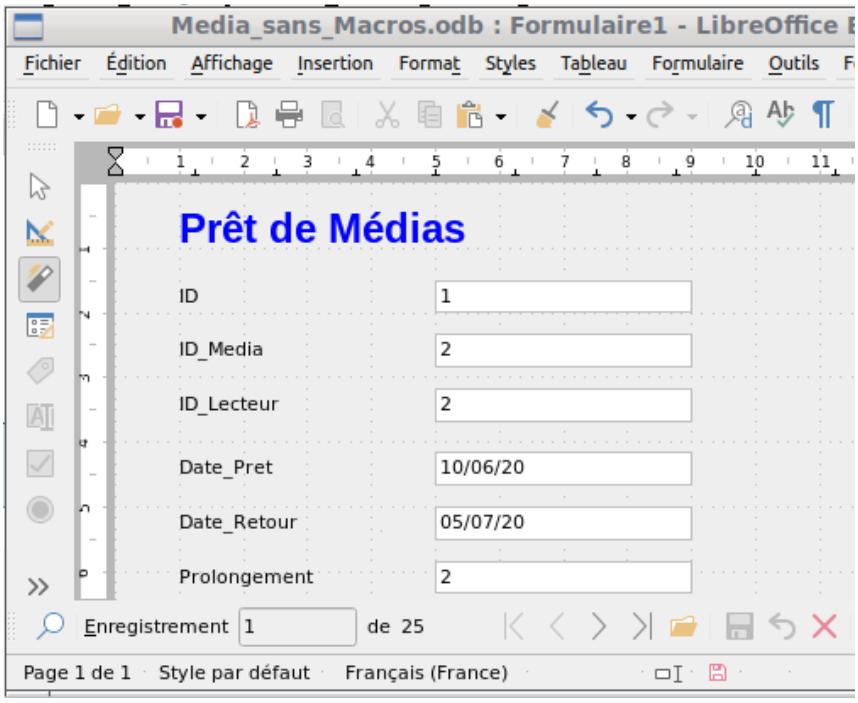


Figure 73: Formulaire simple créé en utilisant Ajouter un champ

Pour chaque champ de la table, un contrôle de formulaire approprié a été automatiquement sélectionné. Les nombres sont dans des champs numériques et sont déclarés comme des entiers sans décimales. Les champs de date sont correctement représentés en tant que contrôles de date. Tous les champs ont la même largeur. Si un contrôle graphique avait été inclus, il aurait reçu un champ carré. Voir l'exemple *Form01-VoirPrets de l'exemple Media_sans_Macros.*

Ajustement des proportions de champ

Nous pouvons maintenant faire des choses créatives, y compris ajuster la longueur des champs et transformer les dates en champs déroulants. Il est encore plus important de rendre les champs ID_Media et ID_Lecteur plus conviviaux, à moins que chaque utilisateur de bibliothèque ne dispose d'une carte d'identité de bibliothèque et que chaque support soit fourni avec un ID lors de l'adhésion. Cela ne sera pas supposé dans ce qui suit.

Pour ajuster les champs individuels, nous devons éditer le groupe. Cela peut être fait avec un clic droit sur le groupe puis en suivant le menu contextuel (Figure 74). Pour les travaux futurs, ce sera plus clair si nous utilisons le Navigateur de formulaires.

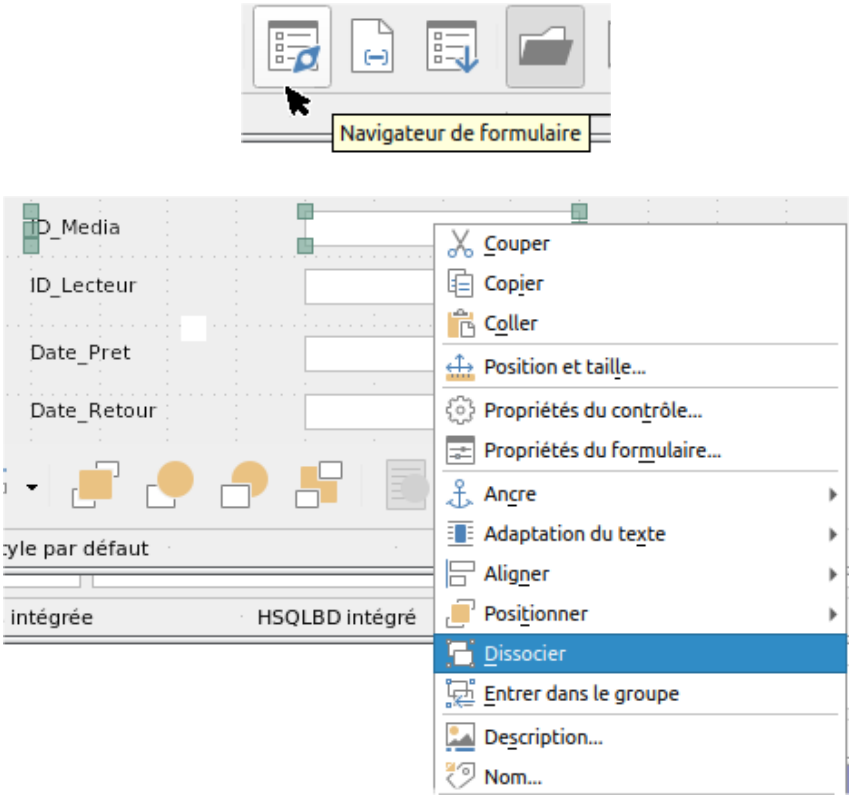


Figure 74: Contrôles : Modification du groupe Contrôle/Etiquette

Le navigateur de formulaire affiche tous les éléments du formulaire avec leurs étiquettes. Pour les contrôles, les noms proviennent directement des noms des champs de la table sous-jacente. Les noms des étiquettes ont le préfixe Étiquette.

Un clic sur ID_Media sélectionne ce champ (Figure 75). Cliquez avec le bouton droit de la souris pour remplacer le champ sélectionné par un autre type de champ, à l'aide du menu contextuel (Figure 76).

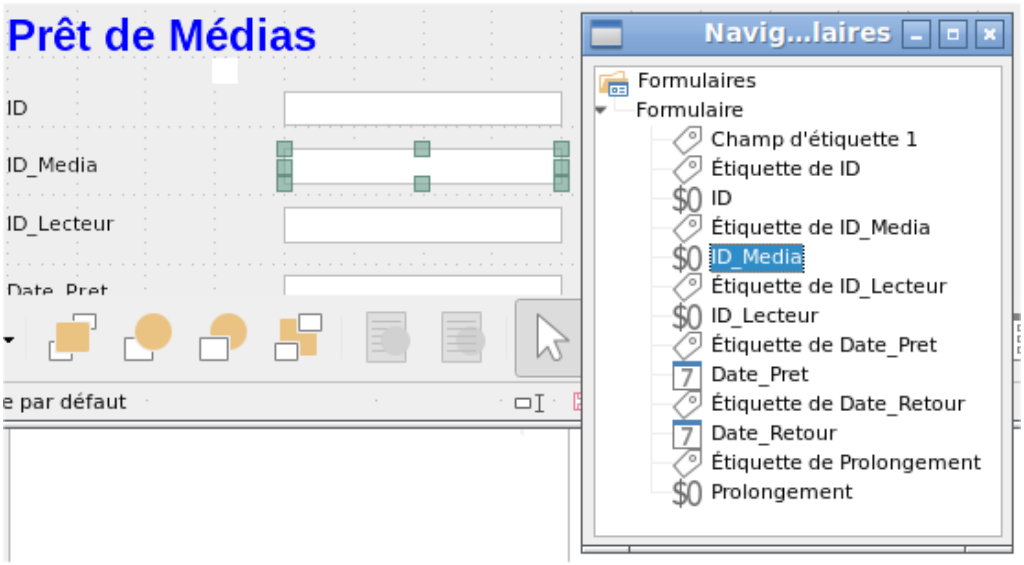


Figure 75: Sélection des contrôles de formulaire directement à l'aide du navigateur de formulaires

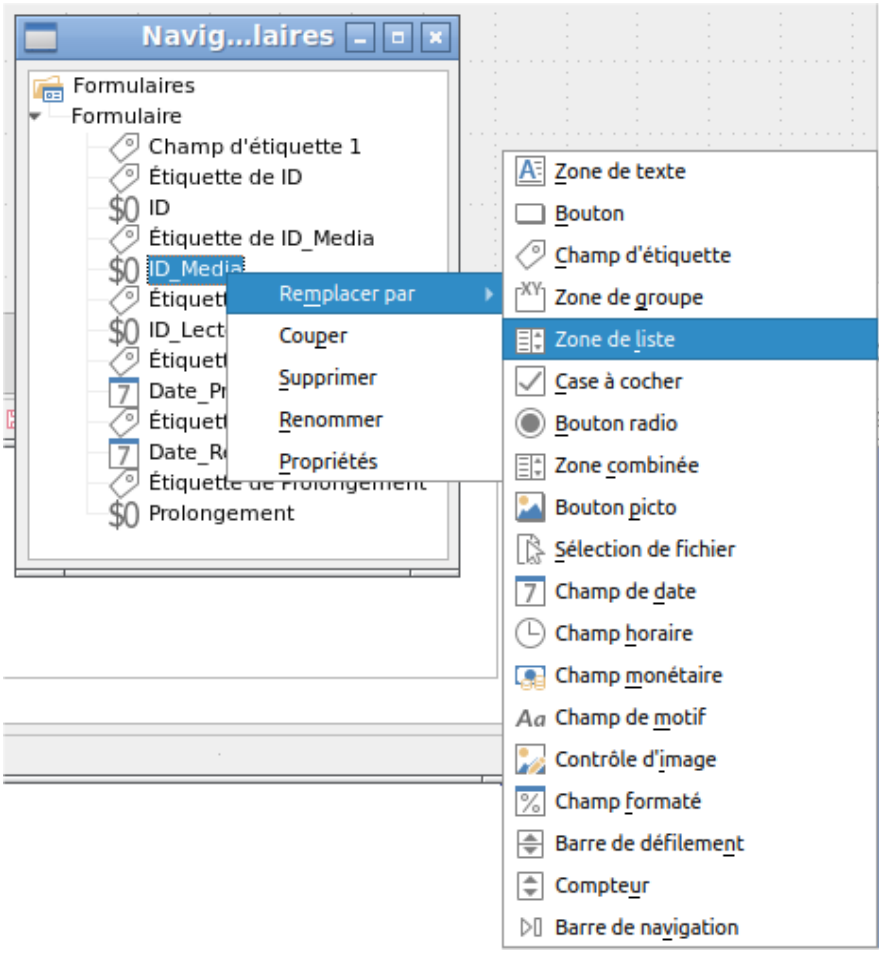
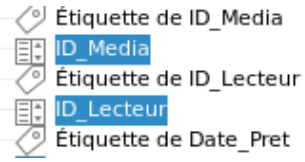


Figure 76: Remplacement d'un type de contrôle par un autre à l'aide du navigateur de formulaires

Ce remplacement est effectué pour les contrôles ID_Media et ID_Lecteur.



La modification est rendue visible dans le Navigateur de formulaires par la modification de l'icône associée.

La requête SQL pour le champ de liste peut maintenant être créée via l'interface utilisateur graphique en cliquant sur le bouton à droite.

Ceci est effectué automatiquement lorsqu'une zone de liste est créée directement, mais pas lorsqu'elle est formée par conversion à partir d'un autre type de contrôle.

Les contrôles Zone de liste permettent d'afficher autre chose que les champs ID sans utiliser de macro.

Les requêtes :

- Pour le champ ID_Media (étiquette "Media")
  ```
  SELECT "Titre", "ID" FROM "Medias"
  ```
- Pour le champ ID_Lecteur (étiquette "Lecteur")
  ```
  SELECT "Nom" || ', ' || "Prenom","ID" FROM "Lecteurs"
  ```

Pour plus de détails sur la commande SQL, reportez-vous au Chapitre 5, Requêtes. Voir l'exemple *Form02-VoirPretsListe de l'exemple Media_sans_Macros.*

Figure 77: Remplacement par zones de liste et requêtes SQL

Étant donné que les listes déroulantes doivent être créées, les défauts suivants peuvent être corrigés en même temps :

- Les étiquettes des zones de liste doivent être Media au lieu de ID_Media et Lecteur au lieu de ID_Lecteur.
- Le contrôle ID doit être déclaré en lecture seule.
- Les champs qui ne sont pas absolument nécessaires pour l'octroi de prêts pour un nouveau support n'ont pas besoin d'un taquet de tabulation. Sans cela, le formulaire peut être parcouru beaucoup plus rapidement. Si nécessaire, la tabulation peut également être ajustée à l'aide de la séquence d'activation (voir page 164). Seuls les champs Media, Lecteur et Date_Pret doivent être accessibles dans tous les cas à l'aide de la touche Tab.
- Si le formulaire est destiné à la réalisation de prêts, il est inutile et également déroutant pour les médias retournés d'être affichés. Les supports avec une date de retour doivent être filtrés. De plus, l'ordre d'affichage pourrait être trié par lecteur, de sorte que les médias prêtés à la même personne soient affichés successivement. Voir la note sur“Propriétés du formulaire” en page 159. Cependant, il y a un problème ici en ce que les lecteurs peuvent être triés uniquement par ID, pas par ordre alphabétique, car la table sous-jacente au formulaire contient uniquement l'ID.

Ajout de champs uniques

L'ajout de champs uniques est un peu plus compliqué. Les champs doivent être sélectionnés, glissés sur la surface du formulaire et le champ approprié de la table sous-jacente spécifié. De plus, le type de champ doit être correctement choisi ; par exemple, les champs numériques ont deux décimales par défaut.

Ce n'est que lors de la création de zones de liste que l'assistant entre en jeu, ce qui permet à un novice d'effectuer plus facilement les étapes de création des champs appropriés, jusqu'à un certain point. Au-delà de ce point, l'assistant cesse de répondre aux exigences car :

- Les entrées ne sont pas triées automatiquement.
- La combinaison de plusieurs champs dans le contenu de la zone de liste n'est pas possible.

Ici encore, nous devons apporter des améliorations rétrospectives, afin que le code SQL requis puisse être créé assez rapidement à l'aide de l'éditeur de requête intégré.

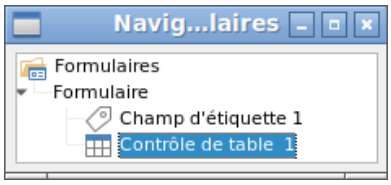
Lors de l'ajout de champs uniques, le champ et son libellé doivent être explicitement associés (voir “Paramètres par défaut pour de nombreux contrôles” en page 164). En pratique, il pourrait être préférable de ne pas associer de champs aux étiquettes, de sorte que vous n'ayez pas à utiliser la modification de groupe avant de modifier les propriétés d'un champ. Cela dit, Un champ d'une

association Champ-étiquette est toujours accessible par ctrl-clic sur ce champ, ce qui facilite les modifications

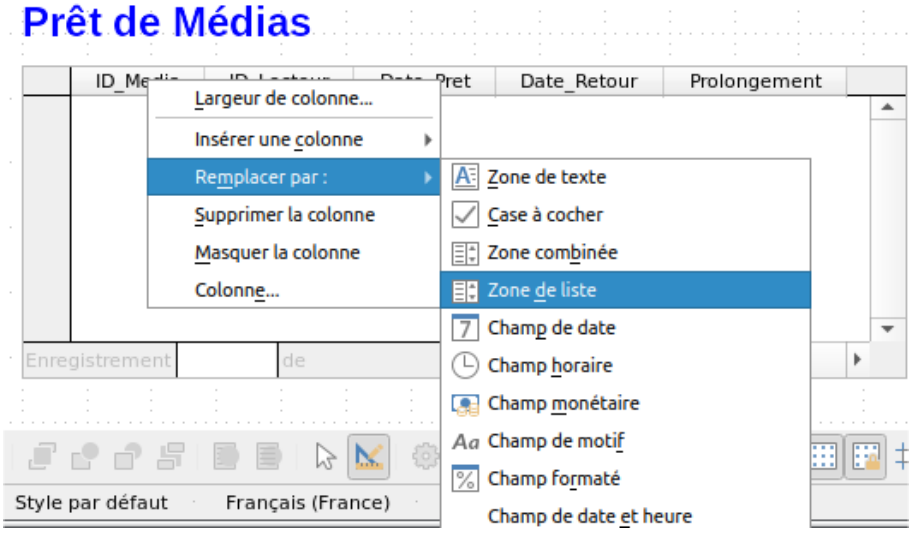
Contrôle de table

L'utilisation de l'assistant de table pour créer un champ de table a déjà été décrite à la page 185. Il présente cependant quelques défauts qui doivent être améliorés :

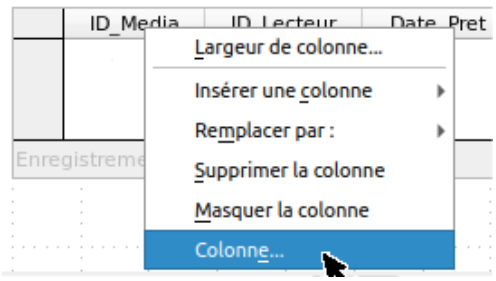
- Les champs ID_Media et ID_Lecteur doivent devenir des zones de liste.
- Les champs numériques doivent être débarrassés de leurs décimales, car l'assistant spécifie toujours deux décimales pour les nombres.



La modification des champs dans le contrôle de table n'est pas possible en utilisant la même méthode que celle décrite pour les autres contrôles. Dans le navigateur, la description des champs se termine par le champ Table. Le Navigateur ne sait rien des contrôles qui se trouvent dans le contrôle de table, faisant référence aux champs de la table sous-jacente. Cela s'applique également plus tard, lorsque des tentatives sont faites pour accéder aux champs à l'aide de macros. Ils ne sont pas accessibles par leur nom.



Les contrôles dans le contrôle de table sont appelés colonnes. Grâce au menu contextuel, il est désormais possible de remplacer un type de champ par un autre. Cependant, toute la gamme des types n'est pas disponible. Il n'y a pas de boutons poussoirs, de boîtes d'options ou de commandes graphiques.



Les propriétés des champs sont masquées dans le menu contextuel derrière le concept de colonnes. Ici, par exemple, le champ numérique Prolongement peut être modifié afin qu'aucune décimale ne soit affichée. De plus, la valeur minimale par défaut de –1 000 000,00 n'a guère de sens pour une prolongation de prêt. Le nombre doit toujours rester petit et positif.

Dès que les propriétés d'une colonne sont affichées, vous pouvez sélectionner une autre colonne sans fermer la boîte de dialogue des propriétés. De cette façon, vous pouvez travailler sur tous les champs, l'un après l'autre, sans avoir à enregistrer entre les deux.

Note

La valeur est enregistrée dans la boîte de dialogue dès qu'un mouvement est effectué entre les propriétés. Tel est également le cas lors du passage d'une colonne à la suivante.

Terminez en sauvegardant le formulaire entier, et enfin la base de données elle-même.

Les propriétés des champs intégrés dans un contrôle de table ne sont pas aussi complètes que celles des champs individuels . La police, par exemple, ne peut être définie que pour l'ensemble du contrôle de table. De plus, vous n'avez pas la possibilité de sauter des colonnes individuelles en supprimant leurs taquets de tabulation.

Conseil

Vous pouvez vous déplacer dans un formulaire à l'aide de la souris ou de la touche Tab. Si vous tabulez dans un contrôle de table, le curseur déplacera un champ vers la droite pour chaque tabulation supplémentaire. À la fin de la ligne, il reviendra au premier champ de l'enregistrement suivant dans le contrôle de table. Pour quitter le contrôle de table, utilisez Ctrl + Tab.

Si seuls certains champs doivent être visibles pendant l'utilisation, vous pouvez utiliser plusieurs contrôles de table différents dans le formulaire, car l'onglet est capturé par défaut par chaque contrôle de table.

Le formulaire illustré à la figure 78 concerne le prêt de médias. Seuls les champs immédiatement nécessaires sont affichés dans le champ de table supérieur. La partie inférieure montre tous les champs, de sorte qu'il soit évident à quelle personne et à quel support le retour est destiné.

Note

Les colonnes ID_Media et ID_Lecteur ont été remplacées par des zones de liste, et leur zone de données remplacées par des requêtes SQL identiques à celles de la page 201. Les étiquettes des colonnes ont été modifiées en conséquence.

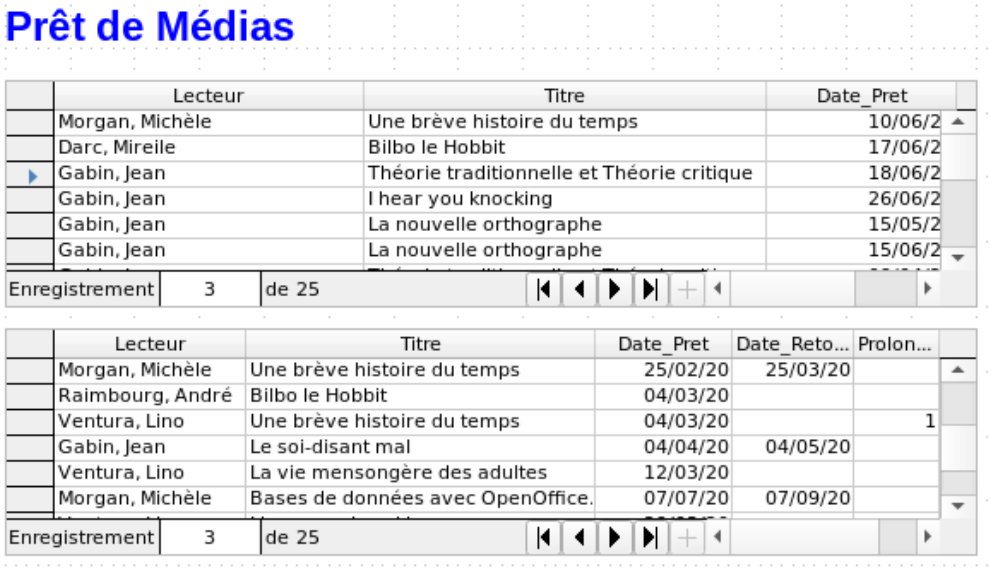
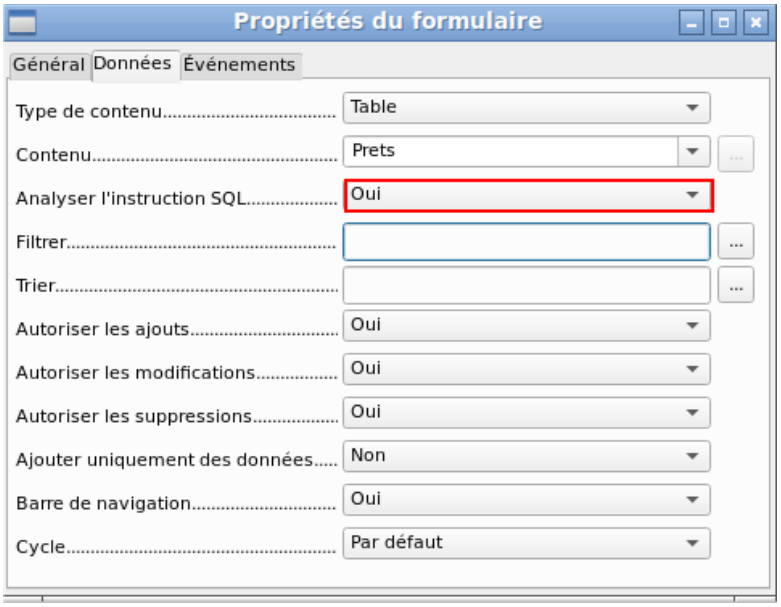


Figure 78: Un formulaire avec plusieurs contrôles de table

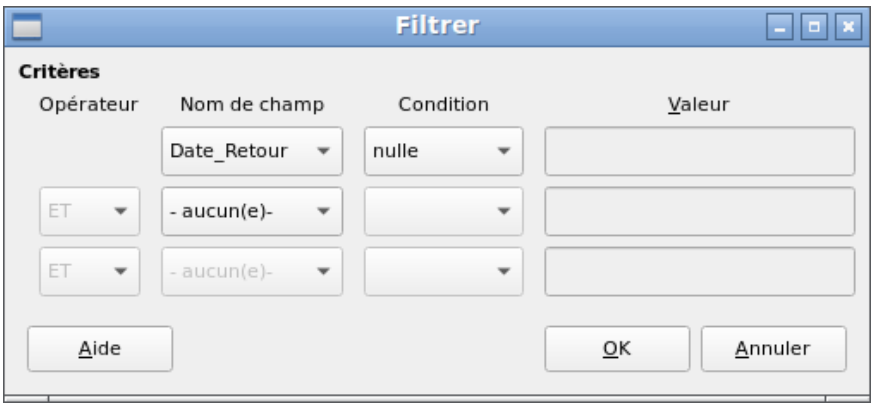
Cette figure montre un défaut esthétique qui nécessite une attention particulière. Dans le contrôle de table supérieur, le même support apparaît parfois plus d'une fois. Cela se produit, car le tableau affiche également les médias qui ont été rendus précédemment. Par conséquent, les données doivent être filtrées pour afficher uniquement les prêts. Les enregistrements avec une date de retour ne doivent pas apparaître.

Ce filtrage peut être réalisé soit par requête, soit directement à l'aide des propriétés du formulaire. Si cela est fait à l'aide des propriétés du formulaire, le filtre peut être temporairement désactivé pendant la saisie. Le filtrage à l'aide d'une requête est décrit au chapitre 5, Requêtes. Nous décrivons ici comment le faire à l'aide des propriétés du formulaire.

Pour avoir accès au Filtre, il faudra valider *Analyser l'instruction SQL*.



Le filtrage est effectué à l'aide du bouton aux trois points (une ellipse, qui ouvre la boîte de dialogue ci-dessous). Vous pouvez également saisir le filtre directement dans le champ de texte Filtre si vous connaissez le codage SQL.



À l'aide de l'interface graphique, vous pouvez maintenant sélectionner le champ nommé Date_Retour. Il affichera uniquement les enregistrements pour lesquels le champ est vide, où "vide" correspond à la désignation SQL NULL.

Le formulaire nettoyé (illustré à la figure 79) semble maintenant plus simple.

Bien sûr, il y a encore place à l'amélioration, mais par rapport à la forme précédente, cette version présente un net avantage en ce que tous les médias sont visibles d'un coup d'œil.

Le traitement des données à l'aide de contrôles de table est similaire à l'utilisation d'une table réelle. Un clic droit sur l'en-tête de l'enregistrement d'un enregistrement existant entraîne sa suppression, et une entrée peut être annulée ou enregistrée dans le cas de nouveaux enregistrements.

Lorsque vous quittez une ligne, l'enregistrement est automatiquement sauvegardé.

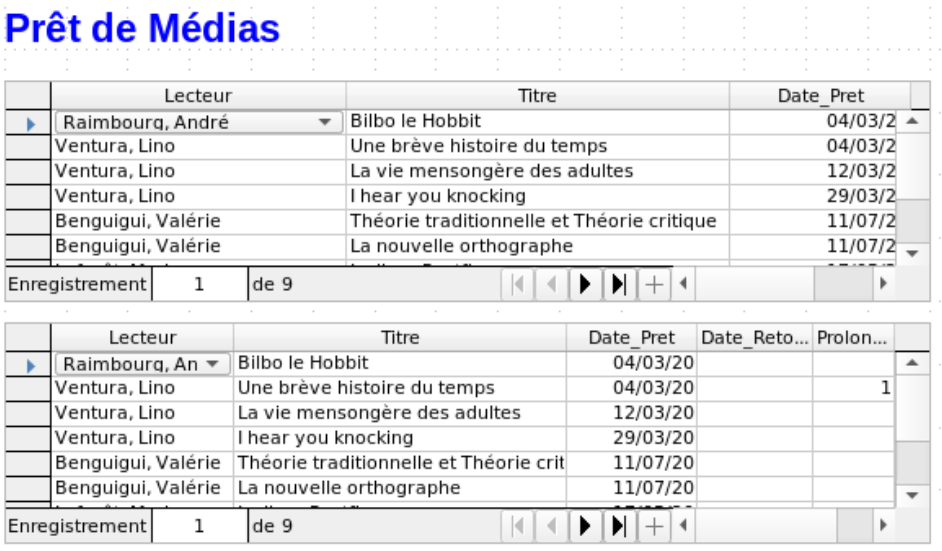


Figure 79: Formulaire modifié avec le filtre

Nous pouvons encore améliorer le formulaire de prêt de médias de plusieurs façons.

- Ce serait bien si la sélection d'un lecteur dans une partie du formulaire provoquait l'affichage du média prêté à ce lecteur dans une autre.
- Dans le tableau ci-dessus, vous pouvez voir un grand nombre d'enregistrements qui ne sont pas nécessaires, car ces supports sont déjà prêtés. Le tableau a été créé pour permettre l'octroi de prêts, il serait donc préférable que seule une page vide apparaisse, qui pourrait ensuite être remplie avec le nouveau prêt.

De telles solutions sont disponibles en utilisant d'autres formulaires qui sont organisés de manière hiérarchique et permettent des vues séparées des données. Voir l'exemple *Form03-VoirPretsFiltre*

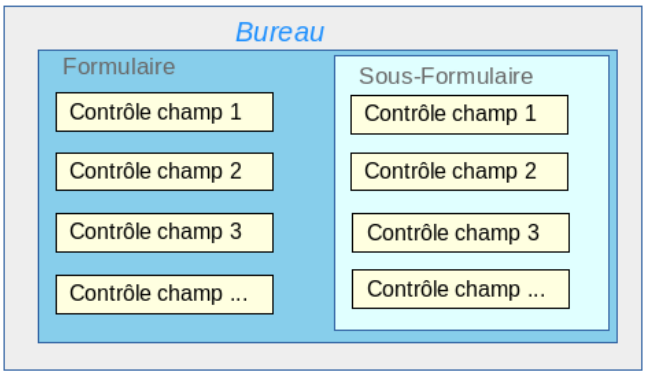
Conseil

Les contrôles de table peuvent être combinés avec d'autres champs du formulaire. Par exemple, le champ de contrôle de table affiche uniquement les titres des supports pour lesquels des modifications ont été apportées dans les champs de formulaire sous-jacents.

Lorsque les contrôles de table sont combinés avec d'autres champs de formulaire, il existe un petit bogue, qui peut être facilement contourné. Si les deux types de champ sont présents ensemble dans un formulaire, le curseur se déplace automatiquement des autres champs dans le contrôle de table même si par défaut ce type de champ a le paramètre **Tabulation > Non**. Cela peut être résolu en basculant la propriété Tabulation sur Oui, puis de nouveau à Non. Cela entraînera l'enregistrement de la valeur "Non".

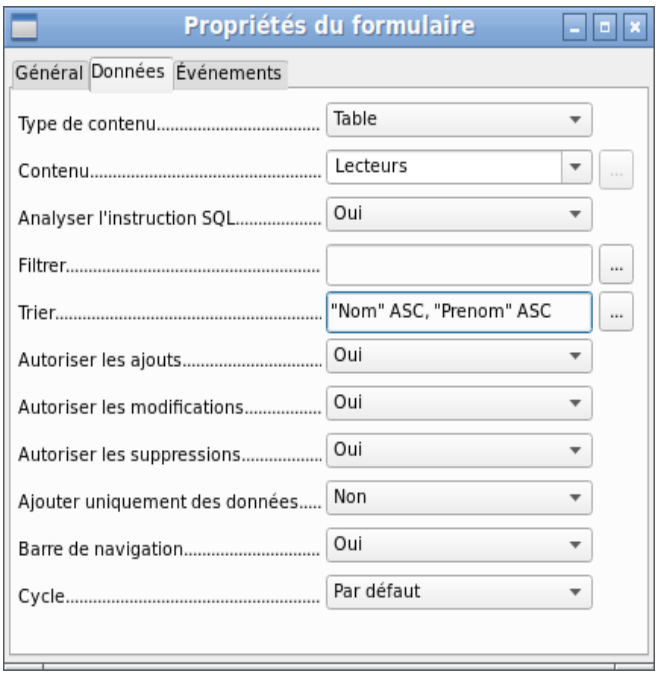
Formulaires principaux et sous-formulaires

Un sous-formulaire se trouve dans un formulaire comme un contrôle de formulaire. Il est lié aux données du formulaire principal. Cependant sa source de données peut être une autre table ou une requête (ou une commande SQL). La chose importante pour un sous-formulaire est que sa source de données est en quelque sorte liée à la source de données du formulaire principal.

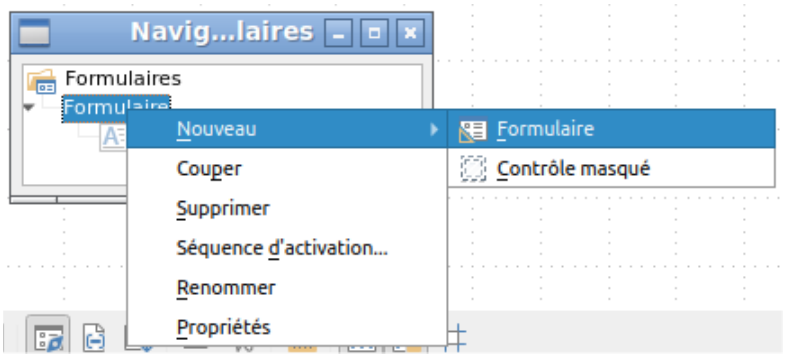


Les structures de table typiques qui se prêtent à être utilisées comme sous-formulaires sont les tables avec une relation un-à-plusieurs (voir Chapitre 3, Tables). Le formulaire principal affiche une table avec des enregistrements auxquels de nombreux enregistrements dépendants du sous-formulaire peuvent être liés et affichés.

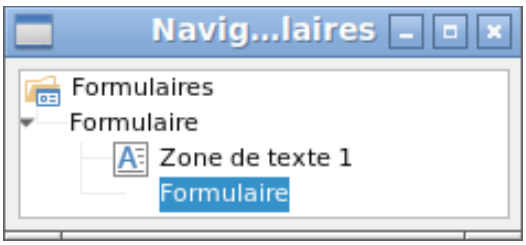
Nous allons d'abord utiliser la relation entre la table Lecteurs et la table Pret (voir Chapitre 3, Tables). La table Lecteurs constituera la base du formulaire principal et la table des Prets sera reproduite dans le sous-formulaire.

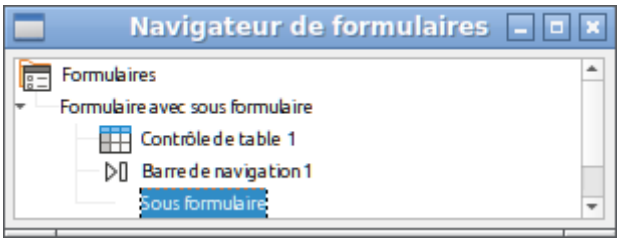


Ici, le formulaire principal est lié à la table Lecteurs. Pour accélérer la recherche de lecteurs, le tableau est trié par ordre alphabétique. Nous nous passerons d'une barre de navigation, puisque le contenu du sous-formulaire se situerait entre le formulaire principal et la barre de navigation. À la place, nous utiliserons le contrôle de formulaire intégré (Figure 70).

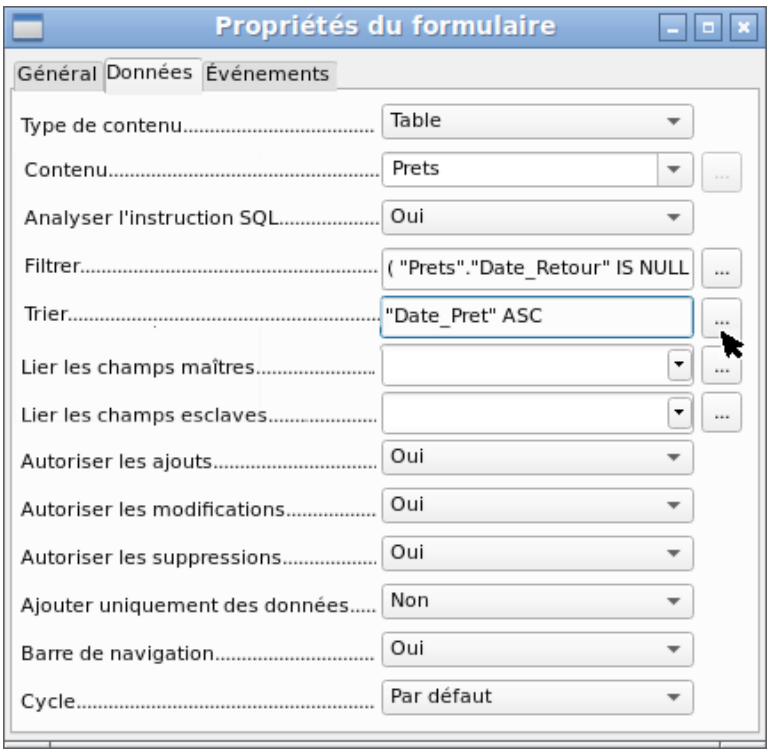


Cliquez avec le bouton droit sur le formulaire principal dans le Navigateur de formulaires pour utiliser le menu contextuel pour créer un nouveau formulaire. Encore une fois, ce formulaire a le nom par défaut de Formulaire, mais il est maintenant un élément dans le sous-dossier du formulaire principal.



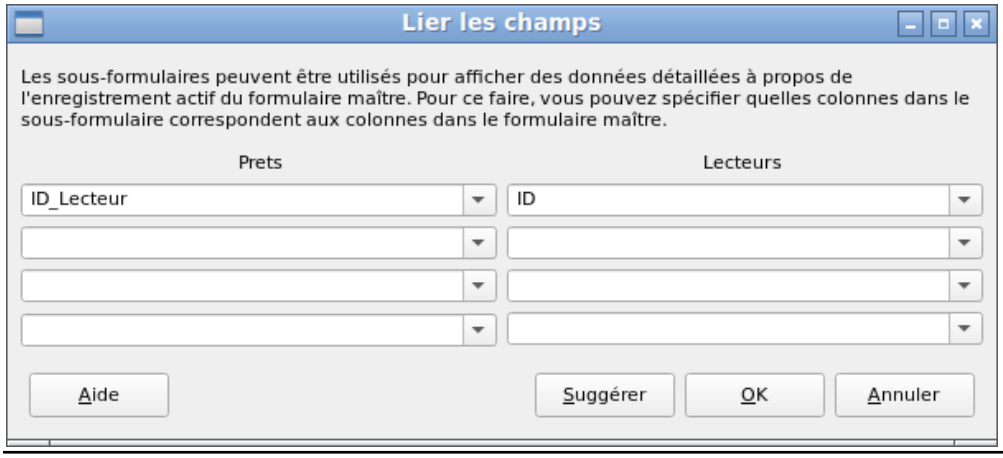


Les propriétés du sous-formulaire doivent maintenant être configurées pour lui donner la bonne source de données, afin de reproduire les données pour le lecteur correct. Il est possible de renommer un formulaire dans l'onglet Général de ses propriétés.



La table Prêts est choisie pour le sous-formulaire. Pour le filtre, nous spécifions que le champ Date_Retour doit être vide (`"Date_Retour" IS NULL`). Cela empêche tout média déjà rendu d'apparaître. Les enregistrements doivent être triés par date de prêt. Le tri croissant montre le support prêté pour la période la plus longue en premier.

Les champs Champs maîtres et Champs esclaves sont utilisés pour créer un lien avec le formulaire principal, dans lequel se trouve le sous-formulaire. Le bouton à trois points montre encore une fois qu'une boîte de dialogue utile est disponible pour les créer.



Sous Prêts, les champs de la table Prêts sont affichés, sous Lecteurs ceux de la table Lecteurs. Le champ ID_Lecteurs de Prets doit être défini comme équivalent au champ ID de la table Lecteurs.

Bien que ce lien ait déjà été créé dans la base de données en utilisant **Outils > Relations** (voir Chapitre 3, Tables), la fonction qui se trouve derrière le bouton **Suggérer** dans cette boîte de dialogue ne fait pas référence à cela et suggère que la première clé étrangère de la table Prêt, à savoir ID_Lecteurs, doit être lié à l'ID de la table Lecteurs. L'assistant de création de formulaire résout mieux ce problème en lisant la relation à partir des relations de la base de données.

Le lien choisi entre la table du sous-formulaire et la table du formulaire principal est désormais spécifié en termes de champs des tables.

Pour créer un contrôle de table dans le sous-formulaire principal, nous devons maintenant sélectionner le sous-formulaire dans le navigateur de formulaires. Ensuite, si l'Assistant Contrôle de table est activé, il affichera les champs disponibles dans le sous-formulaire.

Une fois les champs de table mis en place, nous devons effectuer les modifications déjà évoquées lors de la création du formulaire plus simple :

- Remplacement du champ numérique ID_Media dans le sous-formulaire par une zone de liste.
- Renommer le champ ID_Media en Media.
- Modification des champs numériques pour un format sans décimales.
- Limiter les valeurs minimales et maximales.
- Renommer d'autres champs, pour économiser de l'espace ou pour ajouter des caractères non ASCII qui ne doivent pas être utilisés dans les noms de champs des tables de base de données.

Les fonctions de tri et de filtrage sont complétées pour le formulaire principal par l'ajout d'une barre de navigation. Les autres champs de la barre de navigation ne sont pas nécessaires, car ils sont pour la plupart disponibles depuis le champ table (affichage des enregistrements, navigation dans les enregistrements) ou bien exécutés par déplacement à travers le champ table (stockage des données). La forme finale pourrait ressembler à la figure 80.

Comme précédemment, les champs ID_Genre et ID_Media ont été remplacés par des zones de liste, le type de valeurs par une requête SQL (pour le genre, `SELECT "Genre","ID" FROM "Genres"`) et les étiquettes remplacées.

Voir l'exemple *Form04-VoirPrêtsSelectionTable du fichier Media_sans_Macros*

Figure 80: Formulaire composé d'un formulaire principal (en haut) et d'un sous-formulaire (en bas).

Si un lecteur est maintenant sélectionné dans le formulaire principal, le sous-formulaire affichera les médias prêtés à ce lecteur. Lorsqu'un élément est rendu, il continue à apparaître sur le formulaire jusqu'à ce que le formulaire lui-même soit actualisé. Cela se produit automatiquement lorsqu'un autre enregistrement est chargé dans le formulaire principal. Si le lecteur d'origine est de nouveau sélectionné, les supports rendus ne sont plus affichés.

Cette mise à jour différée est en fait souhaitable dans ce cas, car elle permet d'inspecter les supports actuellement placés sur le compteur de la bibliothèque et de voir immédiatement s'ils ont été enregistrés.

Cette structure de formulaire est beaucoup plus facile à utiliser que la précédente avec un seul formulaire. Cependant, il y a encore des détails qui peuvent être améliorés :

- Les dates de prêt et de média peuvent être modifiées lorsque le média doit être prêté plus longtemps. La modification de la date du support peut rendre impossible le suivi de l'élément encore disponible dans la bibliothèque et de celui qui est prêté. La modification de la date du prêt peut entraîner des erreurs. Les avis de rappel n'ont pas pu être vérifiés.
- Si un enregistrement de lecteur n'est pas sélectionné en cliquant sur l'en-tête d'enregistrement à gauche, seule la petite flèche verte sur l'en-tête montre quel enregistrement est actuellement actif. Il est tout à fait possible que l'enregistrement actif défile directement hors de la fenêtre de contrôle de table.
- Au lieu du texte "Médias prêtés au lecteur sélectionné", il serait préférable d'indiquer le nom du lecteur.
- Il est possible de prêter deux fois le même support sans qu'il ne soit restitué.
- Il est possible de supprimer assez facilement l'enregistrement d'un article prêté.
- Les données peuvent être modifiées ou supprimées dans le formulaire principal. Cela peut être utile pour les petites bibliothèques avec peu de trafic public.
  Cependant, lorsque les choses deviennent mouvementées au comptoir des prêts, l'édition des données des utilisateurs ne doit pas avoir lieu en même temps que l'octroi de prêts. Il

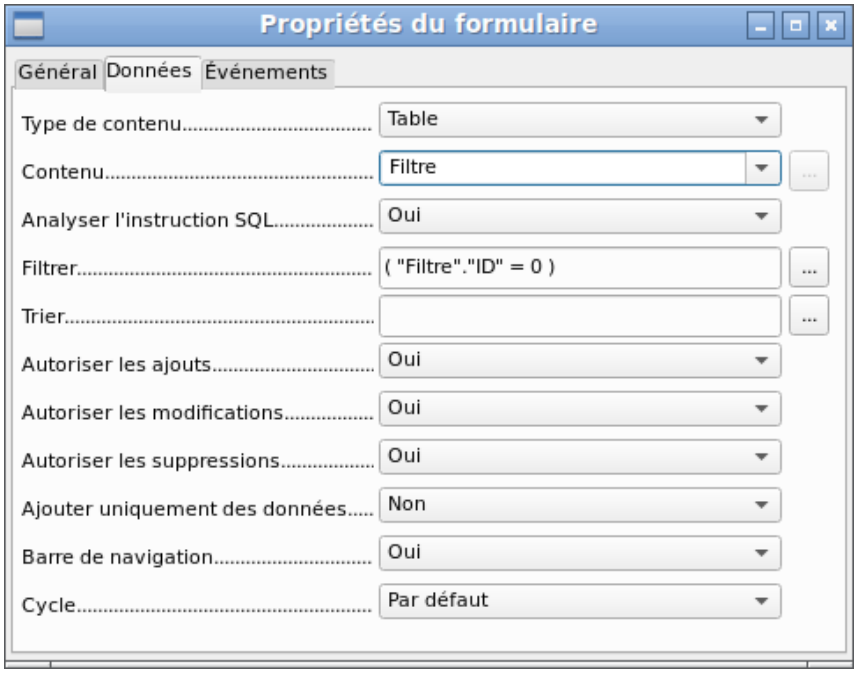
serait préférable que les nouveaux utilisateurs puissent être enregistrés mais que les données utilisateur existantes ne soient pas modifiées. Pour les bibliothèques, cela s'applique également aux suppressions ou aux changements de nom complets.

Améliorons d'abord la sélection des lecteurs. Cela devrait nous protéger des modifications apportées aux registres de prêt. Une solution simple serait de ne permettre aucune modification sauf l'entrée de nouveaux enregistrements. Cela nécessite toujours une fonction de recherche lorsqu'un lecteur souhaite emprunter un article. Il serait préférable d'utiliser une zone de liste pour trouver le lecteur et effectuer les opérations d'émission et de retour dans des champs de table séparés.

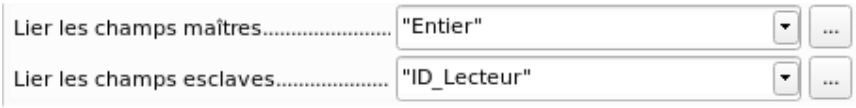
Pour le formulaire principal, nous avons besoin d'une table, dans laquelle la zone de liste peut écrire une valeur liée à cette table. La table nécessite un champ entier et une clé primaire. Il ne contiendra toujours qu'un seul enregistrement, de sorte que l'ID de champ de clé primaire peut être déclaré en toute sécurité comme Tiny Integer. La table suivante nommé Filtre doit donc être créée.

Nom de table : Filtre	
Nom de Champ	***Type de Champ***
ID	Tiny Integer, Primary key
Entier	Integer

La table reçoit une clé primaire avec la valeur 0. Cet enregistrement sera lu et réécrit à plusieurs reprises par le formulaire principal.

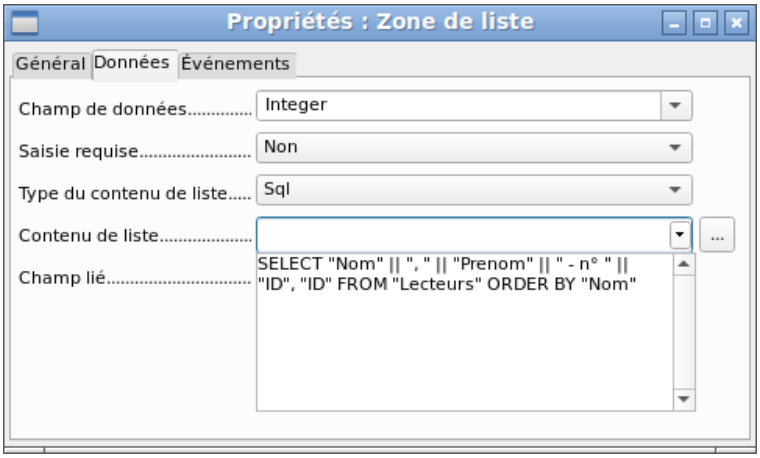


Le formulaire principal est basé sur la table Filtre. Il lira simplement la valeur de la table qui est associée à la clé primaire (ID) de 0. Aucune donnée ne sera ajoutée, l'enregistrement actuel sera simplement réécrit à plusieurs reprises. Comme seules les modifications d'un seul enregistrement sont autorisées, une barre de navigation serait superflue.



Ce formulaire principal est lié au sous-formulaire de telle sorte que la valeur du champ Entier dans la table Filtre est la même que la valeur du champ ID_Lecteur dans la table Prets. Les propriétés du sous-formulaire sont inchangées par rapport à la version ci-dessus.

Avant de créer une zone de liste dans le formulaire principal, nous devons désactiver les assistants. L'assistant de zone de liste vous permet uniquement de créer une zone qui affiche le contenu d'un seul champ. Il serait impossible d'avoir le nom, le prénom et un numéro supplémentaire dans la zone d'affichage d'une zone de liste. Comme dans le formulaire plus simple, nous entrons maintenant pour le contenu de la zone de liste Nom, Prenom – ID Numéro. La zone de liste transmet l'ID à la table sous-jacente.



À côté de la zone de liste, un bouton est créé. Ce bouton fait en fait partie du sous-formulaire. Il reprend deux fonctions : sauvegarder l'enregistrement dans le formulaire principal et mettre à jour la table dans le sous-formulaire. Il suffit de confier la mise à jour au bouton du sous-formulaire. Le processus de sauvegarde du formulaire principal modifié est alors exécuté automatiquement.

Le bouton peut simplement être étiqueté OK. L'action qui lui est assignée dans les propriétés générales du bouton est *Rafraîchir le formulaire*.

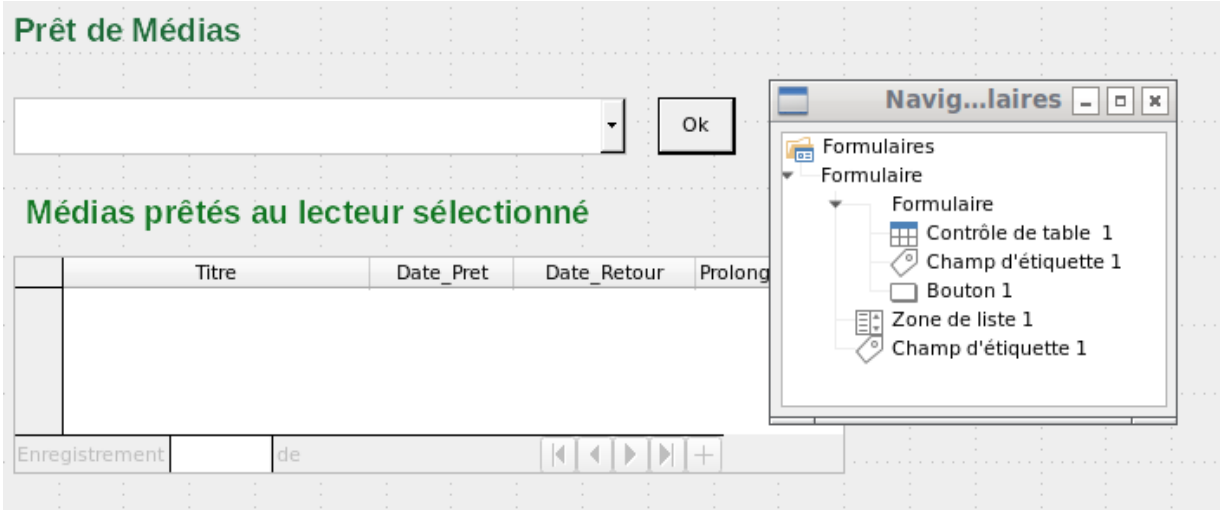


Figure 81: Formulaire principal comme filtre pour un sous-formulaire

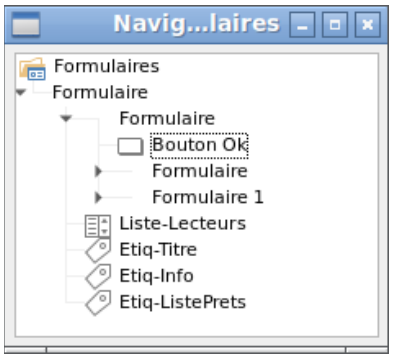
Le formulaire principal se compose uniquement de l'en-tête et de la zone de liste ; le sous-formulaire contient un autre en-tête, le champ table de la version précédente et le bouton.

Ce formulaire fonctionne désormais mieux en ce sens que :

- Aucun lecteur ne peut maintenant être édité, modifié ou supprimé, et
- Les lecteurs peuvent être trouvés plus rapidement en tapant une initiale dans le contrôle qu'en utilisant un filtre.

Cet exemple est disponible sous le nom *Form05-VoirPrêtsSelectionListe.*

Pour un plus grand degré de fonctionnalité (retours sans altération des données précédentes) un second sous-formulaire doit être créé, lié à la même table Prêts. Pour garantir la fonctionnalité de la zone de liste de la figure 81, les deux sous-formulaires doivent être placés un niveau plus bas, en tant que sous-formulaires d'un sous-formulaire. Les données sont mises à jour de manière hiérarchique depuis le formulaire principal jusqu'aux sous-formulaires. Le bouton dans le formulaire décrit précédemment doit maintenant être placé dans le premier sous-formulaire et non dans les deux sous-formulaires qui se trouvent en dessous.



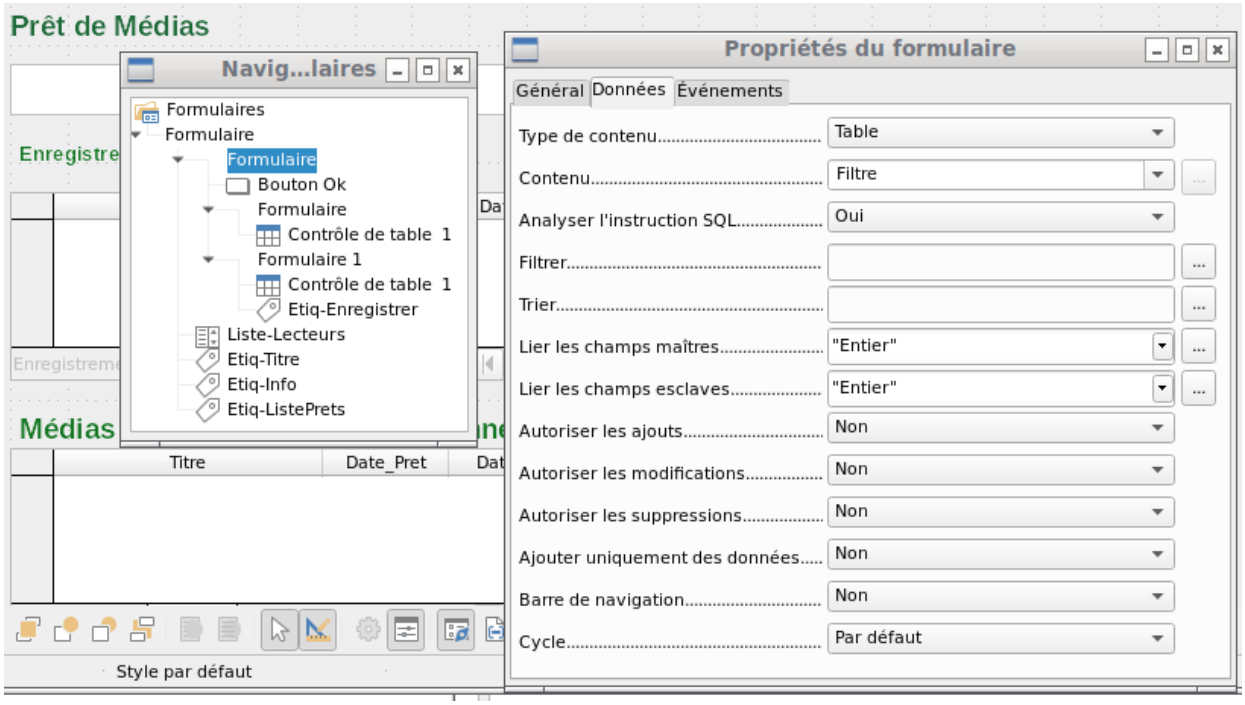
Ici, le navigateur de formulaires est utilisé pour afficher les différents niveaux. Dans le formulaire principal, nous avons le champ d'étiquette pour le titre du formulaire et la zone de liste pour trouver le lecteur. La zone de liste apparaît en bas du formulaire, telle qu'elle est déclarée après le sous-formulaire. Malheureusement, cette séquence d'affichage ne peut pas être modifiée. Le sous-formulaire n'a qu'un seul bouton, pour mettre à jour son contenu et en même temps enregistrer le formulaire principal. Un niveau plus bas se trouvent les deux sous-formulaires supplémentaires.

Ceux-ci reçoivent des noms différents lors de leur création afin qu'aucune confusion ne survienne à aucun niveau de l'arbre.

Note

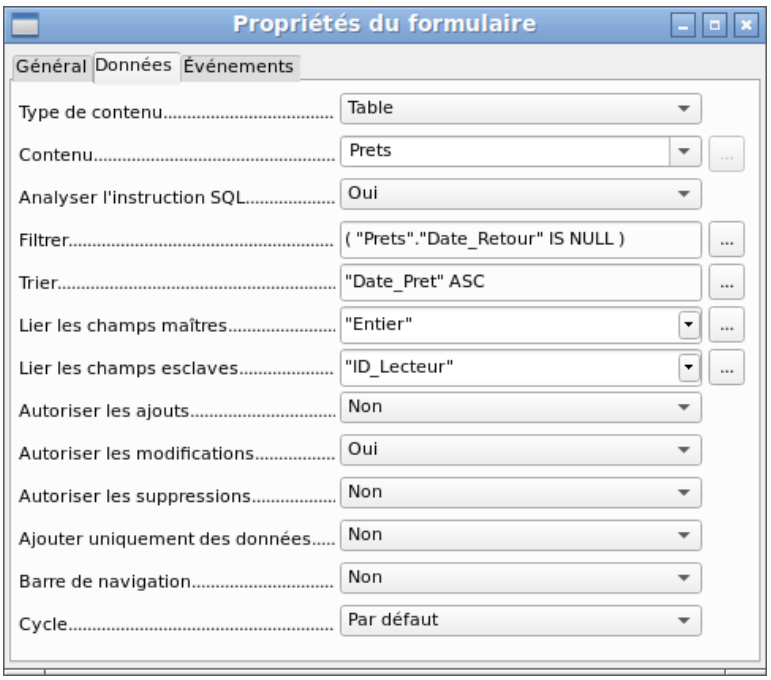
Fondamentalement, les noms des formulaires et des contrôles sont sans importance. Cependant, si ces noms doivent être accessibles par des macros, ils doivent pouvoir être distingués. Vous ne pouvez pas distinguer des noms identiques au même niveau.

Naturellement, il est logique, lors de la création de structures de formulaire plus grandes, d'avoir des noms significatifs pour les formulaires et leurs contrôles. Sinon, trouver le bon champ pourrait rapidement devenir problématique.



Le formulaire principal et le sous-formulaire utilisent la même table. Dans le sous-formulaire, aucune donnée n'est saisie. C'est pourquoi toutes les options de ce formulaire sont définies sur Non. Le formulaire principal et le sous-formulaire sont liés via le champ dont la valeur doit être transmise aux sous-sous-formulaires : le champ Entier dans la table Filtre.

Dans le contrôle table du premier sous-sous-formulaire, aucune donnée existante n'est affichée, il n'est utilisé que pour créer de nouvelles données. Pour cela, le filtre suggéré est adéquat. Seuls les enregistrements correspondant au ID_Lecteur et avec un champ de date de prêt vide ("Date_Pret" IS NULL) seront affichés, donc aucun. En pratique, cela signifie un champ de table vide. Comme le contrôle de table n'est pas mis à jour en permanence, le média nouvellement prêté y restera jusqu'à ce que le bouton de mise à jour OK soit utilisé pour sélectionner un nouveau nom ou pour transférer les données dans le deuxième sous-sous-formulaire.



Le deuxième sous-sous-formulaire nécessite plus de paramètres. Ce formulaire contient également des données de la table “Prets”. Ici, les données sont filtrées pour une date de retour vide. ("Date_Retour" IS NULL). Les données sont triées comme dans le formulaire précédent, de sorte que les supports prêtés depuis plus longtemps soient immédiatement visibles.

Les points suivants sont également importants. Les anciens enregistrements peuvent être modifiés, mais aucun nouvel enregistrement ne peut être ajouté. La suppression est également impossible. Il s’agit de la première étape nécessaire pour éviter que les dossiers de prêt ne soient simplement supprimés ultérieurement. Mais il serait toujours possible de changer le support et la date du prêt. Par conséquent, les propriétés des colonnes devront être ajustées. Finalement, le média et la date du prêt devraient être affichés mais protégés de toute modification.

Le contrôle table est simplement dupliqué après la création du formulaire. Cela se fait en le sélectionnant, en le copiant, en le désélectionnant, puis en le collant à partir du presse-papiers. La copie sera à la même position que l’original et devra donc être déplacée. Après cela, les deux contrôles de table peuvent être modifiés séparément. Le contrôle de table pour le retour de média peut rester pratiquement le même. Seul l’accès en écriture pour les colonnes de média et de date du prêt doit être modifié.

Alors que pour Date_Pret, il est seulement nécessaire de choisir Lecture seule, ce n’est pas suffisant pour les zones de liste. Ce paramètre n’empêche pas la zone de liste d’être utilisée pour apporter des modifications. Cependant, si Activé est défini sur Non, aucun choix ne peut y être effectué. Une zone de liste contenue dans le contrôle de table est ensuite affichée sous la forme d’un champ de texte en lecture seule.

Dans le contrôle de table ci-dessus, tous les champs qui n’ont rien à voir avec le prêt sont supprimés. Seuls le support comme champ de sélection et la date du prêt Date_Pret restent.

Si finalement la requête pour la zone de liste dans le champ de table supérieur est sélectionnée, seuls les médias qui peuvent être prêtés seront affichés. Pour en savoir plus, consultez le chapitre 5, Requêtes.

Voir l’exemple : *Form06-VoirPrêtsEnregistrer du fichier Media_Sans Macros.odb*

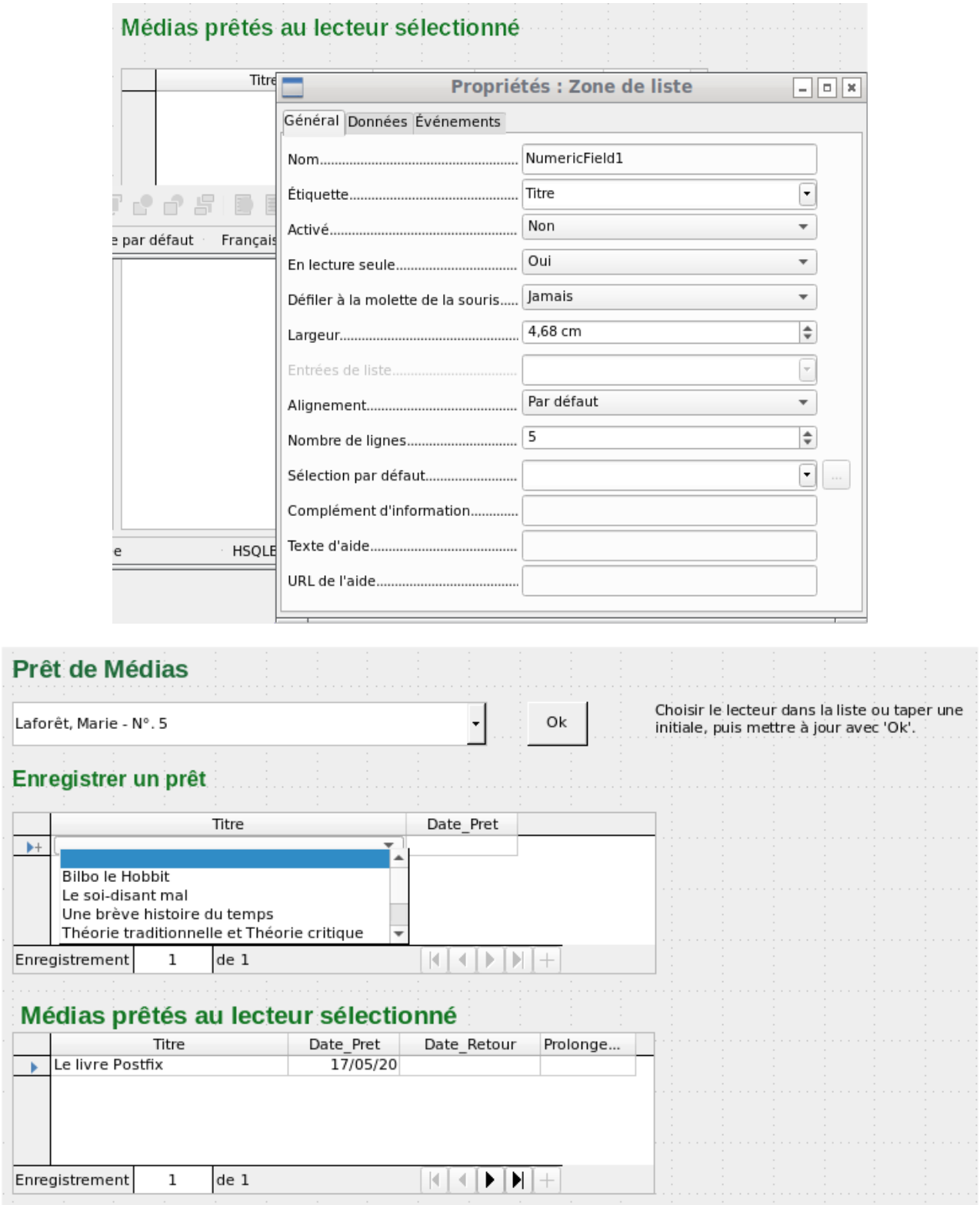
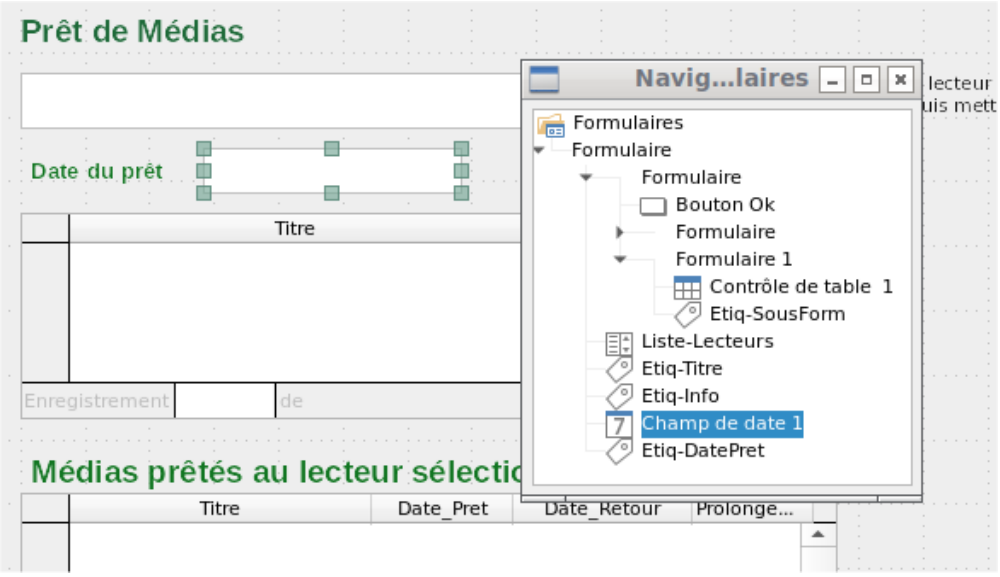


Figure 82: Le champ de sélection dans le sous-formulaire supérieur affiche les supports susceptibles d'être prêtés et permet l'enregistrement.

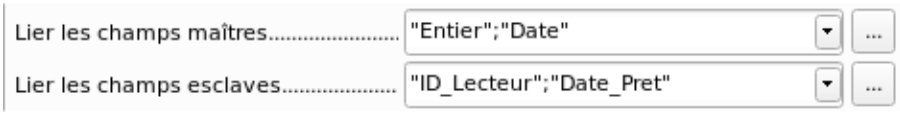
Le formulaire de prêt média est déjà nettement plus utile. Lorsqu'un lecteur arrive au comptoir de prêt, son nom est recherché. Le support à prêter peut-être sélectionné à l'aide de la zone de liste et de la date de prêt saisie. La touche Tab vous amène ensuite à l'enregistrement suivant.

Une dernière amélioration est également souhaitable : actuellement, la date du prêt doit être choisie à chaque fois. Imaginez une journée à la bibliothèque avec peut-être 200 opérations de prêt, peut-être avec une seule personne, qui doit prêter environ 10 médias à chaque fois. Cela nécessiterait la même entrée pour le champ de date encore et encore. Il doit y avoir un moyen de simplifier cela.

Notre formulaire principal est lié à la table Filtre. Le formulaire principal fonctionne uniquement avec l'enregistrement qui a comme clé primaire le "ID" 0. Mais des champs supplémentaires peuvent être créés dans la table Filtre. À l'heure actuelle, aucun champ ne peut stocker une date, mais nous pouvons facilement créer un nouveau champ avec le nom de champ Date et le type de champ Date. Dans la table Filtre, nous avons maintenant stocké non seulement le ID_Lecteur ("Filtre"."Entier") mais également le Date_Pret ("Filtre"."Date").



Dans le formulaire principal, un champ de date supplémentaire apparaît, ainsi qu'une étiquette faisant référence à son contenu. La valeur du champ de date est stockée dans la table Filtre et transférée par les liens du sous-formulaire au sous-sous-formulaire.



Le lien entre les deux formulaires renvoie désormais à deux champs. Le champ Entier est lié au champ ID_Lecteur du sous-sous-formulaire. Le champ Date est lié au champ Date_Pret. Cela garantit que Date_Pret est automatiquement transféré de la table Filtre vers la table Prets lorsque le prêt est effectué.

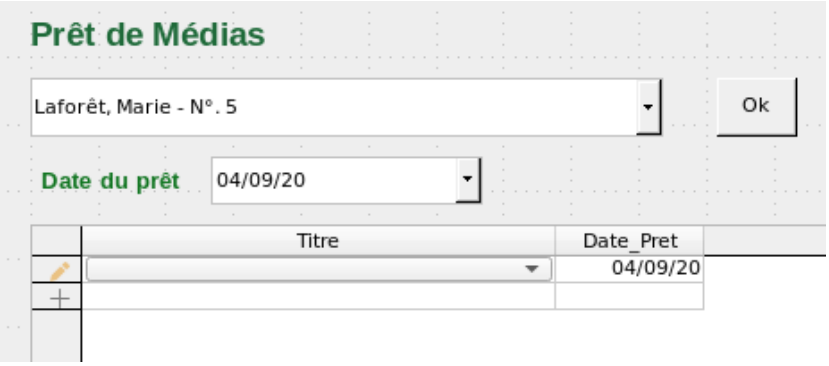
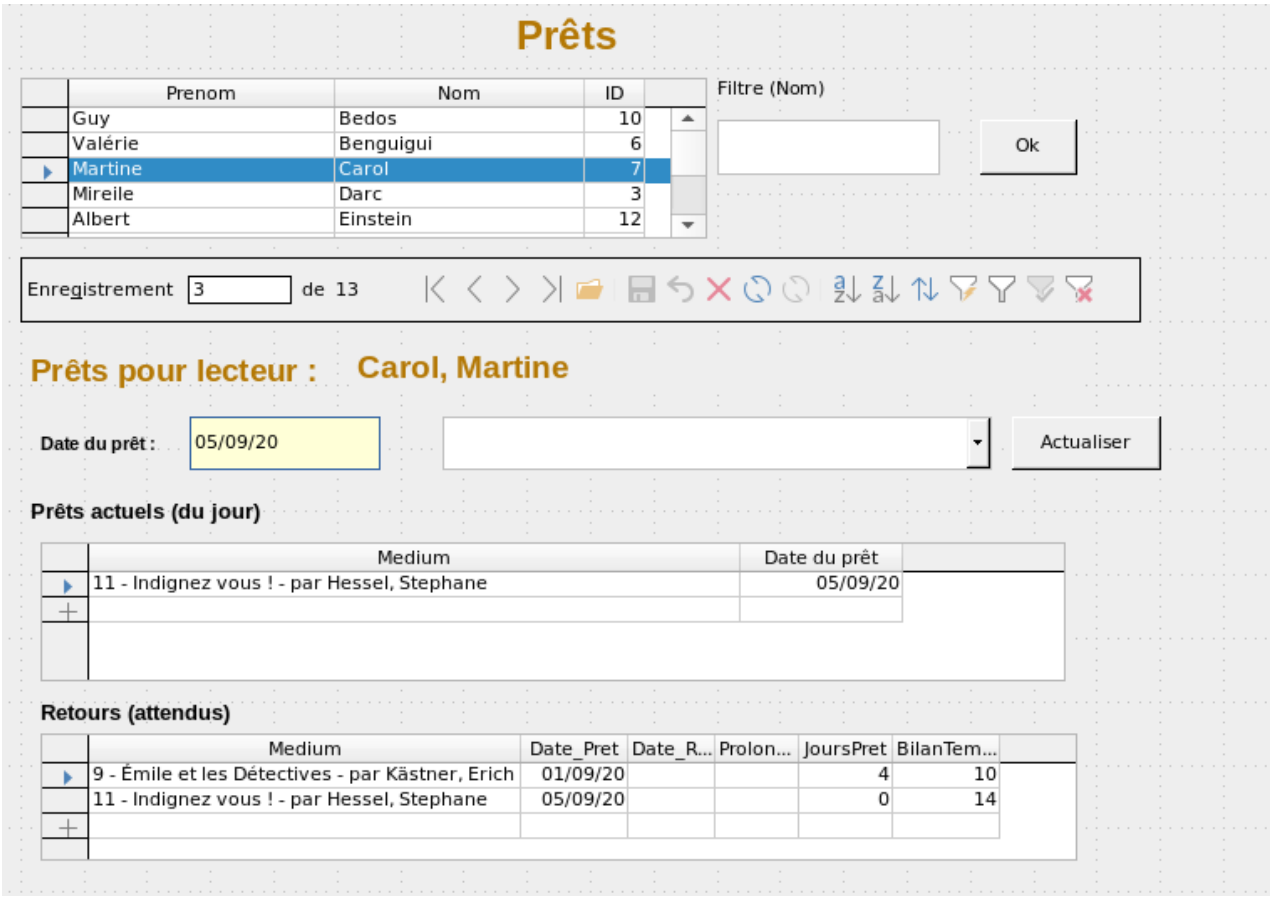


Figure 83: La date du prêt n'est saisie qu'une seule fois. Lorsque le lecteur change, elle doit être ressaisie.

Exemple : *Form07-VoirPrêtsEnregitrerDate*

Le champ de date peut maintenant être supprimé du champ de table, de sorte que ce dernier ne contienne qu'un seul champ de recherche. Ce serait la condition idéale pour accélérer encore plus le travail de la bibliothèque. Car en fait, chaque support aura un numéro d'identification imprimé, alors pourquoi faut-il chercher cela ? Vous pouvez simplement entrer le numéro directement. Ou, mieux encore, il pourrait être numérisé avec un lecteur de code-barres. Ensuite, les médias peuvent être prêtés aussi rapidement que l'emprunteur peut les mettre dans son sac.

Ceci est illustré dans le dernier exemple de formulaire dans la base de données. L'exemple ci-dessus devrait suffire pour comprendre la conception initiale du formulaire mais, comme ce dernier exemple de la base de données, **Media_sans_Macros.odb**, développe le formulaire davantage, les améliorations supplémentaires sont brièvement présentées ci-dessous.



Le formulaire de prêt présente les propriétés suivantes :

- Les lecteurs sont affichés dans un champ de table. Ici, vous pouvez également entrer de nouveaux lecteurs.
- À l'aide d'un filtre, lié à la table Filtre, les noms peuvent être filtrés à l'aide de leur lettre initiale. Ainsi, si vous saisissez A, seules les personnes dont le nom commence par A seront affichées. Ce filtrage est indépendant de la casse.
- Le sous-titre montre à nouveau le nom de la personne à qui le prêt doit être fait. Si un verrou a été placé sur cet emprunteur, il est également affiché.
- La date du prêt est définie sur la date du jour. Cela se fait dans la table de filtrage en utilisant SQL de telle sorte que, lorsqu'aucune date n'est entrée, la valeur par défaut à stocker est la date actuelle.
- Les supports empruntables sont sélectionnés à l'aide d'une zone de liste. Lorsque vous appuyez sur le bouton Mettre à jour, le prêt est transféré vers le contrôle de table au-dessous.
- Le champ de table au milieu sert uniquement à afficher la date réelle du prêt pour le média. Ici, il est également possible de corriger une erreur rétrospectivement en supprimant la ligne.
- Dans le contrôle de table inférieur, comme dans l'exemple ci-dessus, la modification des supports et des dates de prêt n'est pas possible. Il n'est pas non plus possible de supprimer des enregistrements.
- Outre la saisie de la date de retour ou, le cas échéant, une prolongation du prêt, ce tableau affiche également le nombre de jours pour lesquels le support peut être prêté et le nombre de jours restants de la période de prêt en cours.
- Si ce temps restant devient négatif, le support doit être renvoyé immédiatement. Le compte est alors verrouillé. Cela ne redevient possible que lorsque le support est renvoyé. Après le retour, le bouton Mettre à jour ne doit être appuyé qu'une seule fois.

Ce formulaire, créé à l'aide de requêtes, est beaucoup plus complexe dans sa structure que la version précédente. Vous pouvez en savoir plus sur l'essentiel dans le chapitre 5, Requêtes.

Une vue – plusieurs formulaires

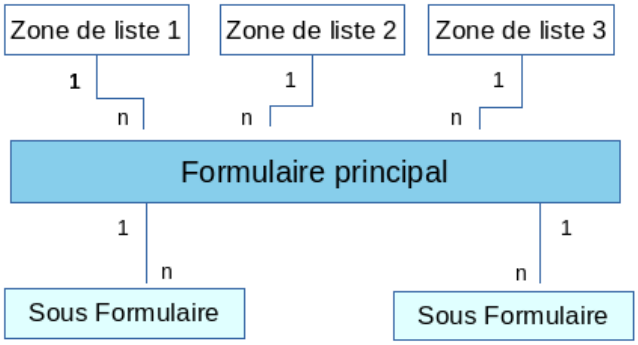
Alors que l'exemple du formulaire Prêts n'implique que des entrées dans une seule table (la table Prets) et permet en outre l'entrée dans le sous-formulaire pour les nouveaux lecteurs, la procédure d'entrée pour les médias est beaucoup plus étendue. Dans sa forme finale, la table des médias est entourée d'un total de huit tables supplémentaires (voir Chapitre 3, Tables).

Les tables SousTitre et relation_Media_Auteur sont liées aux sous-formulaires du formulaire Media via une relation n:1. En revanche, les tables avec une relation 1:n avec la table Media doivent être représentées par des formulaires situés au-dessus de la table Media. Comme il existe plusieurs de ces tables, leurs valeurs sont saisies dans le formulaire principal à l'aide de zones de liste.

La table d'un formulaire principal et la table d'un sous-formulaire sont dans une relation 1:n, ou dans certains cas exceptionnels 1:1. Par conséquent, après une longue utilisation de la base de données, le formulaire principal gère généralement une table qui contient beaucoup moins d'enregistrements que la table appartenant au sous-formulaire.

Plusieurs formulaires principaux ne peuvent pas inclure le même sous-formulaire. Par conséquent, il n'est pas possible de créer une disposition de formulaire pour de nombreuses relations 1;n simultanées dans lesquelles le sous-formulaire a le même contenu. Lorsqu'il existe une relation 1:n pour la table appartenant à un formulaire, vous pouvez utiliser une zone de liste.

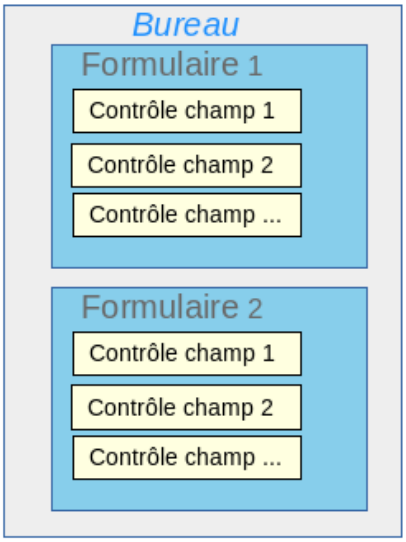
Ici, seuls quelques termes sont disponibles à partir d'une autre table, ceux dont les clés étrangères peuvent être saisies dans la table du formulaire principal de cette manière.



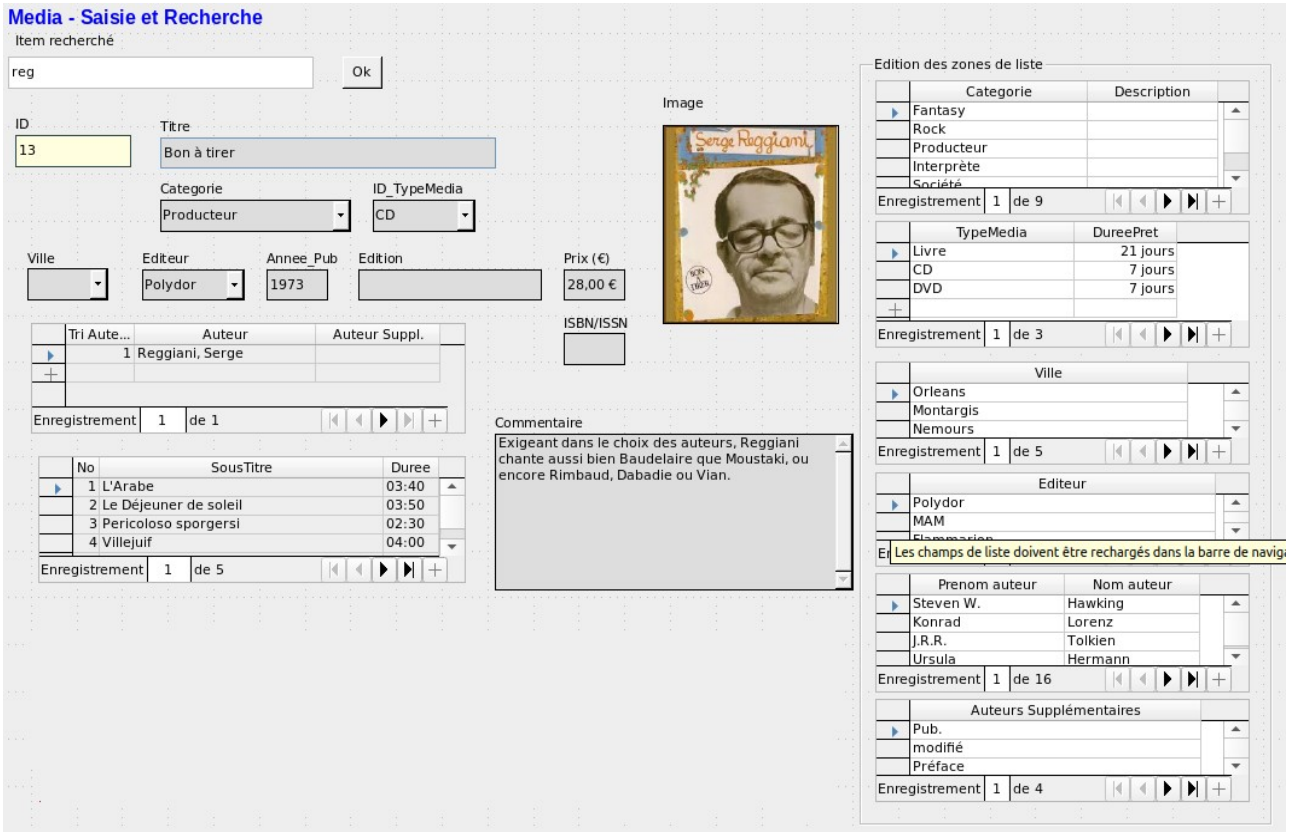
À l'aide de zones de liste, le formulaire principal basé sur la table Media peut recevoir des valeurs des tables Categories, Ville ou Editeurs. Les sous-formulaires sont utilisés pour lier les tables relation_Media_Auteur et SousTitre avec le formulaire principal et à travers celui-ci avec la table Media.

Le sous-formulaire de la table relation_Media_Auteur se compose à nouveau de deux zones de liste afin que les clés étrangères des Auteurs et ID_Auteur_Suppl (les ajouts peuvent être par exemple éditeur, photographe, etc.) ne soient pas entrées directement sous forme de nombres.

Pour le formulaire de saisie des médias, les zones de liste doivent généralement être remplies progressivement au cours du processus de saisie. À cette fin, d'autres formulaires sont intégrés à côté du formulaire principal. Ils existent indépendamment de la forme principale.

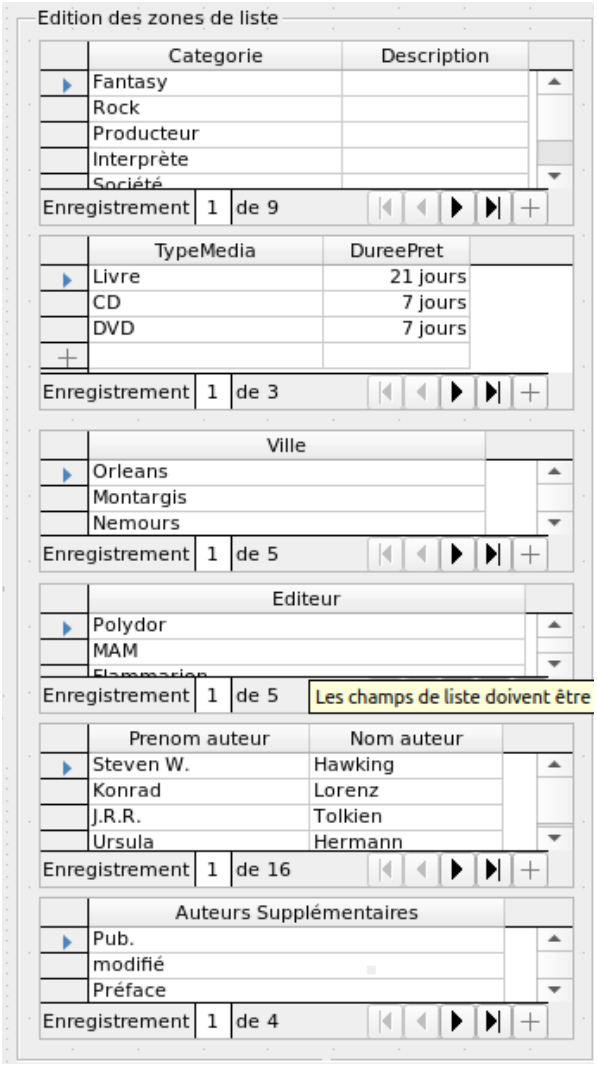


La forme générale de saisie des médias ressemble à ceci :



Sur le côté gauche se trouve le formulaire principal en vue de la recherche et de la saisie de nouveaux médias. Sur le côté droit se trouve une zone de groupe avec l'étiquette Modifier le contenu de la zone de liste, fournissant une zone séparée destinée à remplir les zones de liste dans le formulaire principal. Si la base de données n'existe pas depuis longtemps, il sera souvent nécessaire de faire des entrées dans ces champs. Toutefois, plus il y a d'entrées disponibles pour les zones de liste du formulaire principal, moins il sera nécessaire d'accéder aux contrôles de table dans la zone de groupe.

Les contrôles de table suivants sont tous subordonnés en tant que formulaires secondaires individuels au formulaire principal, le formulaire d'entrée.



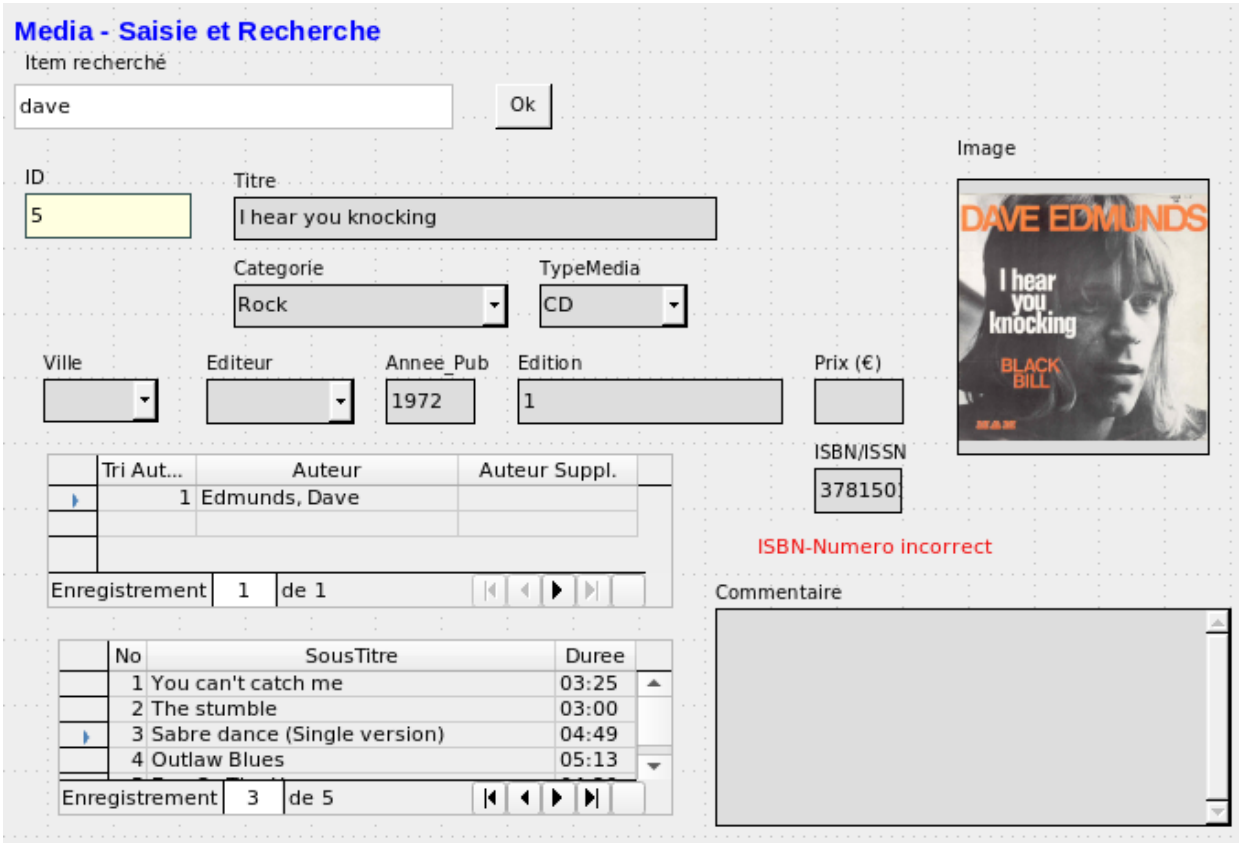
Ici, dans chaque cas, les données complètes d'une table sont saisies. Dans les premiers temps, il est souvent nécessaire d'avoir recours à ces formulaires annexes, car peu d'auteurs sont encore stockés dans la table correspondante.

Lorsqu'un nouvel enregistrement est stocké dans l'un des contrôles de table, il est nécessaire de rechercher la zone de liste correspondante dans le formulaire principal et d'utiliser le contrôle de mise à jour (voir Barre de navigation) pour lire les nouvelles valeurs.

Le Navigateur de formulaires affiche une grande liste correspondante de formulaires.

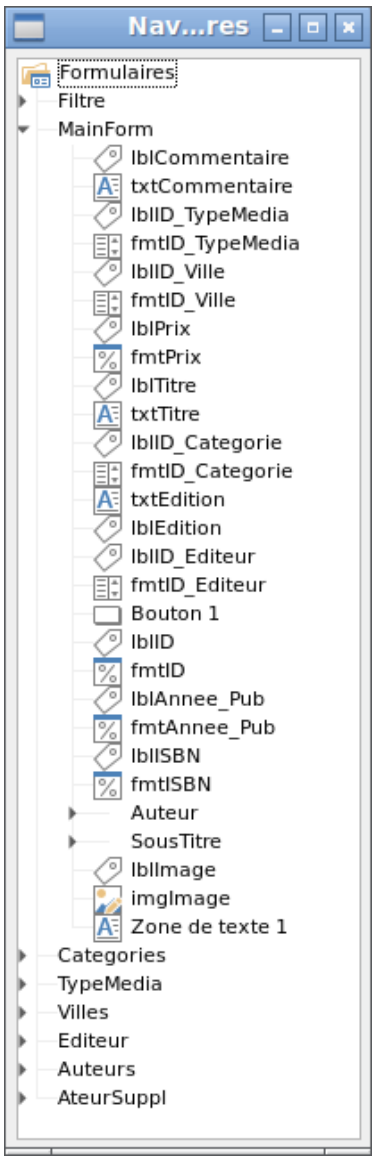
Les formulaires ont été nommés individuellement pour faciliter la reconnaissance. Seul le formulaire principal a toujours le nom de MainForm qui lui a été donné par l'assistant. Au total, il existe huit formulaires parallèles. Le formulaire Filtre héberge une fonction de recherche tandis que le formulaire MainForm est l'interface d'entrée principale. Tous les autres formulaires concernent l'un ou l'autre des champs de table présentés ci-dessus.

Sans les contrôles de table, le formulaire principal semble un peu plus simple.



Le champ du terme recherché se trouve dans le formulaire Filtre, les deux champs de table (pour l'auteur et le sous-titre) se trouvent dans le sous-formulaire du formulaire principal Saisie média.

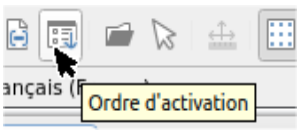
Dans le navigateur de formulaires, ce formulaire semble beaucoup plus complexe, car tous les contrôles et étiquettes y apparaissent. Dans le formulaire précédent, la plupart des champs étaient en fait des colonnes dans les contrôles de table et étaient donc invisibles pour le navigateur de formulaire.



Malheureusement, la séquence dans le navigateur de formulaire ne peut pas être facilement modifiée. Ainsi, par exemple, il semblerait plus judicieux de placer les sous-formulaires SousTitre et Auteur en tant que branches de MainForm dès le début. Mais dans le navigateur de formulaires, les contrôles et sous-formulaires individuels sont simplement affichés dans l'ordre dans lequel ils ont été créés.

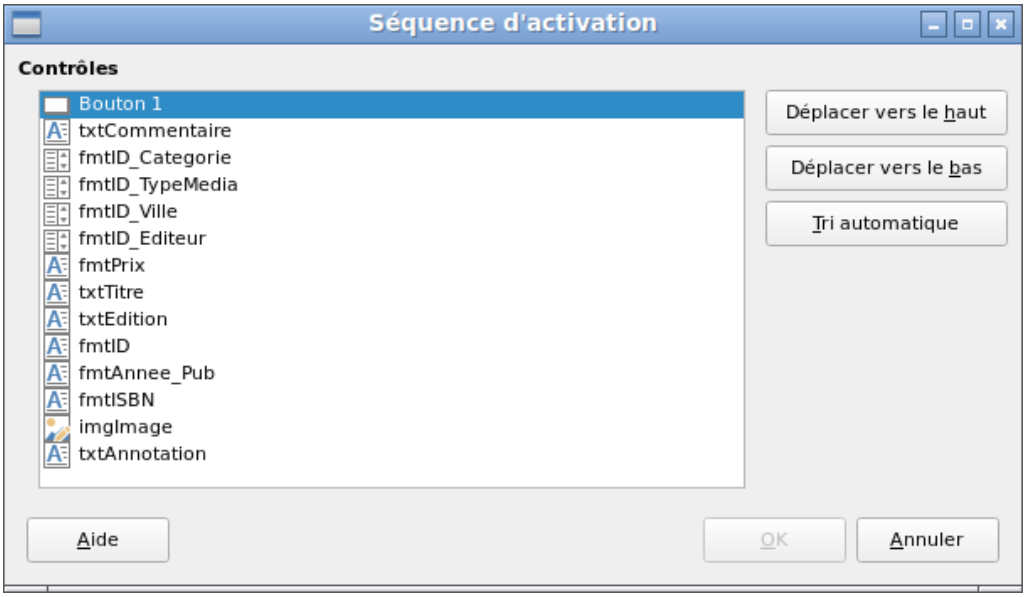
L'Assistant Formulaire fournit aux contrôles des abréviations particulières qui apparaissent à côté des icônes et indiquent de quel type de contrôle il s'agit. Les libellés (étiquettes) commencent par 'lbl', les champs de texte par 'txt' et ainsi de suite. Dans l'ensemble, les étiquettes ne fournissent que des informations secondaires pour la saisie des données. Ils peuvent également être placés directement au-dessus des blocs de texte et n'apparaissent pas dans le navigateur de formulaires.

La séquence des éléments dans le navigateur n'a rien à voir avec la séquence de tabulations. Cela est déterminé par l'ordre d'activation.



La séquence d'activation pour un formulaire particulier est appelée lorsqu'un élément du formulaire est sélectionné et que le bouton Ordre d'activation est cliqué.

Pour un formulaire simple, il n'est pas nécessaire de faire les choses dans cet ordre, mais là où il existe de nombreux formulaires parallèles, la fonction a besoin de savoir quel formulaire est prévu. Sinon, par défaut, il s'agit du premier formulaire du navigateur de formulaires.



La séquence d'activation permet à tous les éléments qui transmettent des données à la table sous-jacente du formulaire ou peuvent effectuer des actions, d'être mis en ordre. Cela correspond au paramétrage de la séquence d'activation dans les propriétés de contrôle listées à la page 164.

Notez que dans la séquence d'activation, certaines commandes apparaissent pour lesquelles le taquet de tabulation est en fait désactivé. Ils sont inclus dans la liste, mais ne sont en fait pas accessibles via le clavier lorsque vous travaillez avec le formulaire.

Les champs peuvent être automatiquement triés dans l'ordre dans lequel ils apparaissent sur l'arrière-plan du formulaire. Plus un champ est haut, plus il arrive tôt dans la séquence lorsque vous utilisez le tri automatique. Pour les champs de même hauteur, le champ le plus à gauche vient en premier. Ce tri fonctionne sans erreur uniquement lorsque les éléments ont été positionnés exactement à l'aide de la grille lors de la création du formulaire. Sinon, vous devrez les ajuster. Sélectionnez simplement un contrôle et utilisez *Déplacer vers le haut* ou *Déplacer vers le bas* pour le déplacer vers le haut ou vers le bas dans la séquence.

S'il existe un sous-formulaire, le tri automatique saute directement dans le sous-formulaire après avoir parcouru le formulaire principal. Dans le cas d'un contrôle de table, pendant la saisie au clavier le curseur est piégé dans ce sous-formulaire ; vous ne pouvez le libérer qu'en utilisant la souris ou en appuyant sur Ctrl + Tab.

Le tri automatique ne fonctionne qu'une seule fois pour un contrôle de table. Un sous-formulaire suivant avec un contrôle de table ne sera pas inclus. Les formulaires parallèles ne sont pas non plus pris en compte. Un tri automatique ne peut pas être effectué rétrospectivement pour un sous-formulaire avec un contrôle de table. Le sous-formulaire doit être complètement supprimé (temporairement déplacé vers un autre formulaire).

Le substrat de données pour le formulaire de saisie multimédia n'est pas dans ce cas une table mais une requête.

Ceci est nécessaire, car le formulaire doit être utilisé non seulement pour saisir des enregistrements, mais également pour la recherche. Le formulaire contient également un champ de texte qui indique, après enregistrement, si le numéro ISBN saisi était correct. Cela aussi n'est possible qu'en utilisant une requête complète. Pour en comprendre le contexte, il est donc nécessaire de discuter des principes fondamentaux des requêtes, traités au chapitre 5.

Messages d'erreur lors de la saisie dans les formulaires

Certains messages d'erreur qui se produisent généralement lors de la conception initiale d'un formulaire sont brièvement expliqués ici.

Attempt to insert null into a non-nullable column : NomColonne : ID table : NomTable in statement…

Certains champs ne peuvent pas être vides. S'ils sont ainsi définis dans la table (*NOT NULL*), vous obtenez ce message d'erreur. S'il a également été déclaré non nul dans le formulaire, vous voyez également un message en langue native avec le nom exact du champ qui doit être rempli.

Le message ci-dessus apparaît très souvent lorsque la clé primaire, dans ce cas ID, n'a pas été importée dans le formulaire. Il a été conçu comme un champ à incrémentation automatique, mais malheureusement le concepteur a oublié de le définir comme Valeur automatique. Vous devez donc corriger cela ou le champ doit apparaître dans le formulaire, afin que vous puissiez saisir une valeur.

Définir un champ rétrospectivement comme Valeur automatique est parfois problématique, si la table a des relations avec d'autres tables ou lorsque la table est accessible à l'aide d'une vue. Tous les liens doivent être supprimés pour rendre le champ de clé primaire de la table modifiable. Le contenu de la commande SQL qui a été utilisée pour créer la vue peut être stocké temporairement dans une requête.

Integrity constraint violation – no parent SYS_FK_95 table : NomTable in statement…

Ici, nous avons un lien entre la table sous-jacente au formulaire et une autre table. Cette table est censée fournir sa clé primaire à utiliser dans un champ de clé étrangère. Normalement, cela se fait en utilisant une zone de liste qui exécute la requête correcte pour récupérer la clé. Si vous n'utilisez pas une zone de liste ou si la zone de liste n'a pas été correctement construite, le champ de clé étrangère peut acquérir une valeur incorrecte qui n'est pas réellement présente dans la table source en tant que clé primaire. C'est ce que l'on entend par "violation de contrainte d'intégrité". '*No parent SYS_FK_95*' signifie que dans la deuxième table, la table source "Parents", la valeur d'index correspondante portant le nom "*SYS_FK_95*" n'est pas présente.

. Errors when entering new records
Errors in running functions

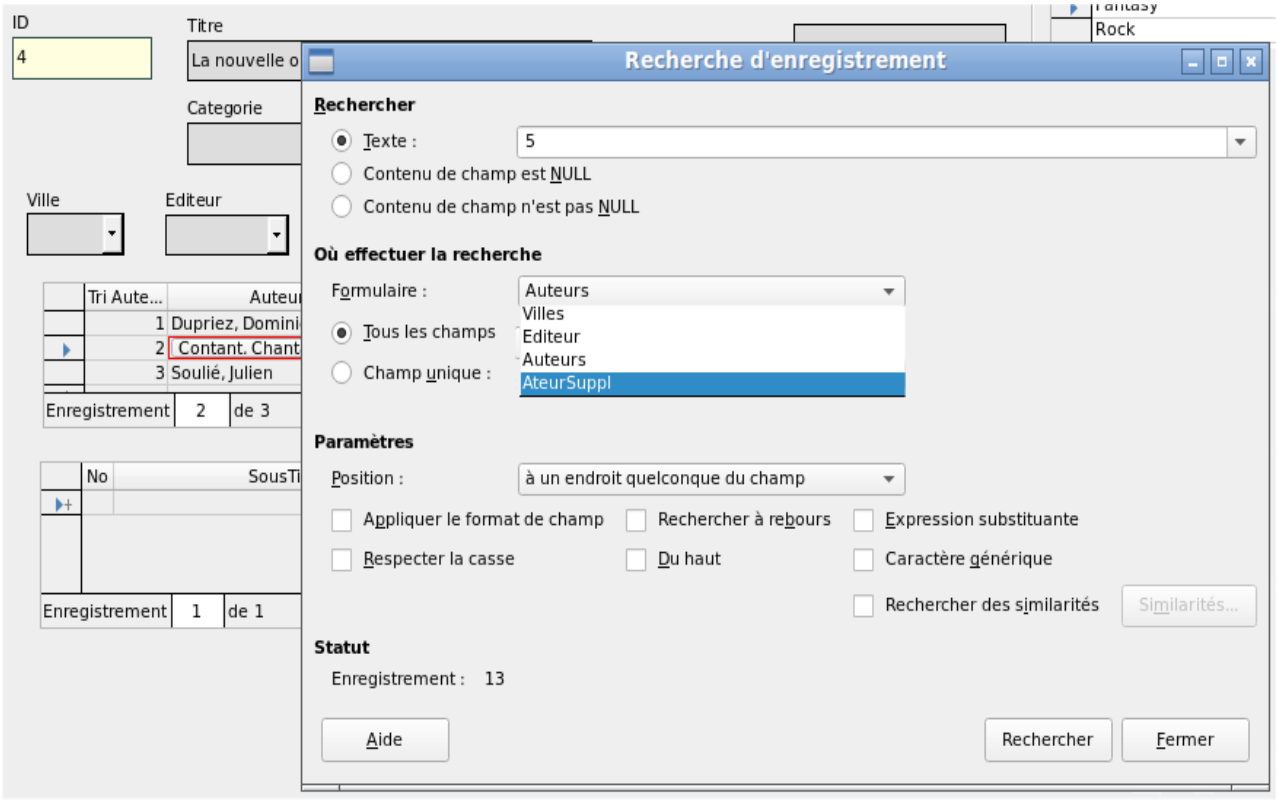
Supposons qu'un formulaire fournisse des données à un sous-formulaire. Base ne voit pas cela comme une modification de la valeur d'un champ. L'interface utilisateur graphique propose de sauvegarder la valeur mais l'enregistrement apparaît vide. La réponse est d'inclure un champ simple, par exemple un champ oui / non, dans la table sous-jacente. Maintenant, avant de sauvegarder, supprimez ce champ et l'enregistrement peut être sauvegardé sans aucun problème.

Les valeurs transmises via d'autres formulaires ne sont apparemment saisies dans les champs correspondants que lorsqu'il y a au moins une action supplémentaire dans le formulaire.

Recherche et filtrage dans les formulaires à l'aide de la barre de navigation

La barre de navigation du formulaire offre diverses possibilités de recherche et de filtrage. Les éléments individuels de la barre de navigation ont déjà été répertoriés ci-dessus.

Recherche d'enregistrement à l'aide de paramètres



Une simple recherche d'enregistrements avec des paramètres est décrite dans le chapitre 3, Tables. Elle offre un nombre considérable de paramètres. Une caractéristique de ce type de recherche est que, fondamentalement, tous les enregistrements sont tamisés plutôt que de limiter les enregistrements affichés à ceux contenant les paramètres de recherche.

Les recherches d'enregistrements dans les formulaires vous permettent de rechercher à la fois dans le formulaire et tous les sous-formulaires, mais il n'est pas possible de rechercher tous les formulaires à la fois.

L'illustration montre l'accès au sous-formulaire Form_Auteur dans le formulaire principal MainForm. Il s'agit d'un champ de liste contenant les noms des auteurs. Cependant, ce qui est

recherché n’est pas le texte dans le champ de liste mais les valeurs qui sont entrées comme clés étrangères lorsque le champ est utilisé.

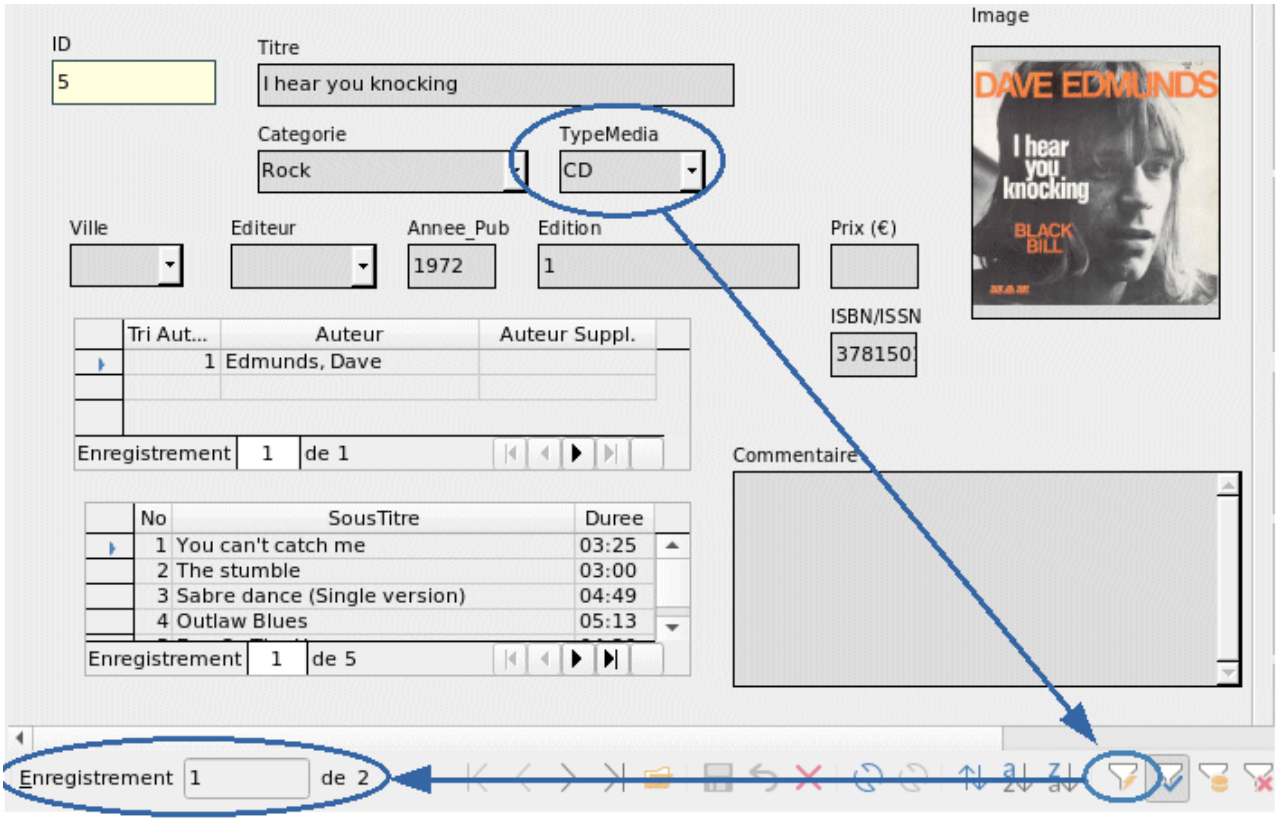
Malheureusement, ce type de recherche présente les inconvénients suivants :

La recherche de paramètres est trop compliquée pour une utilisation normale.

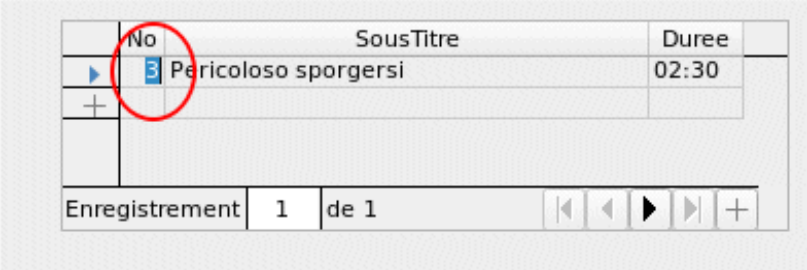
- La fonction de recherche elle-même est très lente, car elle n’utilise pas les fonctions de requête de la base de données.
- La recherche ne fonctionne que pour un formulaire à la fois, mais pour tous les champs. Les sous-formulaires doivent être examinés séparément.
- Nous recherchons dans un sous-formulaire qui appartient au formulaire principal actuel. Les inscriptions dans d’autres sous-formulaires ne seront pas révélées.
  Ainsi, par exemple, il n’est pas possible de trouver un sous-titre d’un support autre que celui actuellement affiché.
- La recherche s’applique aux fonctions de champ de liste sur la base de champs clés (clés étrangères) stockées dans la table. Les données affichées ne peuvent pas être utilisées.

Filtrer avec l’autofiltre

Le filtre automatique peut être sélectionné directement dans la barre de navigation. Cliquez sur un champ dans le formulaire principal et il filtrera sur le contenu de ce champ. Le filtre automatique fonctionne également avec des champs de liste et présente un avantage évident par rapport à la saisie manuelle de clés étrangères.



Dans l’exemple ci-dessus, un enregistrement contenant “CD” dans le champ TypeMedia est récupéré. Ensuite, le bouton de filtre automatique est basculé. Sur les 9 enregistrements d’origine de la table Media, seuls 2 enregistrements s’affichent désormais dans le compteur d’enregistrements.



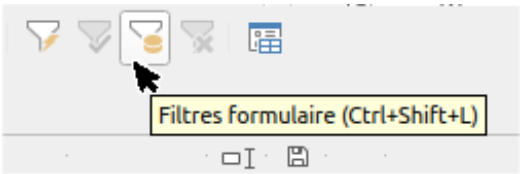
Bien que seuls deux enregistrements aient un “3” dans le champ No°, les principaux formulaires de tous les autres enregistrements continuent d’être affichés. Dans le sous-formulaire, seuls les enregistrements correspondant à “3” sont affichés.

Cependant, l’utilisation du filtre automatique ne peut pas résoudre les problèmes suivants :

- Si vous recherchez dans un sous-formulaire, seule la valeur spécifiée y sera affichée, mais le formulaire principal est affiché pour tous les enregistrements même s’ils ne contiennent pas cette valeur. Ainsi, par exemple, si vous recherchez un support avec un deuxième auteur, vous verrez toujours tous les supports qui n’ont qu’un seul auteur. Dans ces enregistrements, le champ auteur sera vide.
- La valeur utilisée pour le filtre doit être spécifique. Il n’y a aucune possibilité de recherche de similitude pour trouver des matériaux connexes.

Filtrage avec le filtre basé sur le formulaire

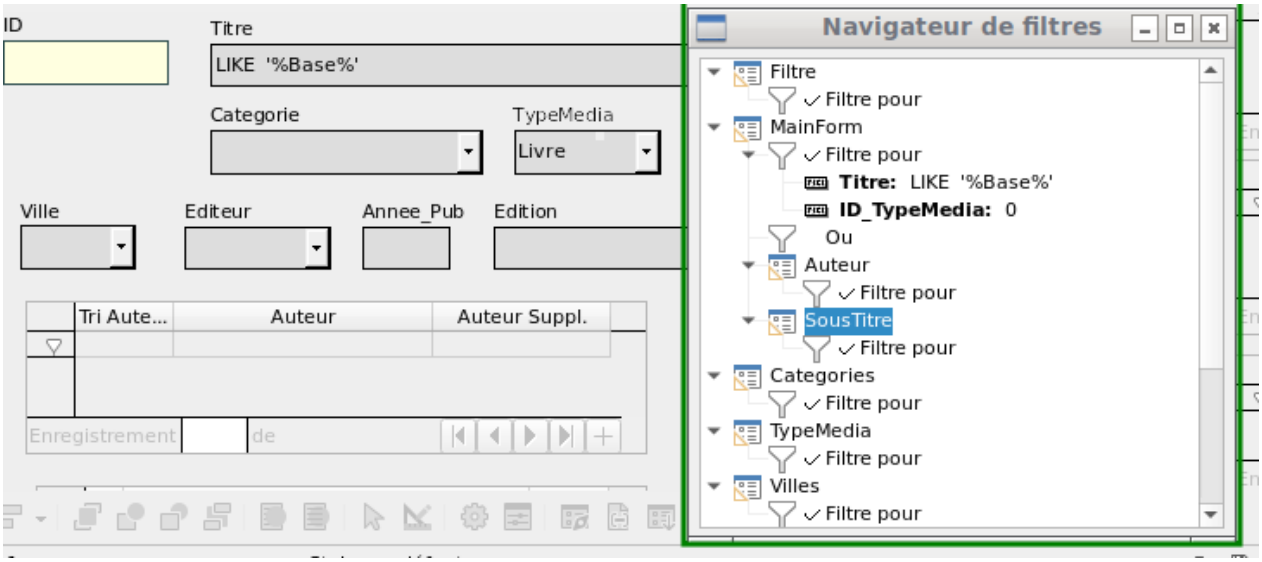
Le filtre basé sur un formulaire offre un formulaire pour insérer des valeurs réelles à utiliser pour le filtrage plutôt qu’un simple bouton de filtre. Ici, vous pouvez filtrer plusieurs valeurs en même temps. Ces valeurs peuvent être combinées sous forme de condition conjointe (AND) ou d’alternatives (OR).



Start Filtre Basé sur un formulaire

Les valeurs de filtre entrées ne doivent pas être exclusivement uniques. Ici donc vous pouvez formuler des conditions comme celles décrites dans le filtrage des tables. Ainsi, LIKE ’% base%’ saisi dans le champ du titre du média donnera tous les enregistrements dont le titre contient le mot ‘base’.

Le filtrage des champs de zones de liste est effectué par sélection directe du contenu des champs de liste affichés. Vous n’avez pas besoin de saisir les clés étrangères sous-jacentes.

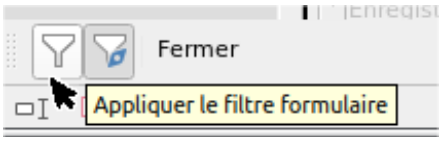


Ici, nous recherchons tous les enregistrements avec le TypeMedia “Livre” (correspondant à une valeur de clé étrangère de “0”) et avons également quelque part dans le titre la séquence de caractères “Base”. Lorsque vous entrez `LIKE '%Base%*'`, l’interface utilisateur graphique remplace les symboles génériques SQL habituels «%» par le caractère «*», un symbole générique plus familier pour la plupart des utilisateurs.

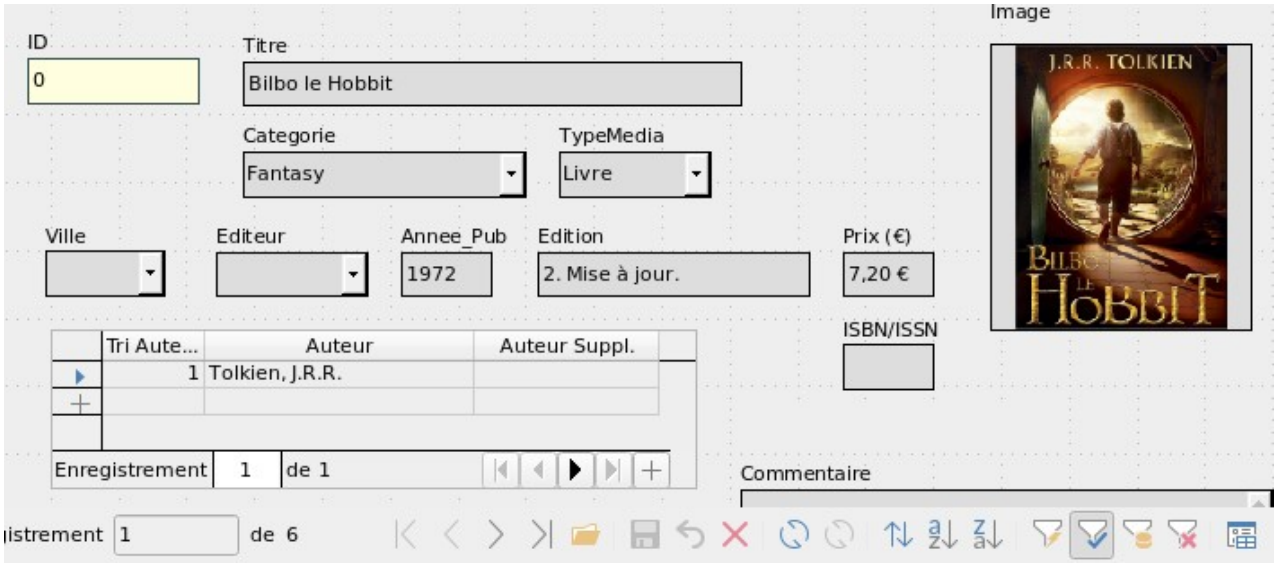
Nous pourrions également ajouter la condition que tous les enregistrements pour lesquels l’année de publication est “1973” soient affichés.

Lors de la saisie, le filtre basé sur un formulaire affiche uniquement les champs qui affichent une coche avant OR. De cette manière, plusieurs conditions peuvent être définies pour le champ. Malheureusement, cela provoque la suppression par le filtre de l’affichage du contenu du champ de liste TypeMedia, alors que cela fonctionne dans le contrôle de table du sous-formulaire.

Lors de la création du filtre, vous ne voyez pas la barre de navigation ; à la place, les choix du filtre basé sur le formulaire sont affichés. Vous filtrez selon vos besoins, insérez les valeurs de filtre, puis fermez simplement l’affichage.



Le filtre est appliqué et le formulaire est filtré selon vos spécifications.



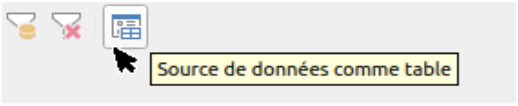
Lorsque le filtrage est activé, le nombre d'enregistrements est limité à ceux qui correspondent. Dans le cas, il y a six correspondances. Dans l'illustration, ni le titre ni le type de média ne correspondent ; la condition que l'année de publication soit inférieure à 1980 fournit la correspondance.

Ce filtre assez réussi présente encore l'inconvénient suivant :

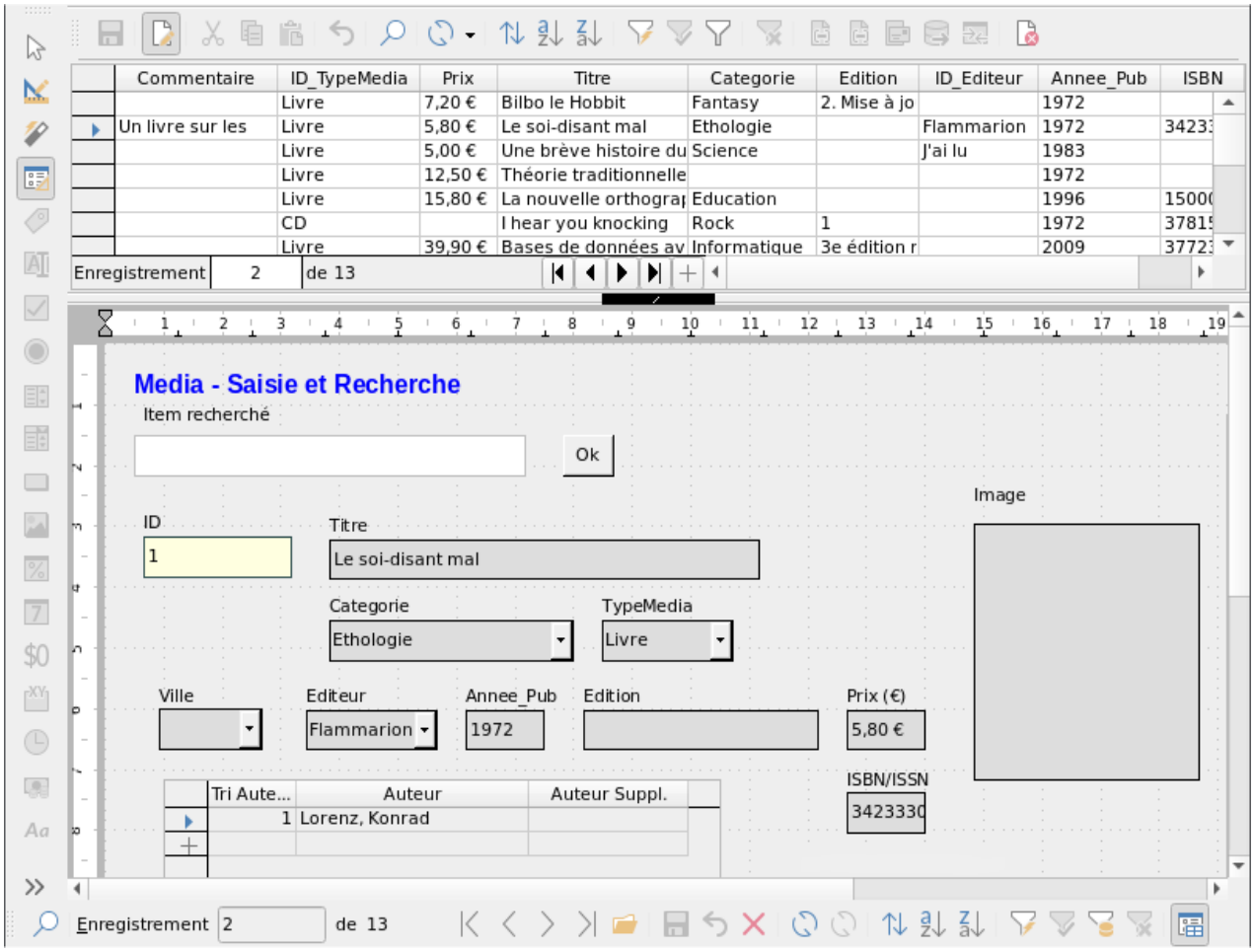
- Vous avez besoin de connaissances détaillées sur la façon de définir des conditions si vous recherchez quelque chose de plus compliqué que des champs complets. La plupart des utilisateurs, même s'ils sont habitués à utiliser les moteurs de recherche, n'ont pas les compétences nécessaires.

Filtrer avec le filtre par défaut

Le filtre par défaut peut être atteint en demandant l'affichage de la source de données sous forme de tableau. Ensuite, les procédures sont les mêmes que lors de l'utilisation du filtre par défaut dans une table.



Lancement du filtre par défaut avec la source de données sous forme de tableau



À droite du navigateur de Formulaire, vous pouvez voir un bouton *Source de données comme table*. En vue tableau, le filtre par défaut devient disponible. Contrairement à la vue de la table Medias réelle, les clés étrangères sont affichées, non pas avec leurs valeurs réelles, mais avec le contenu qui y correspond. Un clic sur le champ TypeMedia montre qu'il s'agit de la source du champ de liste qui apparaît dans le formulaire.

Note

Malheureusement, la vue ci-dessus montre un bogue qui remonte au début de Base. Le numéro ISBN à 13 chiffres n'est pas correctement formaté. Le champ correspondant reçoit à la place le nombre minimum de chiffres possible. Si un numéro a plus de 9 chiffres, attention : la modification peut entraîner une perte de données (bogue 82411).

Ce bogue apparaît également dans les contrôles de table. Ils s'affichent correctement si vous utilisez un champ formaté au lieu d'un champ numérique.

La vue tableau d'un formulaire présente les inconvénients suivants pour cette méthode de recherche :

- La recherche fonctionne uniquement dans la seule table ou requête qui sous-tend le formulaire et non dans un sous-formulaire qui lui appartient.
- Dans la vue des données, les champs de liste sont affichés, mais vous ne pouvez pas les utiliser pour le filtrage. Une fois de plus, vous devez connaître les valeurs de clé étrangère réelles pour ces champs.

Par exemple, si le type de média est “Livre” et qu’il a la valeur de clé primaire “1”, alors la table Medias a “1” stocké dans ce champ. Vous devez rechercher cette valeur, même si la vue composite affiche la valeur “Livre”.

Résumé

Les fonctions de recherche et de filtrage sont d’une utilité limitée pour les formulaires. La recherche est beaucoup trop lente et ne limite pas le nombre total d’enregistrements à ceux qui correspondent vraiment. Les filtres ne fonctionnent que dans un formulaire à la fois. Si vous souhaitez filtrer dans un sous-formulaire, seul le sous-formulaire est affecté. Vous ne pouvez tout simplement pas obtenir le nombre correct d’enregistrements correspondants.

Par conséquent, dans les formulaires de la base de données d’exemple, une fonction de recherche conviviale a été intégrée dans le formulaire, une procédure qui doit être élaborée à la main et nécessite une certaine connaissance de SQL.

Saisie d’enregistrements et navigation

Si un formulaire a été conçu de telle sorte que son ouverture active le focus de contrôle, le curseur apparaîtra dans le premier champ de saisie. Au lieu d’utiliser la souris pour passer d’un champ à l’autre, vous utilisez la touche de tabulation pour passer au champ suivant dans la séquence.

Un formulaire vous permet de passer à un autre champ, soit avec la souris, soit avec un raccourci.

En utilisant **Propriétés : Zone de texte > Champ d’étiquette**, un champ d’étiquette est attribué. L’étiquette de ce champ d’étiquette passe de “Nom” à “No~m”. Dans le formulaire, la lettre “m” de l’étiquette est maintenant soulignée. Vous pouvez maintenant accéder instantanément à ce champ de texte en utilisant le raccourci Alt + m. Cela ne fonctionne que si le curseur est déjà dans un champ du formulaire.

En principe, n’importe quelle lettre peut être utilisée comme raccourci, car le saut a lieu entièrement dans le formulaire. Cependant, si le curseur n’est pas initialement dans un contrôle de formulaire, les raccourcis activent plutôt l’interface utilisateur de LibreOffice. Ainsi, par exemple, le raccourci “a” ne passerait pas au champ Nom mais ouvrirait à la place le menu Tableau.

Malheureusement, les sauts de champ ne fonctionnent que dans un seul formulaire, pas dans les structures qui impliquent des sous-formulaires. Un saut d’un sous-formulaire vers le formulaire principal n’est actuellement pas possible.

Lorsque le curseur est dans un contrôle de table, il sort lorsque vous utilisez *Ctrl+Tab*. Parfois, il arrive que cette combinaison de touches n’ait pas l’effet souhaité. Voici donc une alternative :[4]

Un bouton est créé dans le sous-formulaire. Son titre est “~Nouveau”. Le caractère tilde est utilisé, comme décrit ci-dessus, afin que *Alt+N* puisse être utilisé pour activer le bouton. Le champ Informations supplémentaires du bouton nomme le champ du formulaire principal vers lequel le

4 Voir aussi l’exemple de base de données Exemple_Saut_Curseur_Formulaires.odb.associée à ce manuel.

curseur doit être transféré lors de l'activation. Le bouton est lié à la macro suivante à l'aide de **Propriétés > Événements > Exécuter l'action :**

```
SUB SautVersFormulairePrincipal(oEvent AS OBJECT)
       DIM oChamp AS OBJECT
       DIM oFormulaire AS OBJECT
       DIM oDoc AS OBJECT
       DIM oController AS OBJECT
       DIM oVue AS OBJECT
       DIM stRaccourci AS STRING
       oChamp = oEvent.Source.Model
       stRaccourci = Mid(oChamp.Label,InStr(oChamp.Label,"~") + 1, 1)
       oFormulaire = oChamp.Parent.Parent
       SELECT CASE stRaccourci
          CASE "n"
             oFormulaire.MoveToInsertRow()
          CASE "a"
             IF oFormulaire.isLast() THEN
                oFormulaire.MoveToInsertRow()
             ELSE
                oFormulaire.Next()
             END IF
       END SELECT
       oDoc = thisComponent
       oController = oDoc.getCurrentController()
       oVue = oController.getControl(oFormulaire.getByname(oChamp.Tag))
       oVue.setFocus
END SUB
```

La macro peut être utilisée simplement pour revenir au formulaire principal. Elle peut également être utilisée avec les paramètres ci-dessus pour créer un nouvel enregistrement immédiatement ou, à l'aide d'un autre bouton de la barre de navigation pour Enregistrement suivant, pour passer à l'enregistrement suivant. Si le curseur dans le formulaire principal est sur le dernier enregistrement, le saut conduira à un nouvel enregistrement. Pour plus d'informations sur les macros, reportez-vous au chapitre 9.

Passer à un bouton nécessite que le bouton soit visible. Cependant, il peut être assez petit ou même avoir une largeur et une hauteur de 0 cm. Si la largeur et la hauteur ne sont pas définies, il peut arriver que le bouton apparaisse comme une ligne parcourant tout l'écran.

Impression à partir de formulaires

Les formulaires peuvent être construits de telle sorte qu'une impression directe du formulaire soit possible. Pour obtenir des résultats utilisables, vous devez veiller à ne pas placer d'éléments en dehors de la zone imprimable. Choisissez **Affichage > Normal**. Les propriétés de la page peuvent ensuite être définies. Un fond coloré n'est pas un avantage ici.

Tout élément individuel peut être exclu de l'impression en utilisant **Propriétés > Général > Imprimable.**

Si la base de données est enregistrée dans LibreOffice (en utilisant **Outils > Options > LibreOffice Base > Base de données**, ou directement dans le cadre du processus de création), le formulaire peut être utilisé pour les publipostages. Le formulaire est ouvert pour modification. Les sources de données sont rendues accessibles en utilisant **Affichage > Sources de données** ou en appuyant sur *F4*. Les champs de base de données peuvent maintenant être glissés dans le

formulaire par leurs en-têtes de tableau. Ensuite, le formulaire est enregistré. Si le même formulaire est maintenant ouvert pour la saisie, LibreOffice reconnaît qu'il s'agit de champs pour un publipostage et vous demande si vous souhaitez imprimer les lettres.

Vous trouverez des détails sur la façon de procéder à un publipostage dans le Chapitre 7, Liaison aux bases de données.

Guide Base

Chapitre 5
Requêtes

Informations générales sur les requêtes

Les requêtes vers une base de données sont l'outil le plus puissant dont nous disposons pour utiliser les bases de données de manière pratique. Elles peuvent rassembler les données de différentes tables, calculer les résultats si nécessaire et filtrer rapidement des enregistrements spécifiques à partir d'une masse de données. Les grandes bases de données Internet que les gens utilisent chaque jour existent principalement pour fournir un résultat rapide et pratique à l'utilisateur à partir d'une énorme quantité d'informations par une sélection judicieuse de mots-clés – y compris les publicités liées à la recherche qui encouragent les gens à faire des achats.

Saisie de requêtes

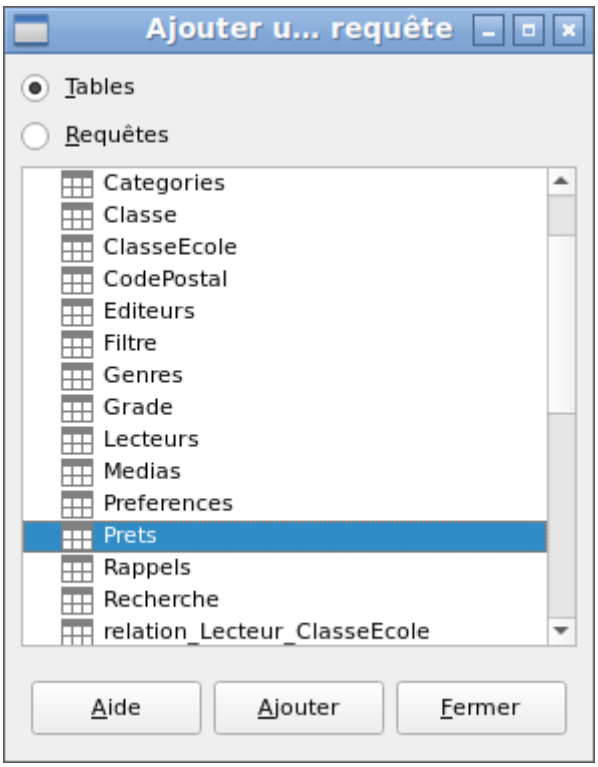
Les requêtes peuvent être saisies à la fois dans l'interface graphique et directement sous forme de code SQL. Dans les deux cas, une fenêtre s'ouvre, dans laquelle vous pouvez créer une requête et également la corriger si nécessaire.

Création de requêtes à l'aide de la boîte de dialogue Ébauche de requête

La création de requêtes à l'aide de l'assistant est brièvement décrite au chapitre 8 du Guide de mise en route, Débuter avec Base. Ici, nous allons expliquer la création directe de requêtes en mode Design.

Dans la fenêtre principale de la base de données, cliquez sur l'icône **Requêtes** dans la section Bases de données, puis dans la section Tâches, cliquez sur **Créer une requête en mode Ébauche**. Deux boîtes de dialogue apparaissent. L'une fournit la base pour une création en vue de conception de la requête ; l'autre sert à ajouter des tables, des vues ou des requêtes à la requête courante.

Comme notre formulaire simple fait référence à la table Prets, nous allons d'abord expliquer la création d'une requête à l'aide de cette table.



Dans les tables disponibles, sélectionnez la table Prets. Cette fenêtre permet de combiner plusieurs tables (ainsi que des vues et des requêtes). Pour sélectionner une table, cliquez sur son nom, puis sur le bouton Ajouter.
Ou bien, double-cliquez sur le nom de la table. Les deux méthodes ajoutent la table à la zone graphique de la boîte de dialogue Ébauche de requête (Figure 84).

Lorsque toutes les tables nécessaires ont été sélectionnées, cliquez sur le bouton Fermer. Des tables et des requêtes supplémentaires peuvent être ajoutées ultérieurement si nécessaire. Cependant, aucune requête ne peut être créée sans au moins une table, une sélection doit donc être effectuée pour commencer.

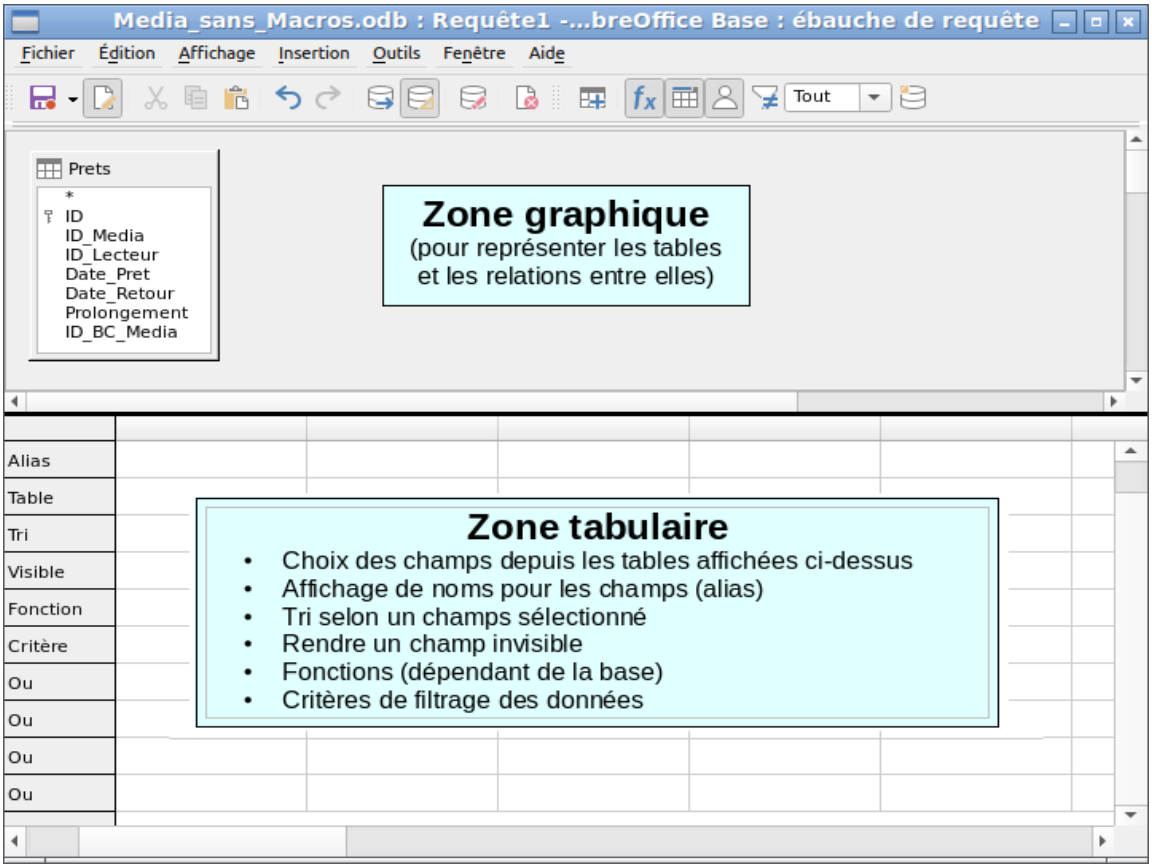
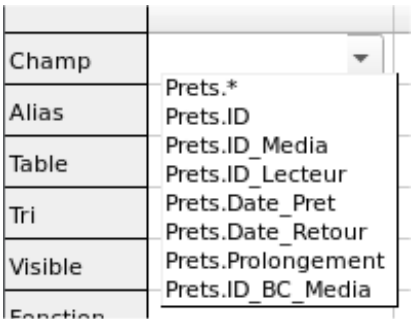


Figure 84: Zones de la boîte de dialogue Ébauche de requête

La Figure 84 affiche les divisions de base de la boîte de dialogue Ébauche de requête : la zone graphique affiche les tables à lier dans la requête. Leurs relations les unes avec les autres par rapport à la requête peuvent également être affichées. La zone tabulaire sert à la sélection des champs à afficher, ou à la définition des conditions liées à ces champs.

Cliquez sur le champ dans la première colonne de la zone tabulaire pour faire apparaître une flèche vers le bas. Cliquez sur cette flèche pour ouvrir la liste déroulante des champs disponibles. Le format est Nom_Table. Nom_Champ – c'est pourquoi tous les noms de champ ici commencent par le mot Prets.

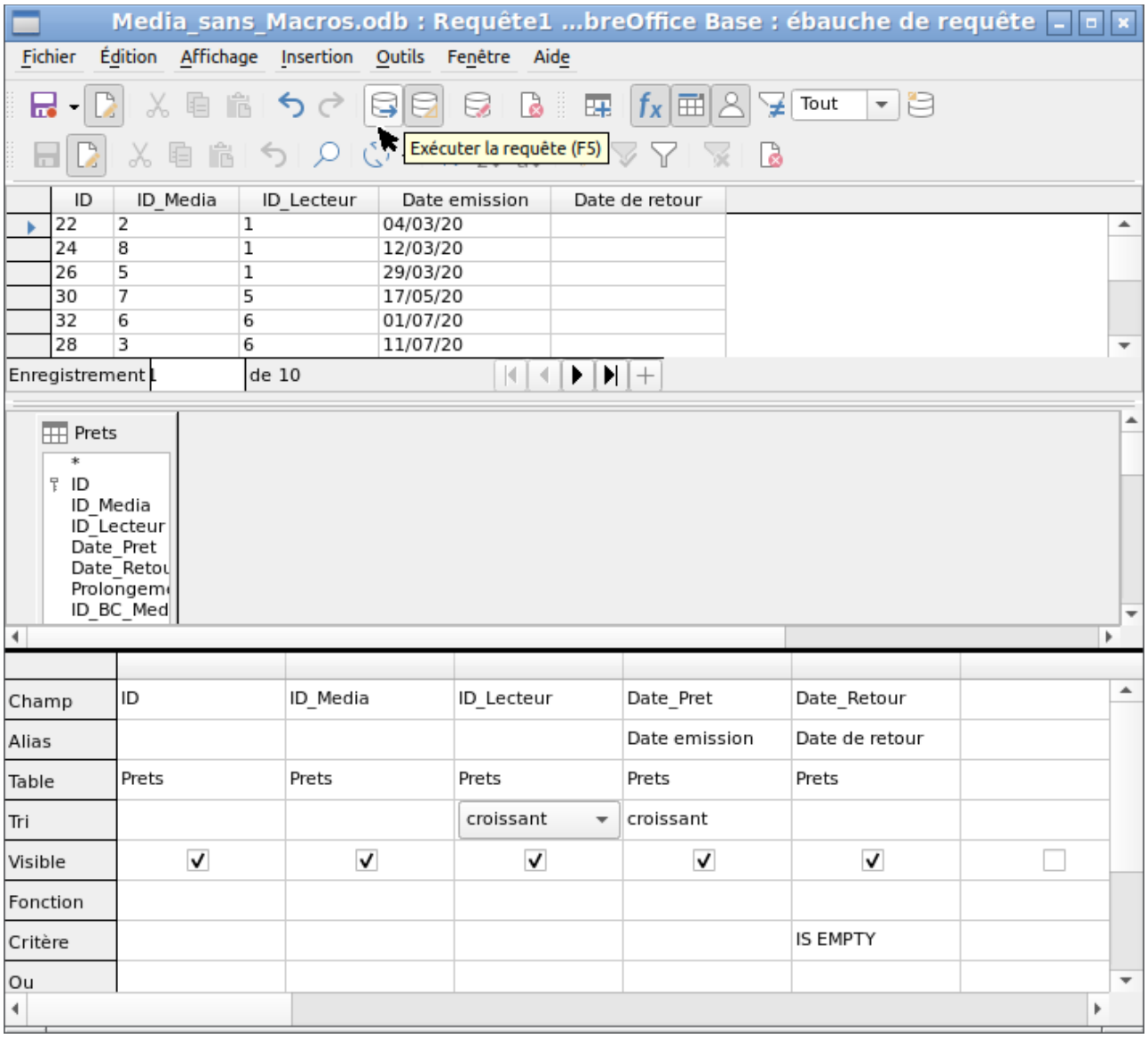


La désignation de champ sélectionnée Prets.* A une signification particulière. Ici, un clic vous permet d'ajouter tous les champs de la table sous-jacente à la requête. Lorsque vous utilisez cette désignation de champ avec le caractère générique * pour tous les champs, la requête ne peut plus être distinguée de la table.

Conseil

Pour un transfert rapide de tous les champs d'une table dans une requête, il suffit de cliquer sur l'étoile dans la vue de table dans l'interface graphique (Prets.* ci-dessus).

Un double-clic sur un champ insère ce champ dans la zone tabulaire à la prochaine position libre.



Les cinq premiers champs de la table Prêt sont sélectionnés. Les requêtes en mode Ébauche peuvent toujours être exécutées en tant que tests (icône Excétuter la requete). Cela fait apparaître une vue tabulaire des données au-dessus de la vue graphique de la table Prêt avec sa liste de champs (pour fermer cette fenêtre, faire F4 ou Affichage/Apercu). Un test d'exécution d'une requête est toujours utile avant de l'enregistrer, pour clarifier si la requête atteint réellement son objectif. Souvent, une erreur logique empêche une requête de récupérer des données. Dans d'autres cas, il peut arriver que les enregistrements que vous souhaitiez exclure s'affichent malencontreusement.

En principe, une requête qui produit un message d'erreur dans la base de données sous-jacente ne peut pas être enregistrée tant que l'erreur n'est pas corrigée.

	ID	ID_Media
▶	1	1
	2	6
	3	3
	9	5
	10	4
*	<Autc	

Figure 85: Une requête modifiable

	ID_Media	ID_Lecteur
▶	2	1
	8	1
	5	1
	7	5
	6	6
	3	6

Figure 86: Une requête non modifiable

Dans le test ci-dessus, portez une attention particulière à la première colonne du résultat de la requête. Le marqueur d'enregistrement actif (ici flèche verte, mais peut être d'une couleur différente) apparaît toujours sur le côté gauche du tableau, pointant ici sur le premier enregistrement comme enregistrement actif. Alors que le premier champ du premier enregistrement de la figure 85 est mis en évidence ou que le curseur clignote dans la cellule, le champ correspondant de la figure 86 montre uniquement une bordure en pointillés. Le surlignage, ou le clignotement du curseur, indique que ce champ peut être modifié. Les enregistrements, en d'autres termes, sont modifiables. La bordure en pointillés indique que ce champ ne peut pas être modifié. La figure 85 contient également une ligne supplémentaire pour l'entrée d'un nouvel enregistrement, avec le champ ID déjà marqué comme <AutoChamp>. Cela montre également que de nouvelles entrées sont possibles.

Conseil

Une règle de base est qu'aucune nouvelle entrée n'est possible si la clé primaire de la table interrogée n'est pas incluse dans la requête.

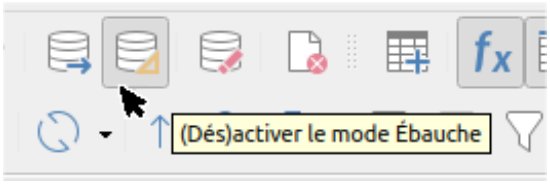
Champ	ID	ID_Media	ID_Lecteur	Date_Pret	Date_Retour
Alias				Date emission	Date de retour

Les champs Date_Pret et Date_Retour reçoivent des alias. Cela ne les conduit pas à être renommés mais uniquement à apparaître sous ces noms pour l'utilisateur de la requête.

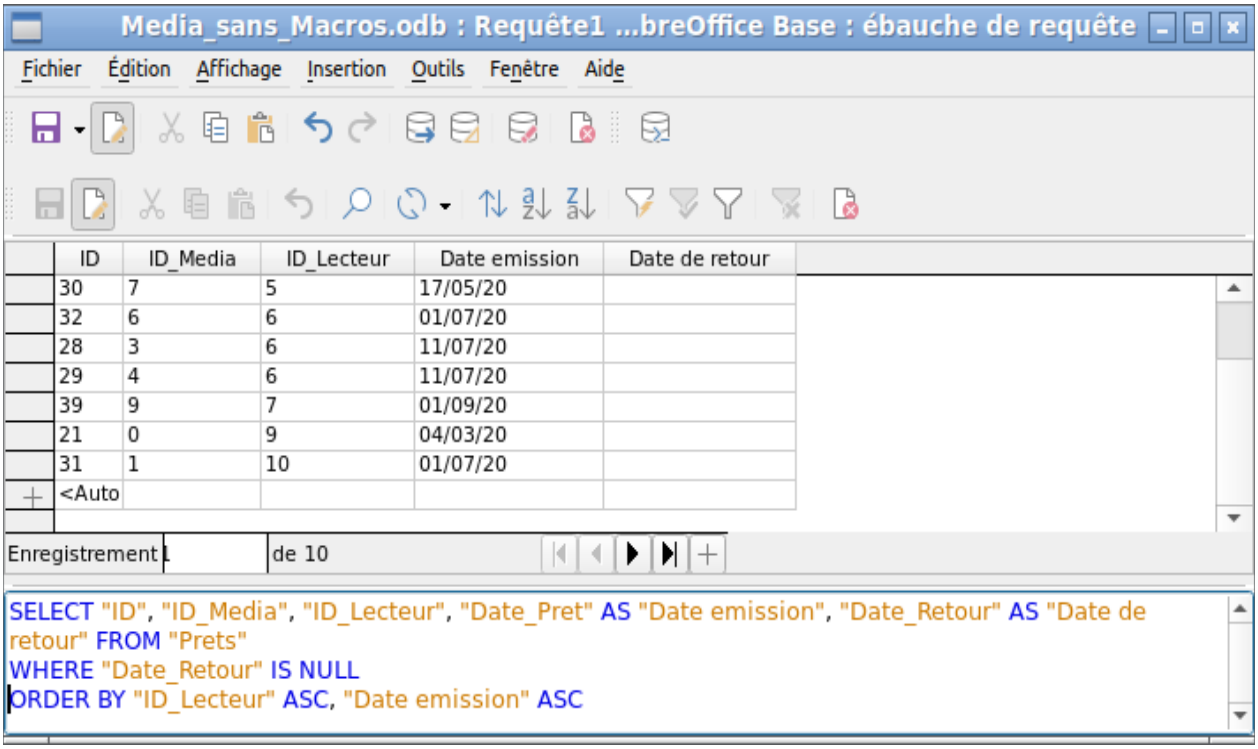
ID	ID_Media	ID_Lecteur	Date emission	Date de retour
22	2	1	04/03/20	

Le tableau ci-dessus montre comment les alias remplacent les noms de champ réels.

Le champ Date_Retour reçoit non seulement un alias, mais aussi un critère de recherche, ce qui entraînera l'affichage uniquement des enregistrements pour lesquels le champ Date_Retour est vide. (Entrez `IS EMPTY` dans le ligne de critère du champ Date_Retour.) Ce critère d'exclusion entraînera uniquement l'affichage des enregistrements relatifs aux supports qui n'ont pas encore été renvoyés.



Pour mieux apprendre le langage SQL, il vaut la peine de basculer de temps en temps entre le mode conception et le mode SQL en désactivant le mode ébauche.



Ici, la formule SQL créée par nos choix précédents est révélée. Pour faciliter la lecture, des sauts de ligne ont été inclus. Malheureusement, l'éditeur ne stocke pas ces sauts de ligne, donc lorsque la requête est à nouveau appelée, elle apparaîtra comme une seule ligne continue jusqu'aux bords de la fenêtre.

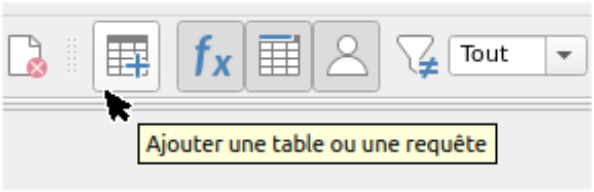
SELECT commence les critères de sélection. AS spécifie les alias de champ à utiliser. FROM montre la table qui doit être utilisée comme source de la requête. WHERE donne les conditions de la requête, à savoir que le champ Date_Retour doit être vide (IS NULL). ORDER BY définit les critères de tri, à savoir l'ordre croissant (ASC – croissant ou ascending) pour les deux champs *ID_Lecteur* et *Date emission*. Cette spécification de tri illustre comment l'alias du champ *Date_Pre*t peut être utilisé dans la requête elle-même.

Conseil

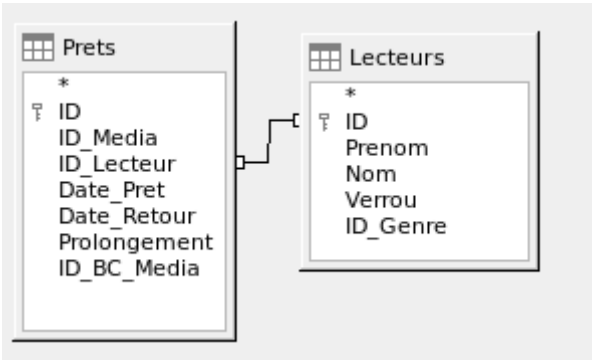
Lorsque vous travaillez en mode d'affichage de conception, utilisez IS EMPTY pour exiger qu'un champ soit vide. Lorsque vous travaillez en mode SQL, utilisez IS NULL, ce qui est requis par SQL (Structured Query Language).

Lorsque vous souhaitez trier par ordre décroissant à l'aide de SQL, utilisez DESC au lieu de ASC.

Jusqu'à présent, les champs ID_Media et ID_Lecteur ne sont visibles que sous forme de champs numériques. Les noms des lecteurs ne sont pas écrits en clair. Pour les afficher dans une requête, la table Lecteurs doit être incluse dans la requête. Pour cela, nous retournons au mode Ébauche. Ensuite, une nouvelle table peut être ajoutée à la vue conception de requête.



Ici, d'autres tables ou requêtes peuvent être ajoutés ultérieurement et rendues visibles dans l'interface utilisateur graphique. Si des liens entre les tables ont été déclarés au moment de leur création (voir Chapitre 3, Tables), alors ces tables sont affichées avec les liens directs correspondants.



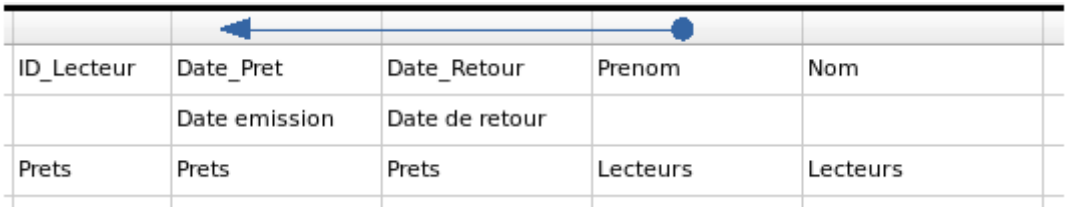
Si un lien est absent, il peut être créé à ce stade en faisant glisser la souris de "Prets". "ID_Lecteur" vers "Lecteurs". "ID".

Note

La liaison des tables ne fonctionne pour le moment que dans la base de données interne ou dans les bases de données relationnelles externes. Par exemple, les tables d'une feuille de calcul ne peuvent pas être liés entre elles. Elles doivent d'abord être importés dans une base de données interne.

Pour créer un lien entre les tables, une simple importation suffit sans création supplémentaire de clé primaire.

Désormais, les champs de la table Lecteurs peuvent être saisis dans la zone tabulaire. Les champs sont initialement ajoutés à la fin de la requête.



ID_Lecteur	Date_Pret	Date_Retour	Prenom	Nom
	Date emission	Date de retour		
Prets	Prets	Prets	Lecteurs	Lecteurs

La position des champs peut être corrigée dans la zone tabulaire de l'éditeur à l'aide de la souris. Ainsi, par exemple, le champ Prenom a été déplacé en position directement avant le champ Date_Pret.

ID	ID_Media	ID_Lecteur	Prenom	Nom	Date emission	Date de retour
22	2	1	Lino	Ventura	04/03/20	
24	8	1	Lino	Ventura	12/03/20	
26	5	1	Lino	Ventura	29/03/20	
30	7	5	Marie	Laforêt	17/05/20	
32	6	6	Valérie	Benguigui	01/07/20	
28	3	6	Valérie	Benguigui	11/07/20	
29	4	6	Valérie	Benguigui	11/07/20	
39	9	7	Martine	Carol	01/09/20	
21	0	9	André	Raimbourg	04/03/20	

Enregistrement 1 de 10

Maintenant, les noms sont visibles. L'ID_Lecteur est devenu superflu. Le tri par Nom et Prenom a également plus de sens que le tri par ID_Lecteur.

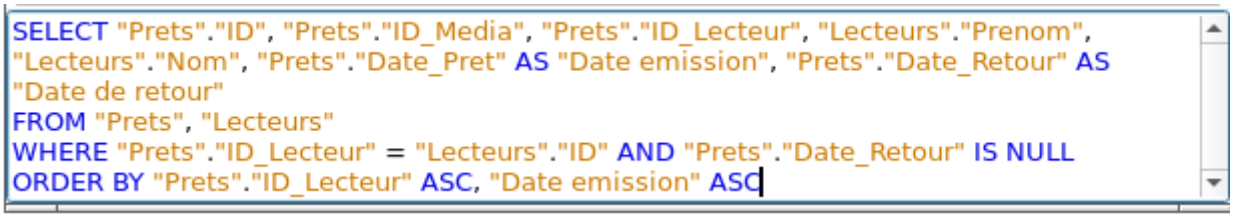
Cette requête ne convient plus pour une utilisation en tant que requête autorisant de nouvelles entrées dans la table résultante, car il lui manque la clé primaire pour la table Lecteurs ajoutée. Les champs de la requête redeviennent modifiables uniquement si cette clé primaire est intégrée. Pour cette raison, rendre les résultats de requête modifiables est une fonctionnalité qui doit être utilisée avec une extrême prudence, si nécessaire sous le contrôle d'un formulaire.

Attention

Avoir une requête que vous pouvez modifier peut créer des problèmes. La modification des données dans la requête modifie également les données de la table sous-jacente et les enregistrements contenus dans la table. Les données peuvent ne pas avoir la même signification. Par exemple, changez le nom du lecteur, et vous avez également changé les livres que le lecteur a empruntés et retournés.

Si vous devez modifier des données, faites-le dans un formulaire afin de voir les effets de la modification des données.

Même lorsqu'une requête peut être modifiée ultérieurement, elle n'est pas si simple à utiliser comme formulaire avec des zones de liste, qui affichent les noms des lecteurs mais contiennent le ID_Lecteur de la table. Les zones de liste ne peuvent pas être ajoutées à une requête ; elles ne sont utilisables que dans les formulaires.



Si nous retournons maintenant à la vue SQL, nous voyons que tous les champs sont maintenant affichés entre guillemets doubles : "Nom_Table". "Nom_Champ". Ceci est nécessaire pour que la base de données sache de quelle table proviennent les champs précédemment sélectionnés. Après tout, les champs de différentes tables peuvent facilement avoir les mêmes noms de champ (ce qui n'est pas recommandé).
Dans la structure de tableau ci-dessus, cela est particulièrement vrai pour le champ ID.

Note

La requête suivante fonctionne sans placer les noms de table devant les noms de champ :

```
SELECT "ID", "Nombre", "Prix" FROM "Stock", "Distribution" WHERE
"Distribution"."ID_Stock" = "Stock"."ID"
```

Ici, l'ID est extrait de la table qui vient en premier dans la définition FROM. La définition de table dans la formule WHERE est également superflue, car ID_Stock n'apparaît qu'une seule fois (dans la table Distribution) et l'ID a été clairement extrait de la table Stock (à partir de la position de la table dans la requête).

Si un champ de la requête a un alias, il peut être référencé – par exemple lors du tri – par cet alias sans qu'un nom de table ne soit donné. Le tri est effectué dans l'interface utilisateur graphique selon la séquence des champs dans la vue tabulaire. Si à la place vous voulez trier d'abord par "Date_Pret" puis par "Prets". "ID_Lecteur", cela peut être fait si :

- La séquence des champs dans la zone de tableau de l'interface utilisateur graphique est modifiée (glisser-déposer "Date_Pret" à gauche de "Prets". "ID_Lecteur", ou
- Un champ supplémentaire est ajouté, défini pour être invisible, juste pour le tri (cependant, l'éditeur ne l'enregistrera que temporairement si aucun alias n'a été défini pour lui) [ajouter un autre champ "Date_Pret" juste avant "Prets". "ID_Lecteur" ou ajouter un autre champ "Prets". "ID_Lecteur" juste après "Date_Pret"], ou
- Le texte de la commande ORDER BY dans l'éditeur SQL est modifié en conséquence (ORDER BY "Date_Pret", "Prets.". "ID_Lecteur").

Conseil

Une requête peut nécessiter un champ qui ne fait pas partie du résultat de la requête. Dans le graphique de la section suivante, Date_Retour est un exemple. Cette requête recherche les enregistrements qui ne contiennent pas de date de retour. Ce champ fournit un critère pour la requête mais aucune donnée visible utile.

Utilisation de fonctions dans une requête

L'utilisation de fonctions permet à une requête de fournir plus qu'une simple vue filtrée des données dans une ou plusieurs tables. La requête suivante calcule le nombre de supports prêtés, en fonction du champ ID_Lecteur.

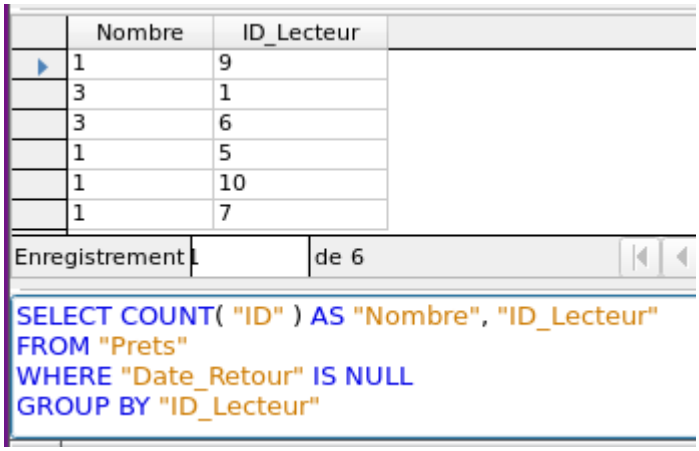
Champ	ID	ID_Lecteur	Date_Retour
Alias	Nombre		
Table	Prets	Prets	Prets
Tri			
Visible	☑	☑	☑
Fonction	Nombre	Groupe	
Critère			IS EMPTY

Figure 87: Fonction de comptage et de groupe

Pour l'ID de la table Prets, la fonction Nombre est sélectionnée. En principe, le champ de la table choisi pour cela ne fait aucune différence. La seule condition est la suivante : le champ ne doit être vide dans aucun des enregistrements. Pour cette raison, le champ de clé primaire, qui n'est jamais vide, est le choix le plus approprié. Tous les enregistrements avec un contenu de champ autre que NULL sont comptés.

Pour le champ ID_Lecteur, qui donne accès aux informations du lecteur, la fonction de regroupement est choisie. De cette manière, les enregistrements avec le même ID_Lecteur sont regroupés. Le résultat montre le nombre d'enregistrements pour chaque ID_Lecteur.

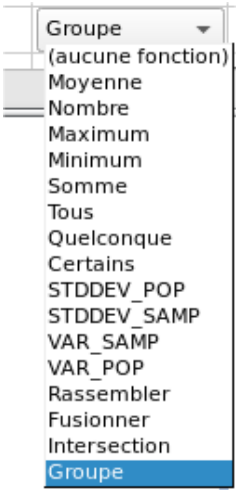
En tant que critère de recherche, Date_Retour est défini sur "IS EMPTY" et en colonne non visible, comme dans l'exemple précédent. (Ci-dessous, le code SQL pour cela est WHERE "Date_Retour" IS NULL.)



Nombre	ID_Lecteur
1	9
3	1
3	6
1	5
1	10
1	7

Enregistrement 1 de 6

```
SELECT COUNT( "ID" ) AS "Nombre", "ID_Lecteur"
FROM "Prets"
WHERE "Date_Retour" IS NULL
GROUP BY "ID_Lecteur"
```



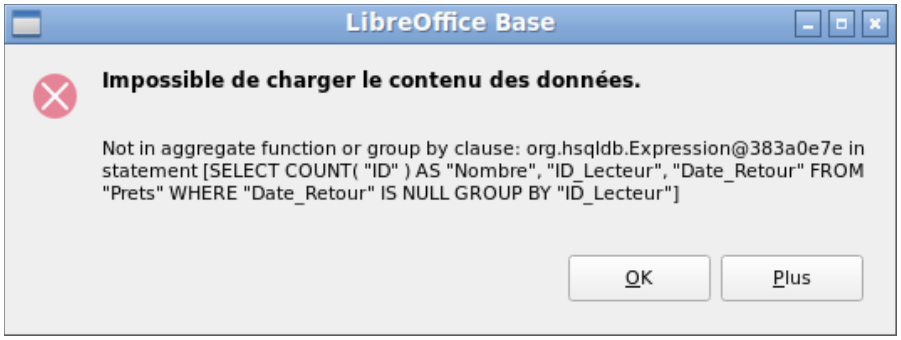
Le résultat de la requête montre que ID_Lecteur '1' a un total de 3 supports en prêt. Si la fonction Nombre avait été affectée à Date_Retour au lieu de l'ID, chaque ID_Lecteur aurait un média '0' en prêt, puisque Date_Retour est prédéfini comme NULL.

La formule correspondante en code SQL est indiquée ci-dessus.

Dans l'ensemble, l'interface utilisateur graphique fournit les fonctions illustrées à droite, qui correspondent aux fonctions de la HSQLDB sous-jacente.

Pour une explication des fonctions, voir "Amélioration des requêtes à l'aide du mode SQL" en page 255.

Si un champ d'une requête est associé à une fonction, tous les champs restants mentionnés dans la requête doivent également être associés à des fonctions pour être affichés. Si cela n'est pas garanti, vous obtenez le message d'erreur suivant :



Une traduction quelque peu libre serait : L'expression suivante ne contient aucune fonction d'agrégation ou de regroupement.

Pour cette raison, dans l'exemple figurant sur la page précédente (figure 87), le champ Date_Retour n'est pas affiché.

Conseil

Lors de l'utilisation du mode d'affichage Ébauche, un champ n'est visible que si la ligne Visible contient une coche pour le champ. Lors de l'utilisation du mode SQL, un champ n'est visible que lorsqu'il suit le mot-clé SELECT.

Note

Lorsqu'aucun champ n'est associé à une fonction, le nombre de lignes dans la sortie de la requête est déterminé par les conditions de recherche. Lorsqu'un champ est associé à une fonction, le nombre de lignes dans la sortie de la requête est déterminé par l'existence d'un regroupement ou non. S'il n'y a pas de regroupement, il n'y a qu'une seule ligne dans la sortie de la requête. S'il y a regroupement, le nombre de lignes correspond au nombre de valeurs distinctes du champ de regroupement. Ainsi, tous les champs visibles doivent être associés à une fonction ou être associés à une instruction de regroupement pour éviter ce conflit dans la sortie de la requête.

Après cela, la requête complète est répertoriée dans le message d'erreur, mais malheureusement sans que le champ incriminé soit spécifiquement nommé. Dans ce cas, le champ Date_Retour a été ajouté en tant que champ affiché. Ce champ n'est associé à aucune fonction et n'est pas non plus inclus dans l'instruction de regroupement.

Les informations fournies à l'aide du bouton Plus ne sont pas très éclairantes pour l'utilisateur normal de la base de données. Il affiche simplement le code d'erreur SQL.

Pour corriger l'erreur, supprimez la coche dans la ligne Visible pour le champ Date_Retour. Sa condition de recherche (critère) est appliquée lors de l'exécution de la requête, mais elle n'est pas visible dans la sortie de la requête.

En utilisant l'interface graphique, des calculs de base et des fonctions supplémentaires peuvent être effectués.

Champ	ID	ID_Media	ID_Lecteur	Date	Nombre("Rappels"."Date")	Date_Retour
Alias	Nombre			NombreRappels	MontantRappels	
Table	Prets	Prets	Prets	Rappels		Prets
Tri						
Visible	☑	☑	☑	☑	☑	☐
Fonction	Groupe	Groupe	Groupe	Nombre		
Critère						IS EMPTY

Supposons qu'une bibliothèque n'émette pas d'avis de rappel lorsqu'un article doit être retourné, mais émet des avis de retard dans les cas où la période de prêt a expiré et que l'article n'a pas été retourné. C'est une pratique courante dans les écoles et les bibliothèques publiques qui n'accordent des prêts que pour de courtes périodes fixes. Dans ce cas, l'émission d'un avis de retard signifie automatiquement qu'une amende doit être payée. Comment calculons-nous ces amendes ?

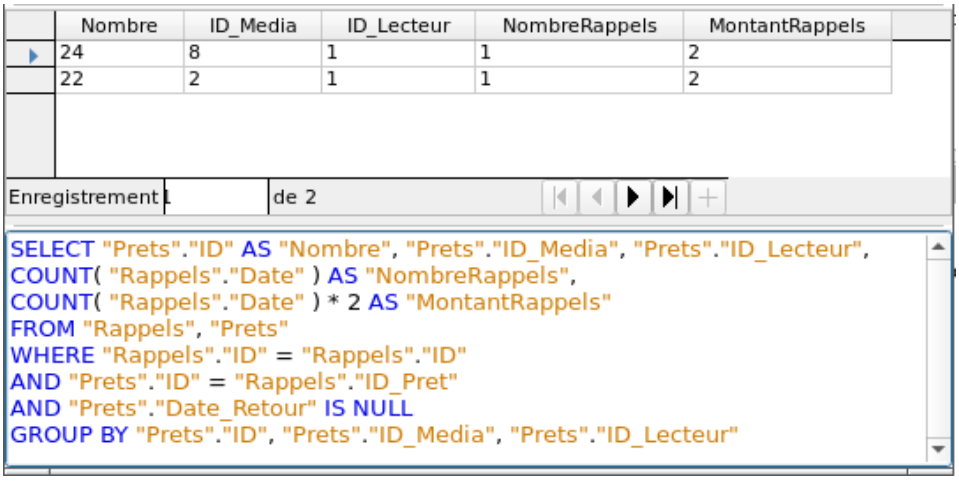
Dans la requête présentée ci-dessus, les tables Prêt et Rappels sont interrogées conjointement. À partir du nombre d'entrées de données dans le tableau Rappels, le nombre total d'avis de rappel est déterminé. L'amende pour les médias en retard est fixée dans la requête à 2,00 €. Au lieu d'un nom de champ, la désignation du champ est donnée comme Nombre (Rappels. Date) * 2. L'interface utilisateur graphique ajoute les guillemets et convertit le terme « compte » en la commande SQL appropriée.

Avertissement

Uniquement pour les personnes qui utilisent une virgule comme séparateur décimal :

Si vous souhaitez entrer des nombres avec des décimales à l'aide de l'interface graphique, vous devez vous assurer qu'un point décimal plutôt qu'une virgule est utilisé comme séparateur décimal dans l'instruction SQL finale. Les virgules sont utilisées comme séparateurs de champ, de sorte que de nouveaux champs de requête seraient créés pour la partie décimale.

Une entrée avec une virgule dans la vue SQL mène toujours à un champ supplémentaire contenant la valeur numérique de la partie décimale.

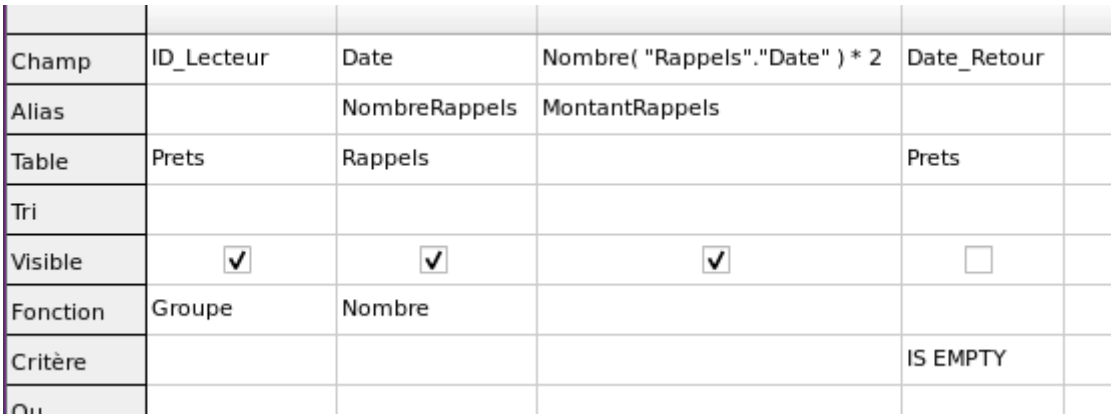


Nombre	ID_Media	ID_Lecteur	NombreRappels	MontantRappels
24	8	1	1	2
22	2	1	1	2

Enregistrement 1 de 2

```
SELECT "Prets"."ID" AS "Nombre", "Prets"."ID_Media", "Prets"."ID_Lecteur",
COUNT( "Rappels"."Date" ) AS "NombreRappels",
COUNT( "Rappels"."Date" ) * 2 AS "MontantRappels"
FROM "Rappels", "Prets"
WHERE "Rappels"."ID" = "Rappels"."ID"
AND "Prets"."ID" = "Rappels"."ID_Pret"
AND "Prets"."Date_Retour" IS NULL
GROUP BY "Prets"."ID", "Prets"."ID_Media", "Prets"."ID_Lecteur"
```

La requête donne maintenant pour chaque support encore en prêt les amendes qui se sont accumulées, en fonction des avis de rappel émis et du champ de multiplication supplémentaire. La structure de requête suivante sera également utile pour calculer les amendes dues par les utilisateurs individuels.



Champ	ID_Lecteur	Date	Nombre("Rappels"."Date") * 2	Date_Retour
Alias		NombreRappels	MontantRappels	
Table	Prets	Rappels		Prets
Tri				
Visible	☑	☑	☑	☐
Fonction	Groupe	Nombre		
Critère				IS EMPTY
Ou				

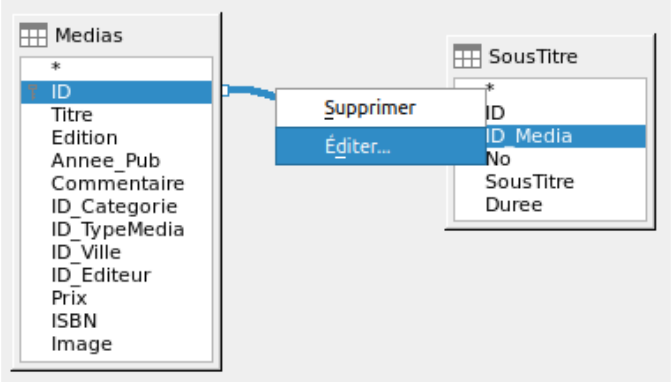
Les champs "Prets"."ID" et "Prets"."ID_Media" ont été supprimés. Ils ont été utilisés dans la requête précédente pour créer, en regroupant, un enregistrement distinct pour chaque support. Maintenant, nous allons regrouper uniquement par lecteur. Le résultat de la requête ressemble à ceci :

Au lieu de lister séparément les supports pour ID_Lecteur = 1, tous les champs "Rappels", "Date" ont été comptés et le total de 4,00 € a été inscrit comme amende due.

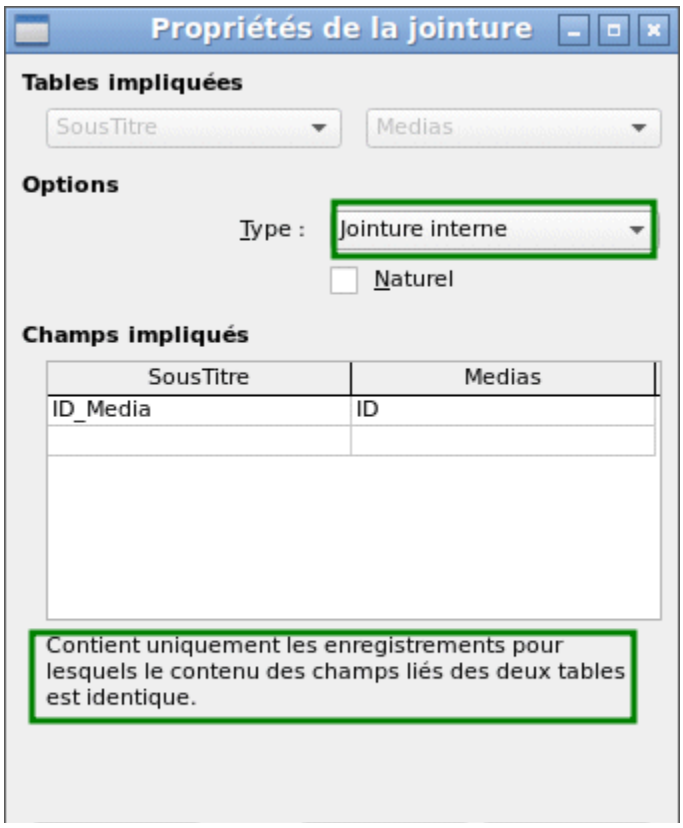
Définition de la relation dans la requête

Lorsque des données sont recherchées dans des tables ou des formulaires, la recherche est généralement limitée à une table ou à un formulaire. Même le chemin d'un formulaire principal vers un sous-formulaire n'est pas navigable par la fonction de recherche intégrée. À ces fins, les données à rechercher sont mieux collectées à l'aide d'une requête.

La requête simple pour le champ Titre de la table Medias affiche les entrées de test pour cette table, 13 enregistrements en tout. Mais si vous entrez SousTitre dans la table de requête, le contenu d'enregistrement de la table Medias est réduit à seulement 2 titres. Seulement pour ces deux titres il y a aussi des sous-titres dans le tableau. Pour tous les autres titres, aucun sous-titre n'existe. Cela correspond à la condition de jointure selon laquelle seuls les enregistrements pour lesquels le champ ID_Media de la table SousTitre est égal au champ ID de la table Medias doivent être affichés. Tous les autres enregistrements sont exclus.



Les conditions de jointure doivent être ouvertes pour être modifiées pour afficher tous les enregistrements souhaités. Nous nous référons ici non pas aux jointures entre les tables dans la conception de relations, mais aux jointures dans les requêtes.



Par défaut, les relations sont définies en tant que jointures internes. La fenêtre fournit des informations sur le fonctionnement de ce type de jointure dans la pratique.

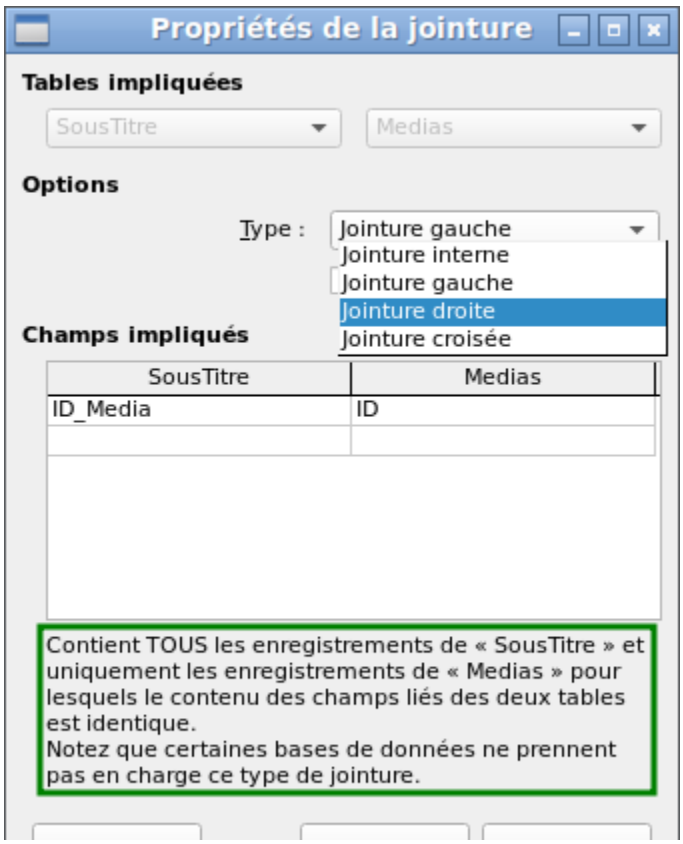
Les deux tables précédemment sélectionnées sont répertoriées comme Tables impliquées. Elles ne sont pas sélectionnables ici. Les champs pertinents des deux tables sont lus à partir des définitions de table. Si aucune relation n'est spécifiée dans la définition de table, une relation peut être créée à ce stade pour la requête. Cependant, si vous avez planifié votre base de données de manière ordonnée à l'aide de HSQLDB, il ne devrait pas être nécessaire de modifier ces champs.

Le paramètre le plus important est l'option Jointure. Ici, les relations peuvent être choisies de manière à ce que tous les enregistrements de la table SousTitres soient sélectionnés, mais uniquement les enregistrements de Médias qui ont un sous-titre entré dans la table SousTitres.

Ou vous pouvez choisir le contraire : que dans tous les cas, tous les enregistrements de la table Medias soient affichés, qu'ils aient ou non un sous-titre.

L'option Naturel spécifie que les champs liés dans les tables sont traités comme égaux. Vous pouvez également éviter d'avoir à utiliser ce paramètre en définissant correctement vos relations au tout début de la planification de votre base de données.

Pour le type Jointure droite, la description montre que tous les enregistrements de la table Medias seront affichés (SousTitre RIGHT JOIN Medias). Comme il n'y a pas de sous-titre qui n'a pas de titre dans Media mais qu'il y a certainement des titres dans Media qui manquent de sous-titre, c'est le choix droit.



Après avoir confirmé la jointure droite, les résultats de la requête semblent tels que nous les voulions. Le titre et le sous-titre sont affichés ensemble dans une seule requête. Naturellement, les titres apparaissent plus d'une fois comme avec la relation précédente. Cependant, tant que les résultats ne sont pas comptés, cette requête peut être utilisée comme base pour une fonction de recherche. Voir les fragments de code dans ce chapitre, au chapitre 8, Tâches de base de données, et au chapitre 9, Macros.

Titre	SousTitre
Bases de données avec OpenOffice.org 3	
Bilbo le Hobbit	
Bon à tirer	Un menuisier dansait
Bon à tirer	Villejuif
Bon à tirer	Pericoloso sporgersi
Bon à tirer	Le Déjeuner de soleil
Bon à tirer	L'Arabe
Émile et les Détectives	
I hear you knocking	Egg Or The Hen
I hear you knocking	Outlaw Blues
I hear you knocking	Sabre dance (Single version)
I hear you knocking	The stumble
I hear you knocking	You can't catch me
Indignez vous !	
La nouvelle orthographe	
La vie mensongère des adultes	
Le livre Postfix	
Les nuits de Reykjavik	

Enregistrement 1 de 21

Définition des propriétés de la requête

Depuis la version 4.1 de LibreOffice, il est possible de définir des propriétés supplémentaires dans l'éditeur de requêtes.

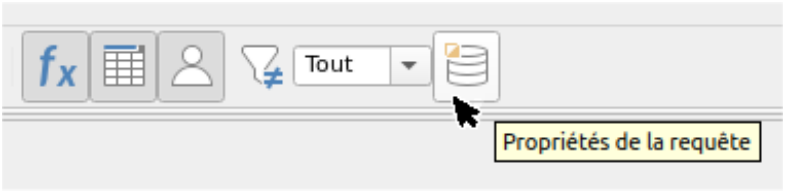
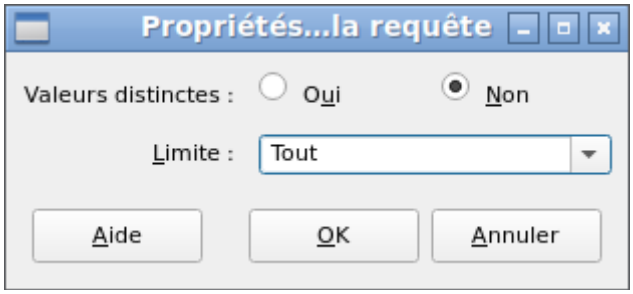


Figure 88: Ouverture des propriétés de requête dans l'éditeur de requête

À côté du bouton pour ouvrir les propriétés de la requête se trouve une zone de liste déroulante pour réguler le nombre d'enregistrements affichés ainsi qu'un bouton pour les valeurs distinctes. Ces fonctions sont répétées dans la boîte de dialogue suivante :



Le paramètre Valeurs distinctes détermine si la requête doit supprimer les enregistrements en double.

	Prenom	Nom	Date_Retour
▶	Guy	Bedos	
	Valérie	Benguigui	
	Valérie	Benguigui	
	Valérie	Benguigui	
	Martine	Carol	
	Marie	Laforêt	
	André	Raimbourg	
	Lino	Ventura	
	Lino	Ventura	
	Lino	Ventura	

Supposons qu'une requête soit faite pour déterminer quels lecteurs ont encore des articles en prêt. Leurs noms sont affichés si le champ de date de retour est vide. Les noms sont affichés plus d'une fois pour les lecteurs qui ont plusieurs articles en prêt.

	Prenom	Nom	Date_Retour
▶	Guy	Bedos	
	Valérie	Benguigui	
	Martine	Carol	
	Marie	Laforêt	
	André	Raimbourg	
	Lino	Ventura	

Si vous choisissez Valeurs distinctes, les enregistrements avec le même contenu disparaissent.

La requête ressemble alors à ceci :

```
SELECT DISTINCT
"Lecteurs"."Prenom", "Lecteurs"."Nom", "Prets"."Date_Retour"
FROM "Prets", "Lecteurs"
WHERE "Prets"."ID_Lecteur" = "Lecteurs"."ID" AND "Prets"."Date_Retour" IS NULL
ORDER BY "Lecteurs"."Nom" ASC
```

La requête d'origine :

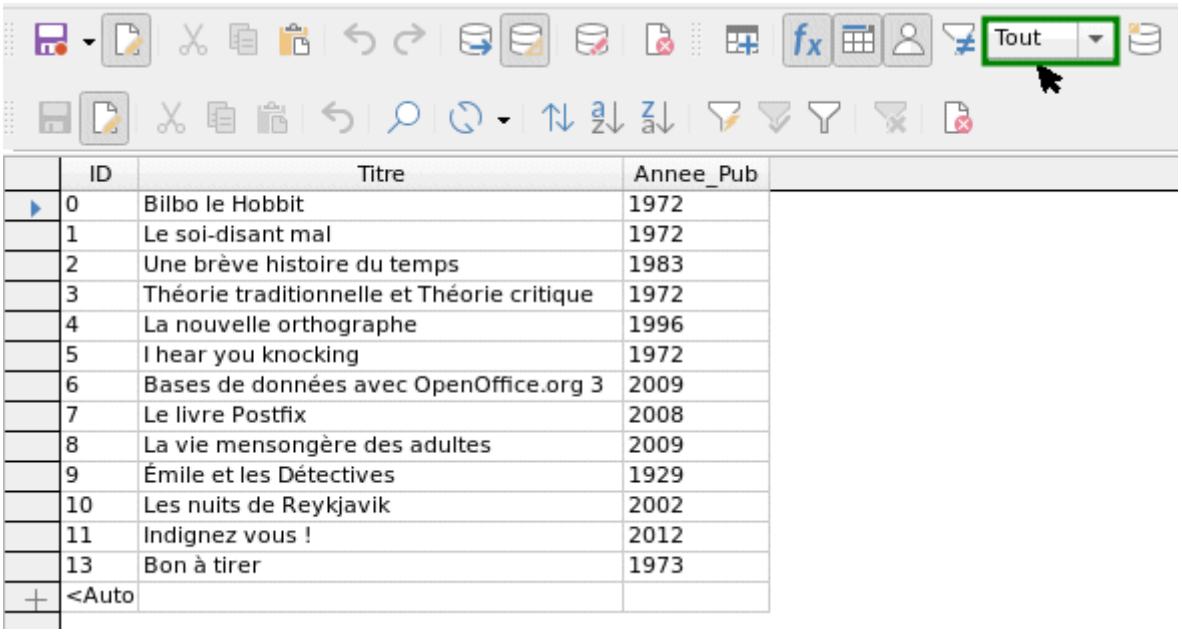
```
SELECT "Lecteurs"."Prenom", "Lecteurs"."Nom"…
```

L'ajout de DISTINCT à la requête supprime les enregistrements en double.

```
SELECT DISTINCT "Lecteurs"."Prenom", "Lecteurs"."Nom"…
```

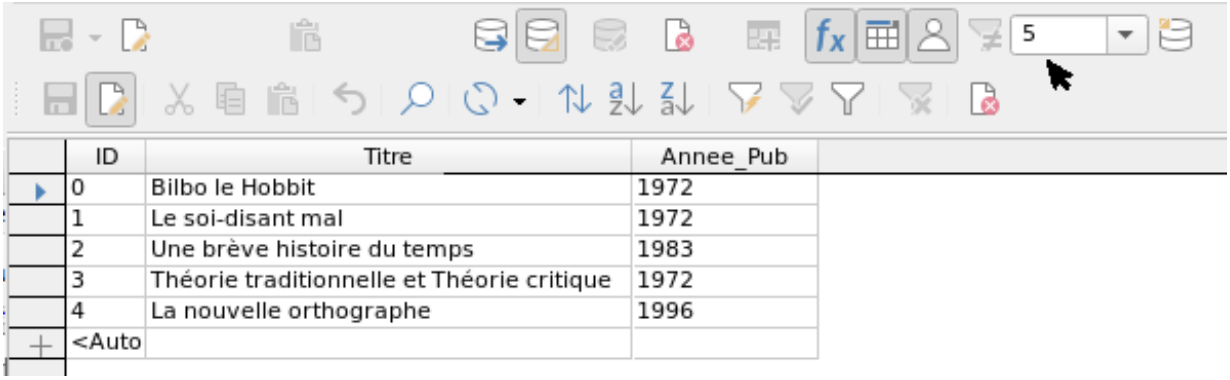
La possibilité de spécifier des enregistrements uniques était également présente dans les versions antérieures. Cependant, il était nécessaire de passer de la vue de conception à la vue SQL pour insérer le qualificatif DISTINCT. Cette propriété est donc rétrocompatible avec les versions précédentes de LO sans poser de problème.

Le paramètre Limite détermine le nombre d'enregistrements à afficher dans la requête. Le nombre d'enregistrements affichés est limité au nombre choisi (5, 10, 20, 50, Tout).



	ID	Titre	Annee_Pub
▶	0	Bilbo le Hobbit	1972
	1	Le soi-disant mal	1972
	2	Une brève histoire du temps	1983
	3	Théorie traditionnelle et Théorie critique	1972
	4	La nouvelle orthographe	1996
	5	I hear you knocking	1972
	6	Bases de données avec OpenOffice.org 3	2009
	7	Le livre Postfix	2008
	8	La vie mensongère des adultes	2009
	9	Émile et les Détectives	1929
	10	Les nuits de Reykjavik	2002
	11	Indignez vous !	2012
	13	Bon à tirer	1973
+	<Auto		

Tous les enregistrements de la table Medias sont affichés. La requête est modifiable car elle inclut la clé primaire.



	ID	Titre	Annee_Pub
▶	0	Bilbo le Hobbit	1972
	1	Le soi-disant mal	1972
	2	Une brève histoire du temps	1983
	3	Théorie traditionnelle et Théorie critique	1972
	4	La nouvelle orthographe	1996
+	<Auto		

Seuls les cinq premiers enregistrements sont affichés (ID 0-4). Un tri n'a pas été demandé, donc l'ordre de tri par défaut par clé primaire est utilisé. Malgré la limitation de la sortie, la requête peut

être modifiée davantage. Cela distingue l'entrée dans l'interface graphique de ce qui, dans les versions précédentes, n'était accessible qu'en utilisant SQL.

```
SELECT "ID", "Titre", "Annee_Pub" FROM "Medias" LIMIT 5
```

La requête d'origine est simplement complétée avec "LIMIT 5". La taille de la limite peut être celle dont vous avez besoin.

Avertissement

La définition de limites dans l'interface graphique n'est pas rétrocompatible. Dans les versions LO antérieures à 4.1, une limite ne pouvait être définie qu'en mode SQL direct. La limite nécessitait alors un tri (`ORDER BY…`) ou une condition (`WHERE…`).

Amélioration des requêtes à l'aide du mode SQL

Si, pendant la saisie graphique, vous utilisez **Affichage> Activer / Désactiver le mode Ébauche**, (ou l'icône correspondante) vous voyez la commande SQL correspondant à ce qui apparaissait précédemment en mode Ébauche. C'est le meilleur moyen pour les débutants d'apprendre le langage de requête standard pour les bases de données. Parfois, c'est aussi le seul moyen de saisir une requête dans la base de données lorsque l'interface graphique ne peut pas traduire vos besoins dans des commandes SQL nécessaires.

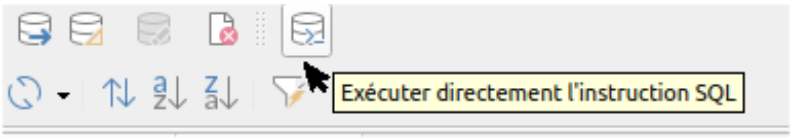
```
SELECT * FROM "Nom_Table"
```

Cela montrera tout ce qui se trouve dans la table Nom_Table. Le caractère"*" représente tous les champs de la table.

```
SELECT * FROM "Nom_Table" WHERE "Nom_Champ" = 'Serge'
```

Ici, il y a une restriction importante. Seuls les enregistrements pour lesquels le champ Nom_Champ contient le mot "Serge" – le terme exact, pas par exemple "Serge Reggiani".

Parfois, les requêtes dans Base ne peuvent pas être effectuées à l'aide de l'interface graphique, car certaines commandes peuvent ne pas être reconnues. Dans de tels cas, il est nécessaire de désactiver le mode Ébauche et d'utiliser la commande **Édition> Exécuter directement l'instruction SQL** pour accéder directement à la base de données. Cette méthode présente l'inconvénient que vous ne pouvez travailler avec la requête qu'en mode SQL.



L'utilisation directe des commandes SQL est également accessible à l'aide de l'interface utilisateur graphique, comme le montre la figure ci-dessus. Cliquez sur l'icône en surbrillance (Exécuter la commande SQL directement) pour désactiver l'icône Ébauche Off / On. Désormais, lorsque vous cliquez sur l'icône Exécuter, la requête exécute directement les commandes SQL.

Voici un exemple des nombreuses possibilités disponibles pour poser des questions à la base de données et spécifier le type de résultat requis :

```
SELECT [{LIMIT <décalage> <limit> | TOP <limit>}][ALL | DISTINCT]
{ <Formule de sélection> | "Nom_Table".* | * } [,…]
[INTO [CACHED | TEMP | TEXT] "Nouvelle_Table"]
```

```
FROM "Liste_Tables"
[WHERE Expression-SQL]
[GROUP BY Expression-SQL [,…]]
[HAVING Expression-SQL]
[{ UNION [ALL | DISTINCT] | {MINUS [DISTINCT] | EXCEPT [DISTINCT] } |
INTERSECT [DISTINCT] } Déclaration de requête]
[ORDER BY Expression-de-Tri [,…]]
[LIMIT <limite> [OFFSET <décalage>]] ;
```

[{LIMIT <décalage> <limite> | TOP <limite>}] :

Cela limite le nombre d'enregistrements à afficher. `LIMIT 10 20` commence au 11e enregistrement et affiche les 20 enregistrements suivants. `TOP 10` affiche toujours les 10 premiers enregistrements. C'est la même chose que `LIMIT 0 10`. LIMIT 10 0 omet les 10 premiers enregistrements et affiche tous les enregistrements à partir du 11e.

Vous pouvez faire la même chose en utilisant la dernière condition SELECT de la formule `[LIMIT <limite> [OFFSET <décalage>]]`. `LIMIT 10` affiche seulement 10 enregistrements. L'ajout de `OFFSET 20` fait que l'affichage commence au 21e enregistrement. Cette forme finale de limitation d'affichage nécessite soit une instruction de tri (`ORDER BY…`) soit une condition (`WHERE…`).

Toutes les valeurs entrées pour la limite doivent être entières. Il n'est pas possible de remplacer une entrée par une sous-requête afin que, par exemple, les cinq derniers enregistrements d'une série puissent être affichés à chaque fois.

[ALL | DISTINCT]

`SELECT ALL` est la valeur par défaut. Tous les enregistrements qui remplissent les conditions de recherche s'affichent. Exemple : `SELECT ALL "Prenom" FROM "Nom_Table"` donne tous les prénoms ; si "Serge" apparaît trois fois et "Valérie" quatre fois dans la table, ces prénoms sont affichés trois et quatre fois respectivement. `SELECT DISTINCT "Prenom" FROM "Nom_Table"` supprime les résultats de requête qui ont le même contenu. Dans ce cas, "Serge" et "Valérie" n'apparaissent qu'une seule fois. `DISTINCT` fait référence à l'ensemble de l'enregistrement auquel la requête accède. Ainsi, si, par exemple, le nom de famille est également demandé, les enregistrements pour "Serge Reggiani" et "Serge Gainsbourg" seront considérés comme distincts. Même si vous spécifiez la condition `DISTINCT`, les deux seront affichés.

<Select-Formulation>

```
{ Expression | COUNT(*) |
{ COUNT | MIN | MAX | SUM | AVG | SOME | EVERY | VAR_POP | VAR_SAMP |
STDDEV_POP | STDDEV_SAMP }
([ALL | DISTINCT]] Expression) } [[AS] "Nom_Affiché"]
```

Les noms de champ, les calculs, les totaux d'enregistrement sont toutes des entrées possibles.

Note

Les calculs au sein d'une base de données conduisent parfois à des résultats inattendus. Supposons qu'une base de données contienne les notes attribuées pour certains travaux en classe et que vous souhaitiez calculer la note moyenne.

Vous devez d'abord additionner toutes les notes. Supposons que la somme soit de 80. Maintenant, cela doit être divisé par le nombre d'élèves (disons 30). La requête donne 2.

Cela se produit car vous travaillez avec des champs de type INTEGER. Le calcul doit donc donner une valeur entière. Vous devez avoir au moins une valeur de type DECIMAL dans votre calcul. Vous pouvez y parvenir soit en utilisant une fonction de conversion HSQLDB ou (plus simplement) vous pouvez diviser par 30,0 plutôt que 30. Cela donnera un résultat avec une décimale (2.6). La division par 30,00 donne deux décimales (2.66). Veillez à utiliser des points décimaux de style anglais. L'utilisation de virgules dans les requêtes est réservée aux séparateurs de champs.

De plus, différentes fonctions sont disponibles pour le champ affiché. À l'exception de COUNT (*) (qui compte tous les enregistrements), aucune de ces fonctions n'accède aux champs NULL (vides).

```
COUNT | MIN | MAX | SUM | AVG | SOME | EVERY | VAR_POP | VAR_SAMP |
STDDEV_POP | STDDEV_SAMP
```

`COUNT("Nom")` compte toutes les entrées pour le champ Nom.
`MIN("Nom")` affiche le premier nom par ordre alphabétique. Le résultat de cette fonction est toujours formaté comme il se présente dans le champ. Le texte est affiché sous forme de texte, les entiers sous forme d'entiers, les décimales sous forme de décimales, etc.
`MAX("Name")` affiche le dernier nom par ordre alphabétique.
`SUM("Nombre")` ne peut ajouter que les valeurs des champs numériques. La fonction échoue pour les champs de date.

Conseil

Cette fonction échoue également sur les champs horaires. Ce qui suit peut être utile :

```
SELECT (SUM(HOUR("Duree"))*3600 + SUM(MINUTE("Duree")))*60 +
SUM(SECOND("Duree")) AS "secondes" FROM "Table"
```

La somme affichée est composée d'heures, de minutes et de secondes distinctes. Ensuite, elle est modifié pour donner le total global en une seule unité, dans ce cas les secondes. Ici, seuls les entiers sont ajoutés par la fonction somme. Maintenant, si vous utilisez :

```
SELECT ((SUM(HOUR("Duree"))*3600 + SUM(MINUTE("Duree")))*60 +
SUM(SECOND("Duree"))) / 3600.0000 AS "heures" FROM "Table"
```

vous obtiendrez une durée en heures avec les minutes et les secondes comme décimales. Avec une mise en forme appropriée, vous pouvez ramener cette valeur à une valeur de temps normale dans une requête ou un formulaire.

`AVG("Nombre")` montre la moyenne du contenu d'une colonne. Cette fonction est également limitée aux champs numériques.
`SOME("Nom_Champ"), EVERY("Nom_Champ")` : Les champs utilisés avec ces fonctions doivent avoir le type de champ Oui/Non [BOOLEAN] (contenant uniquement 0 ou 1). De plus, elles produisent un résumé du contenu du champ auquel ils sont appliqués.

`SOME` renvoie *TRUE* (ou 1) si au moins une entrée pour le champ est 1, et il renvoie *FALSE* (ou 0) uniquement si toutes les entrées sont 0. `EVERY` renvoie 1 uniquement si chaque entrée du champ est 1, et renvoie *FALSE* si au moins une entrée est 0.

Conseil

Le type de champ booléen est Oui/Non [BOOLEAN]. Cependant, ce champ ne contient que 0 ou 1. Dans les conditions de recherche de requête, utilisez TRUE, 1, FALSE ou 0. Pour la condition Oui, vous pouvez utiliser TRUE ou 1. Pour la condition Non, utilisez FALSE ou 0. Si vous essayez d'utiliser "Oui" ou "Non" à la place, vous obtenez un message d'erreur. Ensuite, vous devrez corriger votre erreur

Exemple :

```
SELECT "Classe", EVERY("Nageur")
FROM "Table1"
GROUP BY "Classe";
```

Le champ Classe contient le nom de la classe de natation. Nageur est un champ booléen décrivant si un élève sait nager ou non (1 ou 0). Le champ Etudiant contient les noms des étudiants. La Table1 contient ces champs : sa clé primaire, Classe, Nageur et Etudiant. Seuls les champs Classe et Nageur sont nécessaires pour cette requête.

Étant donné que la requête est regroupée par les entrées du champ Classe, EVERY renverra une valeur pour le champ Nageur pour chaque classe. Lorsque chaque personne dans une classe de natation sait nager, EVERY renvoie TRUE. Sinon EVERY renvoie FALSE car au moins un élève de la classe ne sait pas nager. Puisque la sortie pour le champ Nageur est une case à cocher, une coche indique TRUE, et aucune coche n'indique FALSE.

`VAR_POP | VAR_SAMP | STDDEV_POP | STDDEV_SAMP` sont des fonctions statistiques et n'affectent que les champs entiers et décimaux. Toutes ces fonctions renvoient 0, si les valeurs du groupe sont toutes égales.

Les fonctions statistiques ne permettent pas d'utiliser la limitation DISTINCT. Fondamentalement, ils calculent toutes les valeurs couvertes par la requête, tandis que DISTINCT exclut les enregistrements avec les mêmes valeurs de l'affichage.

`[AS] "Nom_Affiché"`: Les champs peuvent recevoir une désignation différente (alias) dans la requête.

"Nom_Table".* | * [,…]

Chaque champ à afficher est donné avec ses noms de champs, séparés par des virgules. Si des champs de plusieurs tables sont saisis dans la requête, une combinaison du nom du champ avec le nom de la table est nécessaire:: `"Nom_Table"."Nom_Champ"`.

Au lieu d'une liste détaillée de tous les champs d'une table, son contenu total peut être affiché. Pour cela, vous utilisez le symbole "*". Il est alors inutile d'utiliser le nom de la table, si les résultats ne s'appliquent qu'à une seule table. Cependant, si la requête inclut tous les champs d'une table et au moins un champ d'une deuxième table, utilisez: `"Nom_Table 1".*, "Nom_Table 2"."Nom_Champ"`.

[INTO [CACHED | TEMP | TEXT] "nouvelle_table"]

Le résultat de cette requête doit être écrit directement dans une nouvelle table qui est nommée ici. Les propriétés de champ de la nouvelle table sont définies à partir des définitions de champ contenues dans la requête. L'écriture dans une nouvelle table ne fonctionne pas à partir du mode SQL car cela ne gère que les résultats affichables. Au lieu

de cela, vous devez utiliser **Outils > SQL**. La table résultante n'est initialement pas modifiable car elle n'a pas de clé primaire.

FROM <Liste_Tables>

```
"Nom_Table 1" [{CROSS | INNER | LEFT OUTER | RIGHT OUTER} JOIN "Nom_Table
2" ON Expression] [,…]
```

Les tables qui doivent être recherchées conjointement sont généralement dans une liste séparée par des virgules. La relation entre les tables est alors définie en plus par le mot-clé `WHERE`.

Si les tables sont liées par une jointure `JOIN` plutôt que par une virgule, leur relation est définie par le terme commençant par `ON` qui apparaît directement après la deuxième table.

Un simple `JOIN` a pour effet que seuls les enregistrements pour lesquels les conditions des deux tables s'appliquent sont affichés..

Exemple :

```
SELECT "Table1"."Nom", "Table2"."Classe"
FROM "Table1", "Table2"
WHERE "Table1"."ID_Classe" = "Table2"."ID"
```

est equivalent à :

```
SELECT "Table1"."Nom", "Table2"."Classe"
FROM "Table1"
      JOIN "Table2"
ON "Table1"."ID_Classe" = "Table2"."ID"
```

Ici, les noms et les classes correspondantes sont affichés. Si un nom (étudiant) n'a aucune classe répertoriée pour lui, ce nom n'est pas inclus dans l'affichage. Si une classe n'a pas de nom (étudiant), elle n'est pas non plus affichée. L'ajout de `INNER` ne change rien à cela.

```
SELECT "Table1"."Nom", "Table2"."Classe"
FROM "Table1"
      LEFT JOIN "Table2"
ON "Table1"."ID_Classe" = "Table2"."ID"
```

Si `LEFT` est ajouté, tous les noms de Table1 sont affichés même s'ils n'ont pas de classe. Si, au contraire, `RIGHT` est ajouté, toutes les classes sont affichées même si elles ne contiennent aucun nom. L'ajout de `OUTER` n'a pas besoin d'être montré ici. (Right Outer Join est la même chose que Right Join ; Left Outer Join est la même chose que Left Join.)

```
SELECT "Table1"."Joueur1", "Table2"."Joueur2"
FROM "Table1" AS "Table1"
      CROSS JOIN "Table2" AS "Table1"
WHERE "Table1"."Joueur1" <> "Table2"."Joueur2"
```

Un `CROSS JOIN` nécessite que la table soit fournie avec un alias, mais l'ajout du terme `ON` n'est pas toujours nécessaire. Tous les enregistrements de la première table sont associés à tous les enregistrements de la seconde table. Ainsi, la requête ci-dessus donne tous les appariements possibles d'enregistrements de la première table avec ceux de la deuxième table à l'exception des appariements entre enregistrements pour le même joueur. Dans le cas d'un `CROSS JOIN`, la condition ne doit pas inclure de lien entre les tables spécifiées dans le terme `ON`. Au lieu de cela, la condition `WHERE` peur être entrée. Si les conditions sont

formulées exactement comme dans le cas d'un simple JOIN, vous obtenez le même résultat :

```
SELECT "Table1"."Nom", "Table2"."Classe"
FROM "Table1"
      JOIN "Table2"
ON "Table1"."ID_Classe" = "Table2"."ID"
```

donne le même résultat que

```
SELECT "Table1"."Nom", "Table2"."Classe"
FROM "Table1" AS "Table1"
      CROSS JOIN "Table2" AS "Table2"
WHERE "Table1"."ID_Classe" = "Table2"."ID"
```

[WHERE Expression-SQL]

L'introduction standard des conditions pour demander un filtrage plus précis des données. Ici aussi, les relations entre les tables sont généralement définies si elles ne sont pas liées entre elles par JOIN.

[GROUP BY Expression-SQL [,…]]

Utilisez cette option lorsque vous souhaitez diviser les données de la requête en groupes avant d'appliquer les fonctions à chacun des groupes séparément. La division est basée sur les valeurs du champ ou des champs contenus dans l'expression `GROUP BY`.

Exemple :

```
SELECT "Nom", SUM("Entree"-"Sortie") AS "Balance"
FROM "Table1"
GROUP BY "Nom";
```

Les enregistrements ayant `le même` contenu dans le champ nom sont additionnés. Dans le résultat de la requête, la somme des entrées – sorties est donnée pour chaque personne. Ce champ doit être appelé Balance. Chaque ligne du résultat de la requête contient une valeur du champ Nom et le solde calculé pour cette valeur spécifique.

Conseil

Lorsque les champs sont traités à l'aide d'une fonction particulière (par exemple `COUNT`, `SUM`…), tous les champs qui ne sont pas traités avec une fonction mais qui doivent être affichés sont regroupés à l'aide de `GROUP BY`.

[HAVING Expression-SQL]

La formule `HAVING` ressemble étroitement à la formule `WHERE`. La différence est que la formule WHERE s'applique aux valeurs des champs sélectionnés dans la requête. La formule HAVING s'applique aux valeurs calculées sélectionnées. Plus précisément, la formule WHERE ne peut pas utiliser une fonction d'agrégation dans le cadre d'une condition de recherche ; la formule HAVING le fait.

La formule HAVING a deux objectifs comme le montrent les deux exemples ci-dessous. Dans le premier, la condition de recherche exige que le temps d'exécution minimum soit inférieur à 40 minutes. Dans le deuxième exemple, la condition de recherche exige que le solde d'un individu soit positif.

Les résultats de la requête pour la première liste les noms des personnes dont le temps d'exécution a été inférieur à 40 minutes au moins une fois et le temps d'exécution minimum. Les personnes dont les temps d'exécution sont tous supérieurs à 40 minutes ne sont pas répertoriées.

Les résultats de la requête pour la seconde liste les noms des personnes qui ont un résultat total supérieur à l'entrée et leur solde. Les personnes dont le solde est égal ou inférieur à 0 ne sont pas répertoriées.

```
Exemples :
SELECT "Nom", "Duree"
FROM "Table1"
GROUP BY "Nom", "Duree"
HAVING MIN("Duree") < '00:40:00';

SELECT "Nom", SUM("Entree"-"Sortie") AS "Balance"
FROM "Table1"
GROUP BY "Nom"
HAVING SUM("Entree"-"Sortie") > 0 ;
```

[Expression-SQL]

Les expressions SQL sont combinées selon le schéma suivant :

```
[NOT] condition [{ OR | AND } condition]
```

Exemple :

```
SELECT *
FROM "Nom_Table"
WHERE NOT "Date_Retour" IS NULL AND "ID_Lecteur" = 2 ;
```

Les enregistrements lus à partir de la table sont ceux pour lesquels une Date_Retour a été entré et le ID-Lecteur est 2. En pratique, cela signifie que tous les supports prêtés à une personne spécifique et retournés peuvent être récupérés. Les conditions ne sont liées qu'à AND. Le NOT se réfère uniquement à la première condition.

```
SELECT *
FROM "Nom_Table"
WHERE NOT ("Date_Retour" IS NULL AND "ID_Lecteur" = 2) ;
```

Les parenthèses autour de la condition, avec NOT à l'extérieur, montrent uniquement les enregistrements qui ne remplissent pas complètement la condition entre parenthèses. Cela couvrirait tous les enregistrements, à l'exception de ceux du ID_Lecteur numéro 2, qui n'ont pas encore été renvoyés.

[Expression-SQL] : conditions

```
{ valeur [|| valeur]}
```

Une valeur peut être une ou plusieurs valeurs concaténées par deux lignes verticales ||. Naturellement, cela s'applique également au contenu des champs.

```
SELECT "NomFamille" || ', ' || "Prenom" AS "Identite"
FROM "Nom_Table"
```

Le contenu des champs NomFamille et Prenom est affiché ensemble dans un champ appelé Identite. Notez qu'une virgule et un espace sont insérés entre le nom et le prénom.

`| valeur { = | < | <= | > | >= | <> |!= } valeur`

Ces signes correspondent aux opérateurs mathématiques bien connus:
`{Égal à | Inférieur à | Inférieur ou égal à | Supérieur à | Supérieur ou égal à | Différent de | Différent de}`

`| valeur IS [NOT] NULL`

Le champ correspondant n'a pas de contenu, car rien n'y a été écrit. Cela ne peut pas être déterminé sans ambiguïté dans l'interface graphique, car un champ de texte visuellement vide ne signifie pas que le champ est complètement sans contenu. Cependant, la configuration par défaut dans Base est que les champs vides de la base de données sont définis sur NULL.

`| EXISTS(Query_result)`

Exemple :

```
SELECT "Nom"
FROM "Table1"
WHERE EXISTS
      (SELECT "Prenom"
      FROM "Table2"
      WHERE "Table2"."Prenom" = "Table1"."Nom")
```

Les noms de Table1 sont affichés pour lesquels les prénoms sont donnés dans Table2.

`| Valeur BETWEEN Valeur AND Valeur`

`BETWEEN valeur1 AND valeur2` renvoie toutes les valeurs comprises entre valeur1 et valeur2 (incluse). Si les valeurs sont des lettres, un tri alphabétique est utilisé dans lequel les lettres minuscules ont la même valeur que les majuscules correspondantes.

```
SELECT "Nom"
FROM "Nom_Table"
WHERE "Nom" BETWEEN 'A' AND 'E';
```

Cette requête renvoie tous les noms commençant par A, B, C ou D (ainsi que par les lettres minuscules correspondantes). Comme E est défini comme limite supérieure, les noms commençant par E ne sont pas inclus. La lettre E seule apparaît juste *avant* les noms commençant par E.

`| valeur [NOT] IN ({valeur [,…] | Query resultat })`

Cela nécessite soit une liste de valeurs, soit une requête. La condition est remplie si la valeur est incluse dans la liste de valeurs ou dans le résultat de la requête.

`| valeur [NOT] LIKE valeur [ESCAPE] valeur }`

L'opérateur LIKE est nécessaire dans de nombreuses fonctions de recherche simples. La valeur est saisie selon le modèle suivant :

'%' représente n'importe quel nombre de caractères (y compris 0), "_" remplace exactement un caractère.

Pour rechercher "%" ou "_" lui-même, les caractères doivent immédiatement suivre un autre caractère "\" défini comme ESCAPE.

```
SELECT "Nom"
FROM "Nom_Table"
WHERE "Nom" LIKE '\_%' ESCAPE '\'
```

Cette requête affiche tous les noms commençant par un trait de soulignement. «\» est défini ici comme le caractère ESCAPE.

[Expression-SQL] : valeurs

```
[+ | -] { Expression [{ + | – | * | / | || } Expression]}
```

Les valeurs peuvent être précédées d'un signe. L'addition, la soustraction, la multiplication, la division et la concaténation d'expressions sont autorisées. Un exemple de concaténation :

```
SELECT "NomFamille"||', '||"Prenom"
FROM "Table"
```

De cette façon, les enregistrements sont affichés par la requête avec un champ contenant "Nom, Prénom". L'opérateur de concaténation peut être qualifié par les expressions suivantes.

```
| (Condition)
```

Pour cela consultez la section précédente [Expression-SQL] : conditions].

```
| Fonction ([Paramètre] [,…])
```

Voir la section Fonctions en annexe.

Les requêtes suivantes sont également appelées sous-requêtes (sous-sélections).

```
| Résultat de la requête qui donne exactement une réponse
```

Comme un enregistrement ne peut avoir qu'une seule valeur dans chaque champ, seule une requête qui produit précisément une valeur peut être affichée dans son intégralité.

```
| {ANY|ALL} (Résultat de la requête qui donne exactement une réponse à
partir d'une colonne entière)
```

Il y a souvent une condition qui compare une expression à tout un groupe de valeurs.

Combinée avec `ANY` cela signifie que l'expression doit se produire au moins une fois dans le groupe. Cela peut également être spécifié en utilisant la condition `IN`. `= ANY` donne le même résultat que `IN`.

Combinée avec `ALL` cela signifie que toutes les valeurs du groupe doivent correspondre à une seule expression.

[Expression – SQ] : Expression

```
{ 'Texte' | Nombre entier | Nombre à virgules flottante
| ["Table".]"Champ" | TRUE | FALSE | NULL }
```

Fondamentalement, les valeurs servent d'arguments pour diverses expressions, en fonction du format source. Pour rechercher le contenu des champs de texte, placez le contenu entre guillemets. Les nombres entiers sont écrits sans guillemets, tout comme les nombres à virgule flottante.

Les champs représentent les valeurs qui apparaissent dans ces champs de la table. Les champs sont généralement comparés entre eux ou avec des valeurs spécifiques. En SQL, les noms de champs doivent être placés entre guillemets, car ils pourraient ne pas être correctement reconnus autrement. En général, SQL suppose que le texte sans guillemets doubles est sans caractères spéciaux, c'est-à-dire un seul mot sans espaces et en majuscules. Si plusieurs tables sont contenues dans la requête, le nom de la table doit être donné en plus du nom du champ, séparé de ce dernier par un point.
TRUE et FALSE dérivent généralement des champs Oui/Non.
NULL signifie pas de contenu. Ce n'est pas la même chose que 0 mais correspond plutôt à "vide".

UNION [ALL | DISTINCT] Résultat_Requête

Cela lie les requêtes afin que le contenu de la deuxième requête soit écrit sous la première. Pour que cela fonctionne, tous les types des champs des deux requêtes doivent correspondre. Cette liaison de plusieurs requêtes fonctionne uniquement en mode commande SQL directe.

```
SELECT "Prenom"
FROM "Table1"
UNION DISTINCT
      SELECT "Prenom"
      FROM "Table2";
```

Cette requête renvoie tous les prénoms de table1 et table2 ; le terme supplémentaire DISTINCT signifie qu'aucun prénom en double ne sera affiché. DISTINCT est la valeur par défaut dans ce contexte. Par défaut, les prénoms sont triés par ordre alphabétique croissant. ALL provoque l'affichage de tous les prénoms dans Table1, suivis du prénom dans Table2. Dans ce cas, la valeur de tri par défaut est par la clé primaire.

L'utilisation de cette technique de requête permet de lister les valeurs d'un enregistrement directement les unes sous les autres dans une colonne. Supposons que vous ayez une table appelée Stock dans laquelle il y a des champs pour Prix_Vente, Prix_Rabais_1 et Prix_Rabais_2. À partir de là, vous souhaitez calculer un champ de combinaison qui listera ces prix directement les uns sous les autres.

```
SELECT
      "Prix_Vente"
FROM "Stock" WHERE "ID_Stock" = 1
      UNION
      SELECT
            "Prix_Rabais_1"
      FROM "Stock" WHERE "ID_Stock" = 1
      UNION
      SELECT
            "Prix_Rabais_2"
      FROM "Stock" WHERE "ID_Stock" = 1 ;
```

La clé primaire de la table Stock doit naturellement être définie à l'aide d'une macro, car le champ de combinaison aura une entrée correspondante.

MINUS [DISTINCT] | EXCEPT [DISTINCT] Résultat_Requête

```
SELECT "Prenom"
FROM "Table1"
EXCEPT
      SELECT "Prenom"
      FROM "Table2";
```

Affiche tous les prénoms de la table 1 à l'exception des prénoms contenus dans la table 2. `MINUS` et `EXCEPT` conduisent au même résultat. Le tri est alphabétique.

INTERSECT [DISTINCT] Query_result

```
SELECT "Prenom"
FROM "Table1"
INTERSECT
      SELECT "Prenom"
      FROM "Table2";
```

Cela affiche les prénoms qui apparaissent dans les deux tables. Le tri est à nouveau alphabétique. À l'heure actuelle, cela ne fonctionne qu'en mode de commande SQL directe.

[ORDER BY Expression-Tri[,…]]

L'expression peut être un nom de champ, un numéro de colonne (commençant par 1 à gauche), un alias (formulé avec AS par exemple) ou une expression de valeur composite (voir [Expression SQL] : valeurs). L'ordre de tri est généralement croissant (ASC). Si vous souhaitez un tri décroissant, vous devez spécifier DESC explicitement.

```
SELECT "Prenom", "NomFamille" AS "Nom"
FROM "Table1"
ORDER BY "NomFamille";
```

est identique à

```
SELECT "Prenom", "NomFamille" AS "Nom"
FROM "Table1"
ORDER BY 2 ;
```

est identique à

```
SELECT "Prenom", "NomFamille" AS "Nom"
FROM "Table1"
ORDER BY "Nom";
```

Utilisation d'un alias dans une requête

Les requêtes peuvent reproduire des champs avec des noms modifiés.

```
SELECT "Prenom", "NomFamille" AS "Nom"
FROM "Table1"
```

Le champ *NomFamille* s'appelle *Nom* dans l'affichage.

Si une requête implique deux tables, chaque nom de champ doit être précédé du nom de la table :

```
SELECT "Table1"."Prenom", "Table1"."NomFamille" AS "Nom", "Table2"."Classe"
FROM "Table1", "Table2"
WHERE "Table1"."ID_Classe" = "Table2"."ID"
```

Le nom de la table peut également recevoir un alias, mais celui-ci ne sera pas reproduit en vue table. Si un tel alias est défini, tous les noms de table de la requête doivent être modifiés en conséquence :

```
SELECT "a"."Prenom", "a"."NomFamille" AS "Nom", "b"."Classe"
FROM "Table1" AS "a", "Table2" AS "b"
WHERE "a"."ID_Classe" = "b"."ID"
```

L'affectation d'un alias pour une table peut être effectuée plus brièvement sans utiliser le terme AS :

```
SELECT "a"."Prenom", "a"."NomFamille" "Nom", "b"."Classe"
FROM "Table1" "a", "Table2" "b"
WHERE "a"."ID_Class" = "b"."ID"
```

Cela rend cependant le code moins lisible. Pour cette raison, la forme abrégée ne doit être utilisée que dans des circonstances exceptionnelles.

Note

Malheureusement, le code de l'éditeur de requête dans l'interface utilisateur graphique a été récemment modifié afin que les désignations d'alias soient créées sans utiliser le préfixe AS. En effet, lorsque des bases de données externes sont utilisées, dont la méthode de spécification des alias ne peut être prédite, l'inclusion d'AS a conduit à des messages d'erreur.

Si le code de requête est ouvert pour modification, pas directement dans SQL mais à l'aide de l'interface graphique, les alias existants perdent leur préfixe AS. Si cela est important pour vous, vous devez toujours **modifier les requêtes dans la vue SQL**.

Un nom d'alias permet également d'utiliser une table avec le filtrage correspondant plus d'une fois dans une requête :

```
SELECT "PaiementCompte"."Balance", "Compte"."Date",
       "a"."Balance" AS "Actuel",
       "b"."Balance" AS "Prevu"
FROM "Compte"
       LEFT JOIN "Compte" AS "a"
       ON "Compte"."ID" = "a"."ID" AND "a"."Balance" >= 0
       LEFT JOIN "Compte" AS "b"
       ON "Compte"."ID" = "b"."ID" AND "b"."Balance" < 0
```

Requêtes pour la création de champs de list box

Les champs de zone de liste affichent une valeur qui ne correspond pas au contenu de la table sous-jacente. Ils sont utilisés pour afficher la valeur attribuée par un utilisateur à une clé étrangère plutôt que la clé elle-même. La valeur qui est finalement enregistrée dans le formulaire ne doit pas apparaître dans la première position du champ de zone de liste.

```
SELECT "Prenom", "ID"
FROM "Table1";
```

Cette requête afficherait tous les prénoms et les valeurs de clé primaire "ID" fournis par la table sous-jacente du formulaire. Bien sûr, ce n'est pas encore optimal. Les prénoms n'apparaissent pas triés et, dans le cas de prénoms identiques, il est impossible de déterminer de quelle personne il s'agit.

```
SELECT "Prenom"||' '||"Nom", "ID"
FROM "Table1"
ORDER BY "Prenom"||' '||"Nom";
```

Maintenant, le prénom et le nom apparaissent tous les deux, séparés par un espace. Les noms se distinguent et sont également triés. Mais le tri suit la logique habituelle de commencer par la première lettre de la chaîne, donc il trie par prénom et seulement ensuite par Nom. Un ordre de tri différent de celui dans lequel les champs sont affichés serait certainement déroutant.

```
SELECT "Nom"||', '||"Prenom", "ID"
FROM "Table1"
ORDER BY "Nom"||', '||"Prenom";
```

Cela conduit désormais à un tri qui correspond mieux à la coutume normale. Les membres de la famille apparaissent ensemble, les uns sous les autres ; cependant différentes familles avec le

même nom de famille seraient imbriquées. Pour les distinguer, nous aurions besoin de les regrouper différemment dans le tableau.

Il y a un dernier problème : si deux personnes ont le même nom de famille et le même prénom, elles ne seront toujours pas distinguables. Une solution pourrait être d'utiliser un suffixe de nom. Mais imaginez à quoi cela ressemblerait si une civilité se lisait M. "Dupond II"!

```
SELECT "Nom"||', '||"Prenom"||' – ID:'||"ID", "ID"
FROM "Table1"
ORDER BY "Nom"||', '||"Prenom"||"ID";
```

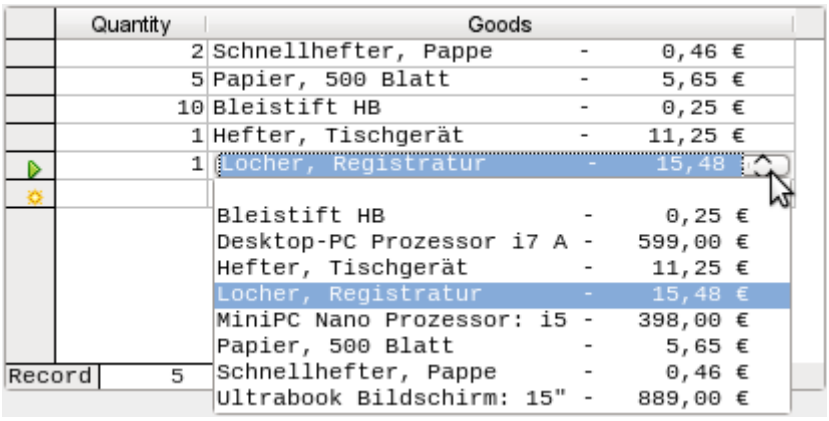
Ici, tous les enregistrements sont apparents. Ce qui est réellement affiché est "Nom, Prenom – ID : valeur ID".

Dans le formulaire de prêt, il y a une zone de liste qui montre uniquement les médias qui n'ont pas encore été prêtés. Il est créé à l'aide de la formule SQL suivante :

```
SELECT "Titre" || ' – No. ' || "ID", "ID"
FROM "Medias"
WHERE "ID" NOT IN
      (SELECT "ID_Media"
      FROM "Prets"
      WHERE "Date_Retour" IS NULL)
ORDER BY "Titre" || ' – No. ' || "ID" ASC
```

Il est important que ce champ de liste soit toujours mis à jour si un média qu'il contient est prêté.

Le champ de liste suivant montre le contenu de plusieurs champs sous forme de tableau afin que les éléments qui appartiennent au même ensemble soient directement les uns sous les autres.



Pour qu'une telle représentation fonctionne, vous devez d'abord choisir une police de largeur fixe appropriée. Courier ou n'importe quelle mono-police telle que Liberation Mono peut être utilisée ici. Vous créez le formulaire tabulaire à l'aide du code SQL :

```
SELECT
      LEFT("Stock"||SPACE(25),25) || ' – ' ||
            RIGHT(SPACE(8)||"Prix",8) || ' €',
      "ID"
FROM "Stock"
ORDER BY ("Stock" || ' – ' || "Prix" || ' $') ASC
```

Le contenu du champ Stock a été rempli d'espaces afin que la chaîne entière ait une longueur minimale de 25 caractères. Ensuite, les 25 premières lettres y sont placées et le surplus est coupé.

Cela devient plus compliqué lorsque le contenu du champ de liste contient des caractères non imprimables comme des retours à la ligne. Ensuite, le code doit être personnalisé pour s'adapter :

```
SELECT
      LEFT(REPLACE("Stock", CHAR(10),' ')||SPACE(25), 25) || ' – ' ||…
```

Cela remplace une fin de ligne sous Linux par un espace. Sous Windows, vous devez en outre supprimer le retour chariot (CHAR (13)).

Le nombre d'espaces nécessaires peut également être déterminé sur une base par requête. Cela évite que la valeur de « Stock » ne soit accidentellement tronquée.

```
SELECT
      LEFT("Stock"||SPACE((SELECT MAX(LENGTH("Stock")) FROM "Stock")),
            (SELECT MAX(LENGTH("Stock")) FROM "Stock"))|| ' – ' ||
            RIGHT(' '||"Prix",8) || ' €',
      "ID"
FROM "Stock"
ORDER BY ("Stock" || ' – ' || "Prix" || ' $') ASC
```

Comme le prix doit être affiché justifié à droite, il est rempli d'espaces à gauche et placé au maximum huit caractères à partir de la droite. La représentation choisie fonctionnera pour tous les prix jusqu'à 99999,99 €.

Si vous souhaitez remplacer le point décimal par une virgule dans SQL, vous aurez besoin d'un code supplémentaire :

```
REPLACE(RIGHT(' '||"Prix",8),'.',',')
```

Requêtes comme base d'informations supplémentaires dans les formulaires

Si vous souhaitez qu'un formulaire affiche des informations supplémentaires qui autrement ne seraient pas visibles, il existe différentes possibilités de requête. Le plus simple est de récupérer ces informations avec des requêtes indépendantes et d'insérer les résultats dans le formulaire. L'inconvénient de cette méthode est que les modifications des enregistrements peuvent affecter le résultat de la requête, mais malheureusement ces modifications ne sont pas affichées automatiquement.

Voici un exemple dans le domaine du contrôle des stocks pour une simple vérification.

La table de vérification contient les totaux et les clés étrangères pour les articles en stock, ainsi qu'un numéro de reçu. L'acheteur a très peu d'informations s'il n'y a pas de résultat de requête supplémentaire imprimé sur le reçu. Après tout, les articles ne sont identifiés que par la lecture d'un code-barres. Sans requête, le formulaire affiche uniquement :

Total	*CodeBarre*
3	17
2	24

Ce qui est caché derrière les nombres ne peut pas être rendu visible à l'aide d'une zone de liste, car la clé étrangère est saisie directement à l'aide du code-barres. De la même manière, il est impossible d'utiliser une zone de liste à côté de l'article pour afficher au moins le prix unitaire.

Ici, une requête peut vous aider.

```
SELECT "Verif"."ID_recu", "Verif"."Total", "Stock"."Element",
"Stock"."Prix_Unitaire", "Verif"."Total"*"Stock"."Prix_Unitaire" AS
"Prix_Total"
FROM "Verif", "Stock"
WHERE "Stock"."ID" = "Verif"."ID_Element";
```

Maintenant, au moins après la saisie des informations, nous savons combien il faut payer pour 3 * Item'17 '. De plus, seules les informations relatives au ID_recu correspondant doivent être filtrées dans le formulaire. Ce qui manque encore, c'est ce que le client doit payer globalement.

```
SELECT "Verif"."ID_Recu", SUM("Verif"."Total"*"Stock"."Prix_Unitaire") AS
"Sum"
FROM "Verif", "Stock"
WHERE "Stock"."ID" = "Verif"."ID_Element"
GROUP BY "Verif"."ID_Recu";
```

Concevez le formulaire pour afficher un enregistrement de la requête à la fois. Étant donné que la requête est regroupée par ID_Recu, le formulaire affiche des informations sur un client à la fois.

Conseil

Si un formulaire doit afficher des valeurs de date qui dépendent d'une autre date (par exemple, une période de prêt pour un support peut être de 21 jours, quelle est la date de retour?), Vous ne pouvez pas utiliser les fonctions intégrées de HSQLDB. Il n'y a pas de fonction « DATEADD ».

La requête

```
SELECT "Date", DATEDIFF('dd','1899-12-30',"Date")+21 AS "Date_Retour" FROM
"Table"
```

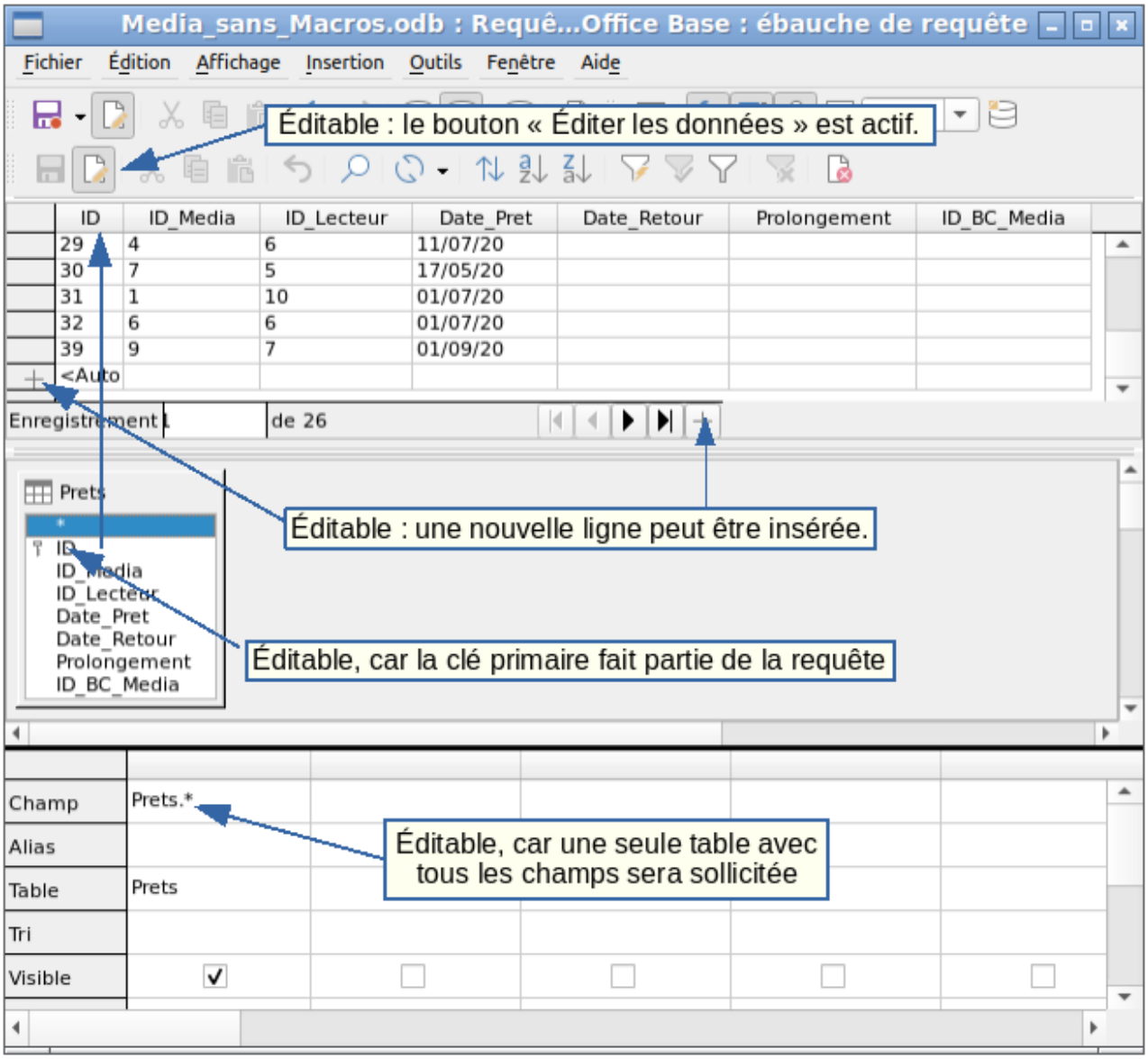
donnera la date cible correcte pour le retour dans un formulaire. Cette requête compte les jours à partir du 30.12.1899. Il s'agit de la date par défaut, que Calc utilise également comme valeur zéro.

Cependant, la valeur renvoyée n'est qu'un nombre et non une date qui peut être utilisée dans une autre requête.

Le nombre renvoyé ne convient pas pour une utilisation dans les requêtes car la mise en forme des requêtes n'est pas enregistrée. Vous devez plutôt créer une vue.

Possibilités de saisie de données dans les requêtes

Pour créer des saisies dans une requête, la clé primaire de la table sous-jacente à la requête doit être présente. Cela s'applique également à une requête qui relie plusieurs tables entre elles.

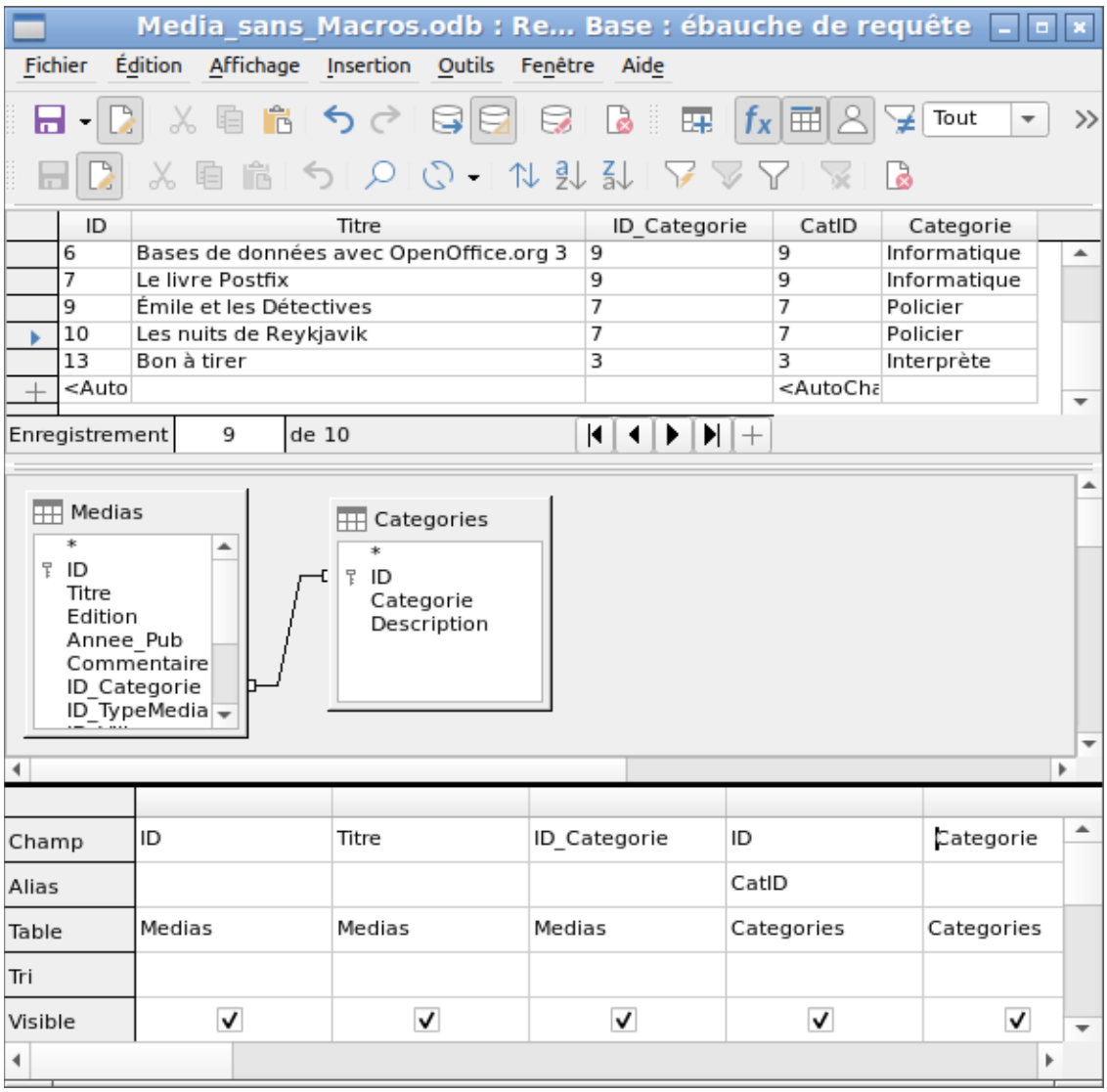


Lors du prêt de supports, cela n'a aucun sens d'afficher pour un lecteur des éléments qui ont déjà été retournés il y a quelque temps.

```
SELECT "ID", "ID_Lecteur", "ID_Media", "Date_Pret"
FROM "Prets"
WHERE "Date_Retour" IS NULL ;
```

De cette manière, un formulaire peut afficher dans un champ de contrôle de table tout ce qu'un lecteur particulier a emprunté au fil du temps. Ici aussi, la requête doit filtrer en utilisant la structure de formulaire appropriée (lecteur dans le formulaire principal, requête dans le sous-formulaire), de sorte que seuls les médias réellement prêtés soient affichés (et non les prêts anciens revenus). La requête convient à la saisie de données car la clé primaire est incluse dans la requête.

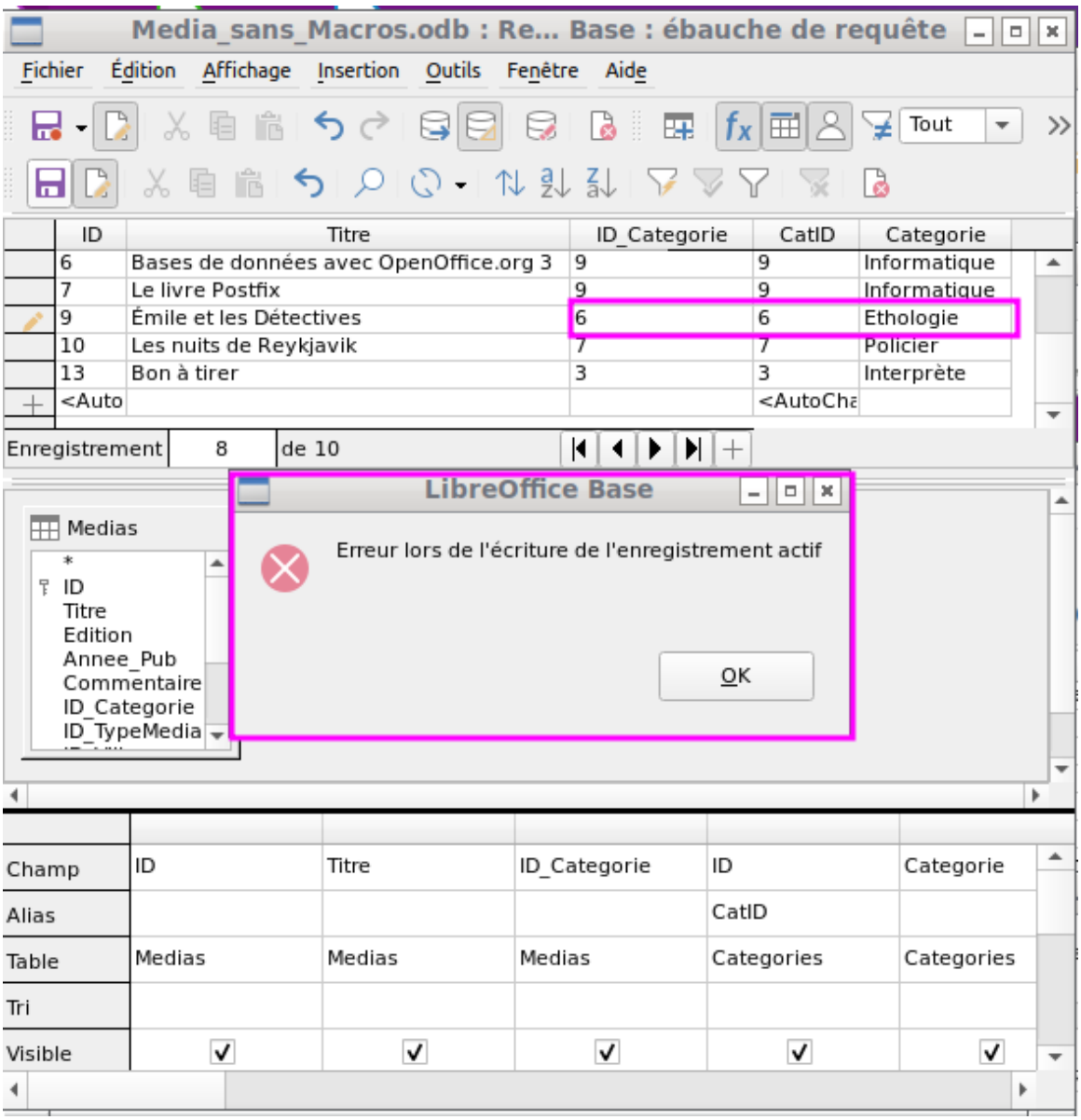
La requête cesse d'être modifiable si elle se compose de plus d'une table et que les tables sont accessibles via un alias. Cela ne fait aucune différence dans ce cas que la clé primaire soit présente ou non dans la requête.




```
SELECT "Medias"."ID", "Medias"."Titre", "Medias"."ID_Categorie",
"Categorie"."ID" AS "CatID", "Categorie"."Categorie"
FROM "Medias", "Categorie"
WHERE "Medias"."ID_Categorie" = "Categorie"."ID";
```

Cette requête est modifiable, car les deux clés primaires sont incluses et sont accessibles dans les tables sans utiliser d'alias. Une requête qui lie plusieurs tables ensemble nécessite également que toutes les clés primaires soient présentes.

Dans une requête qui implique plusieurs tables, il n'est pas possible de modifier un champ de clé étrangère dans une table qui fait référence à un enregistrement dans une autre table. Dans l'enregistrement ci-dessous, une tentative a été faite pour changer la catégorie du titre « Emilie et les détectives ». Le champ ID_Categorie est passé de 7 à 6. Le changement a semblé avoir lieu et la nouvelle catégorie est apparue dans l'enregistrement. Cependant, il s'est avéré impossible de l'enregistrer.

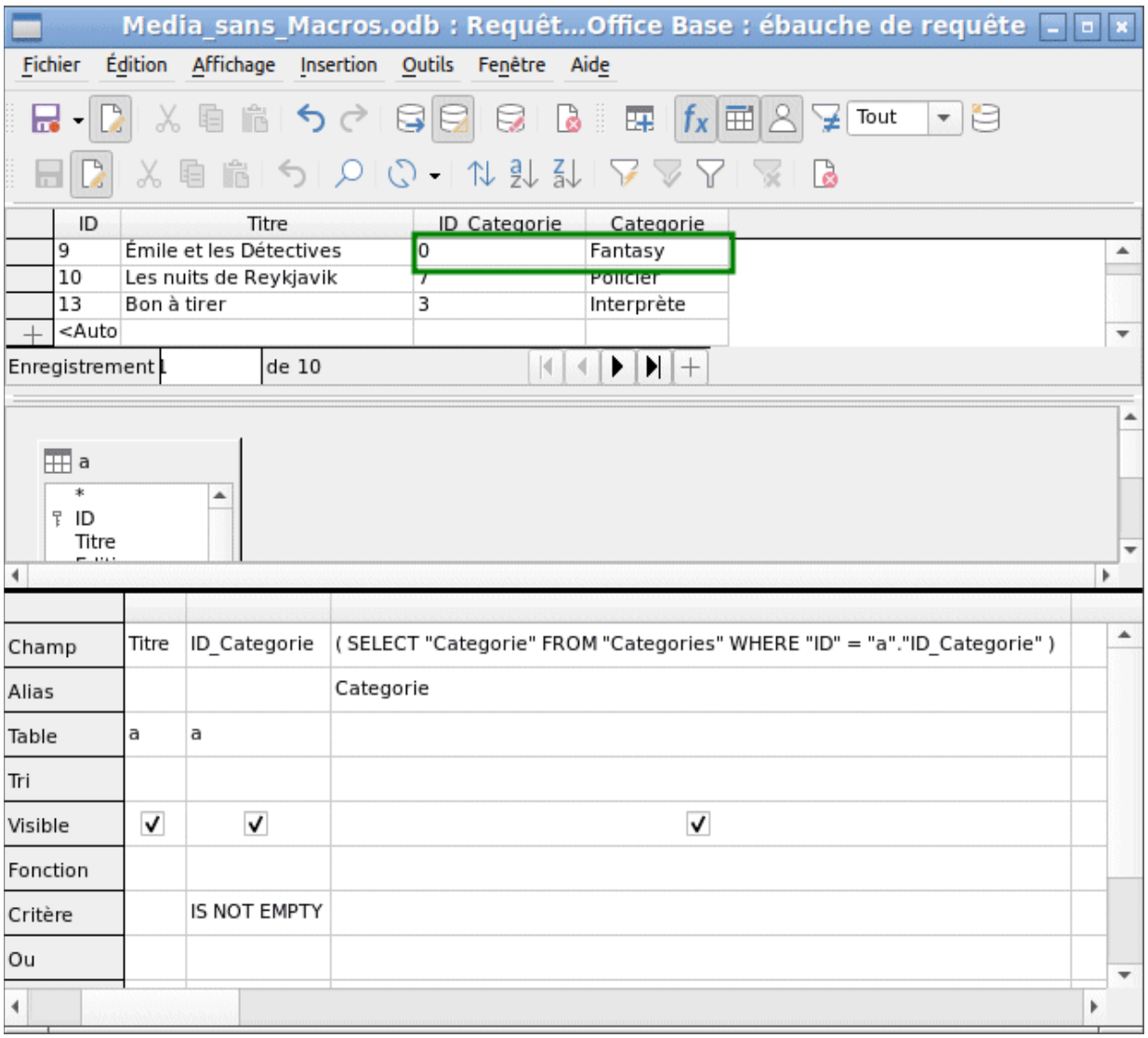


Cependant, il est possible d'éditer le contenu de l'enregistrement de la catégorie correspondante, par exemple pour remplacer « Policier » par « Suspense ». Le nom de la catégorie sera alors modifié dans la table Categories, donc pour tous les enregistrements liés à cette catégorie.

```
SELECT "m"."ID", "m"."Titre", "Categories"."Categorie", "Categorie"."ID" AS
"CatID"
FROM "Medias" AS "m", "Categories"
WHERE "m"."ID_Categorie" = "Categories"."ID";
```

Dans cette requête, la table Medias est accessible à l'aide d'un alias. La requête ne peut pas être modifiée.

Dans l'exemple ci-dessus, ce problème est facilement évité. Si, toutefois, une sous-requête corrélée (voir page 276)est utilisée, vous devez utiliser un alias de table. Une requête n'est modifiable dans ce cas que si elle ne contient qu'une seule table dans la requête principale.



En mode Ébauche, un seul tableau apparaît. La table Medias reçoit un alias afin que le contenu du champ ID_ Categorie soit accessible à l'aide d'une sous-requête corrélée.

```
SELECT "ID", "Titre", "ID_Categorie", (SELECT "Categorie" FROM "Categories"
WHERE "ID" = "a"."ID_Categorie") AS "Categorie" FROM "Medias" AS "a" WHERE
"ID_Categorie" IS NOT NULL
```

Dans une requête comme celle-ci, il est désormais possible de changer le champ de clé étrangère ID_Categorie en une autre catégorie. Dans l'exemple ci-dessus, le champ ID_Categorie passe de 7 à 0. Le titre « Emilie et les détectives » est ainsi affecté à la catégorie "Fantasy".

Cependant, il n'est plus possible de modifier une valeur dans le champ qui a reçu son contenu via une sous-requête corrélée. Une tentative de changement de catégorie de "Fantasy" à "Suspense" est montrée. Cette modification n'est pas enregistrée et ne peut pas non plus être enregistrée, puisque dans le tableau que la vue de conception affiche, le champ Categorie n'est pas présent, mais seulement un alias.

Utilisation des paramètres dans les requêtes

Si vous utilisez souvent la même requête de base mais avec des valeurs différentes à chaque fois, des requêtes avec des paramètres peuvent être utilisées. En principe, les requêtes avec paramètres fonctionnent comme les requêtes pour un sous-formulaire :

```
SELECT "ID", "ID_Lecteur", "ID_Media", "Date_Pret"
FROM "Prets"
WHERE "Date_Retour" IS NULL AND "ID_Lecteur"=2 ;
```

Cette requête n'affiche que les médias prêtés au lecteur avec le numéro 2.

```
SELECT "ID", "ID_Lecteur", "ID_Media", "Date_Pret"
FROM "Prets"
WHERE "Date_Retour" IS NULL AND "ID_Lecteur"= :NumeroLecteur ;
```

Désormais, lorsque vous exécutez la requête, un champ de saisie apparaît. Il vous invite à entrer un numéro de lecteur. Quelle que soit la valeur que vous entrez ici, les médias actuellement prêtés à ce lecteur seront affichés.

```
SELECT
      "Pret"."ID",
      "Lecteurs"."Nom"||', '||"Lecteurs"."Prenom",
      "Prets"."ID_Media",
      "Prets"."Date_Pret"
FROM "Prets", "Lecteurs"
WHERE "Prets"."Date_Retour" IS NULL
      AND "Lecteurs"."ID" = "Prets"."ID_Lecteur"
      AND "Lecteurs"."Nom" LIKE '%' ||: NomLecteur || '%'
ORDER BY "Lecteurs"."Nom"||', '||"Lecteurs"."Prenom" ASC ;
```

Cette requête est clairement plus conviviale que la précédente. Il n'est plus nécessaire de connaître le numéro du lecteur. Tout ce dont vous avez besoin est de saisir une partie du nom et tous les médias prêtés aux lecteurs correspondants sont affichés. La casse est importante.

Si vous remplacez

```
"Lecteurs"."Nom" LIKE '%' ||: NomLecteur || '%'
```

par

```
LOWER("Lecteurs"."Nom") LIKE '%' || LOWER(: NomLecteur) || '%'
```

Peu importe que le nom soit saisi en majuscules ou en minuscules.

Si le paramètre dans la requête ci-dessus est laissé vide, toutes les versions de LibreOffice jusqu'à 4.4 afficheront tous les lecteurs, car ce n'est que dans la version 4.4 qu'un champ de paramètre vide se lit comme NULL plutôt que comme une chaîne vide. Si vous ne voulez pas ce comportement, vous devez utiliser une astuce pour l'empêcher :

```
LOWER ("Lecteurs"."Nom") LIKE '%' || IFNULL(NULLIF (LOWER (: NomLecteur),
''), '§§') || '%'
```

Le champ de paramètre vide renvoie une chaîne vide et non une valeur NULL à la requête. Par conséquent, le champ de paramètre vide doit être affecté à la propriété NULL à l'aide de NULLIF. Ensuite, puisque l'entrée de paramètre donne maintenant NULL, elle peut être réinitialisée à une valeur qui ne se produit normalement dans aucun enregistrement. Dans l'exemple ci-dessus, il s'agit de «§§». Cette valeur ne sera bien sûr pas trouvée dans la recherche.

Depuis la version 4.4, des adaptations à cette technique de requête sont nécessaires :

```
LOWER ("Lecteurs"."Nom") LIKE '%' || LOWER (: NomLecteur) || '%'
```

doit, en l'absence d'entrée, obligatoirement donner pour la combinaison :

```
'%' || LOWER (: NomLecteur) || '%' la valeur NULL.
```

Pour éviter cela, ajoutez une condition supplémentaire, à savoir que pour un champ vide, toutes les valeurs sont réellement affichées :

```
(LOWER ("Lecteurs"."Nom") LIKE '%' || LOWER (: NomLecteur) || '%' OR :
NomLecteur IS NULL)
```

Le tout doit être mis entre parenthèses. Ensuite, soit un nom est recherché, soit, si le champ est vide (NULL de LibreOffice 4.4), la deuxième condition s'appliquera.

Lors de l'utilisation de formulaires, le paramètre peut être passé du formulaire principal à un sous-formulaire. Cependant, il arrive parfois que les requêtes utilisant des paramètres dans les sous-formulaires ne soient pas mises à jour, si les données sont modifiées ou nouvellement saisies.

Souvent, il serait bien de modifier le contenu des zones de liste en utilisant les paramètres du formulaire principal. Ainsi, par exemple, nous pourrions empêcher le prêt de supports de bibliothèque à des personnes actuellement interdites d'emprunter des supports. Malheureusement, il n'est pas possible de contrôler les paramètres de la zone de liste de cette manière personnalisée à l'aide de paramètres.

Sous-requêtes

Les sous-requêtes intégrées aux champs ne renvoient toujours qu'un seul enregistrement. Le champ ne renvoie également qu'une seule valeur.

```
SELECT "ID", "Revenu", "Depense",
      (SELECT SUM("Revenu") – SUM("Depense")
```

```
            FROM "Caisse") AS "Balance"
FROM "Caisse";
```

Cette requête permet la saisie de données (clé primaire incluse). La sous-requête renvoie précisément une valeur, à savoir le solde total. Cela permet de lire le solde de la caisse après chaque entrée. Ce n'est toujours pas comparable avec le formulaire de paiement du supermarché décrit dans "Requêtes comme base d'informations supplémentaires dans les formulaires" en page 268. Naturellement, il manque les calculs individuels de Total * Prix_Unitaire, mais aussi la présence du numéro de reçu. Seule la somme totale est donnée. Au moins le numéro de reçu peut être inclus à l'aide d'un paramètre de requête :

```
SELECT "ID", "Revenu", "Depense",
      (SELECT SUM("Revenu") - SUM("Depense")
            FROM "Caisse")
            WHERE "ID_Recu" = : Numero_Recu) AS "Balance"
FROM "Caisse" WHERE "ID_Recu" =: Numero_Recu ;
```

Dans une requête avec paramètres, le paramètre doit être le même dans les deux instructions de requête s'il doit être reconnu comme paramètre.

Pour les sous-formulaires, ces paramètres peuvent être inclus. Le sous-formulaire reçoit alors, au lieu d'un nom de champ, le nom de paramètre correspondant. Ce lien ne peut être saisi que dans les propriétés du sous-formulaire, et non lors de l'utilisation de l'assistant.

Note

Les sous-formulaires basés sur des requêtes ne sont pas automatiquement mis à jour en fonction de leurs paramètres. Il est plus approprié de transmettre le paramètre directement depuis le formulaire principal.

Sous-requêtes corrélées

En utilisant une requête encore plus raffinée, une requête modifiable vous permet même de porter le solde courant de la caisse :

```
SELECT "ID", "Revenu", "Depense",
(SELECT SUM("Revenu") - SUM("Depense")
      FROM "Caisse"
      WHERE "ID" <= "a"."ID") AS "Balance"
FROM "Caisse" AS "a"
ORDER BY "ID" ASC
```

La table de paiement est le même que la table « a ». "a" ne donne cependant que la relation avec les valeurs actuelles de cet enregistrement. De cette manière, la valeur actuelle de l'ID de la requête externe peut être évaluée dans la sous-requête. Ainsi, en fonction de l'ID, le solde précédent au moment correspondant est déterminé, si vous partez du fait que l'ID, qui est une valeur automatique, s'incrémente de lui-même.

Note

Si la sous-requête doit être filtrée pour le contenu à l'aide de la fonction de filtre de l'éditeur de requête, cela ne fonctionne actuellement que si vous utilisez des parenthèse doubles au lieu de simples au début et à la fin de la sous-requête : `((SELECT…)) AS "Saldo"`

Requêtes comme tables source pour les requêtes

Une requête est nécessaire pour verrouiller tous les lecteurs qui ont reçu un troisième avis de retard pour un support.

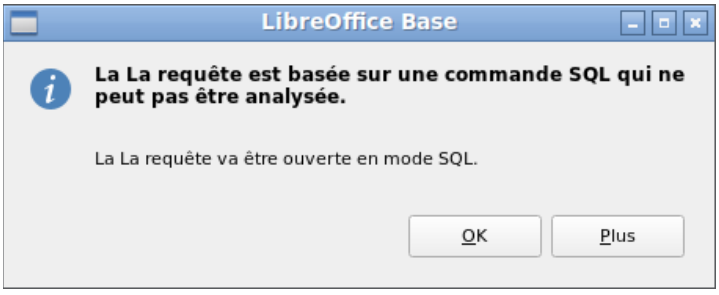
```
SELECT "Prets"."ID_Lecteur", '3e retard – le lecteur est sur la liste noire'
AS "Verrou"
FROM
      (SELECT COUNT("Date") AS "Compte_Total", "ID_Pret"
            FROM "Rappels" GROUP BY "ID_Pret") AS "a",
      "Prets"
WHERE "a"."ID_Pret" = "Prets"."ID" AND "a"."Compte_Total" > 2
```

Examinons d'abord la requête interne à laquelle se rapporte la requête externe. Dans cette requête, le nombre d'entrées de date regroupées par la clé étrangère ID_Pret est déterminé. Cela ne doit pas dépendre de l'ID_Lecteur, car cela entraînerait non seulement trois avis en retard pour un seul support, mais aussi trois médias avec un avis en retard chacun à compter. La requête interne reçoit un alias afin qu'elle puisse être liée à l'ID_Lecteur à partir de la requête externe.

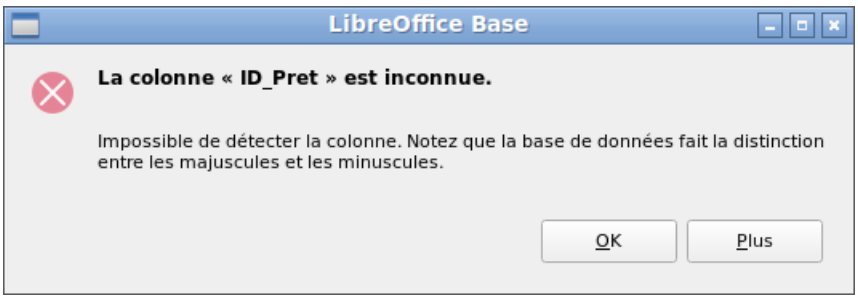
La requête externe concerne dans ce cas uniquement la formule conditionnelle de la requête interne. Il affiche uniquement un ID_Lecteur et le texte du champ Verrou lorsque le "Pret". "ID" et "a". "ID_Pret" sont égaux et "a". "Compte_Total"> 2.

En principe, tous les champs de la requête interne sont disponibles pour la requête externe. Ainsi, par exemple, la somme "a". "Compte_Total" peut être fusionnée dans la requête externe pour donner le total réel des amendes.

Cependant, il peut arriver, dans la boîte de dialogue Ébauche de requête, que le mode Ébauche ne fonctionne plus après une telle construction. Si vous essayez à nouveau d'ouvrir la requête pour la modifier, vous obtenez ce qui suit warning :



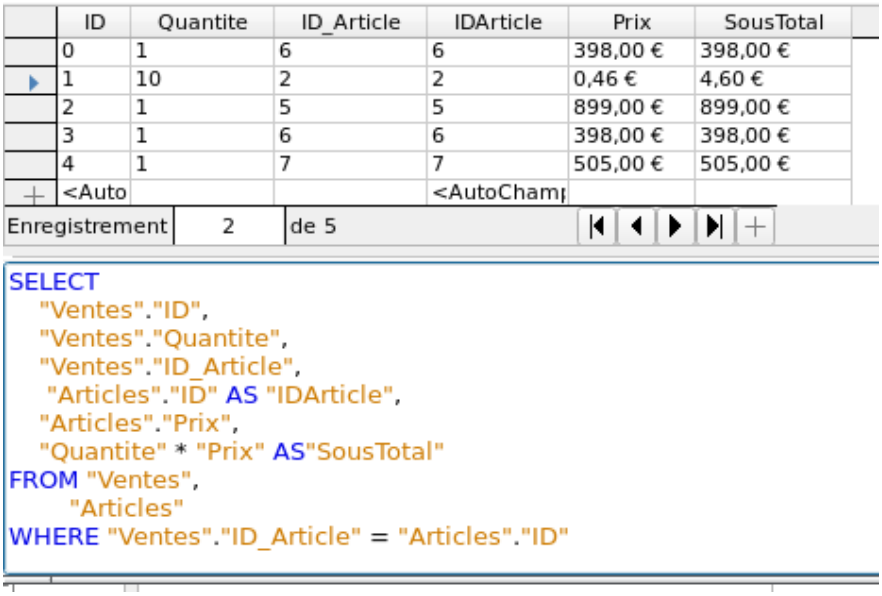
Si vous ouvrez ensuite la requête pour la modifier en mode SQL et que vous essayez de passer de là en mode Ébauche, vous obtenez le message d'erreur :



Le mode d'affichage de conception ne peut pas trouver le champ contenu dans la requête interne "ID_Pret", qui régit la relation entre les requêtes internes et externes. Lorsque la requête est exécutée en mode SQL, le contenu correspondant de la sous-requête est reproduit sans erreur. Par conséquent, vous n'êtes pas obligé d'utiliser le mode SQL direct dans ce cas.

La requête externe a utilisé les résultats de la requête interne pour produire les résultats finaux. Voici une liste des valeurs "ID_Pret" qui doivent être verrouillées et pourquoi. Si vous souhaitez limiter davantage les résultats finaux, utilisez les fonctions de tri et de filtrage de l'interface utilisateur graphique.

Les captures d'écran suivantes montrent comment les résultats d'une requête peuvent être obtenus de différentes manières avec des sous-requêtes. Ici, une requête vers une base de données de stock tente de déterminer ce que le client doit payer à la caisse. Les prix individuels sont multipliés par le nombre d'articles achetés donnant un sous-total. Ensuite, la somme de ces sous-totaux doit être déterminée. Tout cela doit être modifiable pour que la requête puisse être utilisée comme base d'un formulaire.



	ID	Quantite	ID_Article	IDArticle	Prix	SousTotal
	0	1	6	6	398,00 €	398,00 €
▶	1	10	2	2	0,46 €	4,60 €
	2	1	5	5	899,00 €	899,00 €
	3	1	6	6	398,00 €	398,00 €
	4	1	7	7	505,00 €	505,00 €
*	<Auto			<AutoChamp		

Enregistrement 2 de 5

```
SELECT
   "Ventes"."ID",
   "Ventes"."Quantite",
   "Ventes"."ID_Article",
    "Articles"."ID" AS "IDArticle",
   "Articles"."Prix",
   "Quantite" * "Prix" AS"SousTotal"
FROM "Ventes",
      "Articles"
WHERE "Ventes"."ID_Article" = "Articles"."ID"
```

Figure 89: Requête utilisant deux tables. Pour la rendre modifiable, les deux clés primaires doivent être incluses.

Note

En raison du bogue 61871, Base ne met pas à jour automatiquement le résultat partiel.

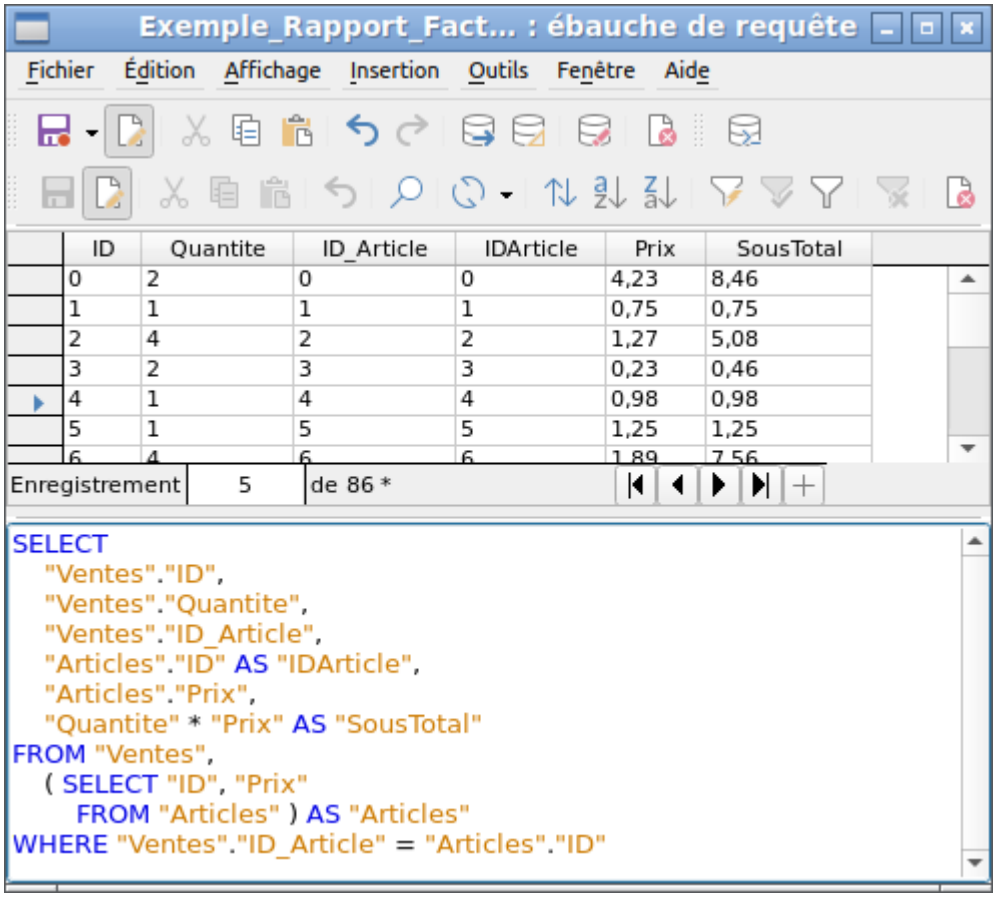


Figure 90: La table Articles est déplacée dans une sous-requête, qui est créée dans la zone de table (après le terme « FROM ») et reçoit un alias. Désormais, la clé primaire de la table Articles n'est plus strictement nécessaire pour rendre la requête modifiable.

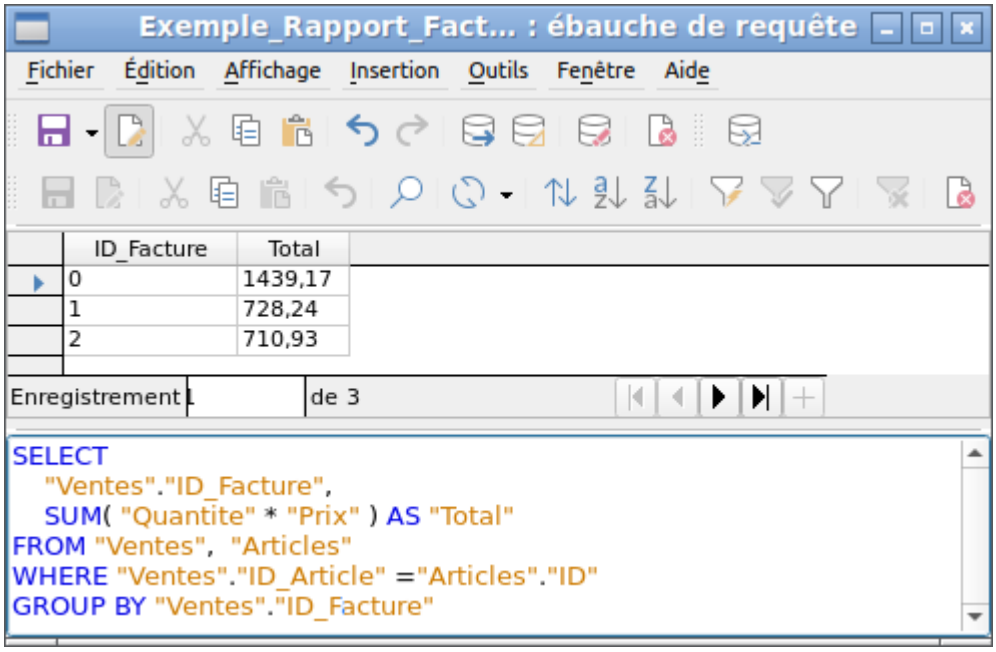


Figure 91: La somme calculée doit maintenant apparaître dans la requête. Déjà, la requête simple pour la somme de calcul n'est pas modifiable, elle est donc regroupée et additionnée ici.

ID	Quantite	ID_Article	Prix	SousTotal	Total
51	1	27	78,89	78,89	1439,17
52	2	28	1,25	2,5	1439,17
53	1	29	0,85	0,85	1439,17
54	2	30	0,65	1,3	1439,17
55	2	0	4,23	8,46	728,24
56	1	1	0,75	0,75	728,24
57	4	2	1,27	5,08	728,24
58	2	3	0,23	0,46	728,24
59	1	4	0,98	0,98	728,24

Enregistrement 1 de 110

```
SELECT
   "Ventes"."ID",
   "Ventes"."Quantite",
   "Ventes"."ID_Article",
   "Articles"."Prix",
   "Quantite" * "Prix" AS "SousTotal",
   "TotalFacture"."Total"
FROM "Ventes",
      ( SELECT
         "ID",
         "Prix"
      FROM "Articles" )
   AS "Articles",
      ( SELECT
         "Ventes"."ID_Facture",
         SUM( "Quantite" * "Prix" ) AS "Total"
      FROM "Ventes",
         "Articles"
      WHERE "Ventes"."ID_Article" = "Articles"."ID"
      GROUP BY "Ventes"."ID_Facture" )
   AS "TotalFacture"
WHERE "Ventes"."ID_Article" = "Articles"."ID"
   AND "Ventes"."ID_Facture" = "TotalFacture"."ID_Facture"
ORDER BY "Ventes"."ID_Facture", "Ventes"."ID"
```

Figure 92: Avec la deuxième sous-requête, l'impossibilité apparente devient possible. La requête précédente est insérée en tant que sous-requête dans la définition de table de la requête principale (après « FROM »). Par conséquent, l'ensemble de la requête reste modifiable. Dans ce cas, les entrées ne sont possibles que dans les colonnes « Quantite » et « ID_Article ». Ceci est ensuite précisé dans le formulaire de requête.

Récapituler les données avec des requêtes

Lorsque des données sont recherchées sur toute une base de données, l'utilisation de fonctions de formulaire simples pose souvent des problèmes. Un formulaire se réfère après tout à une seule table et la fonction de recherche se déplace uniquement dans les enregistrements sous-jacents de ce formulaire.

Obtenir toutes les données est plus simple lorsque vous utilisez des requêtes, qui peuvent fournir une image de tous les enregistrements. La section sur “Définition de la relation dans la requête” suggère une telle construction de requête. Ceci est construit pour l'exemple de base de données comme suit :

```
SELECT "Medias"."Titre", "SousTitre"."SousTitre", "Auteurs"."Nom"
FROM "Medias"
      LEFT JOIN "SousTitre"
            ON "Medias"."ID" = "SousTitre"."ID_Media"
      LEFT JOIN "relation_Media_Auteur"
            ON "Medias"."ID" = "relation_Media_Auteur"."ID_Media"
      LEFT JOIN "Auteurs"
            ON "relation_Media_Auteur"."ID_Auteur" = "Auteurs"."ID"
```

Ici, tous les titres, sous-titres et auteurs sont affichés ensemble.

La table Medias contient un total de 13 titres. Pour deux de ces titres, il y a un total de 10 sous-titres. Sans une jointure à gauche (LEFT **JOIN)**, les deux tables affichées ensemble ne donnent que 10 enregistrements. Pour chaque sous-titre, le titre correspondant est recherché, et c'est la fin de la requête. Les titres sans sous-titres ne sont pas affichés.

Maintenant, pour afficher tous les médias, y compris ceux sans sous-titre : le média est sur le côté gauche de l'affectation, sous-titre sur le côté droit. Une jointure à gauche affichera tous les titres des médias, mais seulement un sous-titre pour ceux qui ont un titre. Le média devient la table décisive pour déterminer quels enregistrements doivent être affichés. Cela avait déjà été prévu lors de la construction de la table (voir Chapitre 3, Tables). Comme des sous-titres existent pour deux des treize titres, la requête affiche désormais 13+ 9 = 22 enregistrements.

Note

La liaison normale des tables, une fois que toutes les tables ont été répertoriées, suit le mot-clé WHERE.

S'il existe une jointure à gauche ou une jointure à droite, l'affectation est définie directement après les deux noms de table à l'aide de ON. La séquence est donc toujours
`Table1 LEFT JOIN Table2 ON Table1.Champ1 = Table2.Champ1 LEFT JOIN Table3 ON Table2.Champ1 = Table3.Champ1…`

Deux titres de la table Medias n'ont pas encore d'entrée d'auteur ou de sous-titre. En même temps, un titre a un total de trois auteurs. Si la table Auteurs est liée sans une jointure à gauche, les médias sans auteur ne seront pas affichés. Mais comme un média a trois auteurs au lieu d'un, le nombre total d'enregistrements affichés sera de 23.

Ce n'est qu'en utilisant LEFT JOIN que la requête sera invitée à utiliser la table Medias pour déterminer les enregistrements à afficher. Maintenant, les enregistrements sans sous-titre ni auteur réapparaissent, ce qui donne un total de 25 enregistrements.

L'utilisation de jointures appropriées augmente généralement la quantité de données affichées. Mais cet ensemble de données élargi peut être facilement examinés, car les auteurs et les sous-titres sont affichés en plus des titres. Dans l'exemple de base de données, toutes les tables dépendantes du support sont accessibles.

Accès plus rapide aux requêtes à l'aide des vues de table

Les vues en SQL sont plus rapides que les requêtes, en particulier pour les bases de données externes, car elles sont directement ancrées dans la base de données et le serveur ne renvoie que les résultats. En revanche, les requêtes sont d'abord envoyées au serveur et y sont traitées.

Si une nouvelle requête est liée à une autre requête, la vue SQL de Base fait que l'autre requête ressemble à une table. Si vous créez une vue à partir de celle-ci, vous pouvez voir que vous travaillez réellement avec une sous-requête (Select utilisé dans un autre Select). Pour cette raison, une requête 2 qui se rapporte à une autre requête 1 ne peut pas être exécutée (en mode vue SQL) à l'aide de la commande **Édition> Exécuter directement l'instruction SQL**, car seule l'interface utilisateur graphique et non la base de données elle-même connaît la requête 1.

La base de données ne vous donne aucun accès direct aux requêtes. Cela s'applique également à l'accès à l'aide de macros. Les vues, en revanche, sont accessibles à partir de macros et de tables. Cependant, aucun enregistrement ne peut être modifié dans une vue. (Ils doivent être modifiés dans un tableau ou un formulaire.)

Conseils

Une requête créée à l'aide de Créer une requête en mode SQL présente l'inconvénient de ne pas pouvoir être triée ou filtrée à l'aide de l'interface graphique. Il y a donc des limites à son utilisation.

Une vue en revanche peut être gérée dans Base comme une table normale – à l'exception qu'aucune modification des données n'est possible. Ici donc, même dans les commandes SQL directes, toutes les possibilités de tri et de filtrage sont disponibles.

De plus, la mise en forme des colonnes dans une vue est conservée lorsque la base de données est fermée, contrairement aux colonnes d'une requête.

Les vues sont une solution pour de nombreuses requêtes, si vous souhaitez obtenir des résultats. Si, par exemple, une sous-sélection doit être utilisée sur les résultats d'une requête, créez une vue qui vous donne ces résultats. Ensuite, utilisez la sous-sélection sur la vue. Des exemples correspondants se trouvent dans le chapitre 8, Trucs et Astuces.

La création d'une vue à partir d'une requête est plutôt simple et directe.

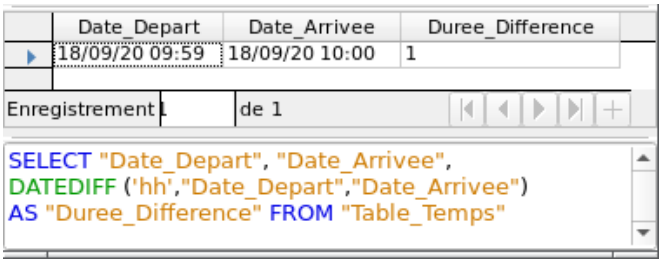
Clic droit sur la requête, puis **Créer en tant que Vue.**

Erreurs de calcul dans les requêtes

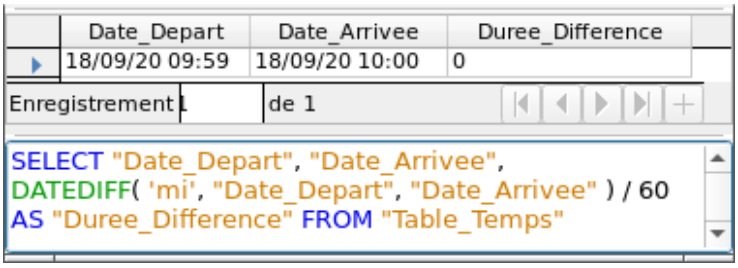
Les requêtes sont également utilisées pour calculer les valeurs. Parfois, la base de données interne HSQLDB produit des erreurs apparentes qui, à un examen plus approfondi, se révèlent être des interprétations logiquement correctes des données. Il existe également des problèmes d'arrondi, qui peuvent facilement prêter à confusion.

Les heures dans HSQLDB ne sont formatées correctement que jusqu'à une différence de 23:59:59 heures. Si plusieurs durées doivent être ajoutées, par exemple pour calculer les heures travaillées, il faut trouver un autre moyen. Ici, il existe plusieurs approches compliquées :

- Le temps est directement exprimé uniquement comme un total de minutes ou même de secondes. Avantage : les valeurs permettent un calcul ultérieur sans problème.
- Le temps est divisé en heures, minutes et secondes et réassemblé sous forme de texte en utilisant «:» comme séparateur. Avantage : le texte apparaît dans les requêtes comme une heure correctement formatée depuis le début.
- L'heure est créée sous forme de nombre décimal. Un jour vaut 1, une heure correspond à 1/24 et ainsi de suite. Avantage : les valeurs peuvent ensuite être reformatées en temps dans la requête et présentées sous forme de champ de formulaire formatable.

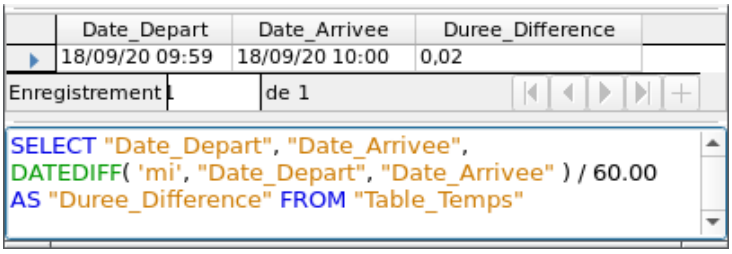


DATEDIFF permet de déterminer les intervalles de temps. Il demande explicitement la différence à déterminer, dans le cas présent les heures ('hh'). . Si, par exemple, le champ Date_Arrivee est défini sur 19/09/20 10:00, cela est calculé comme 25 heures.



Si au contraire l'intervalle de temps est calculé en minutes puis divisé par 60, le décalage horaire en heures devient nul. Cela ressemble plus à la valeur correcte (puiqu'il y a bien 0 heure de décalage) , sauf : où est passée cette minute ?

En utilisant `MOD(DATEDIFF('mi',"Date_Depart",Date_Arrivee"),60`, on obtient bien 1 minute.

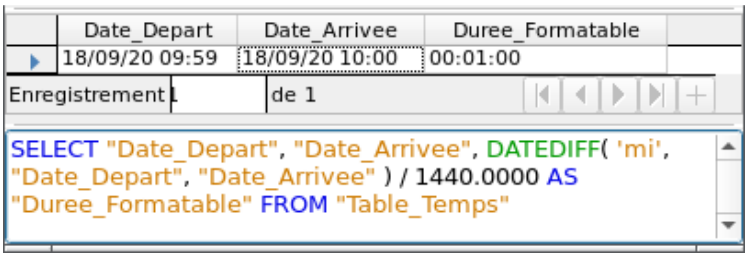


L'intervalle de temps a été donné sous forme de nombre entier. Un entier a été divisé par un entier. Le résultat de la requête dans de tels cas doit également être un entier et non un nombre décimal. Cela peut être facilement corrigé. Le décalage horaire en minutes est divisé par un nombre décimal avec deux décimales (60,00). Cela donne un résultat qui a également deux décimales. 0,02 heure n'est toujours pas exactement une minute, mais c'est beaucoup plus proche qu'avant. Le nombre de décimales peut être augmenté en utilisant plus de zéros. Une valeur de 0,016 est une approximation encore plus proche, mais des erreurs de calcul ultérieures ne peuvent pas toujours être exclues.

Au lieu d'avoir à travailler avec beaucoup de zéros ajoutés, le type de données de DATEDIFF peut être directement influencé. En utilisant `(CONVERT(DATEDIFF('mi', "Date_Depart", "Date_Arrivee"), DECIMAL(50,49)))/60` vous pouvez obtenir une précision de 49 décimales.

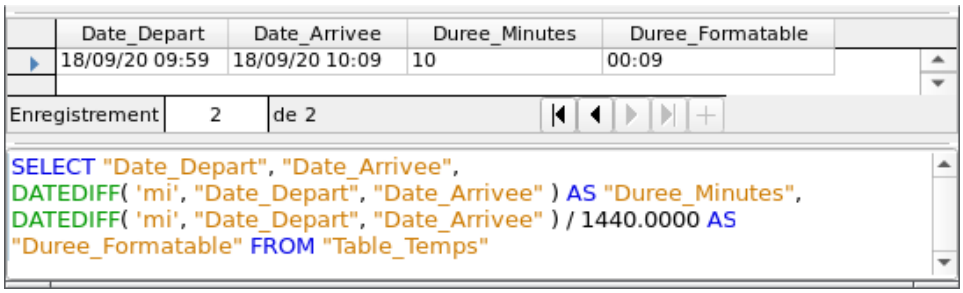
Lorsque vous utilisez des fonctions de calcul, vous devez toujours comprendre que les types de données dans HSQLDB n'ont qu'une précision limitée. Quel que soit le nombre de décimales que vous utilisez, il n'en reste pas moins que les résultats intermédiaires impliquant un décompte du temps ne peuvent être utilisés que dans une mesure limitée pour des calculs ultérieurs.

Si une valeur d'horodatage doit par la suite être utilisée dans un formulaire ou un état en tant qu'heure formatée, vous devez vous assurer que le jour est valide comme base du format d'heure.



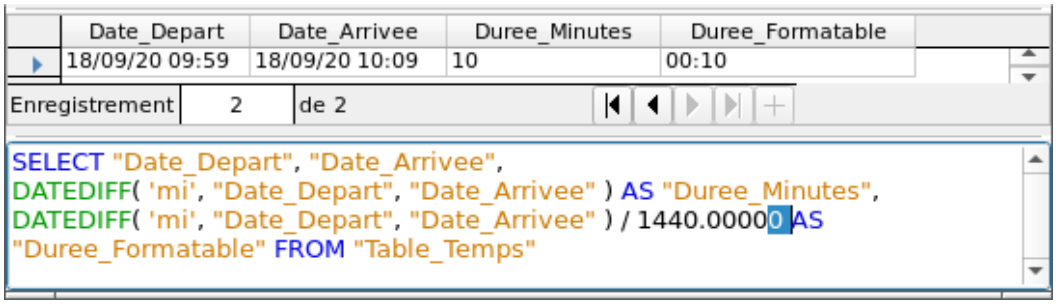
Ci-dessus, la différence est calculée en minutes. Le résultat est donné sous forme de fraction de jour. Un jour a 60 * 24 minutes. Si vous divisez simplement par 1440, le résultat serait zéro, donc encore une fois, vous devez indiquer explicitement les décimales. Il apparaît alors comme une heure formatée de 0 heure et 1 minute. Le formatage est effectué sur la colonne du tableau, après l'affichage.

Le code de format pour une durée supérieure à un jour est [HH] : MM. Si vous utilisez le mauvais format, un décalage horaire de 1 jour et 1 minute peut être affiché comme seulement 1 minute.



Le démon de l'erreur frappe à nouveau ! Un décalage horaire de 10 minutes ne doit pas apparaître comme 9 minutes lorsqu'il est correctement formaté. Pour savoir où se situe le problème, nous devons considérer exactement comment le calcul est effectué :

1. 10/1440 = 0,00694. Le résultat est arrondi à 0,0069 car seules quatre décimales ont été spécifiées.
2. 0,0069 * 1440 = 9,936 minutes, soit 9 minutes 56,16 secondes. Et les secondes ne sont pas affichées dans le format choisi !. Si on choisit le format hh : mm : ss on obtient 00:09:56.



Date_Depart	Date_Arrivee	Duree_Minutes	Duree_Formatable
18/09/20 09:59	18/09/20 10:09	10	00:10

Enregistrement 2 de 2

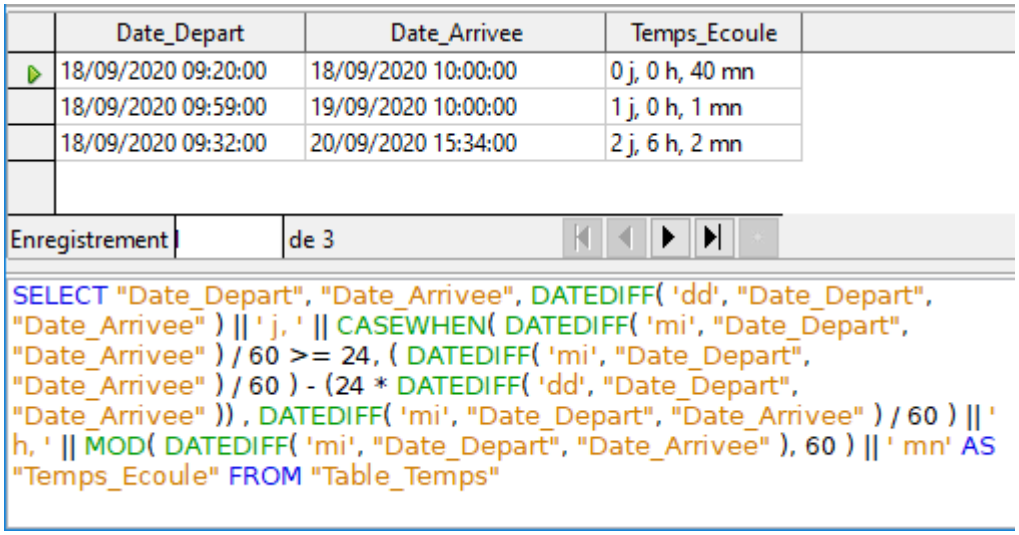
```
SELECT "Date_Depart", "Date_Arrivee",
DATEDIFF( 'mi', "Date_Depart", "Date_Arrivee" ) AS "Duree_Minutes",
DATEDIFF( 'mi', "Date_Depart", "Date_Arrivee" ) / 1440.00000 AS
"Duree_Formatable" FROM "Table_Temps"
```

L'allongement du diviseur d'une seule décimale (de 1440,0000 à 1440,00000) corrige cette erreur. Maintenant, l'arrondi est à 0,00694. 0,00694 * 1440 donne 9,9936, soit 59,616 secondes. Le nombre de secondes est arrondi à 60 secondes, donc les 9 minutes ont 1 minute ajoutée, soit 10 minutes en tout.

Ici aussi, il pourrait y avoir d'autres problèmes. Peut-il y avoir d'autres décimales qui, une fois formatées, ne donnent pas 1 minute ? Pour résoudre ce problème, un court calcul utilisant Calc avec des chiffres arrondis de la même manière peut aider. La colonne A contient une séquence de nombres à partir de 1 (pour les minutes). La colonne B contient la formule = ROUND (A1/ 440 ; 4) et est formatée pour afficher les heures et les minutes. Si cela continue vers le bas, nous pouvons voir, à côté de 10 minutes dans la colonne A, 00:09 dans la colonne B. De même pendant 28 minutes, etc. Si vous arrondissez à 5 places, ces erreurs disparaissent.

Aussi agréable que cela puisse être d'avoir un affichage correctement formaté dans un formulaire, vous devez être conscient que vous avez affaire à des valeurs arrondies qui ne conviennent pas pour une utilisation ultérieure dans des calculs mécaniques aveugles. Si une valeur doit être utilisée pour un calcul ultérieur, il est préférable d'utiliser uniquement une représentation du décalage horaire dans les plus petites unités disponibles, dans ce cas les minutes.

Une alternative est fournie ci- dessous



Date_Depart	Date_Arrivee	Temps_Ecoule
18/09/2020 09:20:00	18/09/2020 10:00:00	0 j, 0 h, 40 mn
18/09/2020 09:59:00	19/09/2020 10:00:00	1 j, 0 h, 1 mn
18/09/2020 09:32:00	20/09/2020 15:34:00	2 j, 6 h, 2 mn

Enregistrement de 3

```
SELECT "Date_Depart", "Date_Arrivee", DATEDIFF( 'dd', "Date_Depart",
"Date_Arrivee" ) || ' j, ' || CASEWHEN( DATEDIFF( 'mi', "Date_Depart",
"Date_Arrivee" ) / 60 >= 24, ( DATEDIFF( 'mi', "Date_Depart",
"Date_Arrivee" ) / 60 ) - (24 * DATEDIFF( 'dd', "Date_Depart",
"Date_Arrivee" )) , DATEDIFF( 'mi', "Date_Depart", "Date_Arrivee" ) / 60 ) || '
h, ' || MOD( DATEDIFF( 'mi', "Date_Depart", "Date_Arrivee" ), 60 ) || ' mn' AS
"Temps_Ecoule" FROM "Table_Temps"
```

La requête affiche :

- les champs Date_Depart et Date_Arrivee. Ils sont de type DATE/Heure [TIMESTAMP] au format JJ/MM/AAAA HH:MM:SS ;

`DATEDIFF ('dd', "DATE_Depart", "Date_Arivee")`
différence en jours entre les deux dates.

`CASEWHEN(DATEDIFF('mi', "Date_Depart", "Date_Arivee")/60 >= 24).`
Dans le cas ou la différence en minutes entre les deux dates, (divisée par 60 pour obtenir des heures) est suppérieure à 24.
Alors, le résultat sera `(DATEDIFF('mi', "Date_Depart", "Date_Arivee")/60 – (24 * DATEDIFF ('dd', "DATE_Depart", "Date_Arivee"))`
La différence en minutes, entre les deux dates, divisée par 60 (pour obtenir des heures) – 24 (heures) multiplié par la différence en jours entre les dates. Exemple (enregistrement n°3) : le résultat de `DATEDIFF('mi', "Date_Depart", "Date_Arivee")/60` donne 54 (heures) alors on obtiendra 54 – (24 * 2[jours]) = 6 heures.
Sinon le résultat sera `DATEDIFF('mi', "Date_Depart", "Date_Arivee")`/60.

`MOD(DATEDIFF('mi', "Date_Depart", "Date_Arivee"),60)`
MOD (Modulo) donne le reste entier de la division de `DATEDIFF('mi', "Date_Depart", "Date_Arivee")`, par 60.
(Nous le répétons : Attention, c'est une chaîne de caractères. Aucun calcul ne peut être réalisé à partir de ces résultats).

Pour plus de précisions sur la sujet, lire le Guide des fonctions intégrées à HSQLDB

Annexe : Notation BNF

BNF signifie Backus Naur Form (voir https://fr.wikipedia.org/wiki/Forme_de_Backus-Naur).

Tableau succinct des signes utilisés dans le présent document.

|	Ou
[ ]	Les éléments compris entre [et] sont facultatifs (optionnels).
{ }	Les accolades { } encadrent des éléments répétitifs.
||	Sert à concaténer plusieurs éléments en une seule chaîne de caractère. Les éléments de type numérique ou de type date, sont automatiquemet convertis en chaînes.

Traduction d'un exemple simple :

[+ | -] { expression [{ + | – | * | / | || } expression]}

+ ou (|) – peuvent facultativement [] se trouver devant ce qui suit ;

Le second élément est constitué à minima d'une expression (une chaîne de caractère, un nombre, un résultat).

Celle-ci peut optionnellement être suivie de :

- + ou - ou * ou / (opérateurs mathématiques pour des expressions constituées par des valeurs numériques) ;
- `ou par un opérateur de concaténation || (pour les expressions de type chaînes de caractères ;`

- et d'une nouvelle expression.

Et ceci autant de fois que souhaité.

Guide Base

Chapitre 6
Rapports

Création de rapports à l'aide du Générateur de rapports

Les rapports sont utilisés pour présenter les données d'une manière qui les rend facilement compréhensibles par des personnes sans connaissance de la base de données. Les rapports peuvent :

- Présenter les données dans des tableaux faciles à lire ;
- Créer des graphiques pour afficher les données ;
- Rendre possible l'utilisation des données pour l'impression d'étiquettes ;
- Produire des lettres types telles que des factures, des avis de rappel ou des notifications aux personnes qui rejoignent ou quittent une association.

La création d'un rapport nécessite un travail préparatoire minutieux sur la base de données sous-jacente. Contrairement à un formulaire, un rapport ne peut pas inclure de sous-rapports et donc incorporer des sources de données supplémentaires. Un rapport ne peut pas non plus présenter des éléments de données différents de ceux qui sont disponibles dans la source de données sous-jacente, comme un formulaire peut le faire à l'aide de zones de liste.

Les rapports sont mieux préparés à l'aide de requêtes. De cette manière, toutes les variables peuvent être déterminées. En particulier, si le tri dans le rapport est requis, utilisez toujours une requête qui prévoit le tri. Cela signifie que les requêtes en mode SQL direct doivent être évitées dans ces conditions. Si vous devez utiliser une requête de ce type dans votre base de données, vous pouvez effectuer le tri en créant d'abord une vue à partir de la requête. Une telle vue peut toujours être triée et filtrée à l'aide de l'interface utilisateur graphique (GUI) de Base.

Attention

Lorsque vous utilisez le Générateur de rapports, vous devez fréquemment enregistrer votre travail pendant l'édition. En plus d'enregistrer dans le Générateur de rapports lui-même après chaque étape importante, vous devez également enregistrer l'ensemble de la base de données.

Selon la version de LibreOffice que vous utilisez, le Générateur de rapports peut parfois planter pendant l'édition.

La fonctionnalité des rapports terminés n'est pas affectée même s'ils ont été créés sous une autre version, dans laquelle le problème ne se produit pas.

Note

Depuis LibreOffice 4.1, le Générateur de rapports a été entièrement intégré et n'apparaît plus dans les Extensions. Il est essentiel que vous n'installiez pas une extension de générateur de rapports qui ne correspond pas à votre version avec le Générateur de rapports intégré.

L'interface utilisateur du Générateur de rapports

Pour démarrer le Générateur de rapports à partir de Base, utilisez **Rapports> Créer un rapport en mode Ébauche.**

La fenêtre initiale du Générateur de rapports (Figure 93) comporte trois zones verticales. Sur la gauche se trouve la division actuelle du rapport comprenant en-tête de page, détail et pied de page.

Au milieu se trouvent les zones correspondantes où le contenu sera saisi ; et, à droite, les propriétés de ces régions et des objets qui y sont placés.

En même temps, la boîte de dialogue **Ajouter des champs** s'affiche. Cette boîte de dialogue correspond à celle de la création de formulaires. Il crée des champs avec leurs étiquettes de champ correspondantes.

Sans contenu de la base de données, un rapport n'a pas de fonction appropriée. Pour cette raison, la boîte de dialogue s'ouvre dans l'onglet Données. Ici, vous pouvez définir le contenu du rapport ; dans l'exemple, il s'agit de la table Vue_Rapports_Rappels. Tant que la commande Analyser SQL est définie sur Oui, le rapport peut être soumis à un tri, un regroupement et un filtrage. Une vue a été choisie pour la base de ce rapport, donc aucun filtre ne sera appliqué ; il a déjà été inclus dans la requête sous-jacente à la vue.

Note

Pour une modification ultérieure de la source de données, il faudra accéder au rapport par le *Navigateur de rapport*.

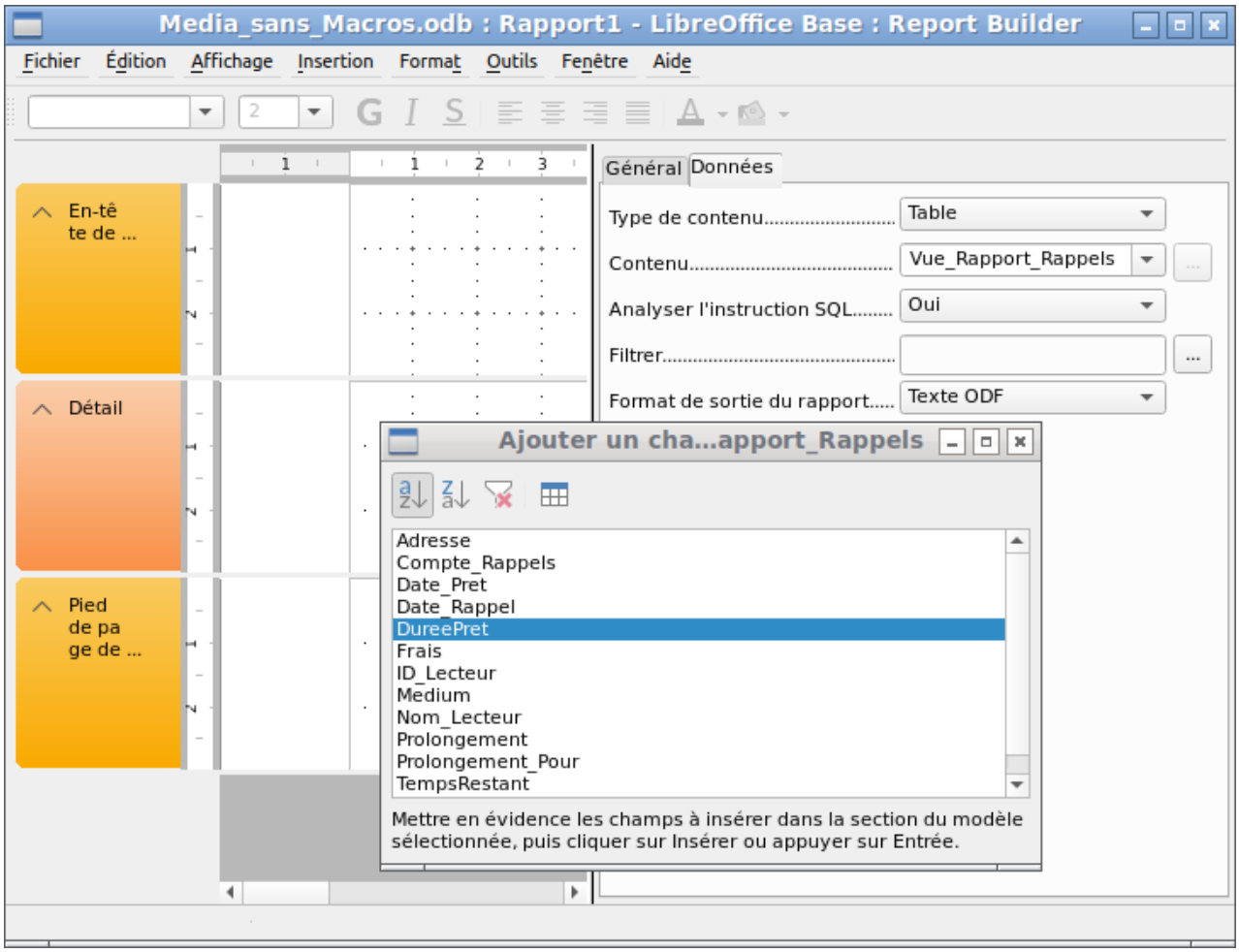


Figure 93: Fenêtre initiale du Générateur de rapports

Note

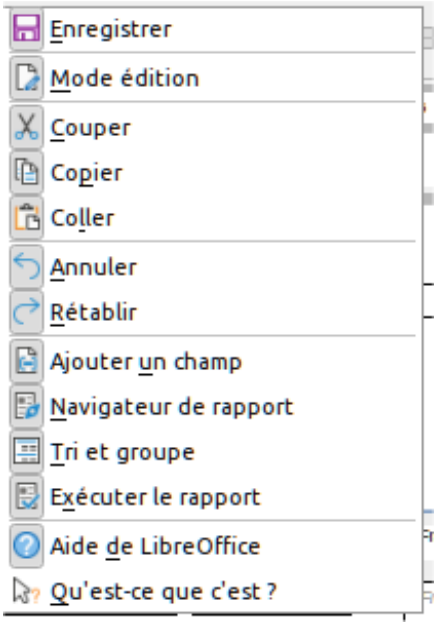
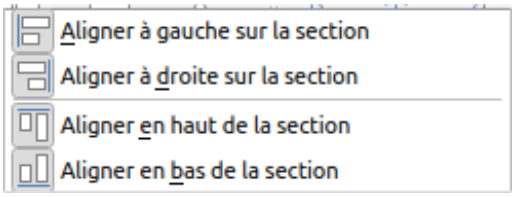
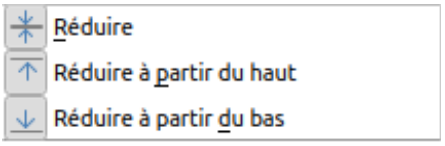
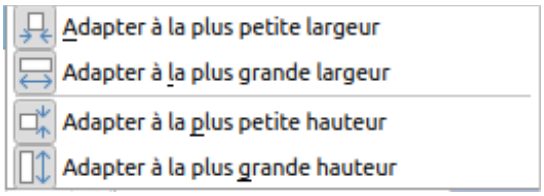
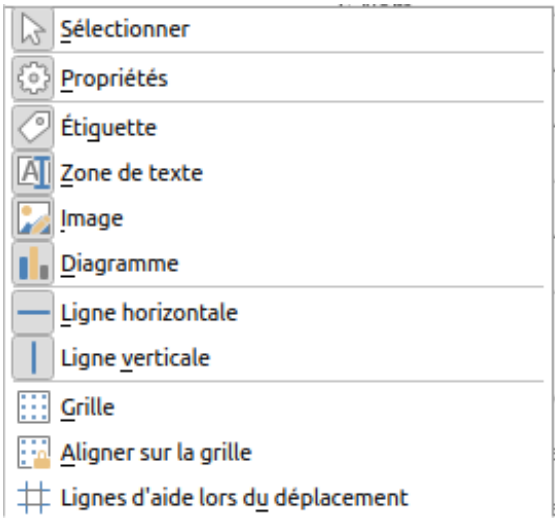
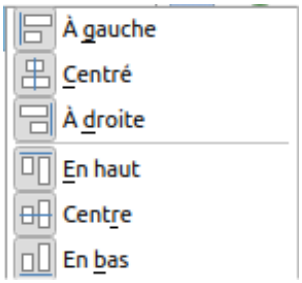
Le type de contenu fourni peut être une table, une vue, une requête ou un codage SQL direct. Le Générateur de rapports fonctionne mieux lorsqu'il reçoit des données qui ont été préparées à l'avance dans la mesure du possible. Ainsi, par exemple, les calculs dans les requêtes peuvent être effectués à l'avance et la portée des enregistrements, qui doivent apparaître dans le rapport, limitée si nécessaire.

Dans les versions antérieures de LibreOffice, il y avait parfois des problèmes lorsque les requêtes impliquant plusieurs tables devaient être regroupées ultérieurement. De tels problèmes peuvent être évités si vous utilisez une vue plutôt qu'une requête. Une vue ressemble à une table de base de données pour programmer des éléments comme le Générateur de rapports. Les noms de champs sont prédéfinis et fixes et un accès ultérieur à l'aide des commandes de tri et de regroupement est possible sans provoquer d'erreurs.

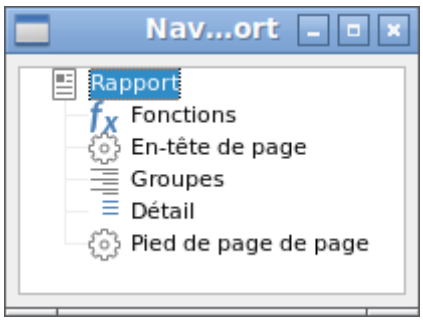
Deux formats de sortie pour les rapports sont disponibles pour la sélection : document texte ODF (un document Writer) ou feuille de calcul ODF (un document Calc). Si vous voulez juste une vue tabulaire de vos données, le document Calc doit certainement être choisi pour votre rapport. Il est beaucoup plus rapide à créer et est également plus facile à formater par la suite, car il y a moins d'options à considérer et les colonnes peuvent facilement être glissées à la largeur requise par la suite.

Par défaut, le Générateur de rapports recherche sa source de données dans la première table de la base de données. Cela garantit qu'au moins un test des fonctions est possible. Une source de données doit être choisie avant que le rapport puisse être garni avec des champs.

Le Générateur de rapports fournit de nombreux boutons supplémentaires. Le tableau de la page suivante montre les boutons avec leurs descriptions. Les boutons d'alignement des éléments ne sont pas décrits plus en détail dans ce chapitre. Ils sont utiles pour l'ajustement rapide des champs dans une seule zone du Générateur de rapports, mais en principe, tout peut être fait par l'édition directe des propriétés des champs.

Boutons d'édition de contenu	***Boutons d'alignement des éléments***
 Barre : Standard	 Barre : Aligner à la section
	 Barre : Adapter à la section
	 Barre : Redimensionnement d'objet
 Barre ; Contrôles de rapport	 Barre : Aligner

Tout comme pour les formulaires, il est utile d'utiliser le navigateur approprié. Ainsi, par exemple, un clic imprudent au début du Générateur de rapports peut rendre difficile la recherche des propriétés des données du rapport. Ces données ne peuvent être accessibles que via le navigateur de rapports. Cliquez avec le bouton gauche sur Rapport (dans le navigateur) et les propriétés du rapport sont à nouveau accessibles.



Dans un premier temps, le navigateur montre, en plus des sections visibles du document (En-tête de page, Groupes, Détail et Pied de page), la possibilité d'inclure des fonctions. Les groupes peuvent être utilisés, par exemple, pour attribuer tous les médias rappelés à la personne qui les a empruntés, afin d'éviter plusieurs avis de rappel. Les zones de détail affichent les enregistrements appartenant à un groupe. Les fonctions sont utilisées pour les calculs tels que les sommes.

Pour obtenir une sortie utile dans l'exemple ci-dessus, le contenu de la vue doit être reproduit avec un regroupement approprié. Chaque lecteur doit être lié aux avis de rappel pour tous ses supports prêtés et en retard.

Affichage> Tri et groupe ou le bouton correspondant lance la fonction de regroupement.

Figure 94: Trier et grouper (sur le nom de lecteur)

Ici, le regroupement et le tri se font par le champ Nom_Lecteur. Des champs supplémentaires pourraient également être inclus dans le tableau ci-dessus. Par exemple, si vous souhaitez également regrouper et trier par le champ Date_Pret, choisissez-le comme deuxième ligne.

Directement sous le tableau, plusieurs actions de regroupement sont disponibles pour la sélection. Vous pouvez déplacer un groupe vers le haut ou le bas de la liste ou le supprimer complètement. Comme un seul groupe est nécessaire pour le rapport planifié, la figure 94 montre uniquement le symbole Supprimer à l'extrême droite des actions de groupe disponibles.

La propriété Tri est explicite et auto-documentée.

Lorsque l'entrée a été créée, le côté gauche du Générateur de rapports montre immédiatement une nouvelle division. À côté de la description du champ Nom_Lecteur, vous pouvez maintenant voir Entête. Cette section concerne l'en-tête de groupe dans le rapport. L'en-tête peut contenir le nom de la personne qui recevra l'avis de rappel. Dans ce cas, il n'y a pas de pied de page de groupe. Un tel pied de page pourrait contenir l'amende due, ou le lieu et la date du jour et un espace pour la signature de la personne qui envoie l'avis.

Par défaut, il existe un nouveau groupe pour chaque valeur. Donc, si le Nom_Lecteur change, un nouveau groupe est démarré. Vous pouvez également grouper par lettre initiale. Dans le cas d'un avis de rappel, cependant, cela regrouperait tous les lecteurs ayant la même initiale dans un même groupe. Ventura, Versini et Vannier recevraient un avis de rappel commun, ce qui serait tout à fait inutile dans cet exemple.

Lors du regroupement par lettre initiale, vous pouvez en outre spécifier combien de lettres plus tard le groupe suivant doit commencer. On peut imaginer par exemple un regroupement pour un petit annuaire téléphonique. Selon la taille de la liste de contacts, on pourrait imaginer un regroupement sur une première lettre sur deux. Ainsi les noms commençant par A et B formeraient le premier groupe, puis C et D, et ainsi de suite.

Un groupe peut être défini pour être conservé avec la première section de détails ou, dans la mesure du possible, en tant que groupe complet. Par défaut, cette option est définie sur Non. Pour les avis de rappel, vous voudrez probablement que le groupe soit organisé de sorte qu'une page distincte soit imprimée pour chaque personne qui doit recevoir une lettre de rappel. Dans un autre menu, vous pouvez choisir que chaque groupe (dans ce cas, chaque nom de lecteur) soit suivi d'un saut de page avant de traiter la valeur suivante.

Si vous avez choisi d'avoir un en-tête de groupe et peut-être un pied de page de groupe, ces éléments apparaîtront sous forme de sections dans le navigateur de rapport sous le nom de champ correspondant Nom_Lecteur. Ici aussi, vous avez la possibilité d'utiliser des fonctions, qui seront alors limitées à ce groupe.

Pour ajouter des champs, utilisez la fonction *Ajouter un champ*, comme avec les formulaires. Cependant, dans ce cas, l'étiquette et le contenu du champ ne sont pas liés. Les deux peuvent être déplacés indépendamment, modifiés en taille et glissés dans différentes sections.

La figure 95 montre la conception du rapport pour l'avis de rappel. Dans l'en-tête de la page se trouve l'en-tête "Bibliothèque Alfred Jarry", inséré comme champ d'étiquette. Ici, vous pouvez également avoir un en-tête avec un logo, car des graphiques peuvent être inclus. Ce niveau est appelé En-tête de page, mais cela n'implique pas qu'il n'y a pas d'espace au-dessus. Cela dépend des paramètres de la page Si une marge supérieure a été définie, elle se trouve au-dessus de l'en-tête de la page.

L'entête Nom_Lecteur est l'en-tête des données groupées et triées. Dans les champs qui doivent contenir des données, les noms des champs de données correspondants sont affichés en gris

clair. Ainsi, par exemple, la vue sous-jacente au rapport comporte un champ nommé Adresse, contenant l'adresse complète du destinataire avec la rue et la ville. Pour le mettre dans un seul champ, il faut des sauts de ligne dans la requête. Vous pouvez utiliser `CHAR (13) || CHAR (10)` pour les créer.

Exemple :

```
SELECT "Civilite"||CHAR(13)||CHAR(10)||"Nom"||' '||"Prenom"||CHAR(13)||
CHAR(10)||"Rue"||' '||"No"||CHAR(13)||CHAR(10)||"CodePostal"||' '||"Ville" AS
"Adresse" FROM "Lecteurs"
```

Le champ = TODAY() représente une fonction intégrée, qui insère la date actuelle dans cette position.

Dans l'entête Nom_Lecteur, en plus de la salutation, nous voyons les en-têtes de colonne pour la vue de tableau suivante. Ces éléments ne doivent apparaître qu'une seule fois, même si plusieurs médias sont répertoriés.

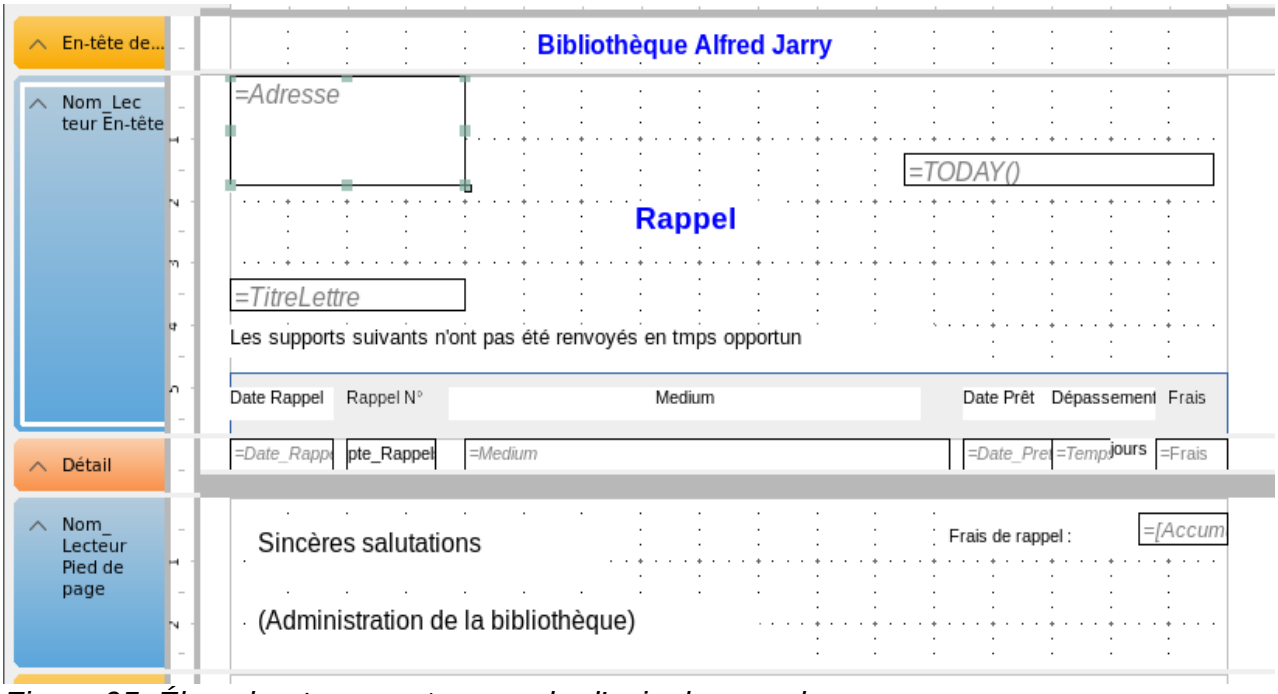


Figure 95: Ébauche de rapport, exemple d'avis de rappel

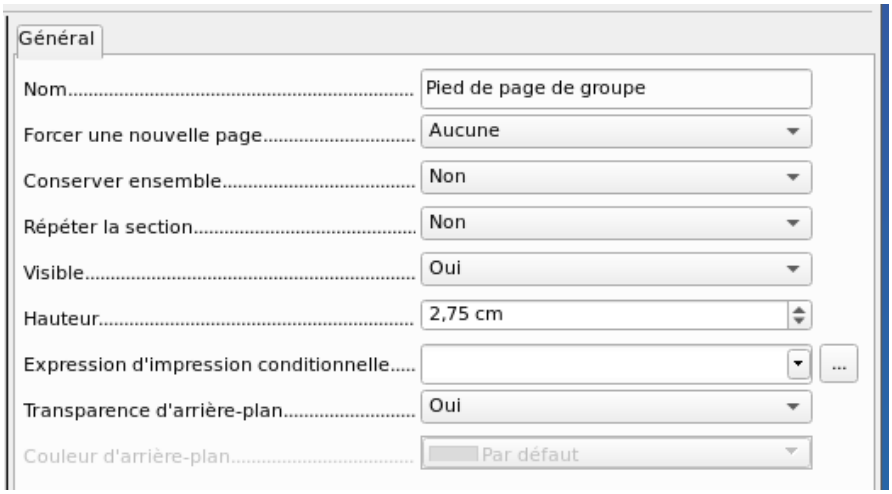
À l'arrière-plan de ces en-têtes de colonne se trouve un rectangle gris, qui sert également de cadre pour les données. La zone Détails est répétée aussi souvent qu'il y a des enregistrements séparés avec les mêmes données Nom_Lecteur. Voici la liste de tous les supports qui n'ont pas été retournés à temps. On peut aussi placer un autre rectangle en arrière-plan pour encadrer le contenu. Ce rectangle serait rempli de blanc plutôt que de gris.

Note

En principe, LibreOffice prévoit la possibilité d'ajouter des lignes horizontales et verticales. Ces lignes ont l'inconvénient d'être interprétées uniquement comme des raies. Elles peuvent être mieux reproduites si des rectangles sont utilisés. Définissez l'arrière-plan du rectangle sur noir et la taille sur, par exemple, 17 cm de large et 0,03 cm de haut. Cela créera une ligne horizontale d'une épaisseur de 0,03 cm et d'une longueur de 17 cm.Malheureusement, cette variante présente également un inconvénient : les éléments graphiques ne peuvent pas être positionnés correctement si la zone s'étend sur plus d'une page.

Le pied de page Nom_Lecteur ferme la lettre avec une formule de salutation et une zone pour la signature. Le pied de page est défini de telle sorte qu'un saut de page supplémentaire se produira après cette zone. De plus, contrairement à la configuration par défaut, il est précisé que cette zone doit être conservée ensemble dans tous les cas. Après tout, il serait plutôt étrange que de nombreux avis de rappel aient la signature sur une page distincte.

La propriété *Conserver ensemble* fait ici référence au saut de page. Si vous souhaitez que le contenu d'un enregistrement soit conservé ensemble indépendamment de la coupure, cela n'est actuellement possible que si l'enregistrement n'est pas lu en tant que Détails mais sert de base à un regroupement. Vous pouvez choisir Conserver ensemble = Oui, mais cela ne fonctionne pas ; la zone Détails est séparée. Vous devez mettre le contenu des détails dans un groupe distinct pour le garder ensemble.



Une fonction intégrée est utilisée pour calculer le total des amendes. Voici à quoi ressemblerait un avis de rappel réel. La zone de détails contient 2 supports que le lecteur a empruntés. Le pied de page du groupe contient l'amende totale due.

Bibliothèque Alfred Jarry

Mr.
Lino Ventura
Rue de la fontaine, 9
F 45100 Orleans

sam. 19 septembre 2020

Rappel

Cher Mr. Ventura,

Les supports suivants n'ont pas été renvoyés en tmps opportun

Date Rappel	Rappel N°	Medium	Date Prêt	Dépassement	Frais
12/03/20	1	8 - La vie mensongère des adultes - par Ferrante, Elena	12/03/20	170 jours	6 €
04/03/20	1	2 - Une brève histoire du temps - par Hawking, Steven W.	04/03/20	171 jours	6 €

Frais de rappel : 12,00 €

Sincères salutations

(Administration de la bibliothèque)

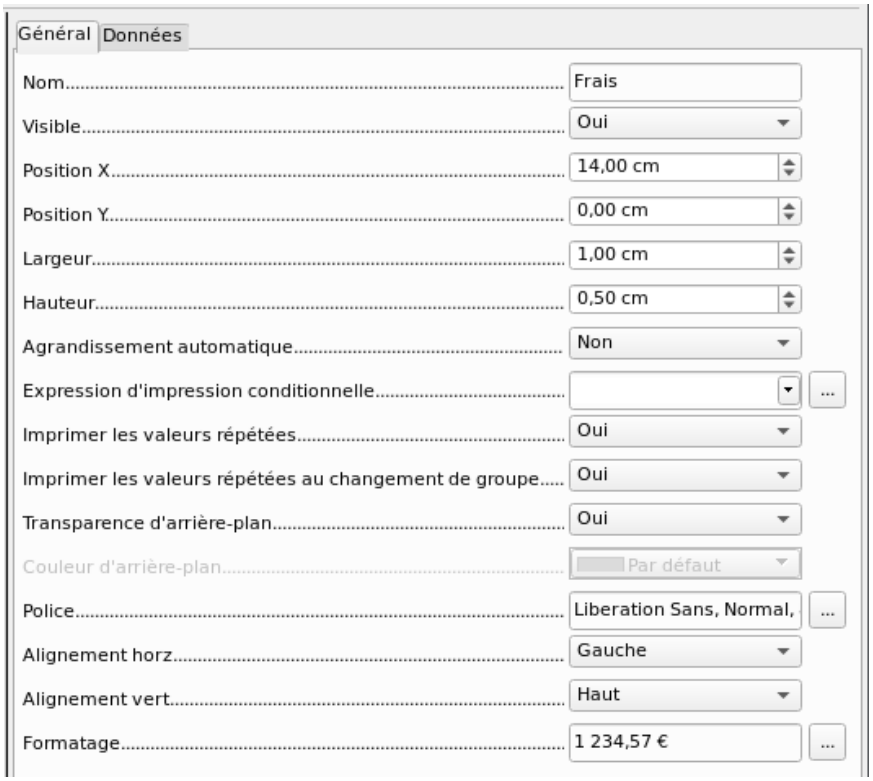
Note

Les rapports pour des enregistrements uniques peuvent également s'étendre sur plus d'une page. La taille du rapport est tout à fait distincte de la taille de la page. Cependant, étirer la zone de détails sur plusieurs pages peut entraîner des coupures défectueuses. Ici, le Générateur de rapports a encore des problèmes pour calculer correctement l'espacement. Si les zones de regroupement et les éléments graphiques sont inclus, cela peut entraîner des tailles imprévisibles pour certaines zones.

Jusqu'à présent, les éléments individuels peuvent être déplacés vers des positions en dehors de la taille d'une seule page uniquement avec la souris et les touches du curseur. Les propriétés des éléments fournissent toujours la même distance maximale à partir du coin supérieur de toute zone située sur la première page.

Propriétés générales des champs

Il n'existe que trois types de champs pour la présentation des données. Outre les champs de texte (qui, contrairement à leur nom, peuvent également contenir des nombres et une mise en forme), il existe également un type de champ pouvant contenir des images de la base de données. Le champ graphique affiche un résumé des données.



Comme pour les formulaires, les champs portent des noms. Par défaut, le nom est celui du champ de base de données (Table, requête ou vue) sous-jacent.

Un champ peut être défini pour être invisible. Cela peut sembler un peu inutile dans le cas des champs mais est utile pour les en-têtes et pieds de page de groupe, qui peuvent être nécessaires pour exécuter d'autres fonctions du groupement sans contenir quoi que ce soit à afficher.

Si l'option Imprimer les valeurs répétées est désactivée, l'affichage du champ est inhibé lorsqu'un champ avec le même contenu est chargé directement avant. Cela fonctionne correctement uniquement pour les champs de données contenant du texte. Les champs numériques ou les champs de date ignorent l'instruction de désactivation, les champs d'étiquette sont complètement estompés lorsqu'ils sont désactivés, même s'ils n'apparaissent qu'une seule fois.

Dans le Générateur de rapports, l'affichage de certains contenus peut être inhibé à l'aide de l'expression d'impression conditionnelle ou la valeur du champ peut être utilisée comme base pour la mise en forme du texte et de l'arrière-plan. Pour plus d'informations sur les expressions conditionnelles, reportez-vous à la section « Impression conditionnelle » à la page 313.

Le paramètre de la molette de la souris n'a aucun effet, car les champs de rapport ne sont pas modifiables. Il semble qu'il s'agisse d'un reste de la boîte de dialogue de définition de formulaire.

La fonction Imprimer lors du changement de groupe n'a pas non plus pu être reproduite dans les rapports.

Si l'arrière-plan n'est pas défini comme transparent, une couleur d'arrière-plan peut être définie pour chaque champ.

Les autres entrées concernent le contenu interne du champ en question. Cela couvre la police (pour la couleur de la police, l'épaisseur de la police, etc., voir Figure 96), l'alignement du texte dans le champ et la mise en forme avec la boîte de dialogue **Police> Propriétés des caractères** correspondante (voir Figure 97).

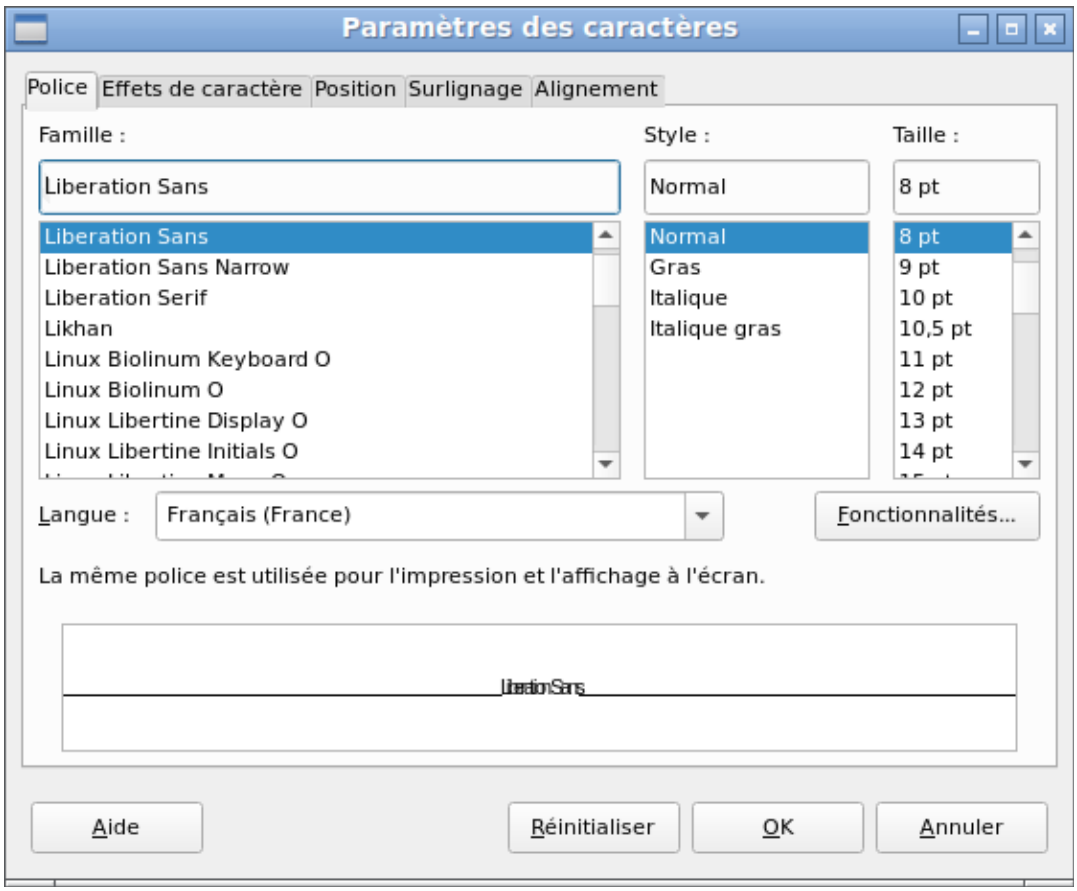


Figure 96: Polices : Paramètres des caractères

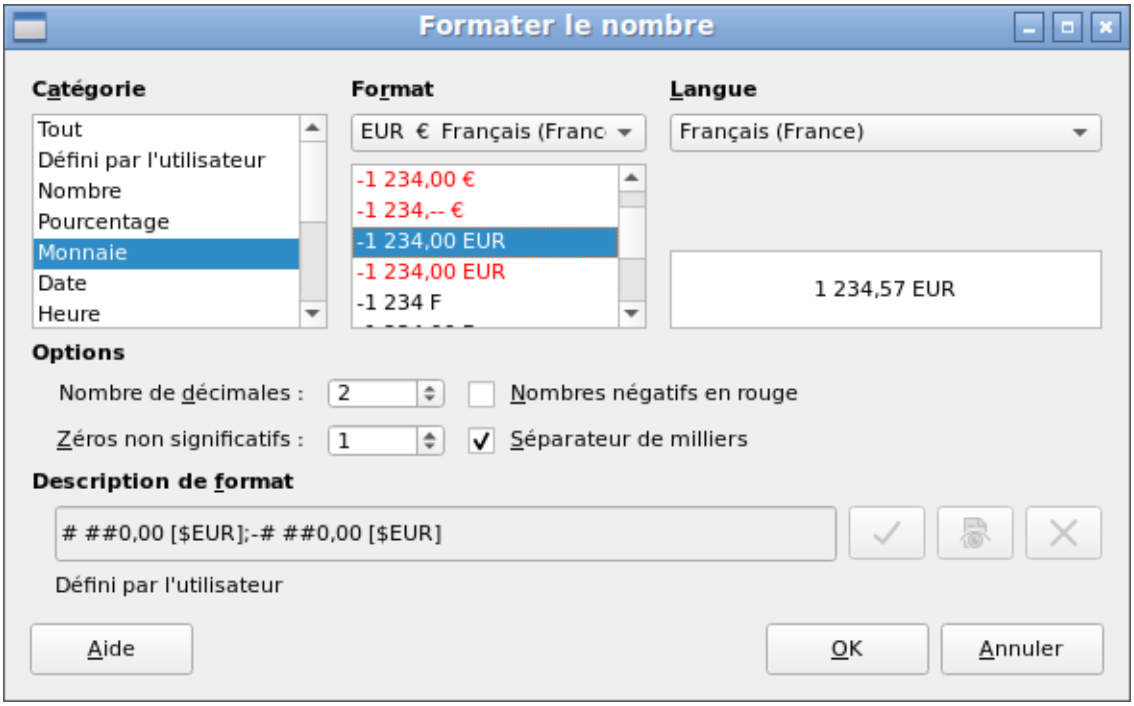
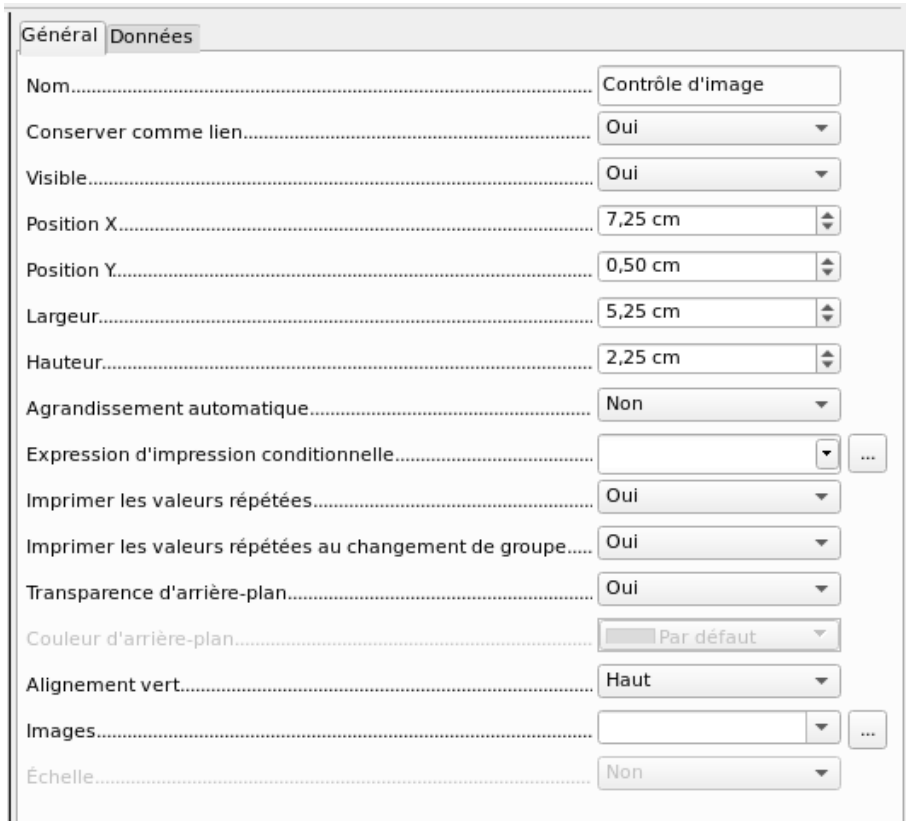


Figure 97: Formatage des nombres

Propriétés spéciales des contrôles graphiques



Un contrôle graphique peut contenir des graphiques à la fois à l'intérieur et à l'extérieur de la base de données. Malheureusement, il n'est actuellement pas possible de stocker un graphisme tel qu'un logo de manière permanente dans Base. Par conséquent, il est essentiel que le graphique soit disponible dans le chemin de recherche, même lorsque le choix d'incorporer plutôt que de lier des images vous est proposé et que le premier champ Configurer comme lien peut être défini (littéralement fermé) vers une fonctionnalité prévue correspondante. C'est l'une des nombreuses fonctions qui sont prévues pour Base et qui se trouvent dans l'interface graphique mais qui n'ont pas encore été implémentées – les boutons et les cases à cocher n'ont donc aucun effet.

Alternativement, bien sûr, un graphique peut être stocké dans la base de données elle-même et devient ainsi disponible en interne. Mais dans ce cas, il doit être accessible via l'un des champs de la requête sous-jacente au rapport.

Pour prendre un graphique externe, utilisez le bouton de sélection à côté du libellé Images pour le charger. Pour charger un champ de base de données graphique, spécifiez le champ sous l'onglet Données.

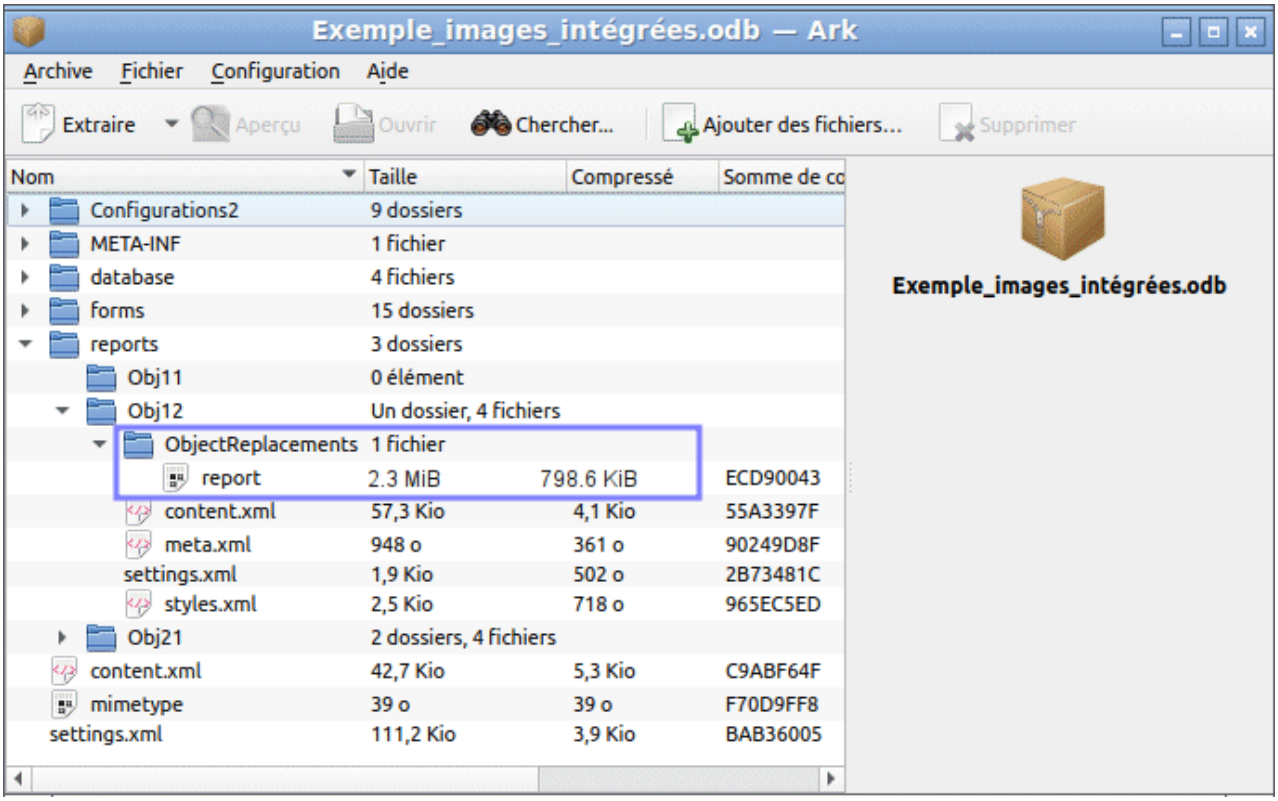
Le paramètre d'alignement vertical ne semble pas avoir d'effet pendant la phase de conception. Cependant, lorsque vous appelez le rapport, le graphique apparaît à la bonne position.

Pour la propriété de mise à l'échelle, vous pouvez sélectionner *Non*, *Conserver le ratio* (hauteur/largeur) ou *Ajuster à la taille*. Cela correspond aux paramètres d'un formulaire :

- **Non :** L'image n'est pas adaptée au contrôle. Si elle est trop grande, une version recadrée est affichée. L'image d'origine n'est pas affectée par cela ;
- **Conserver le ratio :** L'image est adaptée au contrôle mais n'est pas déformée ;
- **Ajuster à la taille :** L'image est adaptée à la commande et dans certains cas peut être déformée.

Lorsque des rapports contenant des images sont modifiés, il peut arriver que la base de données s'agrandisse considérablement. Dans le fichier *.odb, Base place dans le répertoire du rapport un

dossier ObjectReplacements, pour des raisons qui ne sont pas entièrement comprises. Ce dossier contient un fichier « report » responsable de l'agrandissement.



Si la base de données est ouverte dans un programme d'archivage, ce dossier avec son contenu est visible dans le sous-répertoire des rapports. Vous pouvez utiliser en toute sécurité le programme d'archivage pour rechercher et supprimer le dossier (àa condition que le base ne soit pas ouverte en même temps).

Note

Si les rapports ne doivent pas être modifiés à plusieurs reprises, la suppression du dossier ObjectReplacements une fois suffit. La taille de ce dossier peut augmenter très rapidement. Cela dépend du nombre et de la taille des fichiers inclus. Un test a montré qu'un seul fichier jpg de 2,8 Mo agrandissait le fichier *.odb de 11 Mo ! Voir le bogue 80320.

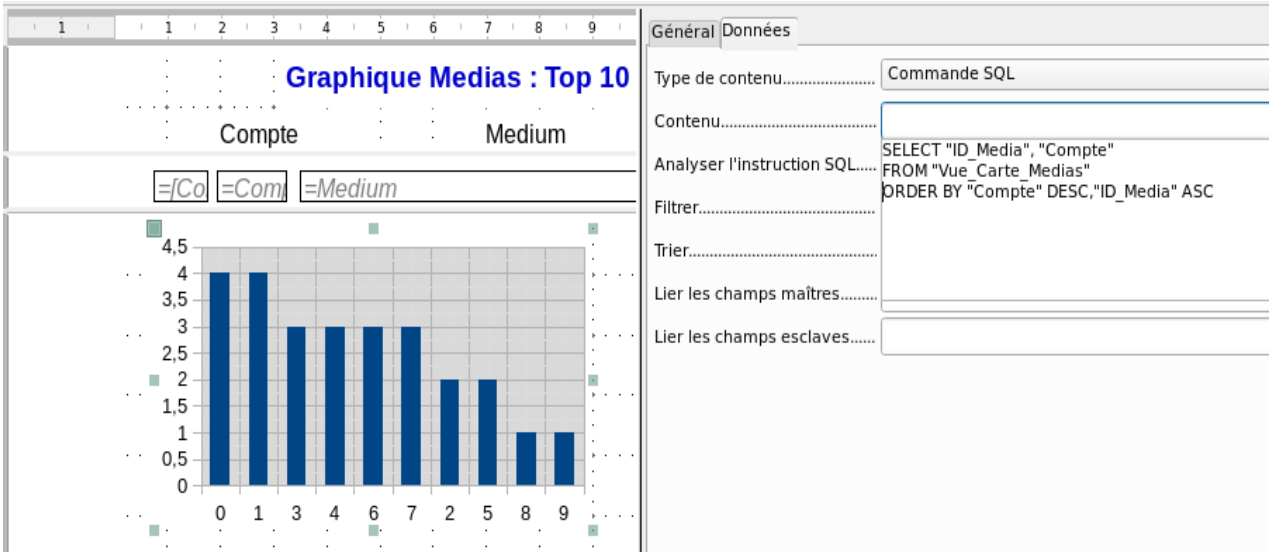
Incorporer des diagrammes dans le rapport

Vous pouvez insérer des graphiques dans un rapport en utilisant le contrôle correspondant ou avec **Insertion> Contrôles de rapport> Diagramme** (ou le bouton équivalent). Un graphique est le seul moyen de reproduire des données qui ne se trouvent pas dans la source de données spécifiée pour le rapport. Un graphique peut donc être considéré comme une sorte de sous-rapport, mais aussi comme un élément autonome du rapport.

Vous devez dessiner l'emplacement du graphique à l'aide de la souris. Dans les propriétés générales, en plus des champs familiers, vous pouvez choisir un type de graphique (voir les types

correspondants dans Calc). De plus, vous pouvez définir un nombre maximum d'enregistrements pour l'aperçu, ce qui donnera une impression de l'apparence finale du graphique.

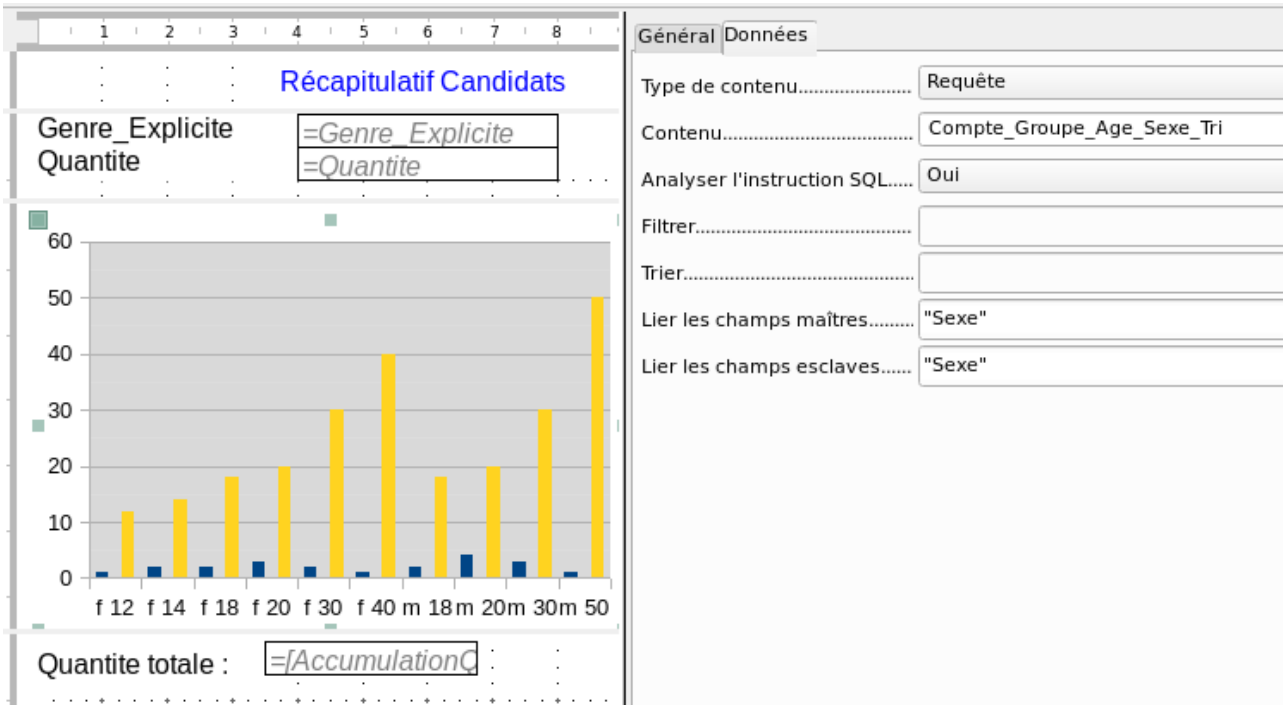
Les graphiques peuvent être formatés de la même manière que dans Calc (double-cliquez sur le graphique). Pour plus d'informations, consultez la description dans le Guide LibreOffice Calc.



Le graphique est lié dans la section Données avec les champs de données nécessaires. Ici, dans un exemple de liste Media Top 10, le graphique montre la fréquence à laquelle des médias particuliers sont empruntés. L'éditeur de requête est utilisé pour créer une commande SQL appropriée, comme vous le feriez pour une zone de liste dans un formulaire.

La première colonne de la requête sera utilisée pour fournir les étiquettes des barres verticales dans le graphique, tandis que la deuxième colonne donne le nombre total de transactions de prêt, indiqué dans la hauteur des barres.

Dans l'exemple ci-dessus, le graphique montre très peu de prêts, car seuls des prêts de test limités ont été effectués avant l'émission de la commande SQL.



Dans les propriétés de données du graphique, une **Requête** a été spécifiée. Ce graphique de la base de données Exemple_sport.odb[5] montre quelque chose de spécial en plus des bases de la création de graphiques dans les rapports : l'aperçu du graphique montre plus de colonnes que prévu. Cela résulte du contenu de la requête, qui crée des colonnes supplémentaires qui n'apparaîtront pas toutes dans le graphique lui-même.

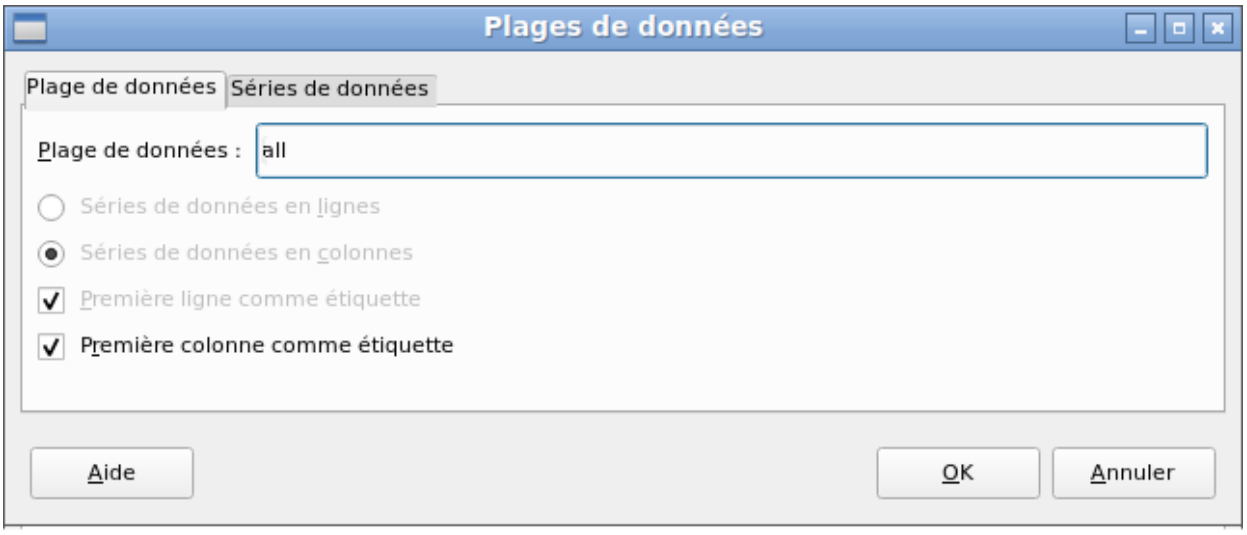
Un filtre et un tri avec les outils internes du Générateur de rapports ne sont pas nécessaires, car cela a déjà été fait dans la mesure du possible dans la requête.

Conseil

Fondamentalement, vous souhaitez supprimer autant de tâches que possible de la création de vos rapports. Tout ce qui peut être géré au début du processus à l'aide de requêtes n'a pas besoin d'être refait pendant le processus relativement lent de création du rapport lui-même.

Comme pour les formulaires principaux et les sous-formulaires, certains champs sont désormais liés entre eux. Dans le rapport proprement dit, les groupes d'âge des participants masculins et féminins aux camps sportifs sont indiqués sous forme de tableau. Ils sont regroupés par sexe. Dans chaque groupe, il y a maintenant un graphique séparé. Pour garantir que le graphique n'utilise que des données pour le sexe correct, les deux champs appelés « Sexe » – dans le rapport et dans le graphique – sont liés.

L'axe X du graphique est automatiquement lié à la première colonne de la table source des données. S'il y a plus de deux colonnes dans le tableau, des colonnes supplémentaires sont automatiquement placées dans le graphique. Vous pouvez accéder à d'autres paramètres pour le graphique en sélectionnant l'ensemble du graphique avec un double-clic. Cliquez sur le bouton droit de la souris pour ouvrir un menu contextuel au-dessus du diagramme, son contenu dépendant de l'élément sélectionné. Il contient les paramètres possibles pour les plages de données :



Séries de données en colonnes est grisé et ne peut donc pas être modifié. Vous ne pouvez pas non plus modifier la case à cocher *Première ligne comme étiquette*. Les paramètres restants de l'onglet Plage de données ne doivent pas être modifiés, car il y a ici plus de possibilités que le Générateur de rapports ne peut en gérer.

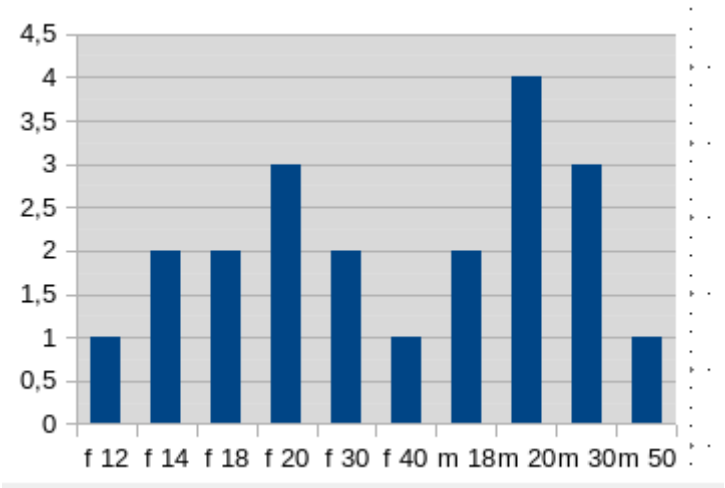
5 Cette base de données est incluse dans l'archive d'exemples de bases de données de ce manuel.

L’onglet Série de données, en revanche, masque quelques paramètres qui peuvent modifier considérablement l’apparence par défaut de votre graphique. Il affiche toutes les séries de données disponibles pour la première colonne de la requête. Tout ce que vous ne souhaitez pas afficher peut être supprimé à ce stade.

Dans l’exemple ci-dessus, il y avait trop de colonnes visibles dans le graphique.

Cela demande une amélioration ! Ni sexe ni Tri_Groupe_Age, dont les noms proviennent de la requête sous-jacente, ne sont d’aucune utilité ici. La série Sexe permet de lier le graphique à la source de données du rapport et ne peut en aucun cas être représentée numériquement. Et Tri_Groupe_Age ne sert qu’à assurer un tri correct des valeurs dans la requête, car sinon un code comme « m8 » viendrait immédiatement avant « m80 » au lieu du début (le tri dans les champs de texte conduit souvent à des résultats indésirables similaires).

Lorsque toutes les lignes ont été supprimées sauf *Quantite*, l’aperçu du graphique ressemble à ceci :



Cet aperçu montre 10 colonnes – les dix premières colonnes de la requête. Dans l’exécution réelle, seules les colonnes appartenant au sexe correct seront affichées : les colonnes « m » pour les hommes et les colonnes « f » pour les femmes.

L’axe Y montre toujours une caractéristique malheureuse. Après tout, il n’y a pas une demi-personne ! Encore une fois, cela pourrait être amélioré. Cependant, dans une exécution

automatisée avec ce paramètre, ces nombres seront remplacés par des entiers si la plage de valeurs ne s'arrête pas, comme dans l'exemple ci-dessus, à « 4 ». Si ce traitement automatique devait être désactivé, une amélioration manuelle serait en effet nécessaire.

Tous les autres paramètres sont similaires à ceux utilisés par Calc pour la création de graphiques.

Propriétés des données des champs

Dans la boîte de dialogue des propriétés, l'onglet Données affiche par défaut uniquement le champ de base de données à partir duquel les données de ce champ de rapport seront lues. Cependant, en plus des types de champ Champ et Formule, les types Fonction, Compteur et Fonction définie par l'utilisateur sont disponibles.

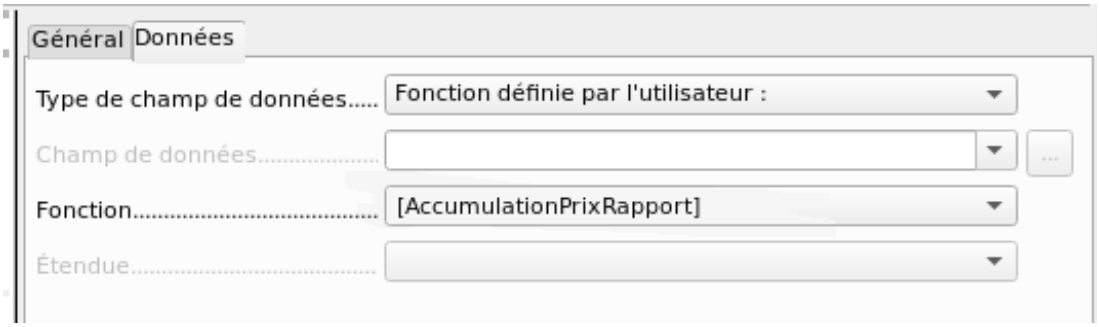
Vous pouvez sélectionner à l'avance les fonctions Accumulation, Minimum et Maximum. Elles s'appliqueront soit au groupe actuel, soit à l'ensemble du rapport. Ces fonctions peuvent entraîner des problèmes si un champ est vide (NULL). Dans ces champs, s'ils ont été formatés sous forme de nombres, NaN apparaît ; c'est-à-dire qu'il n'y a pas de valeur numérique présente. Pour les champs vides, aucun calcul n'est effectué et le résultat est toujours 0.

Ces champs peuvent être reformatés pour afficher une valeur de 0 à l'aide de la formule suivante dans la zone Données de la vue.

IF([Champ_Numérique] ;[Champ_Numérique] ; 0)

Cette fonction calcule avec la valeur réelle d'un champ qui n'a pas de valeur. Il semblerait plus simple de formuler la requête sous-jacente pour le rapport afin que 0 soit donné au lieu de NULL pour les champs numériques.

Le compteur compte uniquement les enregistrements qui se produiront dans le groupe ou dans le rapport dans son ensemble. Si le compteur est inséré dans la zone Détails, chaque enregistrement recevra un numéro courant. La numérotation s'appliquera uniquement aux enregistrements du groupe ou de l'ensemble du rapport.



Enfin, une fonction détaillée définie par l'utilisateur est disponible. Il peut arriver que le Générateur de rapports choisisse lui-même cette variante si un calcul a été demandé, mais pour une raison quelconque, il ne peut pas interpréter correctement la source de données.

Fonctions du générateur de rapports

Le Générateur de rapports fournit une variété de fonctions, à la fois pour afficher les données et pour définir les conditions. Si celles-ci ne sont pas suffisantes, des fonctions définies par l'utilisateur peuvent être créées à l'aide d'étapes de calcul simples, qui sont particulièrement utiles dans les pieds de page de groupe et les résumés.

Saisir des formules

Le Générateur de rapports est basé sur le Générateur de rapports Pentaho. Une petite partie de sa documentation est àhttp://wiki.pentaho.com/display/Reporting/9.+Report+Designer+Formula+Expressions.

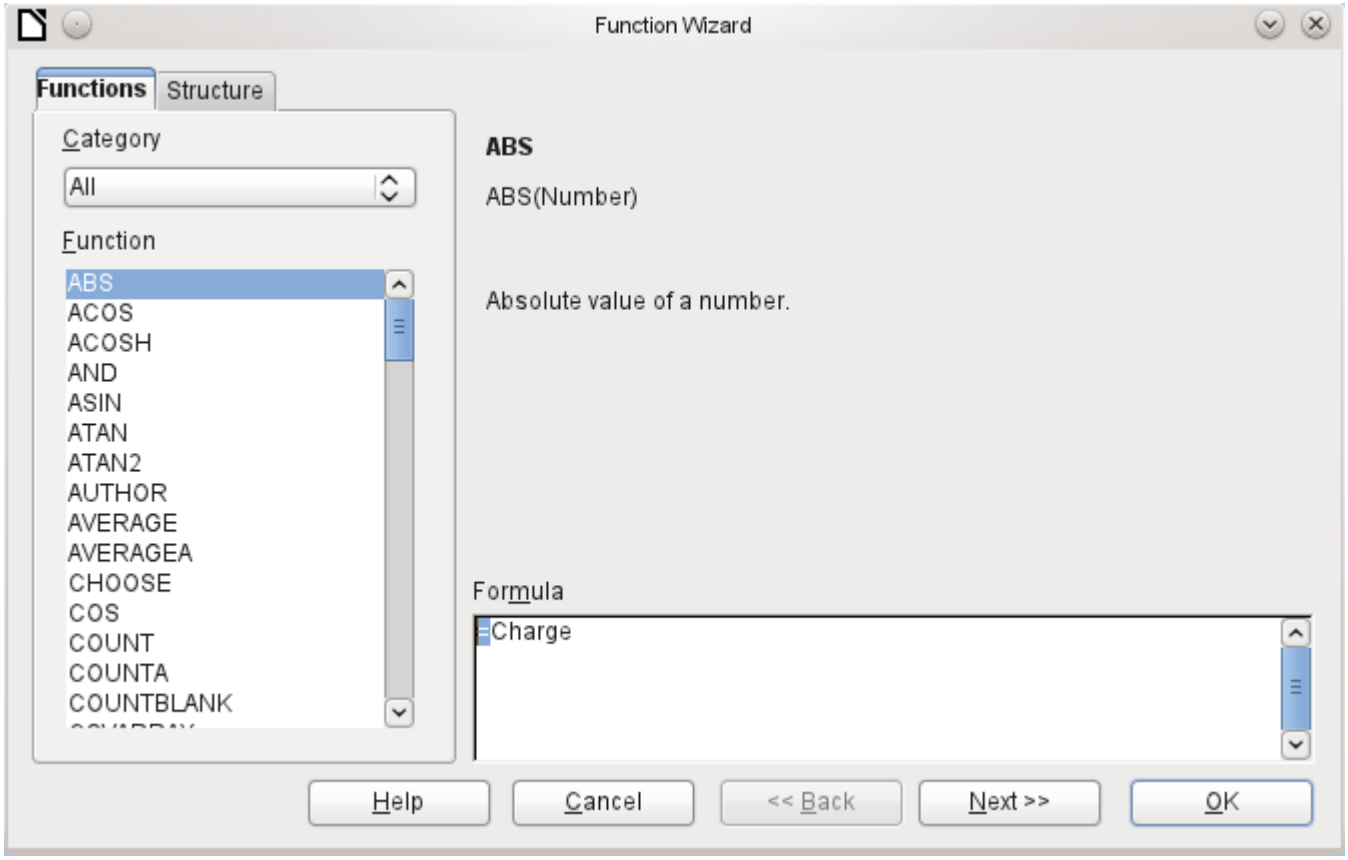
Une autre source est les spécifications du standard OpenFormula : http://www.oasis-open.org/committees/download.php/16826/openformula-spec-20060221.html

Principes de base :

Les formules commencent par un signe égal.	`=`
Les références aux champs de données sont placées entre crochets.	`[Nom_Champ]`
Si les champs de données contiennent des caractères spéciaux (y compris des espaces), le nom du champ doit également être placé entre guillemets.	`["Ce nom de champ doit être entre guillemets"]`
La saisie de texte doit toujours être entre guillemets.	`"Entrée texte"`
Les opérateurs suivants sont autorisés.	+, –, * (Multiplication), / (Division), % (divise le nombre précédent par 100), ^ (Élever-à la puissance du nombre suivant), & (concaténer du texte),
Les comparaisons suivantes sont possibles.	`=, <>, <, <=, >, >=`
Les parenthèses sont autorisées.	`()`
Message d'erreur par défaut.	`NA` (Indisponible)
Message d'erreur pour un champ vide défini comme un nombre.	`NaN` (« not a number »?)

Toutes les entrées de formule s'appliquent uniquement à l'enregistrement actuel. Les relations avec les enregistrements précédents ou suivants ne sont donc pas possibles.

À côté du *Champ de données* se trouve un bouton avec trois points chaque fois qu'une formule peut être saisie. Ce bouton lance l'assistant de fonction.



Cependant, il y a beaucoup moins de fonctions que dans Calc. De nombreuses fonctions ont des équivalences dans Calc. Là, l'assistant calcule directement le résultat de la fonction.

L'assistant de fonction ne fonctionne pas toujours parfaitement. Par exemple, les entrées de texte ne sont pas prises avec des guillemets doubles. Cependant, seules les entrées avec des guillemets doubles sont traitées lors de l'exécution de la fonction.

Les fonctions suivantes sont disponibles :

Fonction	***Description***
Fonctions de date et d'heure	
DATE	Produit une date valide à partir de valeurs numériques pour l'année, le mois et le jour.
DATEDIF (DAY | MONTH | YEAR)	Renvoie le nombre total d'années, de mois ou de jours entre deux valeurs de date.
DATEVALUE	Convertist une entrée de date américaine sous forme de texte (entre guillemets) en une valeur de date. La variante américaine produite peut ensuite être reformatée.
DAY	Renvoie le jour du mois pour une date donnée. DAY ([Champ Date])
DAYS	Renvoie le nombre de jours entre deux dates.
HOUR	Renvoie les heures d'une heure donnée au format 24 heures. HOUR ([ChampHorodatage]) calcule les heures dans le champ.

MINUTE	Renvoie les minutes d'une date au format numérique interne MINUTE ([Champ Date]) calcule la partie minutes du temps.
MONTH	Renvoie le mois d'une date saisie sous forme de nombre. MOIS ([Champ Date])
NOW	Renvoie la date et l'heure actuelles.
SECOND	Renvoie les secondes d'une date au format numérique interne SECOND (NOW ()) montre la partie des secondes au moment de l'exécution de la commande.
TIME	Affiche l'heure actuelle.
TIMEVALUE	Convertis une entrée de texte pour une durée en une valeur de temps pour les calculs.
TODAY	Affiche la date actuelle.
WEEKDAY	Renvoie le jour de la semaine sous forme de nombre. Le jour numéro 1 est le dimanche.
YEAR	Renvoie la partie année d'une entrée de date.
Fonctions logiques	
AND	Renvoie TRUE(vrai) lorsque tous ses arguments sont TRUE.
FALSE	Définis la valeur logique comme FALSE (faux).
IF	Si une condition est TRUE, alors cette valeur, sinon une autre valeur.
IFNA	Renvoie la valeur si la cellule ne contient pas la valeur d'erreur #N/A (valeur non disponible) ou la valeur alternative si c'est le cas.
NOT	Inverse la valeur logique de l'argument.
OR	Renvoie TRUE lorsque l'une de ses conditions est TRUE.
TRUE	Définis la valeur logique comme TRUE.
XOR	Renvoie TRUE lorsqu'une seule des valeurs liées est TRUE.
Fonctions d'arrondi	
INT	Arrondis à l'entier précédent.
Fonctions mathématiques	
ABS	Renvoie la valeur absolue (non négative) d'un nombre.
ACOS	Calcule l'arc cosinus d'un nombre. – arguments entre -1 et 1.
ACOSH	Calcule l'aire cosinus (cosinus hyperbolique inverse) – argument> = 1.
ASIN	Calcule l'arc sinus d'un nombre – argument entre -1 et 1.
ATAN	Calcule l'arc tangente d'un nombre.
ATAN2	Calcule l'arc tangente d'une coordonnée x et d'une coordonnée y.

AVERAGE	Donne la moyenne des valeurs saisies.
AVERAGEA	Donne la moyenne des valeurs entrées. Le texte est traité comme valeur nulle.
COS	L'argument est l'angle en radians dont le cosinus doit être calculé.
EVEN	Arrondis un nombre positif vers le haut ou un nombre négatif vers le bas au prochain entier pair.
EXP	Calcule la fonction exponentielle (Base "e").
LN	Calcule le logarithme naturel d'un nombre.
LibreOfficeG10	Calcule le logarithme d'un nombre (Base '10').
MAX	Renvoie le maximum d'une série de nombres.
MAXA	Renvoie la valeur maximale dans une ligne. Tout texte vaut zéro.
MIN	Renvoie la plus petite d'une série de valeurs.
MINA	Renvoie la valeur minimale dans une ligne. Tout texte vaut zéro.
MOD	Renvoie le reste d'une division lorsque vous entrez le dividende et le diviseur.
ODD	Arrondis un nombre positif vers le haut ou un nombre négatif jusqu'à l'entier impair suivant.
PI	Donne la valeur du nombre "π".
POWER	Élève la base à la puissance de l'exposant.
SIN	Calcule le sinus d'un nombre.
SQRT	Calcule la racine carrée d'un nombre.
SUM	Somme une liste de valeurs numériques.
SUMA	Somme une liste de valeurs numériques. Les champs Texte et Oui/Non sont autorisés. Malheureusement, cette fonction se termine (toujours) par un message d'erreur.
VAR	Calcule la variance à partir d'un échantillon.

Fonctions Texte	
EXACT	Indique si deux chaînes de texte sont exactement égales.
FIND	Donne le décalage d'une chaîne de texte dans une autre chaîne.
LEFT	Le nombre spécifié de caractères d'une chaîne de texte est extrait à partir de la gauche.
LEN	Donne le nombre de caractères dans une chaîne de texte.
LOWER	Convertis le texte en minuscules.
MESSAGE	Formate la valeur dans le format de sortie donné.

MID	Le nombre spécifié de caractères d'une chaîne de texte est extrait à partir d'une position de caractère spécifiée.
REPLACE	Remplace une sous-chaîne par une autre sous-chaîne. La position de départ et la longueur de la sous-chaîne à remplacer doivent être indiquées.
REPT	Répète le texte un nombre spécifié de fois.
RIGHT	Le nombre spécifié de caractères d'une chaîne de texte est extrait à partir de la droite.
SUBSTITUTE	Remplace des parties spécifiques d'une chaîne de texte donnée par un nouveau texte. De plus, vous pouvez spécifier quelle occurrence de la chaîne cible parmi plusieurs doit être remplacée.
T	Renvoie le texte, ou une chaîne de texte vide si la valeur n'est pas du texte (par exemple un nombre).
TEXT	Conversion de nombres ou d'heures en texte.
TRIM	Supprime les espaces initiaux et les espaces terminaux et réduit plusieurs espaces en un seul espace.
UNICHAR	Convertit un nombre décimal Unicode en caractère Unicode. Par exemple, 196 devient « Ä » (« Ä » a la valeur hexadécimale 00C4).
UNICODE	Convertis un caractère Unicode en un nombre décimal Unicode. « Ä » devient 196.
UPPER	Renvoie une chaîne de texte en majuscules.
URLENCODE	Convertis un texte donné en un texte conforme à une URL valide. Si aucune norme particulière n'est spécifiée, l'ISO-8859-1 est suivie.
Fonctionsd'information	
CHOOSE	Le premier argument est un index, suivi d'une liste de valeurs. La valeur représentée par l'index est renvoyée. `CHOOSE(2 ;"Pomme";"Poire";"Banane")` `renvoie Poire.` `CHOOSE([Champ_Age] ;"Lait";"Soda";"Bière")` renvoie une boisson possible pour le « Champ_Age » donné.
COUNT	Seuls les champs contenant un nombre ou une date sont comptés. `COUNT([duree] ;[nombre])` renvoie 2, si les deux champs contiennent une valeur (non NULL) ou bien 1 ou 0.
COUNTA	Comprends également des champs contenant du texte. Même NULL est compté, ainsi que les champs booléens.
COUNTBLANK	Compte les champs vides dans une région.
HASCHANGED	Vérifie si la colonne nommée a changé. Cependant, aucune information sur la colonne n'est fournie.
INDEX	Fonctionne avec les régions
ISBLANK	Teste si le champ est NULL (vide).
ISERR	Renvoie TRUE si l'entrée contient une erreur autre que NA. `ISERR(1/0)` renvoie TRUE

ISERROR	Comme ISERR, sauf que NA renvoie également TRUE.
ISEVEN	Teste si un nombre est pair.
ISLOGICAL (ISTLOG)	Teste s'il s'agit d'une valeur Oui/Non. `ISLOGICAL(TRUE())` ou `ISLOGICAL(FALSE())` rendent TRUE, les valeurs textuelles telles que `ISLOGICAL("TRUE")` rendent FALSE.
ISNA	Teste si l'expression est une erreur de type NA.
ISNONTEXT	Teste si la valeur n'est pas du texte.
ISNUMBER	Teste si quelque chose est numérique. `ISNUMBER(1)` donne TRUE, `ISNUMBER("1")` donne FALSE
ISODD	Teste si un nombre est un nombre impair.
ISREF	Teste si quelque chose est une référence de champ. `ISREF([Nom_Champ])` donne TRUE, `ISREF(1)` donne FALSE.
ISTEXT	Teste si le contenu du champ est du texte.
NA (NV)	Renvoie le code d'erreur NA.
VALUE	Renvoie la valeur numérique d'une chaîne
Défini par l'utilisateur	
CSVARRAY	Convertis le texte CSV en un tableau.
CSVTEXT	Convertis un tableau en texte CSV.
NORMALIZEARRAY	Convertis un tableau, base de données, ou séquence en tableau à une dimension.
NULL	Renvoie NULL.
PARSEDATE	Convertis le texte en date. Utilise le SimpleDateFormat. Nécessite une date sous forme de texte comme décrit dans ce format de date. Exemple : `PARSEDATE("9.10.2012";"dd. MM.yyyy")` renvoie le nombre utilisable en interne pour la date.
Informations sur le document	
AUTHOR	Auteur, tel que lu dans **Outils> Options> LibreOffice> Données d'identité**. Il ne s'agit donc pas de l'auteur réel mais de l'utilisateur actuel de la base de données.
TITLE	Renvoie le titre du rapport.

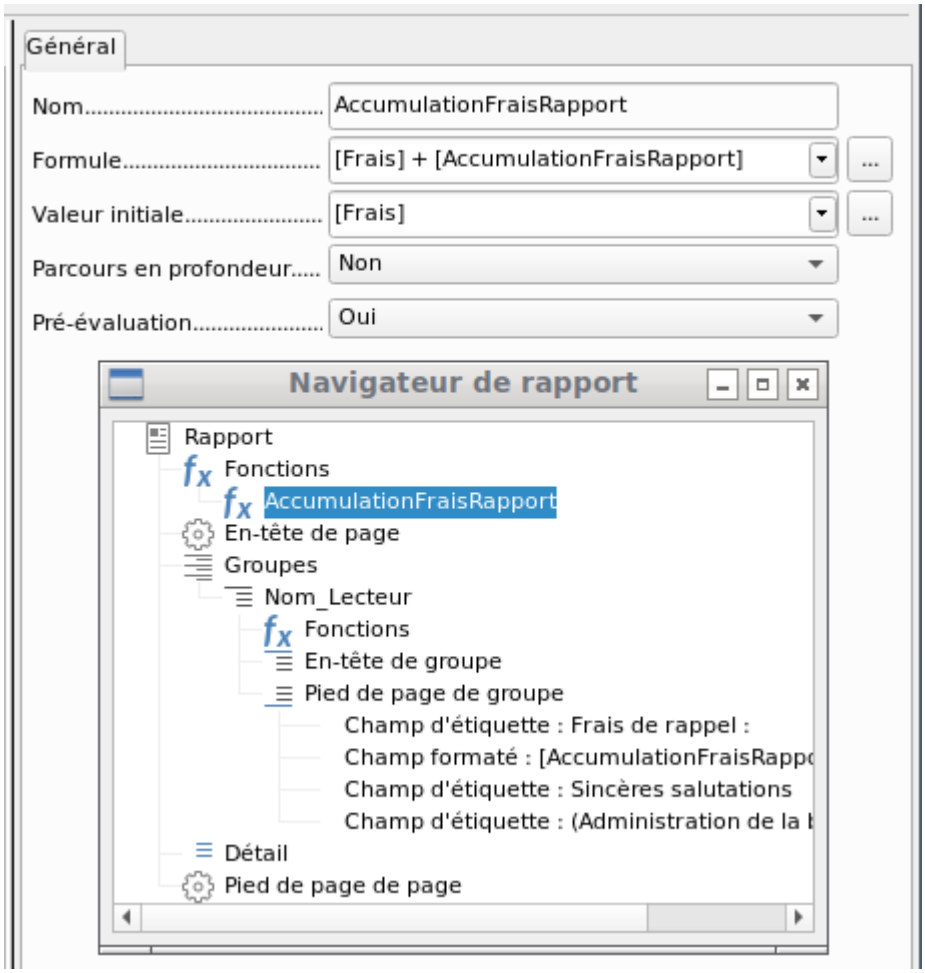
Fonctions définies par l'utilisateur

Vous pouvez utiliser des fonctions définies par l'utilisateur pour renvoyer des résultats intermédiaires spécifiques pour un groupe d'enregistrements.

Dans l'exemple ci-dessous, une fonction de ce type a été utilisée pour calculer les amendes dans la zone Nom_Lecteur Pied de page.

Dans le navigateur de rapports, la fonction est affichée sous le nom du rapport. En cliquant avec le bouton droit de la souris sur l'entête Fonctions, vous pouvez définir des fonctions supplémentaires par leur nom.

Les propriétés de la fonction *AccumulationFraisRapport* sont indiquées ci-dessous. La formule ajoute le champ Frais à la valeur déjà stockée dans la fonction elle-même. La valeur initiale est la valeur du champ Frais lors du premier parcours du groupe. Cette valeur est stockée dans la fonction sous le nom de la fonction et est réutilisée dans la formule, jusqu'à ce que la boucle soit terminée et que le pied de page du groupe soit écrit.



Le parcours en profondeur semble n'avoir aucune fonction pour le moment, à moins que les graphiques ne soient traités ici comme des sous-rapports.

Si la pré-évaluation est activée pour la fonction, le résultat peut également être placé dans l'en-tête de groupe. Sinon, l'en-tête de groupe contient uniquement la valeur correspondante du premier champ du groupe.

Les fonctions définies par l'utilisateur peuvent également faire référence à d'autres fonctions définies par l'utilisateur. Dans ce cas, vous devez vous assurer que les fonctions utilisées ont déjà été créées. Le pré-calcul dans les fonctions faisant référence à d'autres fonctions doit être exclu.

[`Somme_Scores_Classe`] / ([`Effectif_Classe`]+1)

Fait référence au groupe Classe. Le contenu du champ Score est additionné et la somme de tous les enregistrements est renvoyée. La somme des Scores est divisée par la somme des enregistrements. Pour obtenir le nombre correct, 1 doit être ajouté comme indiqué avec [Effectif_Classe]. Cela donnera alors les notes moyennes.

Entrée de formule pour un champ

À l'aide de Données> Champ de Données, vous pouvez saisir des formules qui n'affectent qu'un seul champ dans la zone Détails.

IF([Champ_Booleen];"Oui";"Non")

définit les valeurs autorisées sur « Oui » ou « Non » au lieu de TRUE et FALSE.

Il peut arriver que dans un champ avec une entrée de formule, un seul nombre apparaisse. Dans les champs de texte, il s'agit d'un zéro. Pour résoudre ce problème, vous devez changer le champ de texte de la valeur par défaut « Nombre » à « Texte ».

Impression conditionnelle

Les propriétés générales des en-têtes de groupe, des pieds de page de groupe et des champs incluent un champ Expression d'impression conditionnelle. Les formules écrites dans ce champ influencent le contenu d'un champ ou l'affichage d'une région entière. Ici aussi, vous pouvez utiliser l'assistant de fonction.

[Nom_Champ]="TRUE"

provoque l'affichage du contenu du champ nommé uniquement s'il est vrai[6].

De nombreuses formes d'affichage conditionnel ne sont pas entièrement déterminées par les propriétés spécifiées. Par exemple, si une ligne de séparation graphique doit être insérée après la dixième place d'une liste de résultats de concours, vous ne pouvez pas simplement utiliser la commande d'affichage conditionnel suivante pour le graphique :

[Place]=10

Cette commande ne fonctionne pas. Au lieu de cela, le graphique continuera à apparaître dans la section Détails après chaque enregistrement suivant. Voir le bogue 73707.

Si vous voulez juste une forme rectangulaire superposée à cet emplacement, cela peut être fait à l'aide d'un contrôle graphique, auquel on peut donner l'adresse d'un fichier graphique (monochrome). Dans les propriétés générales de ce contrôle, **Agrandissement automatique** doit être sélectionné. Ensuite, le graphique s'adaptera au formulaire et la condition sera remplie.

Il est plus sûr de lier l'affichage conditionnel à un pied de page de groupe plutôt qu'au graphique, si cela n'est pas nécessaire autrement. La ligne est positionnée dans le pied de page du groupe. Ensuite, la ligne apparaît après la 10e place, lorsqu'elle est formulée comme ci-dessus. Étant donné que la condition est associée au pied de page du groupe, la ligne apparaît uniquement si elle est vraiment nécessaire. Sinon, l'affichage conditionnel peut entraîner l'apparition d'un blanc à la place de la ligne.

Ici, vous pouvez également utiliser les formes fournies par **Insertion> Formes> Formes de base**, par exemple la ligne horizontale pour se fondre avec le pied de page du groupe.

6 Voir aussi la base de données Exemple_Rapport_Affichage_Images_Conditionnel, qui est incluse dans les exemples de bases de données pour ce livre.

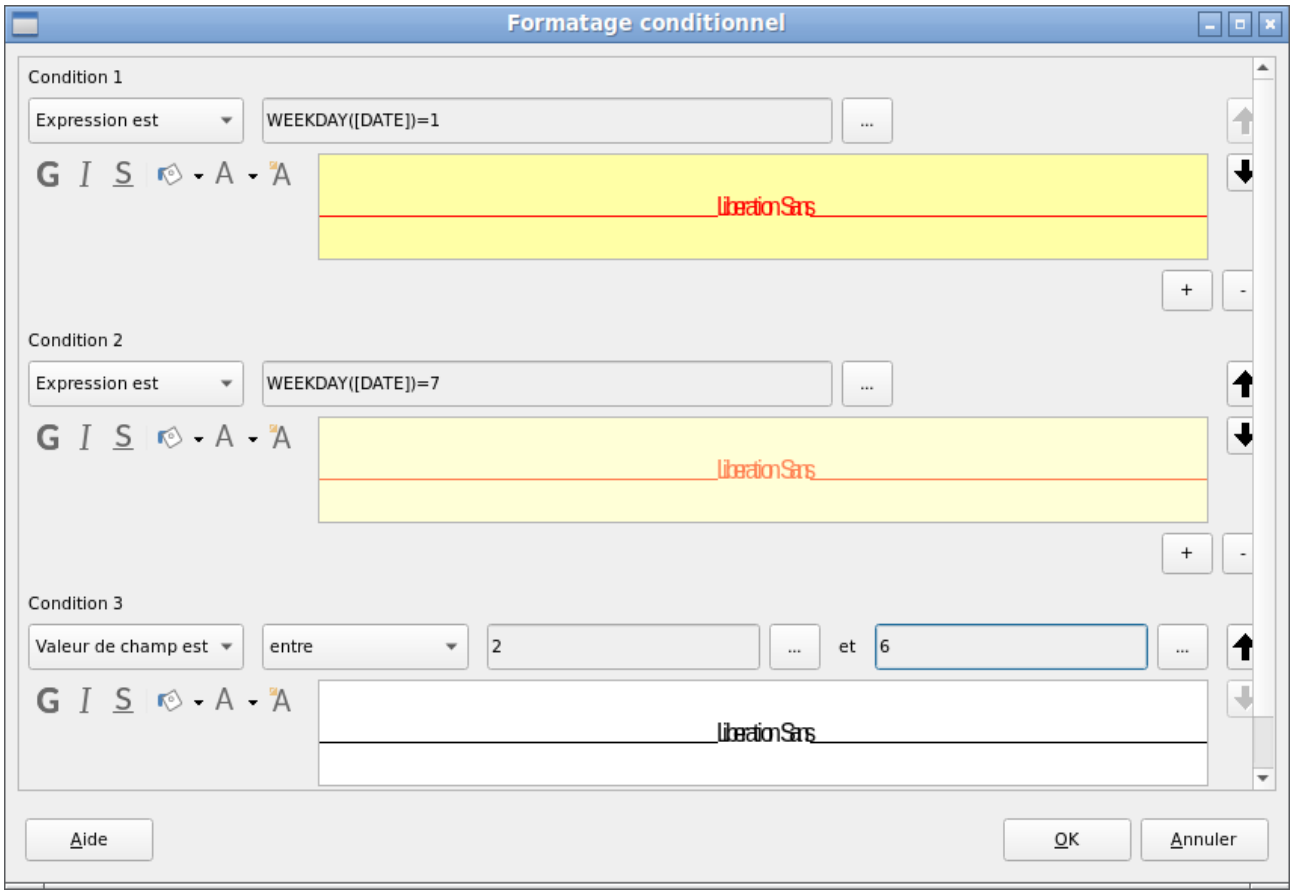
Mise en forme conditionnelle

Le formatage conditionnel peut être utilisé, par exemple, pour mettre en forme un calendrier afin que les week-ends soient affichés différemment. Choisissez Format> **Formatage conditionnel** et entrez :

WEEKDAY([Date])=1

et le formatage correspondant pour les dimanches.

Si vous utilisez « Expression est » dans le formatage conditionnel, vous pouvez saisir une formule. Comme d'habitude dans Calc, plusieurs conditions peuvent être formulées et sont évaluées séquentiellement. Dans l'exemple ci-dessus, le premier dimanche est testé, puis le samedi. Enfin, il peut y avoir une requête sur le contenu du champ. Ainsi, par exemple, le contenu « Vacances » conduirait à un format différent.



Note

Si des erreurs incorrigibles supplémentaires se produisent (formules non implémentées, texte trop long affiché sous forme de champ vide, etc.), il est parfois nécessaire de supprimer des parties du rapport ou simplement de le créer à nouveau.

Exemples de rapports créés avec le Générateur de rapports

L'utilisation du Générateur de rapports est quelque peu perfide, car certaines fonctions sont disponibles en théorie mais ne fonctionnent pas dans la pratique.

De plus, LibreOffice Help a très peu à dire à ce sujet. Pour cette raison, quelques exemples sont fournis ici, montrant comment le Générateur de rapports peut être utilisé pour différents types de rapport[7].

Impression de factures

La création d'une facture nécessite les considérations suivantes :

- Les éléments individuels doivent être numérotés ;
- Les factures qui nécessitent plus d'une page doivent être numérotées ;
- Les factures qui nécessitent plus d'une page doivent avoir un total cumulé sur chaque page, qui est reporté à la page suivante.

Plusieurs bogues actuels semblent rendre cela impossible :

Bogue 51452 : Si un groupe est défini sur « Répéter la section », un saut de page est inséré automatiquement avant et après le groupe ;

Bogue 51453 : les groupes avec pagination individuelle sont autorisés mais ne fonctionnent pas réellement ;

Bogue 51959 : un pied de page de groupe ne peut pas être répété. Cela ne peut se produire qu'à la fin du groupe, pas à la fin de chaque page. Et si vous choisissez « Répéter la section », il disparaît complètement.

En outre, l'insertion de lignes (graphiques) dans les rapports pose des problèmes. Les lignes horizontales et verticales intégrées n'apparaissent que dans les versions LibreOffice 4.0.5 et 4.1.1 respectivement. Vous pouvez utiliser des rectangles comme substitut, mais ils ne peuvent pas être correctement positionnés en cas de saut de page dans la section.

Le rapport sous une forme simple devrait ressembler à ceci :

7 Ces exemples sont disponibles dans la base Exemple_Rapport_Facturation.odb ddes bases exemples disponibles avec ce manuel

La Boutique Office
123 rue des Moulins 94000 Créteil

Numero de Facture : 2020-0
Date : ven. 11 septembre 2020

Quantite	Article	Prix	Quantite*Prix
2	agrafes	0,85 €	1,70 €
1	gomme	0,75 €	0,75 €
4	taille crayon	1,27 €	5,08 €
2	crayon HB	0,23 €	0,46 €
1	bloc-notes à spirale	0,98 €	0,98 €
1	envelopes, 25 pcs.	1,25 €	1,25 €
4	pochettes CD, 100 pcs.	1,89 €	7,56 €
1	crayons de cire, 6 pcs.	3,85 €	3,85 €
1	dossier, largeur 5 cm	1,89 €	1,89 €
2	Presse-papiers	4,45 €	8,90 €
1	reliure à anneauxA4	5,76 €	5,76 €
1	Stylos CD, 4 pcs.	4,56 €	4,56 €
2	CD-vierges, 50 pcs.	12,34 €	24,68 €
1	papier, recyclage, 500 feuilles	3,76 €	3,76 €
4	mallette avec registre	6,87 €	27,48 €
2	carnet de reçus, 50 feuilles	1,35 €	2,70 €
10	carnet de factures, 50 feuilles	3,15 €	31,50 €
1	horodatage, simple	18,50 €	18,50 €
1	fiches bristol 100 pces	12,00 €	12,00 €
12	dossier suspendu, 25 pcs.	14,75 €	177,00 €
2	étiquettes universelles 70x32, 2700 pcs.	20,50 €	41,00 €
1	étiquettes universelles format 12x30, 700 pcs.	4,15 €	4,15 €
2	tête d impression, noir	24,00 €	48,00 €
1	tête d impression, couleur	26,30 €	26,30 €
3	cartouche d encre, noir, 32 ml	5,40 €	16,20 €
5	cartouche d encre, couleur, 17 ml	5,20 €	26,00 €
		Report :	502.01

Page 1

Détails Banque: Banque Fantastique - BIC DABCFR2X - IBAN FR6123456756785000000456

La Boutique Office
123 rue des Moulins 94000 Créteil

Page 2

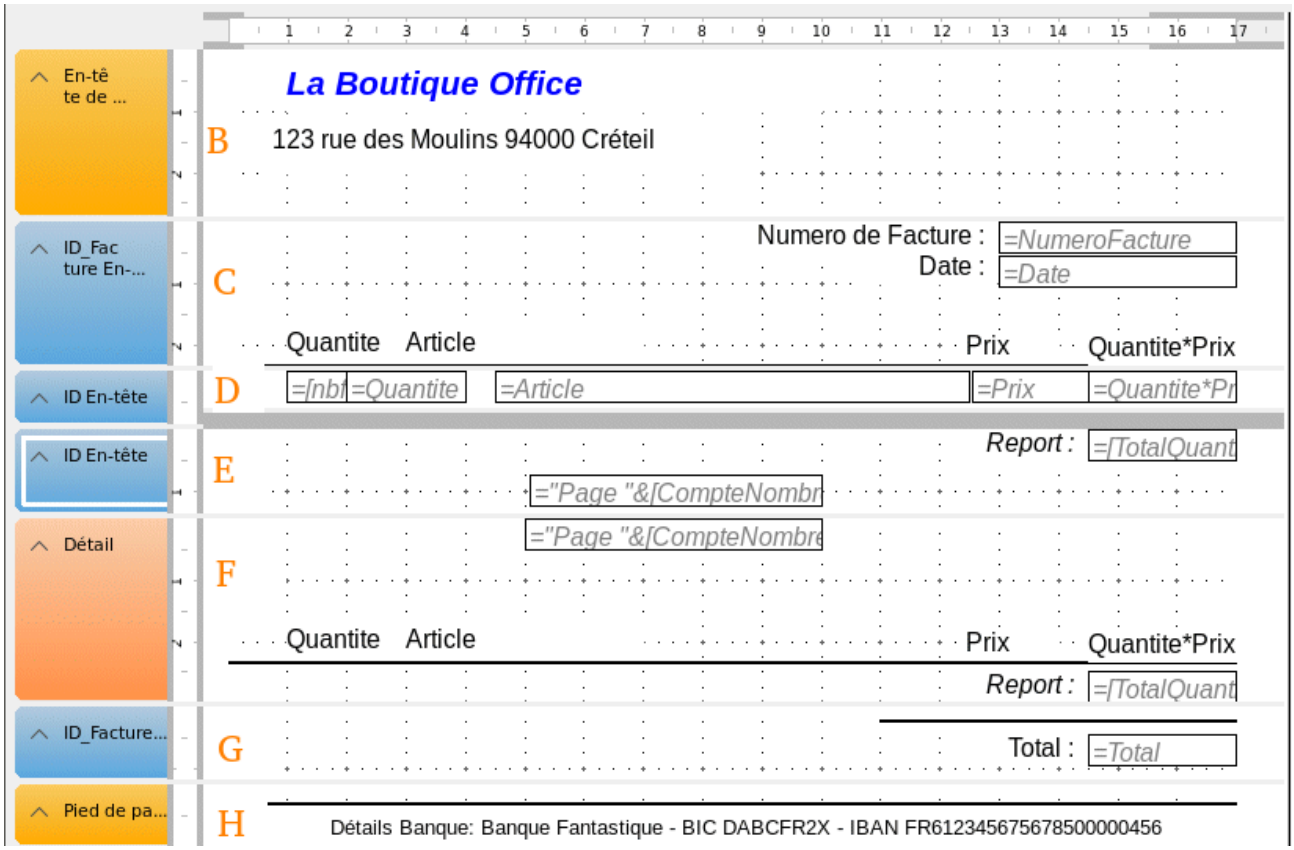
Quantite	Article	Prix	Quantite*Prix
		Report :	502,01 €
2	toner imprimante laser noir	65,89 €	131,78 €
1	toner imprimante laser couleur	78,89 €	78,89 €
2	enregistrement pour dossiers, A-Z, A-Z	1,25 €	2,50 €
1	agrafes	0,85 €	0,85 €
2	recyclage de fichiers	0,65 €	1,30 €
1	papier, recyclage, 500 feuilles, couleur	4,15 €	4,15 €
1	crayons, 10 pièces, div. Epaisseur	4,85 €	4,85 €
2	stylo à bille	1,35 €	2,70 €
1	aquarelles, 12 couleurs	8,75 €	8,75 €
1	aquarelle, 24 colors	17,15 €	17,15 €
2	pinceau aquarelle, 3 pièces, div. Epaisseur	8,34 €	16,68 €
1	Bloc à dessin, A3, 20 feuilles	3,85 €	3,85 €
4	Sous-main de bureau 20 * 30 pouces	15,67 €	62,68 €
2	papier, div. couleurs, 20 * 30 pouces	0,45 €	0,90 €
10	carton, div. couleurs, 20 * 30 pouces	0,89 €	8,90 €
1	horodatage, simple	18,50 €	18,50 €
1	fiches bristol 100 pces	12,00 €	12,00 €
12	dossier suspendu, 25 pcs.	14,75 €	177,00 €
2	étiquettes universelles 70x32, 2700 pcs.	20,50 €	41,00 €
1	étiquettes universelles format 12x30, 700 pcs.	4,15 €	4,15 €
2	tête d impression, noir	24,00 €	48,00 €
1	tête d impression, couleur	26,30 €	26,30 €
3	cartouche d encre, noir, 32 ml	5,40 €	16,20 €
5	cartouche d encre, couleur, 17 ml	5,20 €	26,00 €
2	toner imprimante laser noir	65,89 €	131,78 €
1	toner imprimante laser couleur	78,89 €	78,89 €
		Report :	1427.76

Page 2

Détails Banque: Banque Fantastique - BIC DABCFR2X - IBAN FR6123456756785000000456

Pour surmonter les restrictions décrites ci-dessus, la création de la facture correspondante nécessite une attention particulière aux dimensions de la page dans le document imprimé final. Cet exemple part d'un format DIN-A4. La hauteur totale de chaque page est donc de 29,7 cm.

Le rapport doit être divisé en plusieurs groupes. Deux groupes se rapportent au même champ de table et contiennent le même élément de table.



Le tableau suivant montre la division de la page en différentes sections du rapport.

A	Marge de haut de page	2.00 cm
B	En-tête de page (apparaît sur chaque page, ne contient pas de contenu de base de données, uniquement du matériel tel qu'un logo d'entreprise ou l'adresse du fournisseur).	3.00 cm
C	En-tête de groupe de la facture (seuls les articles appartenant à un numéro de facture doivent être ajoutés ultérieurement. L'en-tête de groupe n'apparaît qu'au début de la facture).	2.50 cm
D	En-tête de groupe pour les articles individuels. (La section Détail est requise dans un autre but. Par conséquent, un groupe vient ici qui trie également le contenu après la saisie. Ce groupe ne contient qu'une seule valeur.)	0.70 cm
E	En-tête de groupe, également lié aux éléments. (Cette section ne s'affiche que s'il y a tellement d'éléments qu'une page supplémentaire est nécessaire. Elle contient le total cumulé et le numéro de page en bas de la page. Il y a un saut de page après cette section,)	2.00 cm
F	Section Détails. (Cette section n'est affichée que s'il y a tellement d'éléments qu'une page supplémentaire est nécessaire. Elle contient la somme reportée et le numéro de page en haut de la page.)	2.50 cm
G	Pied de page de groupe pour le numéro de facture. (Voici le montant total de la facture, éventuellement avec TVA ajoutée. Le pied de page du groupe apparaît uniquement sur la dernière page de la facture.)	1.60 cm
H	Pied de page (par exemple, coordonnées bancaires)	1.00 cm
I	Marge de bas de page	1.00 cm

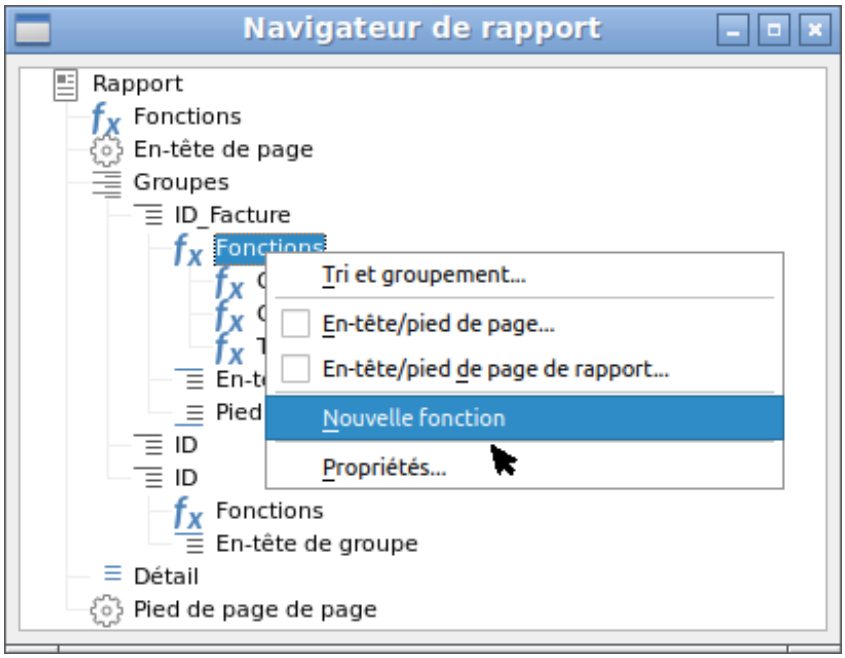
Un saut de page suit uniquement s'il y a trop d'éléments. Pour les articles, nous avons les espaces libres suivants :

	29.70 cm	(DIN A 4)
-	2.00 cm	(Pos. A)
-	3.00 cm	(Pos. B)
-	2.50 cm	(Pos. C)
-	1.60 cm	(Pos. G)
-	1.00 cm	(Pos. H)
-	1.00 cm	(Pos. I)
=	18.60 cm	

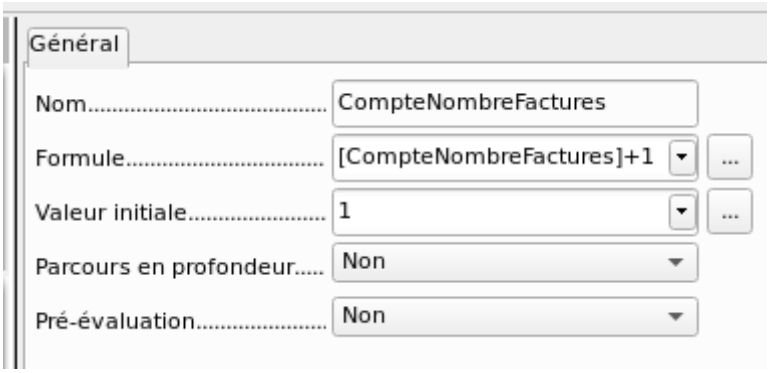
L'espace libre restant est donc au maximum de 18,60 cm / 0,70 cm = 26,57. L'arrondi donne 26 lignes d'articles.

Dès que le 27e élément apparaît, un saut de page doit immédiatement suivre. Cela indique que l'en-tête de groupe E et la section Détails doivent être affichés. Nous avons donc besoin d'un compteur pour les éléments (section C). Lorsque ce compteur atteint 27, la section Détails (F) s'affiche.

Le compteur des articles est défini comme suit :



À l'aide du navigateur de rapports, nous recherchons le groupe ID_Facture. Notre nouvelle fonction s'appelle CompteNombreFactures. La formule est [CompteNombreFactures] + 1. La valeur initiale est 1. Aucun sous-état n'est lié (il n'y a pas de telle fonction). Il n'est pas non plus pré-évalué. Pour les calculs ultérieurs, nous utilisons un compteur distinct, CompteNombreFacturesTotal.



L'en-tête de groupe E et la section Détails F sont tous deux affichés si un total de plus de 26 articles sont dans la facture et que la position d'enregistrement actuelle a atteint 26. L'expression pour l'affichage conditionnel est donc la même pour les deux :

AND([CompteNombreFactures]=26 ;[CompteNombreFacturesTotal]>26)

Le contenu de cette section n'apparaît donc que si au moins 27 éléments sont attendus. L'en-tête de groupe E apparaît sur la première page. Un saut de page suit et le contenu de la section Détails apparaît sur la page suivante.

Il faut maintenant calculer le nombre de sections qui doivent apparaître sur la première page.

	29.70 cm	(DIN A 4)
-	2.00 cm	(Pos. A)
-	3.00 cm	(Pos. B)
-	2.50 cm	(Pos. C)
-	18.20 cm	(Pos. D * 26)
-	1.00 cm	(Pos. H)
-	1.00 cm	(Pos. I)
=	2.00 cm	

Le pied de page de groupe n'est pas requis pour la première page. Il y a 26 lignes d'articles au total. L'en-tête de groupe E peut donc occuper au maximum 2 cm sur la première page. Ces 2 cm doivent contenir le total cumulé et le numéro de page. Pour garantir un saut de page correct en toutes circonstances, cette section doit en fait être un peu plus petite que 2 cm. Dans l'exemple, nous utilisons 1,90 cm.

La section Détails vient en haut de la page suivante. Comme l'en-tête de groupe pour les éléments (C) n'apparaît pas sur cette page, la section Détails peut occuper autant d'espace que l'en-tête, à savoir 2,50 cm. Ensuite, commencez le prochain lot d'articles avec le même arrangement que sur la page précédente.

La somme reportée est calculée en additionnant simplement les éléments précédents.

Le navigateur de rapport est utilisé pour trouver le groupe Numero_Facture. La nouvelle fonction à créer s'appellera TotalQuantitePrix. La formule est [TotalQuantitePrix] + [Quantite * Prix]. La valeur initiale est [Quantite * Prix]. Aucun sous-rapport n'est lié. Ce nombre ne doit pas non plus être calculé à l'avance.

La somme à reporter est affichée à la fois dans l'en-tête de groupe E et dans la section Détails. Dans l'en-tête du groupe, il se trouve juste en haut de la page. Sur la première page, il apparaît en bas. Dans la section Détails, la somme se trouve tout en bas. Il apparaît à la page 2 directement sous l'en-tête du tableau.

La requête pour déterminer le numéro de page est similaire à celle pour déterminer l'affichage de l'en-tête de groupe et de la section détails :

IF([CompteNombreFactures]=26 ;"Page 1";"")

Cela donne le numéro de page de la première page. D'autres requêtes IF peuvent être utilisées pour les autres pages.

Le numéro de page de la page suivante est facilement réglé sur « Page 2 » de la même manière.

Si les formules se poursuivent de la même manière, elles peuvent couvrir autant de pages que vous le souhaitez.

L'expression pour l'**affichage conditionnel** passe de

AND([CompteNombreFactures]=26 ;[CompteNombreFacturesTotal]>26)

à

AND(MOD([CompteNombreFactures] ; 26)=0 ;
[CompteNombreFacturesTotal]>[CompteNombreFactures])

L'en-tête de groupe E et la section Détails F n'apparaissent que lorsque la division du compteur d'articles par 26 ne donne pas de reste et le nombre total d'articles est supérieur au compteur d'articles.

L'expression du **numéro de page** passe de

IF([CompteNombreFactures]=26 ;"Page 1";"")

à

"Page" & [CompteNombreFactures]/26

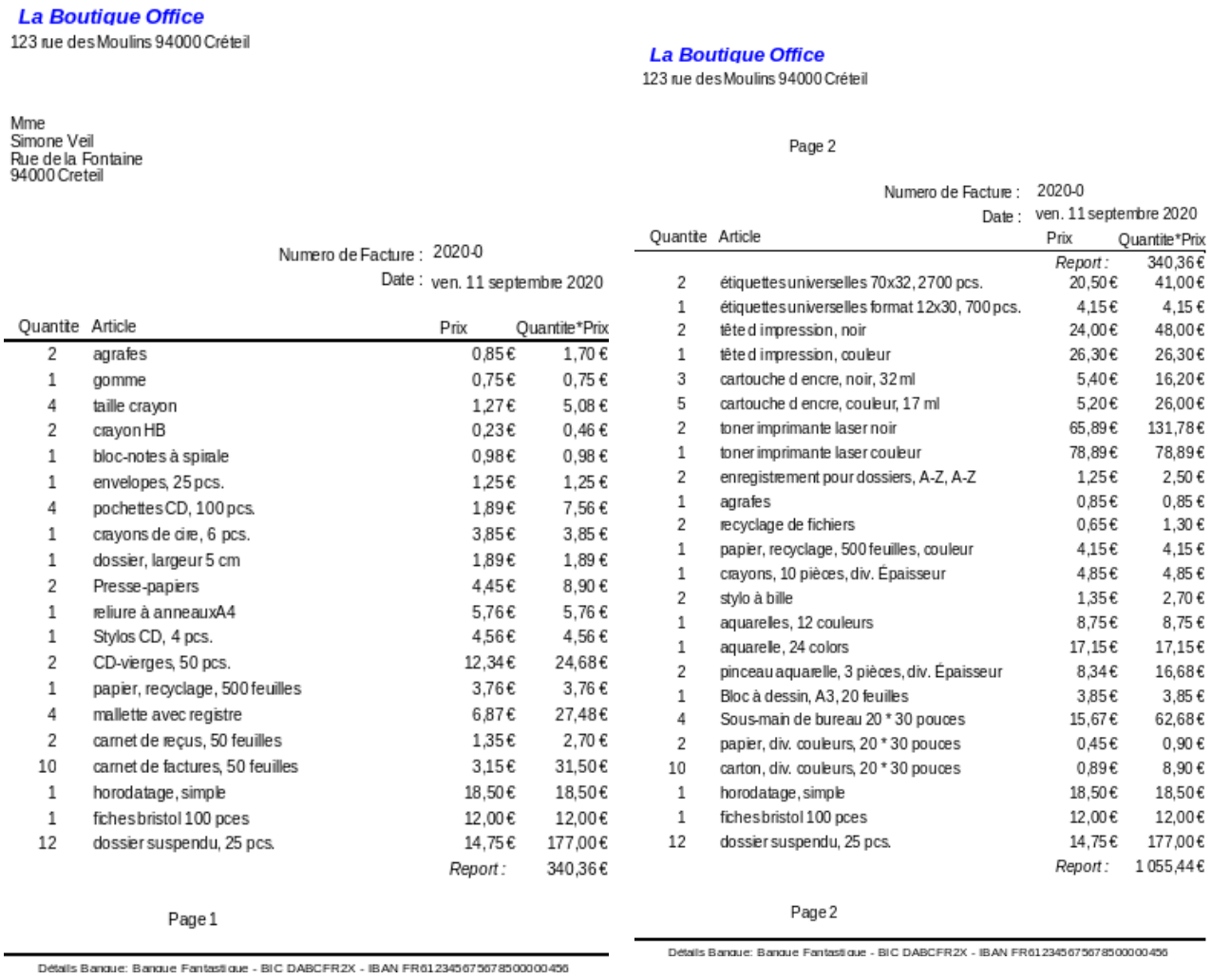
pour la page en cours et

"Page" & ([CompteNombreFactures]/26)+1

pour la page suivante.

Ne pas oublier les sauts de page après les sections E et G. Dans les propriétés de la section, **Forcer une nouvelle page> Après la section**.

L'impression de rapport suivante utilisant ces paramètres n'est toujours pas prête à être utilisée :



La Boutique Office
123 rue des Moulins 94000 Créteil

Mme
Simone Veil
Rue de la Fontaine
94000 Creteil

Numero de Facture : 2020-0
Date : ven. 11 septembre 2020

Quantite	Article	Prix	Quantite*Prix
2	agrafes	0,85 €	1,70 €
1	gomme	0,75 €	0,75 €
4	taille crayon	1,27 €	5,08 €
2	crayon HB	0,23 €	0,46 €
1	bloc-notes à spirale	0,98 €	0,98 €
1	envelopes, 25 pcs.	1,25 €	1,25 €
4	pochettes CD, 100 pcs.	1,89 €	7,56 €
1	crayons de cire, 6 pcs.	3,85 €	3,85 €
1	dossier, largeur 5 cm	1,89 €	1,89 €
2	Presse-papiers	4,45 €	8,90 €
1	reliure à anneauxA4	5,76 €	5,76 €
1	Stylos CD, 4 pcs.	4,56 €	4,56 €
2	CD-vierges, 50 pcs.	12,34 €	24,68 €
1	papier, recyclage, 500 feuilles	3,76 €	3,76 €
4	mallette avec registre	6,87 €	27,48 €
2	carnet de reçus, 50 feuilles	1,35 €	2,70 €
10	carnet de factures, 50 feuilles	3,15 €	31,50 €
1	horodatage, simple	18,50 €	18,50 €
1	fiches bristol 100 pces	12,00 €	12,00 €
12	dossier suspendu, 25 pcs.	14,75 €	177,00 €
		Report :	340,36 €

Page 1

Détails Banque: Banque Fantastique - BIC DABCFR2X - IBAN FR6123456756785000000456

La Boutique Office
123 rue des Moulins 94000 Créteil

Page 2

Numero de Facture : 2020-0
Date : ven. 11 septembre 2020

Quantite	Article	Prix	Quantite*Prix
		Report :	340,36 €
2	étiquettes universelles 70x32, 2700 pcs.	20,50 €	41,00 €
1	étiquettes universelles format 12x30, 700 pcs.	4,15 €	4,15 €
2	tête d impression, noir	24,00 €	48,00 €
1	tête d impression, couleur	26,30 €	26,30 €
3	cartouche d encre, noir, 32 ml	5,40 €	16,20 €
5	cartouche d encre, couleur, 17 ml	5,20 €	26,00 €
2	toner imprimante laser noir	65,89 €	131,78 €
1	toner imprimante laser couleur	78,89 €	78,89 €
2	enregistrement pour dossiers, A-Z, A-Z	1,25 €	2,50 €
1	agrafes	0,85 €	0,85 €
2	recyclage de fichiers	0,65 €	1,30 €
1	papier, recyclage, 500 feuilles, couleur	4,15 €	4,15 €
1	crayons, 10 pièces, div. Épaisseur	4,85 €	4,85 €
2	stylo à bille	1,35 €	2,70 €
1	aquarelles, 12 couleurs	8,75 €	8,75 €
1	aquarelle, 24 colors	17,15 €	17,15 €
2	pinceau aquarelle, 3 pièces, div. Épaisseur	8,34 €	16,68 €
1	Bloc à dessin, A3, 20 feuilles	3,85 €	3,85 €
4	Sous-main de bureau 20 * 30 pouces	15,67 €	62,68 €
2	papier, div. couleurs, 20 * 30 pouces	0,45 €	0,90 €
10	carton, div. couleurs, 20 * 30 pouces	0,89 €	8,90 €
1	horodatage, simple	18,50 €	18,50 €
1	fiches bristol 100 pces	12,00 €	12,00 €
12	dossier suspendu, 25 pcs.	14,75 €	177,00 €
		Report :	1 055,44 €

Page 2

Détails Banque: Banque Fantastique - BIC DABCFR2X - IBAN FR6123456756785000000456

En raison du champ d'adresse, il y a moins d'articles sur la première page de la facture que sur la deuxième page. La section Détails en haut de la deuxième page est donc nettement plus petite que l'en-tête de groupe de la facture (C).

Pour permettre un nombre différent d'éléments sur les deux premières pages, les formules doivent être ajustées.

Le calcul suivant garantit que les sections correspondantes sont correctement affichées.

AND(MOD([CompteNombreFactures] - 20 ; 24)=0 ;
[CompteNombreFacturesTotal]>[CompteNombreFactures])

Nous soustrayons le nombre d'articles sur la première page du compteur d'articles. Cette différence est divisée par le nombre total possible d'éléments sur la deuxième page. Si la division est exacte sans reste, la première condition d'affichage de l'en-tête de groupe E et de la section Détails F est remplie. De plus, comme indiqué précédemment, la valeur actuelle du compteur d'articles doit être inférieure au total attendu. Sinon, il y aurait de la place pour la somme totale calculée sur la page en cours.

Le nombre total possible d'articles sur la deuxième page est désormais plus petit, car cette page contient désormais le numéro de facture et la date.

Le numéro de page peut désormais être calculé plus simplement :

"Page" & INT([CompteNombreFactures]/24)+1

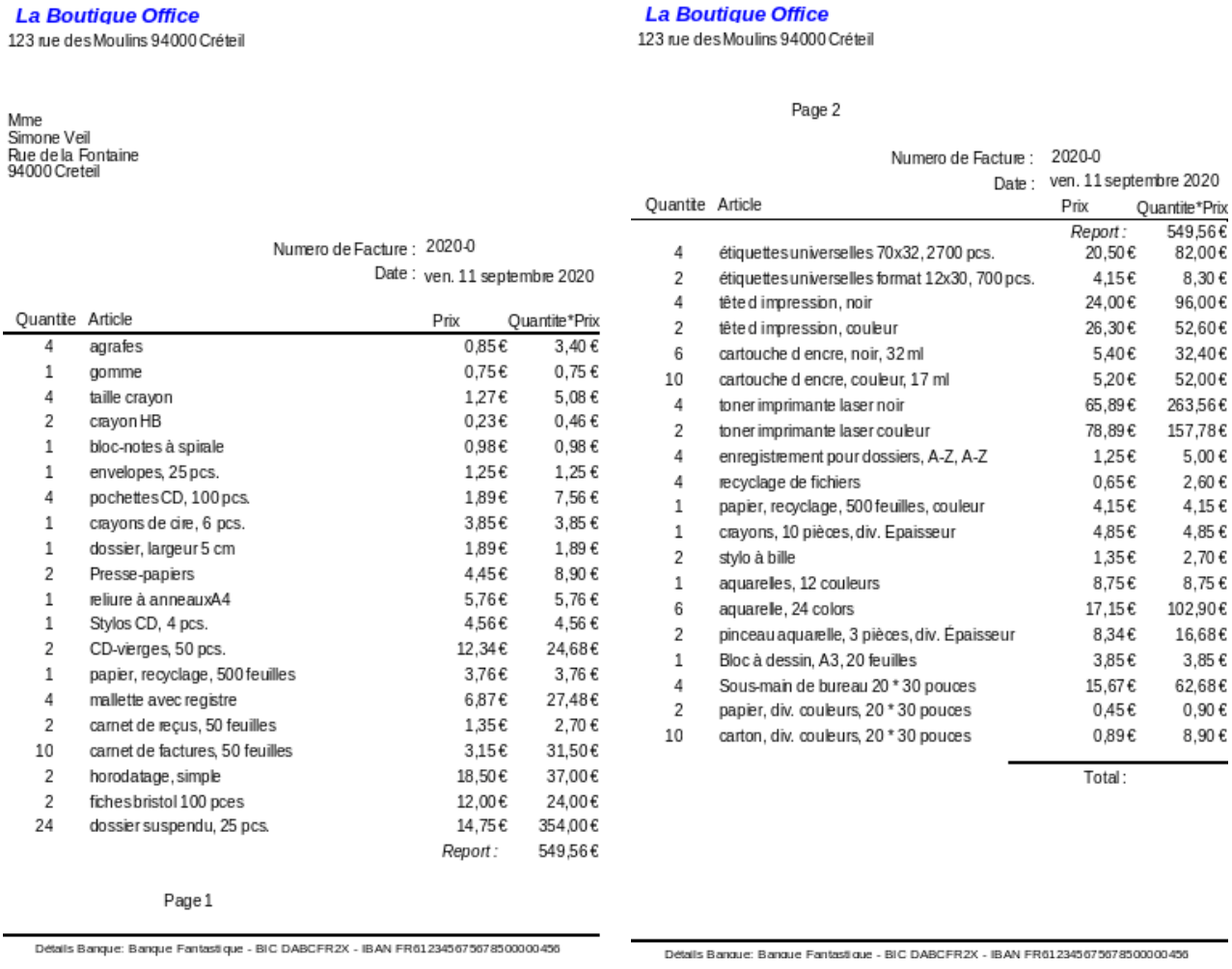
La fonction INT arrondit à l'entier inférieur le plus proche. La première page contient un maximum de 20 éléments. La division donne donc un résultat <1 pour cette première page. Cela s'arrondit à zéro. Nous devons donc ajouter 1 au numéro de page calculé pour que 1 apparaisse sur la première page. De même pour la page 2.

"Page" & INT([CompteNombreFactures]/24)+2

Le rapport montre encore un défaut esthétique. Un examen attentif des éléments de facturation montre que les trois éléments du bas des pages sont les mêmes. Ce n'est pas parce que les enregistrements ont simplement été copiés. Ce ne sont pas en fait les mêmes enregistrements, mais des enregistrements différents pour le même produit qui ont été traités indépendamment les uns des autres. Il serait préférable ici de regrouper par type de produit afin que chaque produit n'apparaisse qu'une seule fois avec le nombre total commandé.

En gros, vous devez essayer de supprimer autant de calculs et de regroupements que possible du Générateur de rapports. Au lieu d'utiliser les groupes du Générateur de rapports, il est préférable d'utiliser les fonctions de regroupement de l'éditeur de requêtes. Pour que le Générateur de rapports traite facilement la requête, transformez-la en vue. Sinon, le Générateur de rapports tentera d'améliorer la requête avec ses propres fonctions de regroupement et de tri, ce qui peut rapidement conduire à un codage peu pratique.

Le résultat final est le suivant :



La Boutique Office
123 rue des Moulins 94000 Créteil

Mme
Simone Veil
Rue de la Fontaine
94000 Creteil

Numero de Facture : 2020-0
Date : ven. 11 septembre 2020

Quantite	Article	Prix	Quantite*Prix
4	agrafes	0,85 €	3,40 €
1	gomme	0,75 €	0,75 €
4	taille crayon	1,27 €	5,08 €
2	crayon HB	0,23 €	0,46 €
1	bloc-notes à spirale	0,98 €	0,98 €
1	envelopes, 25 pcs.	1,25 €	1,25 €
4	pochettes CD, 100 pcs.	1,89 €	7,56 €
1	crayons de cire, 6 pcs.	3,85 €	3,85 €
1	dossier, largeur 5 cm	1,89 €	1,89 €
2	Presse-papiers	4,45 €	8,90 €
1	reliure à anneauxA4	5,76 €	5,76 €
1	Stylos CD, 4 pcs.	4,56 €	4,56 €
2	CD-vierges, 50 pcs.	12,34 €	24,68 €
1	papier, recyclage, 500 feuilles	3,76 €	3,76 €
4	mallette avec registre	6,87 €	27,48 €
2	carnet de reçus, 50 feuilles	1,35 €	2,70 €
10	carnet de factures, 50 feuilles	3,15 €	31,50 €
2	horodatage, simple	18,50 €	37,00 €
2	fiches bristol 100 pces	12,00 €	24,00 €
24	dossier suspendu, 25 pcs.	14,75 €	354,00 €
		Report :	549,56 €

Page 1

Détails Banque: Banque Fantastique - BIC DABCFR2X - IBAN FR6123456756785000004 56

La Boutique Office
123 rue des Moulins 94000 Créteil

Page 2

Numero de Facture : 2020-0
Date : ven. 11 septembre 2020

Quantite	Article	Prix	Quantite*Prix
		Report :	549,56 €
4	étiquettes universelles 70x32, 2700 pcs.	20,50 €	82,00 €
2	étiquettes universelles format 12x30, 700 pcs.	4,15 €	8,30 €
4	tête d impression, noir	24,00 €	96,00 €
2	tête d impression, couleur	26,30 €	52,60 €
6	cartouche d encre, noir, 32 ml	5,40 €	32,40 €
10	cartouche d encre, couleur, 17 ml	5,20 €	52,00 €
4	toner imprimante laser noir	65,89 €	263,56 €
2	toner imprimante laser couleur	78,89 €	157,78 €
4	enregistrement pour dossiers, A-Z, A-Z	1,25 €	5,00 €
4	recyclage de fichiers	0,65 €	2,60 €
1	papier, recyclage, 500 feuilles, couleur	4,15 €	4,15 €
1	crayons, 10 pièces, div. Epaisseur	4,85 €	4,85 €
2	stylo à bille	1,35 €	2,70 €
1	aquarelles, 12 couleurs	8,75 €	8,75 €
6	aquarelle, 24 colors	17,15 €	102,90 €
2	pinceau aquarelle, 3 pièces, div. Épaisseur	8,34 €	16,68 €
1	Bloc à dessin, A3, 20 feuilles	3,85 €	3,85 €
4	Sous-main de bureau 20 * 30 pouces	15,67 €	62,68 €
2	papier, div. couleurs, 20 * 30 pouces	0,45 €	0,90 €
10	carton, div. couleurs, 20 * 30 pouces	0,89 €	8,90 €
		Total :	

Détails Banque: Banque Fantastique - BIC DABCFR2X - IBAN FR6123456756785000004 56

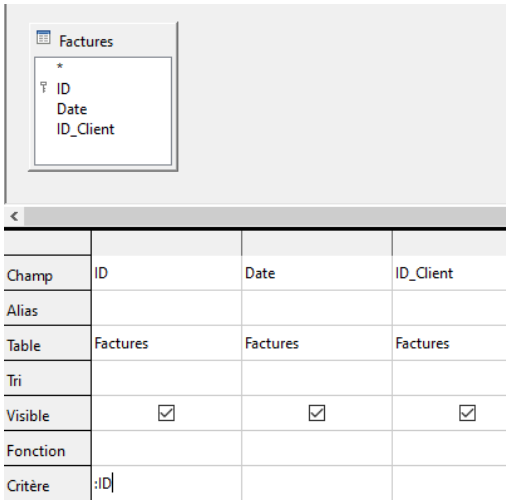
Ici, tous les éléments n’apparaissent qu’une seule fois. Les 12 dossiers suspendus d’origine au bas de la page 1 sont désormais 24 dossiers suspendus. Quantité*Prix a été recalculé en conséquence.

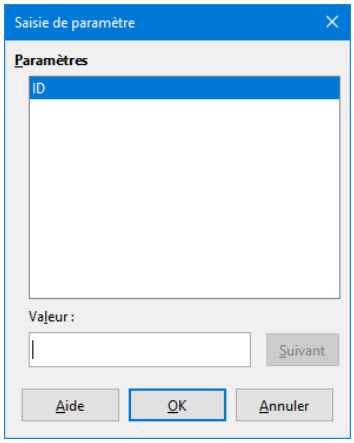
Impression de rapports pour l’enregistrement en cours dans un formulaire

Surtout dans le type de production de factures montré dans l’exemple précédent, il peut être utile de préparer et de prévisualiser une nouvelle impression après chaque entrée d’un article. Le contenu du projet de facture doit être déterminé à l’aide d’un formulaire et l’impression des documents individuels doit suivre.

Les rapports ne peuvent pas être lancés en utilisant des macros avec un filtre. Cependant, la requête qui sert de base au rapport peut être filtrée au préalable. Cela peut se faire soit sous la forme d'une requête paramétrée :

SELECT * FROM "Factures" WHERE "ID" = :ID





soit par une requête qui est fournie avec des données via une table de filtrage à une ligne :

SELECT * FROM "Factures" WHERE "ID" = (SELECT "Entier" FROM "Filtre" WHERE "ID" = TRUE)

Dans une requête paramétrée, le contenu doit être inséré dans le champ de dialogue correspondant après son lancement.

Lorsqu'une table de filtres est utilisée pour contrôler le processus, son contenu doit être écrit par une macro. Par conséquent, une entrée séparée n'est plus nécessaire, ce qui facilite la vie de l'utilisateur. Le processus est décrit ci-dessous.

Construire la table des filtres

La table de filtrage ne doit contenir qu'un seul enregistrement à la fois. Cela signifie que sa clé primaire peut être un champ Oui/Non. Les autres champs du tableau sont nommés de manière à indiquer clairement le type de contenu qu'ils contiennent. Dans l'exemple, le champ qui doit filtrer la clé primaire de la table Factures est appelé Entier, car la clé elle-même est de ce type. Pour d'autres possibilités de filtrage, d'autres champs peuvent être ajoutés ultérieurement. Le filtre Entier peut être utilisé à des moments différents pour plusieurs tables différentes, car l'ancien contenu est toujours écrasé par le nouveau avant l'impression. Mais cette utilisation multiple ne fonctionne que dans une base de données mono-utilisateur (Base avec HSQLDB interne). Dans une base de données multi-utilisateurs, il y a toujours la possibilité qu'un autre utilisateur change la valeur du filtre dans l'une des tables normales pendant que la requête qui utilise le filtre est en cours d'exécution,

Nom de Champ	***Type de Champ***
ID	Oui/Non [BOOLEAN]
Entier	Integer [INTEGER]

Ce tableau est rempli au début avec un enregistrement. Pour cela, le champ ID doit être défini pour avoir la valeur Oui (" TRUE" en SQL). Ceci est réalisé dans la macro ci-dessous à la ligne stSql = "UPDATE .... (3 lignes avant End Sub).

Création de la macro pour lancer le rapport filtré

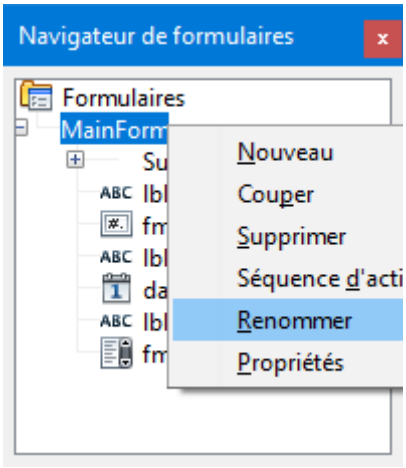
Pour appeler un seul état, le formulaire doit contenir quelque part la clé primaire de la table Factures. Cette clé primaire est lue et transférée vers la table de filtrage par une macro. Ensuite, la macro lance le rapport souhaité.

```
SUB Filtre_et_Imprime
	DIM oDoc AS OBJECT
	DIM oDrawpage AS OBJECT
	DIM oFormulaire AS OBJECT
	DIM oChamp AS OBJECT
	DIM oSourceDonnees AS OBJECT
	DIM oConnexion AS OBJECT
	DIM oSQL_Commande AS OBJECT
	DIM stSQL AS STRING
	oDoc = thisComponent
	oDrawpage = oDoc. Drawpage
	oFormulaire = oDrawpage. Forms.getByName("MainForm")
	oChamp = oFormulaire.getByName("fmtID")
	oSourceDonnees = ThisComponent. Parent. CurrentController
```

```
        If NOT (oSourceDonnees.isConnected()) THEN
            oSourceDonnees.connect()
        END IF
        oConnexion = oSourceDonnees. ActiveConnection()
        oSQL_Commande = oConnexion.createStatement()
        stSql = "UPDATE ""Filtre"" SET ""Entier"" = " & oChamp.GetCurrentValue() & _
"WHERE ""ID"" = TRUE"
        oSQL_Commande.executeUpdate(stSql)

    ThisDatabaseDocument.ReportDocuments.getByName("Factures_articles_groupes_avec_Client
    ").open
    END SUB
```

Dans cet exemple, le formulaire est appelé par son nom "MainForm" (Nom par défaut donné par base lors de la conception). Si l'on change ce nom, lors de la conception du formulaire ou par la suite, il sera nécessaire de remplacer MainForm par le nom que l'on aura choisi. Pour rappel un clic sur l'icône "Navigateur de formulaire" ouvre la fenêtre suivante qui permet de renommer (Clic droit sur "MainForm"). On pourra aussi le renommer dans les propriétés du formulaire.



La clé primaire est appelée fmtID. Ce champ clé n'a pas besoin d'être visible pour que la macro y accède. La valeur du champ est lue et la commande `UPDATE` l'écrit dans le champ "Entier" de la table "Filtre" . Ensuite, le rapport est lancé. La vue Rapport_Articles_Groupes dans cet exemple à laquelle il fait référence est développée à une condition :

```
… WHERE "ID_Facture" = IFNULL((SELECT “Entier” FROM “Filtre” WHERE “ID” =
TRUE),"ID_Facture")…
```

Le champ Entier est lu. S'il n'a aucune valeur, il est défini sur ID_Facture. Cela signifie que tous les enregistrements sont affichés, pas seulement l'enregistrement de filtre. Ainsi, à partir de la même vue, tous les enregistrements stockés peuvent être imprimés.

Coloration alternée du fond des lignes

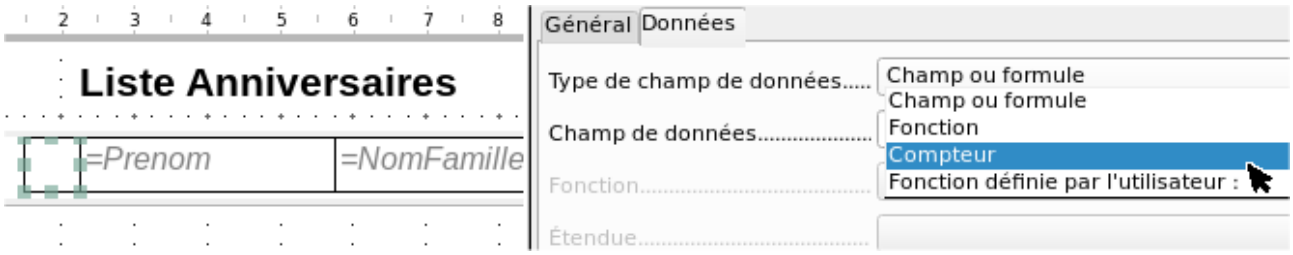
Lorsque vous lisez des lignes individuelles dans un tableau dans un rapport, il est facile pour l'œil de glisser vers le haut ou vers le bas d'une ligne. La coloration en arrière-plan d'au moins une ligne permet d'éviter cela. Dans l'exemple suivant, les lignes sont simplement colorées en alternance. Le résultat ressemble à ceci :

	Liste Anniversaires		
1	Alain	Daixe	01/05/64
2	Amélie	Oration	07/04/78
3	Fred	Déplacement	04/07/81
4	Clara	Hocquet	25/12/84
5	Gaspard	Alizan	17/03/85
6	César	Yennes	28/09/87
7	Agathe	Zeublouse	06/07/89
8	Bruno	Zieuvair	07/06/95
9	Cécile	Cligne	27/02/98
10	Anna	Logis	01/01/99

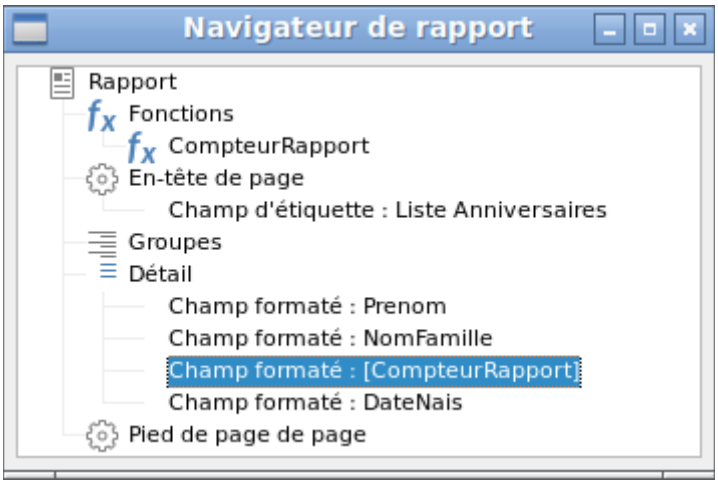
La base du rapport est une requête avec des noms et des dates. La table d'origine est interrogée avec un tri par date (année, mois et jour) pour afficher la séquence des anniversaires tout au long de l'année. Ceci est fait par :

```
... ORDER BY YEAR ("DateNais") ASC, MONTH("DateNais") ASC, DAY("DateNais") ASC
```

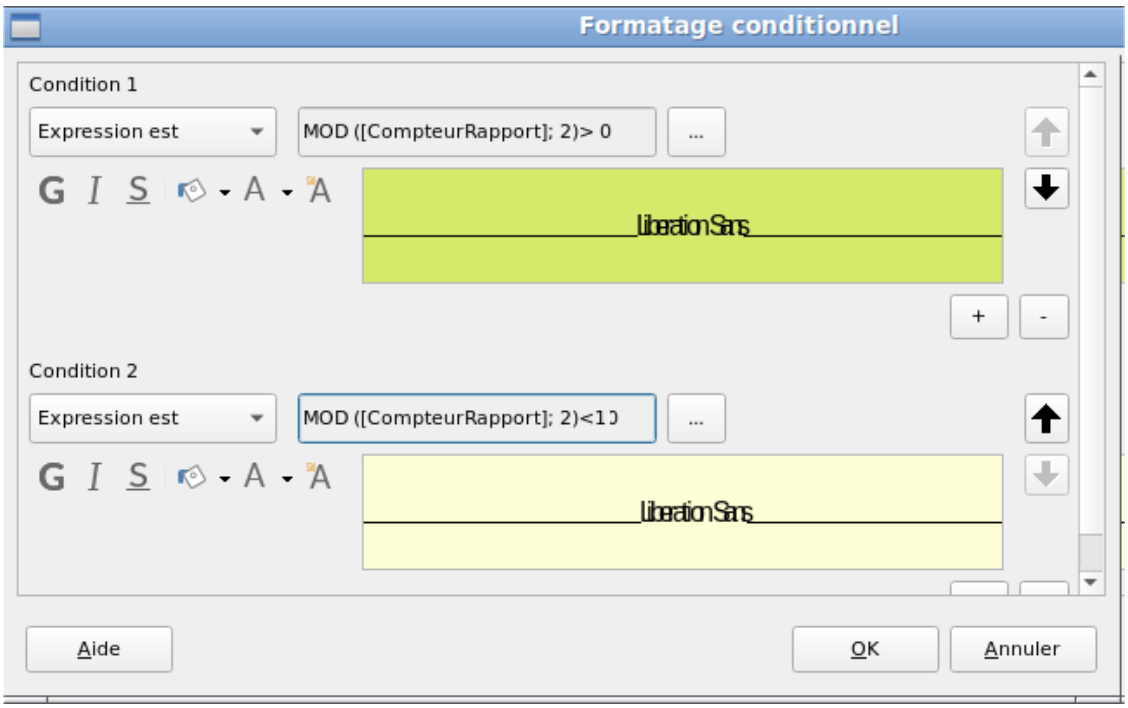
Pour obtenir les couleurs alternées, nous devons créer une fonction qui peut plus tard utiliser une valeur pour définir les conditions, qui à leur tour déterminent la couleur d'arrière-plan. Nous créons un champ de texte dans le rapport et, en utilisant **Propriétés> Données> Type de champ de données**, nous définissons un compteur.



Nous avons besoin du nom de la fonction pour la mise en forme conditionnelle. Le compteur réel n'a pas besoin d'apparaître dans l'expression. Le nom peut être lu directement depuis le champ. Si le champ est à nouveau supprimé, le nom de la fonction est accessible via **Affichage> Navigateur de rapport.**

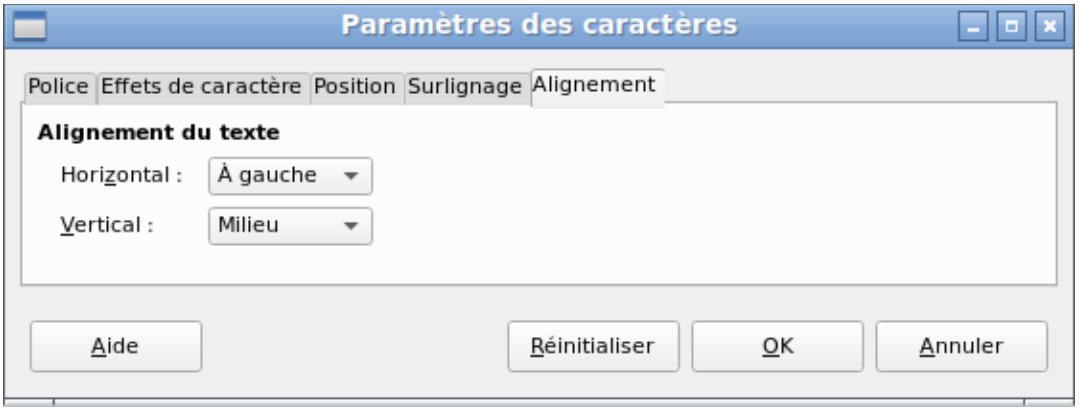


Maintenant, le compteur doit être utilisé pour donner à chaque champ de texte son propre format. La condition est une expression qui n'est pas directement connectée au champ. Par conséquent, il est configuré comme **Condition 1> L'expression est>** MOD([CompteurRapport] ; 2)>0. MOD calcule le reste d'une division. Pour tous les nombres impairs, c'est supérieur à 0 ; pour les nombres pairs, il vaut 0 précisément. Les lignes 1, 3 et 5 se verront donc attribuer ce format.



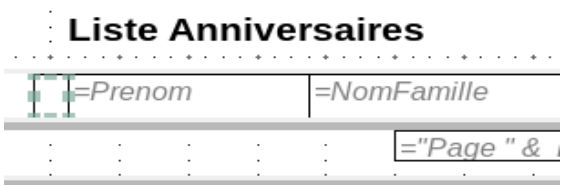
La deuxième condition est formulée comme l'exact opposé et un format correspondant est attribué. En fait, cette deuxième condition pourrait être omise et un format par défaut défini dans les propriétés de chaque champ. Le format conditionnel spécifié dans la première condition remplacerait alors cette valeur par défaut chaque fois que la condition était satisfaite.

Comme le format conditionnel remplace tous les formats par défaut, un alignement de texte pour ce format doit être inclus à l'aide des paramètres de caractères dans les propriétés du contrôle.



Ici, les lettres doivent être centrées verticalement sur le champ de texte coloré. L'alignement horizontal est l'alignement standard, mais un retrait doit être ajouté, afin que les lettres ne soient pas entassées sur le bord gauche du champ de texte.

Les expériences d'ajout d'espaces au contenu d'une requête ou d'une formule n'ont pas conduit à une indentation correcte du texte. Les espaces de tête sont simplement découpés.



Une méthode plus efficace consiste à positionner un champ de texte avant le texte réel, mais sans le lier à un champ de données. Ce champ de texte est mis en forme conditionnellement de la

même manière que les autres, de sorte qu'une indentation visible cohérente apparaît à l'impression.

Rapports à deux colonnes

Grâce à des techniques de requête intelligentes, vous pouvez créer un rapport avec plusieurs colonnes de sorte que la séquence horizontale corresponde à des enregistrements successifs :

Liste Anniversaires

Mehdi	01/01/02	Anna	01/01/99
Adele	05/01/01	Barbie	03/02/01
Bruno	27/02/98	Anna	01/03/06
Teddy	04/03/00	Gaspard	17/03/85
Vladimir	03/04/00	Amélie	07/04/78
Alain	01/05/64	Cécile	03/06/01
Elise	03/06/81	Alain	03/07/00
Agathe	05/07/89	Clémence	15/07/07
Fred	04/08/61	César	28/09/87
Ivan	05/10/99	Clara	23/12/84

Le premier enregistrement va dans la colonne de gauche, le second dans la droite. Les enregistrements sont triés selon la séquence des anniversaires au cours de l'année.

Le tri par anniversaire rend la requête de ce rapport assez longue. Si le tri était effectué par la clé primaire de la table sous-jacente, ce serait beaucoup plus court. Le critère de tri utilise un bloc de texte récurrent qui est expliqué plus en détail ci-dessous.

Voici la requête qui sous-tend le rapport :

```
SELECT "T1"."Prenom" AS  "Prenom1", "T1"."DateNais" AS "DateNais1",
         "T2"."Prenom" AS "Prenom2", "T2"."DateNais" AS "DateNais2"
FROM
      (SELECT "Prenom", "DateNais", "LignesNr" AS "Ligne"
      FROM
         (SELECT "a".*,
            (SELECT COUNT("ID")
            FROM "Anniversaires"
            WHERE  RIGHT('0' || MONTH("DateNais"), 2) || RIGHT('0' ||
DAY("DateNais"), 2) ||
                  "ID" <= RIGHT('0' || MONTH("a"."DateNais"), 2) ||
                  RIGHT('0' || DAY("a"."DateNais"), 2) || "a"."ID"
            ) AS "LignesNr"
         FROM "Anniversaires" AS "a"
         )
      WHERE MOD("LignesNr", 2) > 0
      ) AS "T1"

LEFT JOIN

      (SELECT "Prenom", "DateNais", "LignesNr" - 1 AS "Ligne"
      FROM
         (SELECT "a".*,
```

```
            (SELECT COUNT("ID")
            FROM "Anniversaires"
            WHERE RIGHT('0' || MONTH("DateNais"), 2) || RIGHT('0' || _
DAY("DateNais"), 2) ||
                  "ID" <= RIGHT('0' || MONTH("a"."DateNais"), 2) ||
                  RIGHT('0' || DAY("a"."DateNais"), 2) || "a"."ID"
            ) AS "LignesNr"
         FROM "Anniversaires" AS "a"
         )
      WHERE MOD("LignesNr", 2) = 0
      ) AS "T2" ON "T1"."Ligne" = "T2"."Ligne"
ORDER BY "T1"."Ligne"
```

Il y a deux sous-requêtes identiques dans cette requête. Les deux premières colonnes concernent la sous-requête avec l'alias T1 et les deux dernières la sous-requête T2.

Les sous-requêtes fournissent un autre champ (LignesNr) en plus de ceux de la table Anniversaires, ce qui permet de distinguer les lignes et de trier les enregistrements. La requête principale l'utilise avec la fonction de numérotation des lignes dans les requêtes (voir chapitre 5, Requêtes).

```
RIGHT('0'||MONTH("DateNais"),2)||RIGHT('0'||DAY("DateNais"),2)|| "ID"
```

Cette formule permet d'établir une séquence unique d'enregistrements. Comme les exemples d'enregistrements doivent être triés par date, il serait facile de penser que seule la date doit être utilisée dans la comparaison. C'est plus difficile en pratique, car ce ne sont pas les dates de naissance elles-mêmes mais leur enchaînement dans une seule année qui compte. Des problèmes supplémentaires sont causés par des valeurs de date identiques qui empêchent l'établissement d'une séquence unique. Par conséquent, le tri ne se fait pas seulement par mois et par jour, mais, dans ce cas, par clé primaire unique. Et pour éviter que le mois 10 ne vienne avant le mois 2, un zéro non significatif est placé avant chaque mois en utilisant | | lorsque le critère de tri est assemblé ; si le mois comporte déjà 2 chiffres, celui-ci est ensuite supprimé en utilisant `RIGHT(…, 2)`.

`SELECT COUNT ("ID")`donne le nombre d'enregistrements dont la combinaison du mois, du jour et de la clé primaire est inférieure ou égale à celle de l'enregistrement actuel dans la table Anniversaires. Ici, nous avons une sous-requête corrélative (voir le chapitre « Requêtes »).

`MOD("LignesNr", 2)` détermine si ce nombre est pair ou impair. MOD donne le reste d'une division, dans l'exemple, la division par 2. Cela amène LignesNr à donner les valeurs « 1 » et « 0 » en alternance. Cela distingue à son tour les requêtes pour T1 et T2.

Au niveau de requête suivant de T2, Ligne est défini comme "LignesNr" –1. De cette manière, T1 et T2 sont directement comparables.

T1 est lié à T2 en utilisant `LEFT JOIN`, de sorte que toutes les dates de la table Anniversaires soient également affichées pour les numéros d'enregistrement impairs. Lorsque T1 et T2 sont combinés, il est désormais possible de se référer directement aux lignes réelles : `"T1"."Ligne" = "T2"."Ligne"`.

Enfin tout le contenu est trié sur la valeur de ligne, qui est la même pour T1 et T2. Le rapport peut également l'utiliser directement à l'aide de sa fonction de regroupement.

Liste Anniversaires

Janvier		Février	
Anna	01/01/99	Barbie	03/02/01
Mehdi	03/01/02	Bruno	27/02/98
Adele	05/01/01		
Mars		**Avril**	
Anna	01/03/06	Vladimir	03/04/00
Teddy	04/03/00	Amélie	07/04/78
Gaspard	17/03/85		
Mai		**Juin**	
Alain	01/05/64	Cécile	03/06/01
		Elise	03/06/81
Juillet		**Août**	
Alain	03/07/00	Fred	04/08/61
Agathe	05/07/89		
Clémence	15/07/07		
Septembre		**Octobre**	
César	28/09/87	Ivan	05/10/99
Novembre		**Decembre**	
		Clara	23/12/84

Il est beaucoup plus compliqué de créer des techniques de requête pour faire des subdivisions en plus du format à 2 colonnes. Dans de tels cas, des lignes vides apparaissent au milieu du rapport, alors que dans l'exemple précédent à 2 colonnes, elles n'apparaissent qu'à la fin. Le Générateur de rapports ne fonctionne pas correctement avec ce type de requête. Par conséquent, au lieu de cela, nous utiliserons deux vues liées.

Nous créons d'abord la vue suivante :

```
SELECT "a"."ID", "a"."Prenom", "a"."DateNais",
      MONTH("a"."DateNais") AS "NumeroMois",
         (SELECT COUNT("ID") FROM "Anniversaires"
            WHERE MONTH("DateNais") = MONTH("a"."DateNais")
            AND RIGHT('0' || DAY("DateNais"), 2) || "ID" <= RIGHT('0' || _
DAY("a"."DateNais"), 2) || "a"."ID"
         ) AS "CompteurMois"
FROM "Anniversaires" AS "a"
```

Tous les champs de la table Anniversaires sont inclus. De plus, le mois est inclus sous forme de nombre. La table Anniversaires reçoit l'alias "a", de sorte que la table est accessible avec une sous-requête corrélée. Cette vue est nommée Vue_Numero_Mois.

La sous-requête compte tous les enregistrements d'un mois dont les valeurs de date ont un numéro de jour inférieur ou égal. Pour des dates identiques, la clé primaire détermine celle qui vient en premier. Cette technique est la même que dans l'exemple précédent.

La vue Vue_nom_mois_deux_colonnes accède à la vue Vue_Numero_Mois. Seuls des extraits de cette vue sont donnés ici :

```
SELECT"Tab1"."Prenom" AS "Prenom1",
      "Tab1"."DateNais" AS "DateNais1",
      1 AS "MoisNumero1",
```

```
        IFNULL("Tab1"."CompteurMois",999) AS "CompteurMois1",
        "Janvier" AS "Mois1",
        "Tab2"."Prenom" AS "Prenom2",
        "Tab2"."DateNais" AS "DateNais2",
        2 AS "NumeroMois2",
        IFNULL("Tab2"."CompteurMois",999) AS "CompteurMois2",
        "Fevrier" AS "Mois2"
FROM
        (SELECT * FROM "Vue_Numero_Mois"
           WHERE "NumeroMois" = 1
        ) AS "Tab1"
        RIGHT JOIN
        (SELECT * FROM "Vue_Numero_Mois" WHERE "NumeroMois" = 2
        ) AS "Tab2" ON "Tab1"."CompteurMois" = "Tab2"."CompteurMois"

UNION

SELECT"Tab1"."Prenom" AS "Prenom1",
          "Tab1"."DateNais" AS "DateNais1",
          1 AS "NumeroMois1",
          IFNULL("Tab1"."CompteurMois",999) AS "CompteurMois1",
          "Janvier" AS "Mois1",
          "Tab2"."Prenom" AS "Prenom2",
          "Tab2"."DateNais" AS "DateNais2",
          2 AS "NumeroMois2",
          IFNULL("Tab2"."CompteurMois",999) AS "CompteurMois2",
          "Février" AS "Mois2"
FROM
        (SELECT *
        FROM "Vue_Numero_Mois"
           WHERE "NumeroMois" = 1
        ) AS "Tab1"
        LEFT JOIN
        (SELECT *
        FROM "Vue_Numero_Mois"
        WHERE "NumeroMois" = 2
        ) AS "Tab2" ON "Tab1"."CompteurMois" = "Tab2"."CompteurMois"

UNION
…
ORDER BY "MoisNumero1", "CompteurMois1", "ConpteurMois2"
```

La sous-requête lit d'abord à partir de Vue_Numero_Mois toutes les dates pour lesquelles NumeroMois = 1. Cette sélection reçoit l'alias Tab1. En même temps, toutes les dates pour NumeroMois = 2 sont lues dans Tab2. Ces tables sont liées avec une JOINTURE DROITE (`RIGHT JOIN`), de sorte que tous les enregistrements de Tab2 sont affichés mais uniquement les enregistrements de Tab1 avec le même compteur de mois que Tab2.

Les colonnes de la vue doivent avoir des noms différents afin que chaque colonne soit dotée d'un alias. Un **1** est également entré directement comme numéro de mois pour Tab1 et janvier comme mois1. Ces entrées apparaissent également lorsqu'il n'y a aucun enregistrement de Tab1 mais toujours certains enregistrements dans Tab2. S'il n'y a pas de compteur mensuel disponible, la valeur **999** est entrée. Comme Tab1 est lié à Tab2 par une JOINTURE DROITE, il peut arriver que lorsqu'il y a moins d'enregistrements dans Tab1, des champs vides soient affichés à la place. Étant

donné que le tri ultérieur placerait ces enregistrements avant tous les enregistrements avec un contenu, un nombre très élevé est entré à la place.

La création de colonnes pour Tab2 est similaire. Ici bien qu'il ne soit pas nécessaire d'utiliser le code `IFNULL("Tab2"."CompteurMois",999)`, puisque, avec un `RIGHT JOIN` pour Tab2, a toutes les lignes de Tab2 seront affichées, mais il ne sera plus possible de créer plus de lignes à partir de Tab1 qu'à partir de Tab2.

C'est précisément ce problème qui est résolu par la liaison de deux requêtes. À l'aide `d'UNION`, tous les enregistrements de la première requête et tous les enregistrements de la deuxième requête sont affichés. Les enregistrements de la deuxième requête n'apparaissent que s'ils ne sont pas identiques à un enregistrement précédent. Cela signifie que `UNION` fonctionne comme `DISTINCT`.

En utilisant UNION, la même requête est à nouveau demandée, seuls Tab1 et Tab2 sont maintenant liés par un `LEFT JOIN`. Cela provoque l'affichage de tous les enregistrements de Tab1 même si Tab2 contient moins d'enregistrements que Tab1.

Pour les mois 3 et 4, 5 et 6, et ainsi de suite, des requêtes exactement équivalentes sont utilisées et à nouveau attachées à la requête précédente à l'aide d'un **`UNION`**.

Enfin, le résultat de la vue est trié par NumeroMois1, CompteurMois1 et CompteurMois2. Il n'est pas nécessaire de trier par NumeroMois2, car NumeroMois1 donne déjà la même séquence d'enregistrement.

Avertissement

Les requêtes avec UNION provoquent une erreur d'exécution si l'icône *Exécuter directement l'instruction SQL* n'est pas activée

Sources d'erreurs dans les rapports

Le Générateur de rapports masque parfois des erreurs dont la cause exacte ne peut pas être facilement déterminée par la suite. Voici quelques sources d'erreur et des contre-mesures utiles.

Le contenu d'un champ d'une requête n'apparaît pas

Une base de données est mise en place pour simuler la vente de stock. Une requête calcule un prix total à partir du nombre d'articles achetés et du prix unitaire.

```
SELECT "Ventes"."Total", "Stock"."Stock", "Stock"."Prix",
"Ventes"."Total"*"Stock"."Prix"
FROM "Ventes", "Stock"
WHERE "Ventes"."ID_Stock" = "Stock"."ID"
SELECT "Ventes"."Total", "Stock"."Stock", "Stock"."Prix",
"Ventes"."Total"*"Stock"."Prix" A
FROM "Ventes", "Stock"
WHERE "Ventes"."ID_Stock" = "Stock"."ID"
```

Cette requête sert de base au rapport. Si toutefois le champ `"Ventes"."Total"*"Stock"."Prix"` est appelé dans le rapport, il reste vide. Si en revanche il est fourni avec un alias dans la requête, le Générateur de rapports peut facilement y accéder :

```
SELECT "Ventes"."Total", "Stock"."Stock", "Stock"."Prix",
"Ventes"."Total"*"Stock"."Prix" AS "PrixTotal"
FROM "Ventes", "Stock"
WHERE "Ventes"."ID_Stock" = "Stock"."ID"
```

Le champ accède maintenant à “PrixTotal” et présente la valeur correspondante.

Un rapport ne peut pas être produit

Parfois, un rapport peut être préparé mais pas produit ni même enregistré. Un message d’erreur assez peu informatif apparaît :

« Le rapport n’a pas pu être produit. Une exception de type com.sun.star.lang. WrappedTargetExeption a été découverte »

Il faut vérifier tout d’abord que l’extension « Report Builder » n’est pas installée. Si c’est le cas il faut la supprimer, car elle n’est plus nécessaire c’est maintenant une fonction interne de Base.

Ici, il peut être utile de lire les informations supplémentaires jointes au message. Si vous y voyez une référence à SQL, cela signifie que le Générateur de rapports ne peut pas interpréter correctement le code SQL dans la source de données.

Il peut être utile ici d’utiliser le Navigateur de rapports pour définir **Données> Analyser Commande SQL> Non**. Malheureusement, cette solution empêche les groupes existants de fonctionner.

Une meilleure façon est d’utiliser une vue plutôt qu’une requête comme base de votre rapport. Ceci est configuré de telle manière qu’il ressemble à une table pour le Générateur de rapports et peut être modifié sans aucune difficulté. Même les exigences de tri de la vue fonctionnent plus aisément.

Chapitre 7
Connexions aux bases

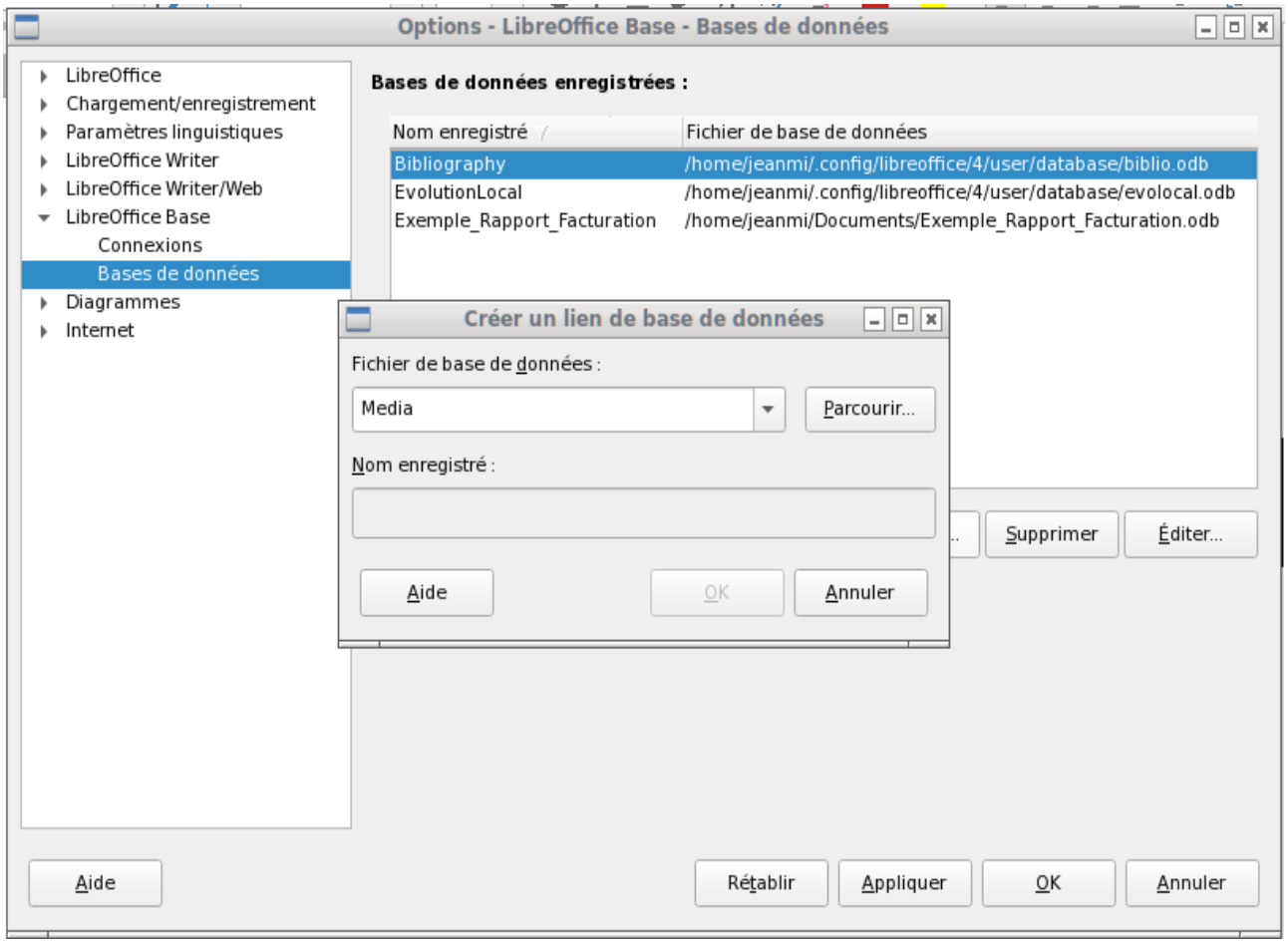
Notes générales sur la liaison avec les bases de données

Avec Base, vous pouvez utiliser des documents dans LibreOffice Writer et Calc de différentes manières comme sources de données. Cela signifie que l'utilisation de Base n'est pas nécessairement liée à l'enregistrement des bases de données dans la configuration de LibreOffice. Les formulaires externes peuvent également interagir directement avec Base, à condition que le chemin d'accès aux sources de données soit fourni.

Enregistrement des bases de données

De nombreuses fonctions, comme l'impression d'étiquettes ou l'utilisation de données pour des lettres types, nécessitent l'enregistrement d'une base de données dans la configuration de LibreOffice.

En utilisant **Outils > Options > LibreOffice Base > Bases de données > Nouveau**, vous pouvez enregistrer une base de données pour une utilisation ultérieure par d'autres composants LibreOffice.



Trouvez la base de données à l'aide d'un navigateur de fichiers et connectez-la à LibreOffice de la même manière que pour un formulaire simple. Donnez à la base de données elle-même un nom enregistré suffisamment informatif, par exemple le nom du fichier de base de données. Le nom sert d'alias, qui peut également être utilisé dans les requêtes adressées à la base de données.

Navigateur de sources de données

Le navigateur de sources de données dans Writer et Calc permet d'accéder aux tables et aux requêtes de toutes les bases de données enregistrées sous leurs noms enregistrés. Pour ouvrir le navigateur, utilisez **Affichage > Sources de données**, ou appuyez sur les touches Ctrl + Maj + F4, ou cliquez sur l'icône dans la barre d'outils Standard (si elle est visible).

L'icône Sources de données n'est généralement pas visible dans la barre d'outils Standard. Pour le rendre visible, cliquez avec le bouton droit sur le bouton Enregistrer pour ouvrir cette barre d'outils. Faites défiler jusqu'à Boutons visibles. Cela ouvre une liste de tous les boutons. Faites défiler vers le bas pour trouver le bouton Sources de données. Cliquez dessus pour le rendre visible à l'extrémité droite de la barre d'outils. Ceci peut également être réalisé en utilisant **Outils > Personnaliser** et cocher l'icône de la barre d'outils Standard.

Conseil

Si vous utilisez un ordinateur portable, vous devrez peut-être appuyer sur Ctrl + fn + Maj + F4. La touche fn (fonction) permet d'utiliser les touches F pour plusieurs fonctions.

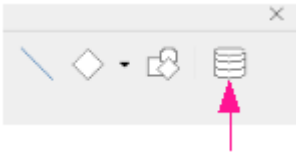
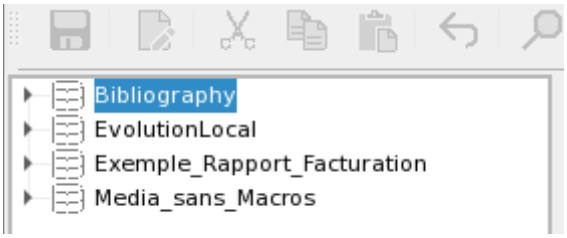


Figure 98: Bouton source de données

Les sources de données enregistrées sont affichées sur le côté gauche du navigateur de sources de données, qui se trouve par défaut en haut de l'espace de travail. La source de données Bibliographie est incluse dans LibreOffice par défaut. Les autres sources de données sont répertoriées par leurs noms enregistrés.



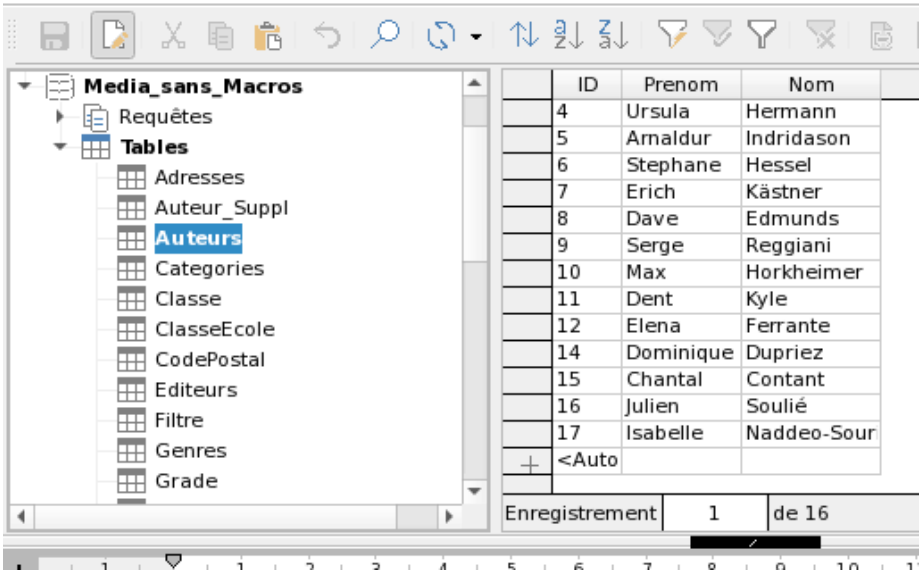
Cliquez sur le signe d'extension devant le nom de la base de données pour ouvrir la base de données et afficher les sous-dossiers pour les requêtes et les tables. Les autres sous-dossiers de la base de données ne sont pas disponibles ici. Les formulaires et rapports internes ne sont accessibles qu'en ouvrant la base de données elle-même.

Ce n'est que lorsque vous cliquez sur le dossier Tables que la base de données est réellement accessible. Pour les bases de données protégées par un mot de passe, le mot de passe doit être saisi à ce stade.

À droite de l'arborescence des noms, vous pouvez voir la table que vous avez sélectionnée. Elle peut être éditée comme dans Base. Cependant, l'entrée directe dans les tables doit être effectuée avec prudence dans les bases de données relationnelles très complexes, car les tables sont liées

entre elles par des clés étrangères. Par exemple, la base de données ci-dessous contient des tables séparées pour les noms de rue, les codes postaux et les villes.

Pour une vue correcte des données (mais sans possibilité de modification), les requêtes ou les vues sont plus adaptées.



La plupart des icônes de la barre d'outils (Figure 99) vous seront familières lors de la saisie de données dans les tableaux. (Les icônes sur votre écran peuvent être différentes de celles illustrées sur la figure, selon le thème choisi pour l'affichage.)

Les principales nouvelles icônes sont celles de la dernière section : Données en texte, Données en champs, Fusion et publipostage, Source de données du document actuel, Explorateur activé / désactivé. Leur utilisation est décrite ci-dessous, à l'aide de la table Lecteurs de la base de données Media.

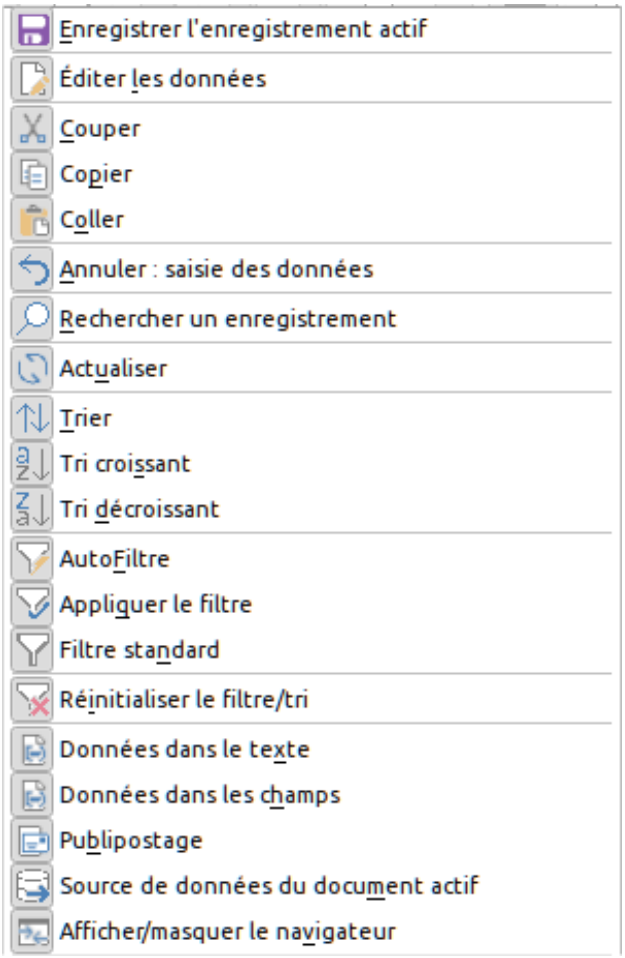


Figure 99: Barre d'outils du navigateur de source de données

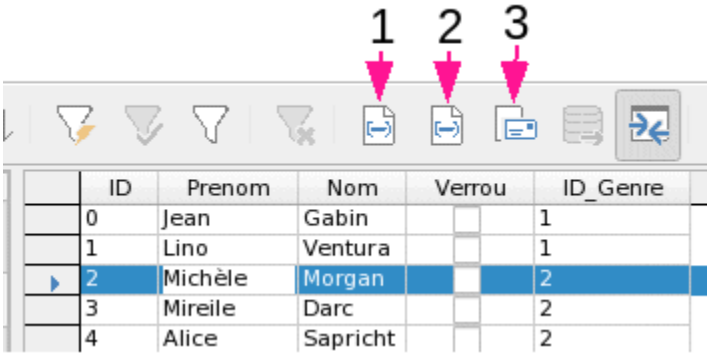
Données dans le texte

Conseil

Avec cette méthode, les données peuvent être insérées directement à des endroits spécifiques dans un document texte ou dans des cellules spécifiques d'une feuille de calcul. Bien que les données puissent être saisies à ces emplacements, leur insertion depuis une base en garantit l'exactitude. Ceci est important lors de l'utilisation du publipostage, qui sera abordé plus loin.

Lors de l'envoi du même document à différentes personnes, cela garantit que tous recevront exactement les mêmes informations.

Sélectionnez un ou plusieurs enregistrements (Ctrl-clic, Maj-clic) pour activer les fonctions Données en texte et Données en champs.



1	*Données dans le texte*	***2***	*Données dans les champs*	***3***	*Publipostage*

Figure 100: Selectionner un enregistrement de données.

Si vous choisissez maintenant **Données dans le texte**, un assistant apparaît pour effectuer le formatage nécessaire (Figure 101).

Les trois choix pour saisir les données sous forme de texte sont les suivants : sous forme de tableau, de champs uniques ou de texte ordinaire.

La figure 101 montre l'option **Insérer les données comme tableau**. Dans le cas des champs numériques et de date, le format de la base de données peut être changé en un format choisi. Sinon, le formatage est effectué automatiquement lorsque les champs de la table sont sélectionnés. La séquence des champs est ajustée à l'aide des touches fléchées.

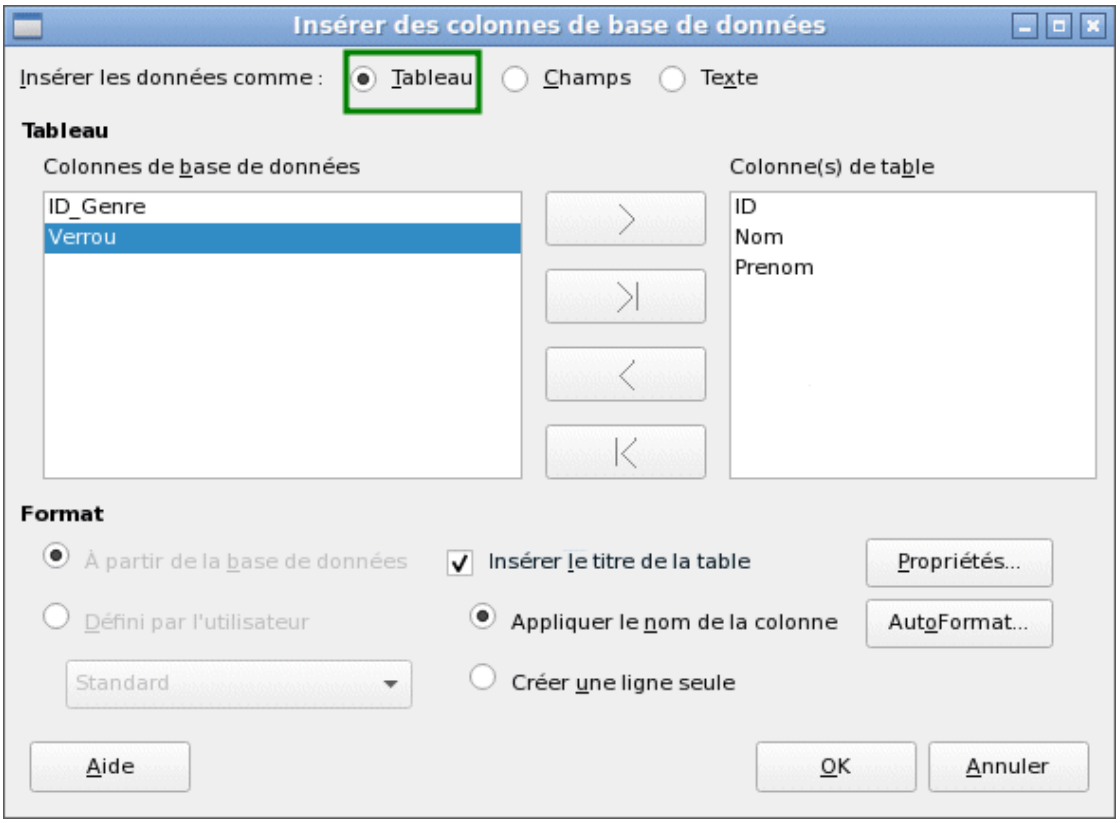


Figure 101: Insérer les données sous forme de tableau

Insérer les données comme tableau

Dès que les colonnes de la table ont été sélectionnées, le bouton **Propriétés** du tableau est activé. Cela vous permet de définir les propriétés de tableau habituelles pour Writer (largeur de tableau, largeur de colonne, etc.).

La case à cocher **Insérer un en-tête de tableau** détermine si une ligne d'en-tête de tableau est requise. Si elle n'est pas cochée, aucune ligne distincte ne sera réservée aux en-têtes.

La ligne choisie pour l'en-tête du tableau peut être extraite des noms de colonne, ou l'enregistrement peut être écrit avec un espace laissé pour les en-têtes à éditer ultérieurement. Choisissez l'option **Créer une ligne seule**.

Vous pouvez utiliser le bouton **AutoFormat** pour ouvrir une boîte de dialogue avec plusieurs styles de tableau pré-formatés. En dehors du style par défaut suggéré, tous les formats peuvent être renommés. (Vous pouvez également ajouter des autoformats ; pour ce faire, créez d'abord un tableau au format requis. Sélectionnez ensuite le tableau et cliquez sur le bouton Ajouter pour ajouter son format à la liste.) Vous pouvez également formater le tableau dans Writer en le sélectionnant et choisir un format dans la liste Styles de tableau dans le volet Styles de la barre latérale ; la liste des styles de tableau est la même que la liste des formats dans la boîte de dialogue AutoFormat.

Pour créer le tableau avec les enregistrements et les colonnes sélectionnés, cliquez sur OK dans la boîte de dialogue Insérer des colonnes de base de données.

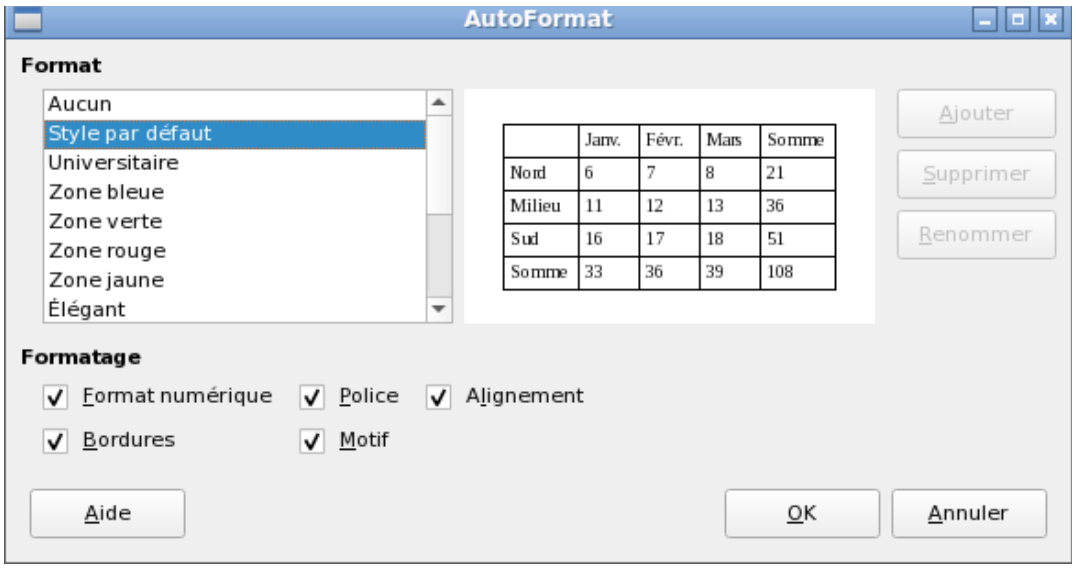


Figure 102: La boîte de dialogue AutoFormat propose un choix de formats de tableau

Insérer les données comme champs

Note

Cette méthode est utile lors de l'envoi d'un document à plusieurs personnes dans lesquelles chacune recevra des données qui leur sont spécifiques. Cela se fait par publipostage.

Par exemple, les bibliothèques envoient des avis aux personnes énumérant les médias qu'ils ont empruntés et qu'ils n'ont pas retournés à temps. La liste sera généralement différente pour chaque personne, mais tous recevront un avertissement. Bien entendu, le type d'avertissement dépendra de la durée écoulée depuis l'échéance du média. Tous ceux qui tombent avec un délai donné recevront le même avertissement.

Insérer des données comme champs offre la possibilité d'utiliser un mini-éditeur pour positionner successivement les différents champs du tableau dans le texte. Le texte ainsi créé peut également être doté d'un style de paragraphe. Dans ce cas également, le formatage des dates et des nombres peut être spécifié séparément, ou peut être lu directement à partir des paramètres de table dans la base de données.

Pour insérer des données dans des champs (voir Figure 101) :

Cliquez à gauche de l'une des lignes de la base pour la mettre en surbrillance;

3) Cliquez sur le bouton **Données dans le texte** pour ouvrir l'assistant **Insérer des colonnes de base de données**;
4. Choisissez le bouton d'option, **Champs**;
5. Déplacez les champs de la base de données que vous souhaitez utiliser de la liste de gauche vers la droite dans l'ordre souhaité. Du texte peut être ajouté ainsi que dans la figure 103;
6. Pour appliquer un style de paragraphe spécifique à ces champs, sélectionnez-le dans la liste déroulante Styles de paragraphe;
7. Cliquez sur **OK**.

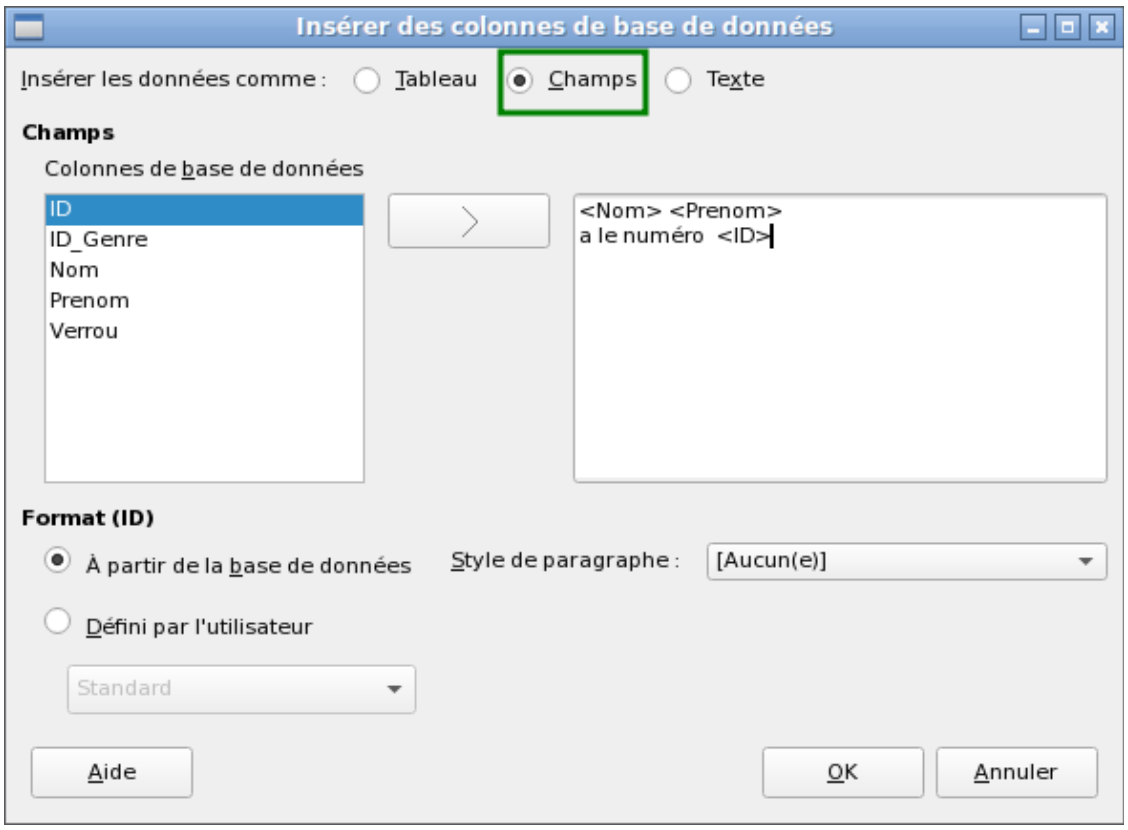


Figure 103: insérer des données sous forme de champs – Est la même que la boîte de dialogue Insérer des données sous forme de texte

Les champs insérés dans le texte de cette manière peuvent ensuite être supprimés individuellement ou utilisés pour un publipostage.

Insérer les données comme Texte

Si vous choisissez **Insérer les données comme texte**, la seule différence par rapport à l'utilisation des champs est que les champs restent liés à la base de données. Lorsque vous insérez sous forme de texte, seul le contenu des champs spécifiés est transféré et non le lien vers la base de données réelle. Cela permet par exemple de récupérer une adresse pour un courrier ponctuel.

Note

L'insertion de données dans du texte se fait de la même manière que l'insertion de données dans des champs. La seule différence est de savoir si vous sélectionnez respectivement Texte ou Champs. La différence réside dans l'apparence du document ou de la feuille de calcul, comme indiqué ci-dessous. Celui de gauche représente données comme champs ; le droit données comme texte.

Les résultats des deux procédures sont comparés ci-dessous.

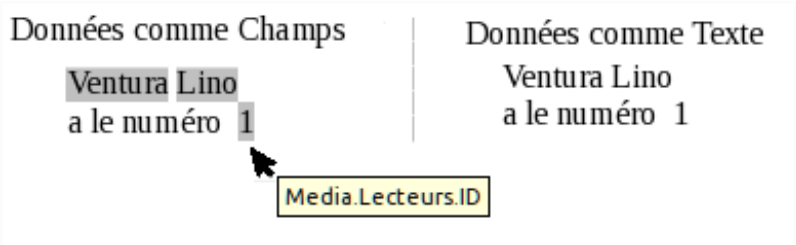


Figure 104: Comparaison données sous forme champs – données sous forme texte

Modifier les champs

Les champs ont un fond gris. Si vous passez le curseur de la souris sur les champs, une info-bulle indique que les champs sont liés à la base de données Medias_sans_Macros, à la table Lecteurs et, dans cette table, et le champ contenant "1" au champ ID.

Ainsi, par exemple, un double-clic sur le champ ID ouvre l'aperçu suivant. Cela indique clairement quel champ a été créé via la procédure Insérer des données comme champs. C'est le même type de champ que celui affiché par le menu **Insertion> Champs> Autres champs> Base de données**.

Il est plus simple de créer un tel champ en sélectionnant l'en-tête de colonne du tableau dans le navigateur de source de données et en le faisant glisser dans le document avec la souris. Vous pouvez créer une lettre type directement de cette manière.

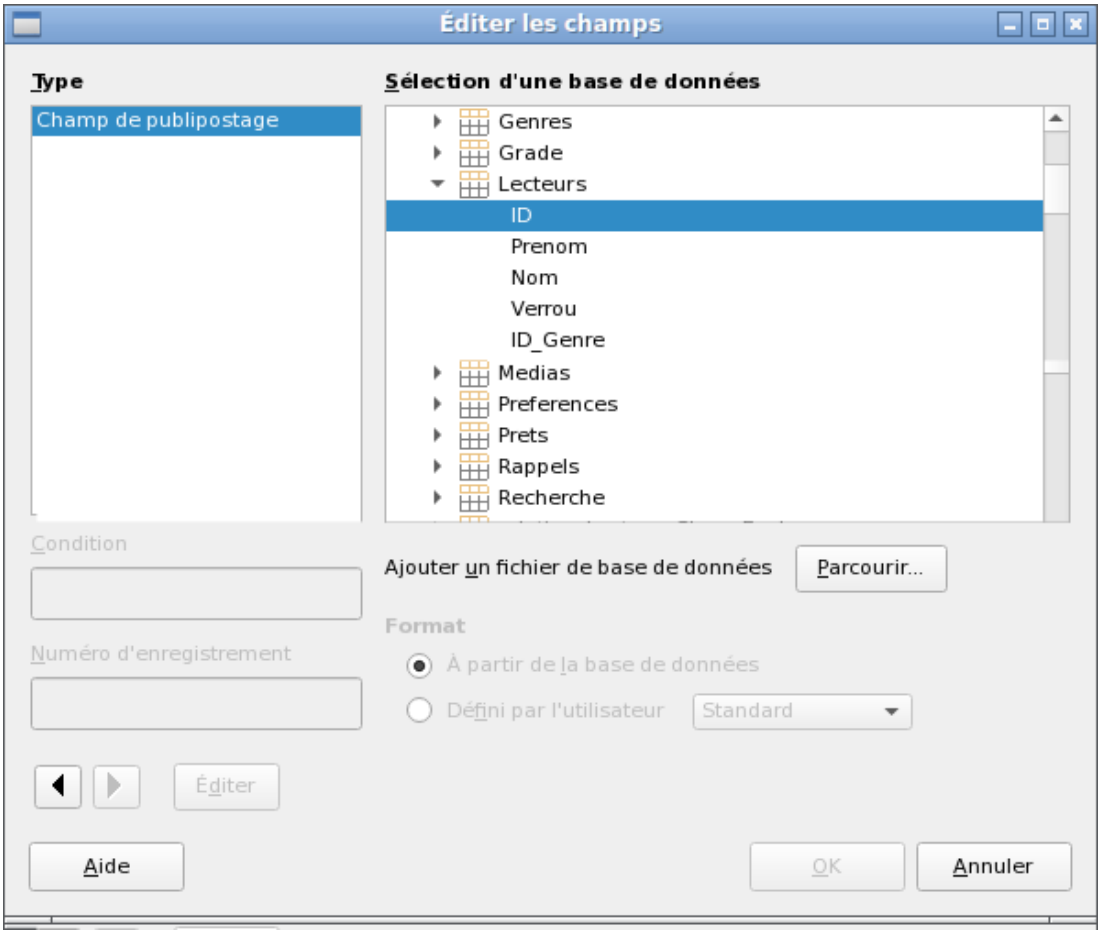


Figure 105: Afficher les champs de publipostage

Astuce

Une fois qu'un champ a été inséré, les valeurs qu'il affiche peuvent être modifiées. Sélectionnez un autre enregistrement (ligne), puis cliquez sur le bouton **Données vers champs** pour actualiser le contenu selon l'enregistrement choisi.

Morgan Michèle a le numéro 2	Ventura Lino a le numéro 1

Fusion et publipostage

Le bouton Publipostage lance l'Assistant publipostage. Une lettre type rassemble ses données à partir de différentes tables, vous devez donc d'abord lancer la base de données. Dans la base de données, vous créez ensuite une nouvelle requête pour rendre les données requises disponibles.

Figure 106: Bouton Assistant publipostage

Pour lancer la base de données, faites un clic droit sur la base de données elle-même ou sur l'une de ses tables ou requêtes ; cela actualise immédiatement l'affichage dans le navigateur de source de données. Ensuite, l'assistant de publipostage peut être appelé en cliquant sur le bouton correspondant.

Source de données du document actif

Cliquez sur le bouton Source de données du document actif pour ouvrir une vue directe du tableau qui forme la base des données insérées dans le document. Dans l'exemple ci-dessus, la table Lecteurs de la base de données Medias_sans_Macros apparaît.

Afficher/Masquer le navigateur

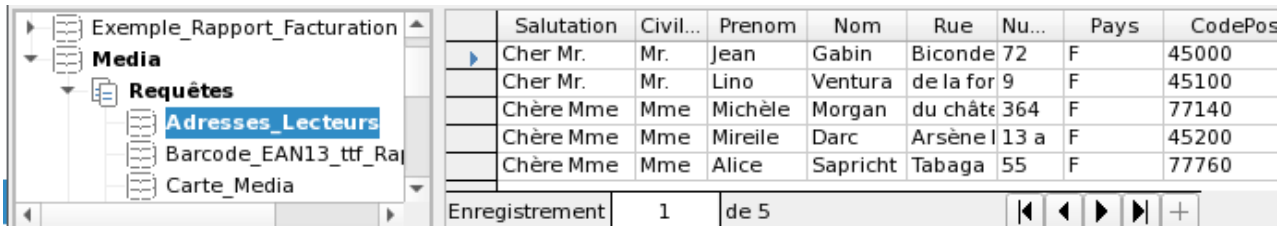
Afficher/Masquer le Navigateur affiche ou masque l'arborescence de répertoires sur la gauche de la vue de table. Cela laisse plus d'espace, si nécessaire, pour un affichage des données. Pour accéder à une autre table, vous devez réactiver l'explorateur.

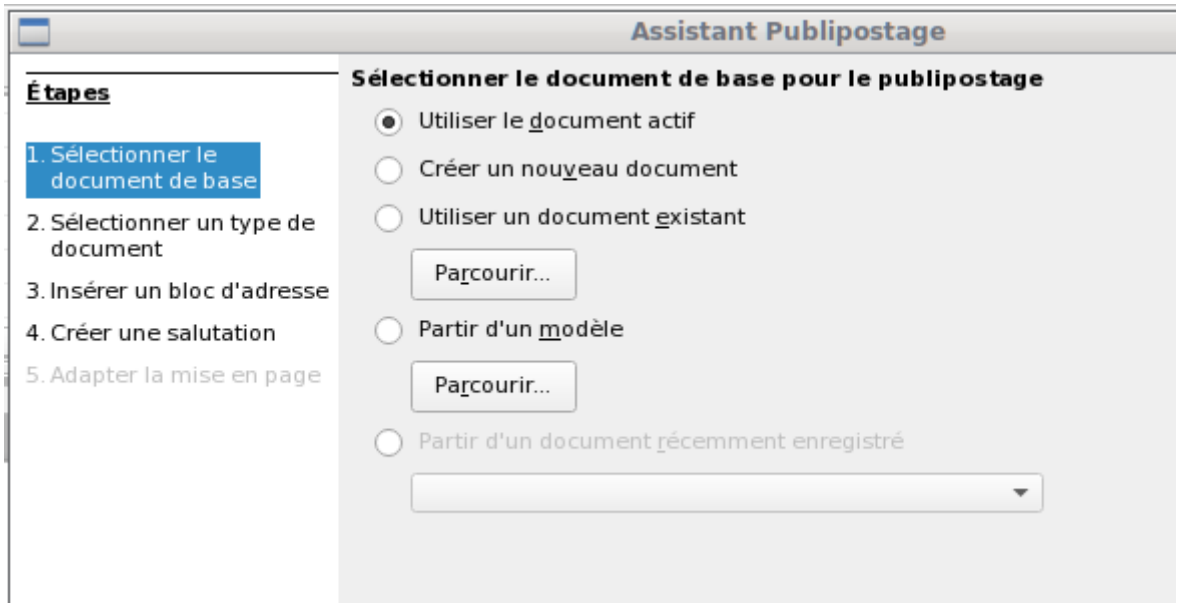
Création de documents de publipostage

L'Assistant Fusion et publipostage est également accessible à partir du navigateur de base de données. Cet assistant permet de construire le champ d'adresse et la civilité à partir d'une source de données en petites étapes. En principe, vous pouvez créer ces champs sans utiliser l'assistant. Ici, nous allons travailler à travers les étapes de l'assistant à titre d'exemple. Les étapes suivantes utilisent l'Assistant pour ce faire. Cette fois, une requête sera la source de données, en particulier les adresses des lecteurs (**Adresses_Lecteurs**). Recherchez-la dans la liste déroulante Requête comme vous l'avez fait précédemment pour la table Lecteurs dans la liste déroulante Table.

Conseil

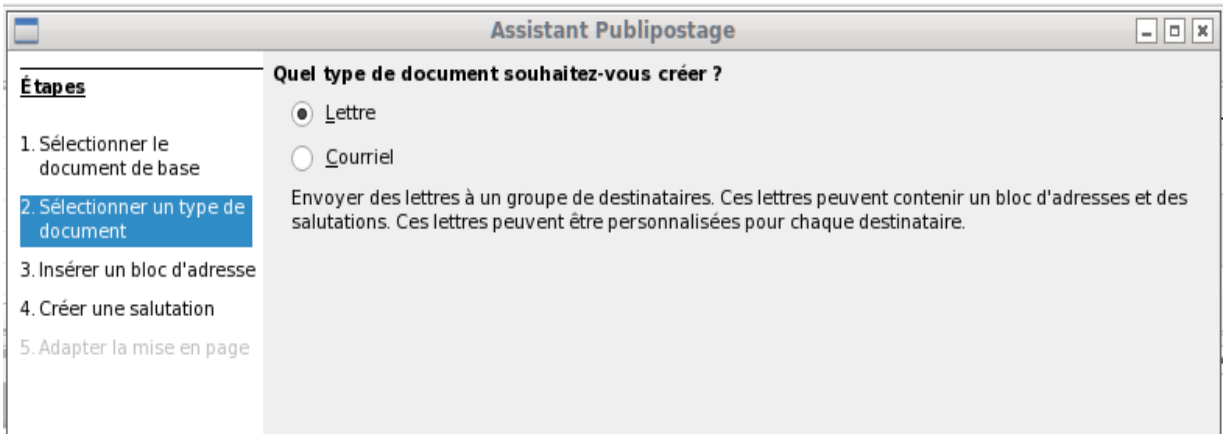
Supprimez tout document texte dans Writer s'il contient des liens vers une base de données. Vous ne pouvez pas établir un nouveau lien vers ce document lorsque l'ancien est toujours actif. Commencez par un nouveau document, qui peut être sans titre, ou un modèle de lettre qui ne contient aucun lien.





Le *document de base* de la lettre type est le document auquel les *champs de la base de données seront liés*.

Le *document fusionné* est celui qui contient les données des différentes personnes qui doivent recevoir les lettres types. Dans le document fusionné, il n'y a *pas de lien* avec la source de données. Il est similaire à la sortie de **Insérer des données comme texte**.



L'assistant de publipostage peut produire des lettres ou des courriels (e-mails) à l'aide d'enregistrements de la base de données. Dans cet exemple, nous allons créer des lettres à l'aide de la table Lecteurs de la base de données Media.

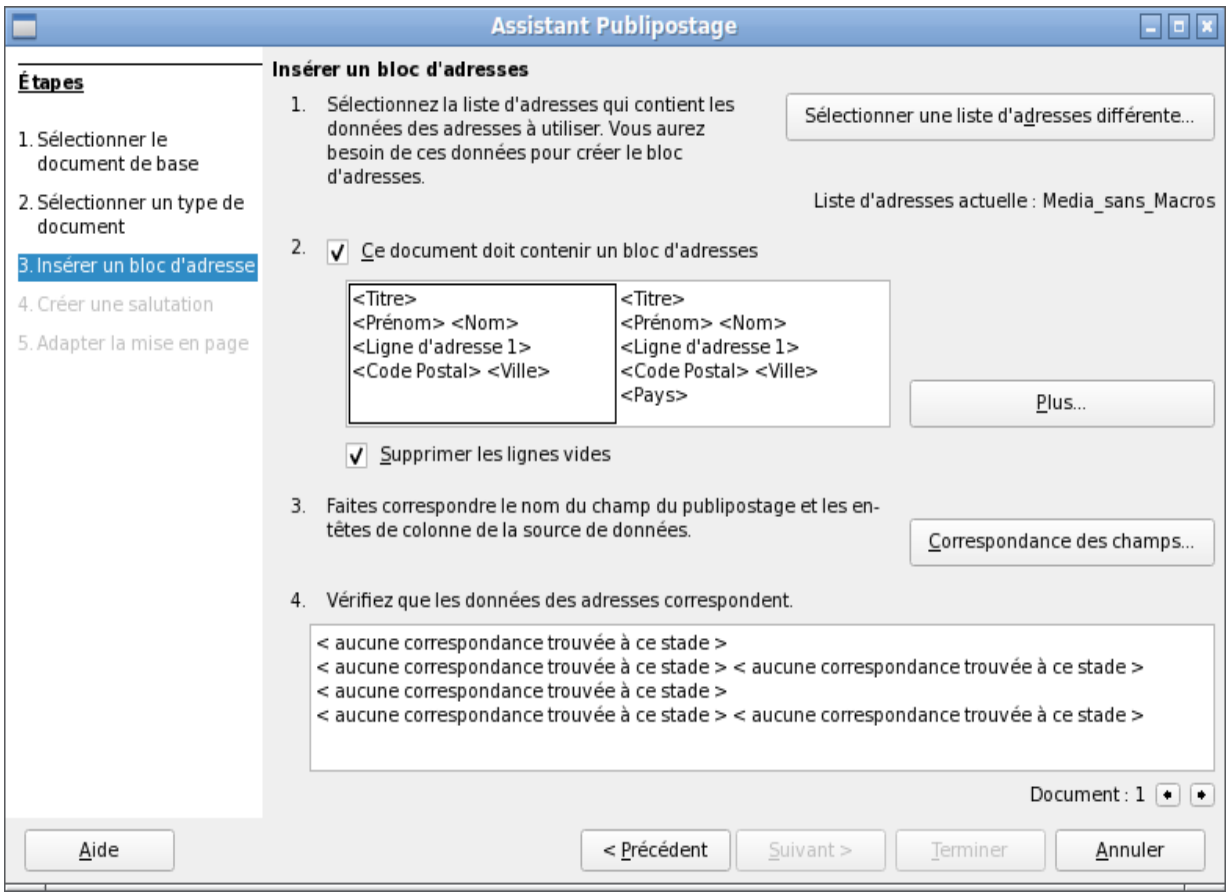
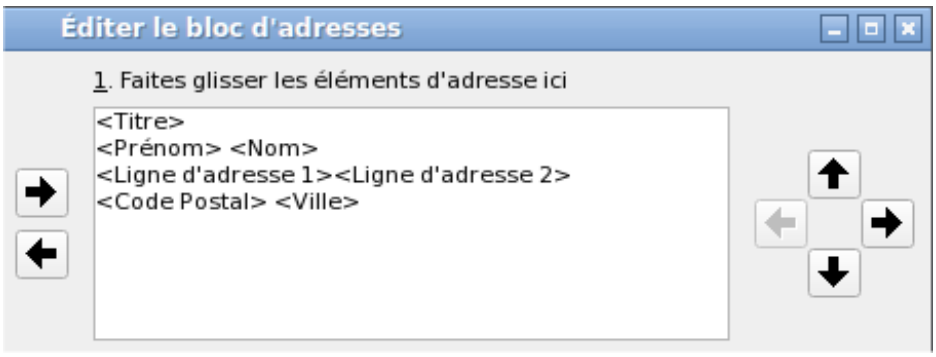


Figure 107: Insertion du bloc d'adresse.

La saisie du bloc d'adresse permet la configuration la plus étendue. La liste d'adresses suggérée provient de la requête ou de la table actuellement sélectionnée dans la base de données actuellement sélectionnée.

L'étape 2 détermine l'aspect général du bloc d'adresses, qui peut être personnalisé davantage en cliquant sur le bouton **Plus**. Voir la Figure 107. L'adresse de gauche est déjà sélectionnée et ce bloc sera utilisé.



Un élément doit être ajouté : <Ligne d'adresse 2> (il faut le numéro et la rue) . Pour faire cela :

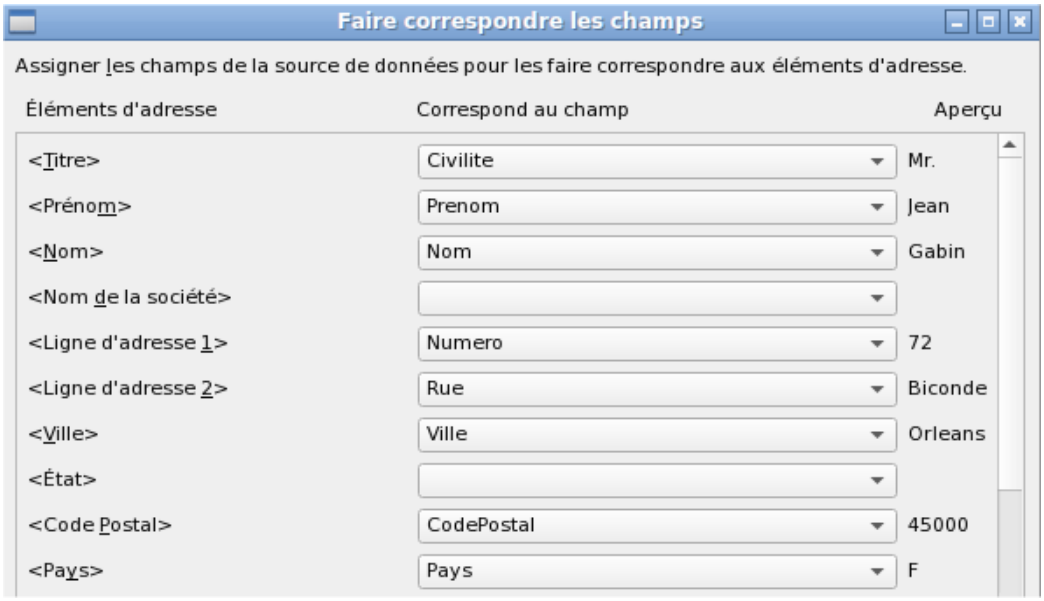
1. Cliquez sur le bouton **Plus**.
2. Cliquez sur le bouton **Editer**.
3) Faites glisser l'élément <Ligne d'adresse 2> dans la liste des éléments d'adresse pour le placer à droite de <Ligne d'adresse 1>.
4. S'il n'y a pas d'espace entre ces deux éléments, cliquez sur la flèche droite sur le côté droit de la boîte de dialogue pour créer l'espace.
5. Cliquez **OK**.
6. Sélectionnez le bloc d'adresse : cliquez **OK**.

L'élément <Titre> doit être déplacé d'une ligne vers le bas en le plaçant avant <Prénom> <Nom>. Assurez-vous de mettre un espace entre <Titre> et <Prénom>. Utilisez les quatre flèches à droite pour déplacer d'abord <Nom>, puis <Prénom>.

L'étape 3 sert à lier les champs nommés dans le bloc d'adresses aux champs corrects de la base de données. L'Assistant reconnaît initialement uniquement les champs de base de données qui ont exactement les mêmes noms que ceux que l'Assistant utilise. Dans cet exemple, quasiment aucun des champs ne correspond, donc tous devront être sélectionnés dans les listes déroulantes de cette étape.

Cliquez le bouton **Correspondance des champs**.

- Pour <Titre>, sélectionnez Civilite.
- Pour <Prénom> sélectionnez Prenom.
- Pour <Nom> sélectionnez Nom.
- Pour <Ligne d'adresse 1> sélectionnez Numero.
- Pour <Ligne d'adresse 2> sélectionnez Rue.
- Pour <Ville> sélectionnez Ville.
- Pour <Code Postal> sélectionnez CodePostal.



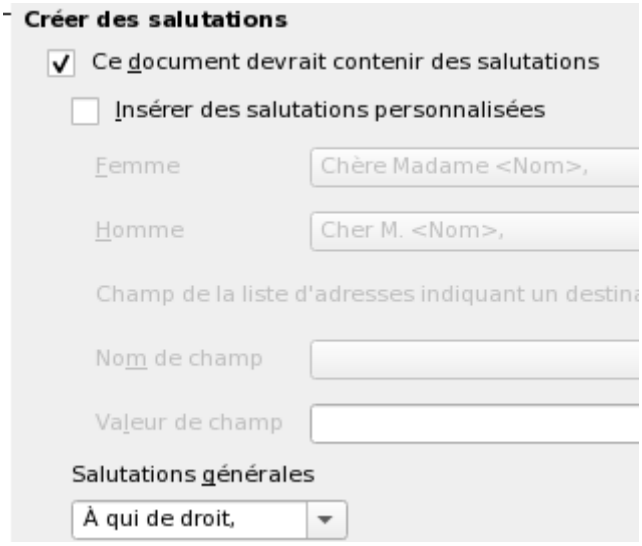
Ici, les éléments d'adresse sont associés aux éléments correspondants de la requête de la base de données transférée avec succès par l'assistant de publipostage. Là encore, le premier enregistrement de la requête est utilisé pour l'aperçu.

Les paramètres de la base de données se terminent essentiellement par l'étape 4. Ici, il s'agit simplement de choisir dans quel champ le sexe du destinataire doit être pris. Ce champ a déjà été nommé, de sorte que seul le contenu du champ pour une destinataire féminine devrait encore être spécifié. Ce sujet est traité à la page 354, Créer des lettres types en sélectionnant des champs

Note

Étant donné que l'assistant a un bogue à ce stade, la salutation personnelle est créée à l'aide de données en texte comme décrit ci-dessus. Plus précisément, le champ salutation fournira le titre approprié pour chaque personne. Le reste de la salutation est créé en le tapant sur le document de fusion ou en y insérant des champs.

1. Pour terminer cette page, décochez *Insérer une salutation personnalisée*.
2. Ne changez rien aux salutations générales. Elles seront remplacées plus tard, mais il est nécessaire d'identifier où les salutations doiventt être dans la lettre.



Cliquez sur **Suivant**>. À l'étape 5, vous pouvez ajuster la position du bloc d'adresse et la salutation sur la page. (Voir la figure 108.) Cliquez ensuite sur Terminer.

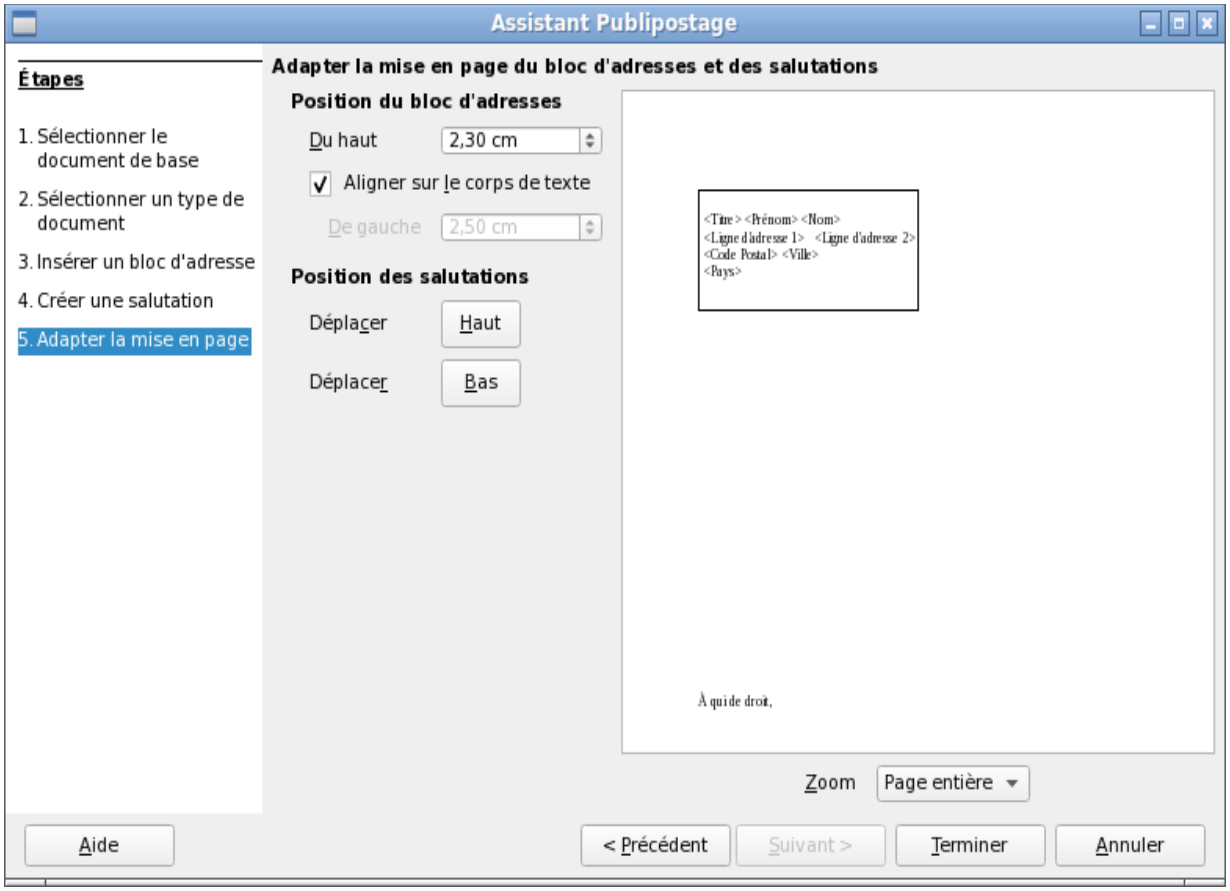
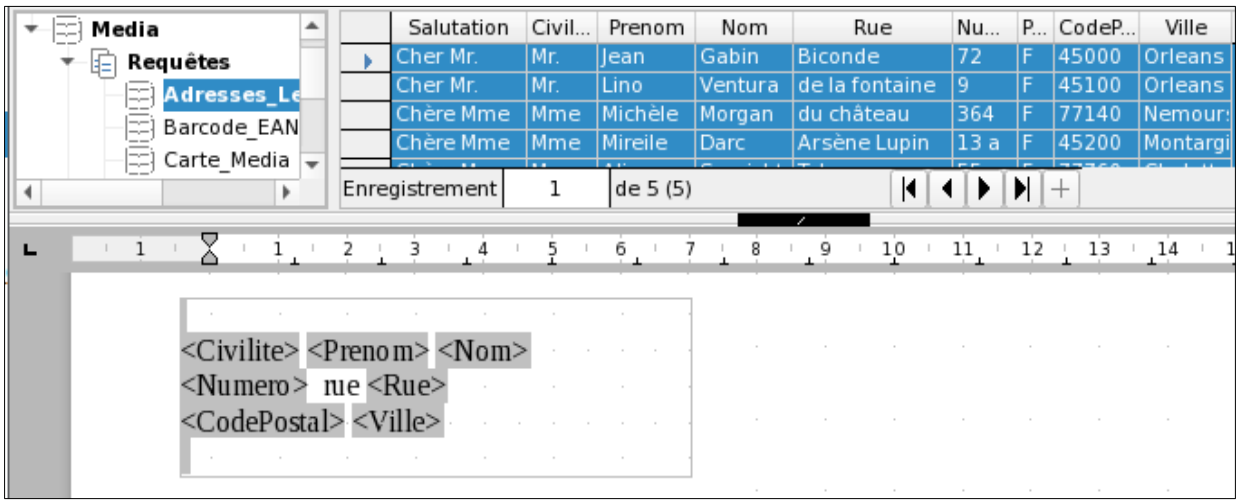


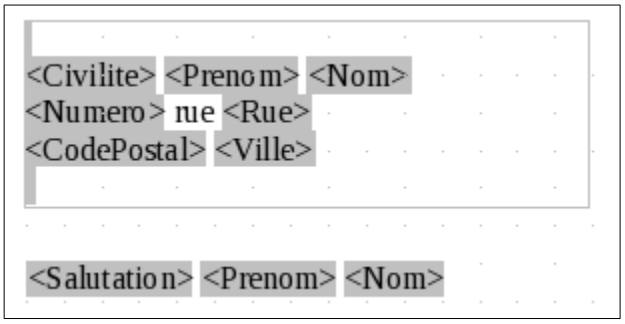
Figure 108: Ajuster la mise en page de la lettre type

Maintenant, pour terminer la mise en page sur le document de fusion et publipostage. Il contient les champs du bloc d'adresses dans lesquels vous les avez placés.

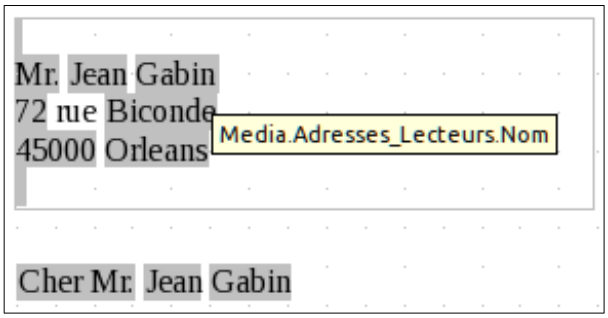


Maintenant, remplacez les salutations.

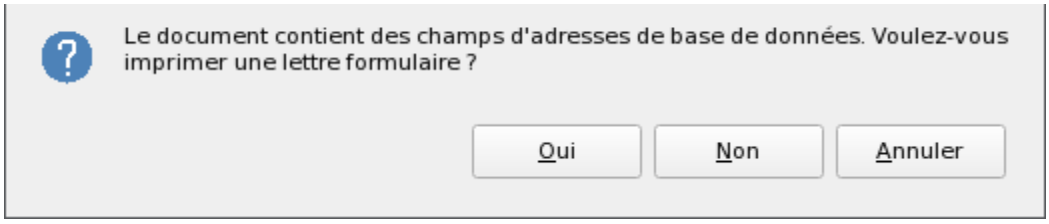
1. Glissez-déposez <Salutation> à la place de "A qui de droit" (depuis lle titre de la colonne)
2. Glissez-déposez <Prenom> dans un espace après <Civilite>.
3. Glissez-déposez <Nom> dans un espace après <Prenom>.



Sélectionnez le premier enregistrement dans la fenêtre Source de données. Cliquez ensuite sur le bouton Données dans les champs pour voir les données saisies dans les champs.



Vous disposez maintenant d'un document Writer dans lequel vous pouvez taper le contenu de la lettre. Pour fusionner les champs et imprimer les lettres, choisissez **Fichier> Imprimer** dans la barre de menus. Le message suivant apparaît. Cliquez sur Oui.



La boîte de dialogue Publipostage (Figure 109) s'affiche maintenant, dans laquelle vous pouvez éventuellement sélectionner les enregistrements à inclure ou exclure, et choisir d'imprimer les lettres ou de les enregistrer dans un fichier. Pour plus de détails, reportez-vous au chapitre Publipostage dans le Guide Writer.

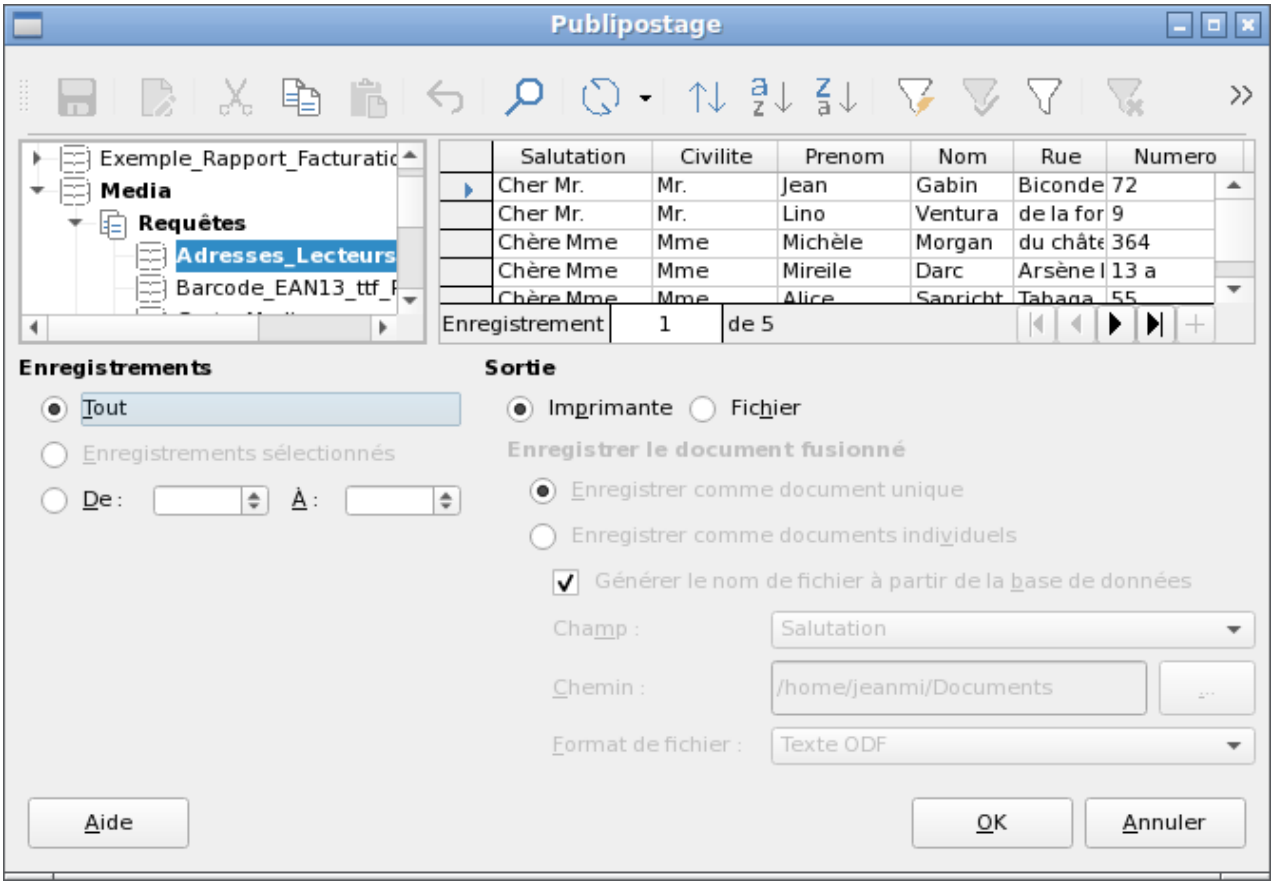


Figure 109: Dialogue publipostage

Impression d'étiquettes

Fichier> Nouveau> Étiquettes lance l'assistant d'étiquettes. Il ouvre une boîte de dialogue, qui comprend toutes les questions de mise en forme et de contenu pour les étiquettes, avant que les étiquettes elles-mêmes ne soient produites. Les paramètres de cette boîte de dialogue sont enregistrés dans les paramètres personnels de l'utilisateur.

Les paramètres de base du contenu se trouvent dans l'onglet Étiquettes (Figure 110). Si pour le texte de l'étiquette vous cochez la case Adresse, toutes les étiquettes auront le même contenu, tiré des paramètres LibreOffice pour l'utilisateur actuel du programme.

A titre d'exemple, nous utiliserons à nouveau la base de données Media_sans_Macros (copiée sous le nom Media pour la circontance). Bien que le champ de sélection suivant soit intitulé Tables,

les tables, les vues et les requêtes sont toutes répertoriées ici, tout comme dans le navigateur de source de données.

Utilisez le bouton fléché pour insérer des champs de base de données individuels dans l'éditeur. Le nom du champ de base de données Nom est défini ici sur <Media. Adresses_Lecteurs.1. Nom>. La séquence est donc <Champ. Table.1. Base>. Les noms de champs étant assez longs, ils sont renvoyés à la ligne suivante après un espace.

Vous pouvez travailler avec le clavier dans l'éditeur. Ainsi, par exemple, vous pouvez insérer un saut de ligne au début, afin que les étiquettes ne soient pas imprimées directement sur le bord supérieur mais que le contenu puisse être imprimé de manière complètement et clairement visible.

Le format peut être sélectionné dans l'onglet Étiquettes. Ici, de nombreuses marques d'étiquettes sont incorporées de sorte que la plupart des autres paramètres de l'onglet Format ne sont pas nécessaires.

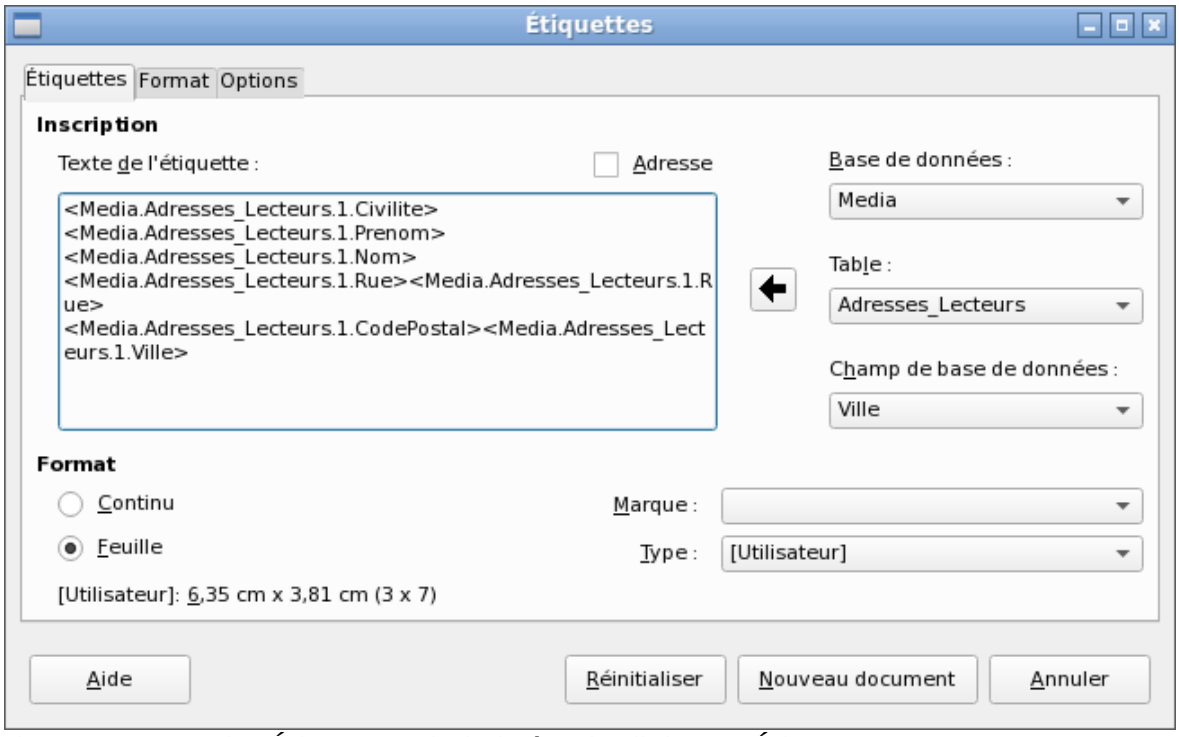


Figure 110: Onglet Étiquettes de la boîte de dialogue Étiquettes

Utilisez l'onglet Format (Figure 111) pour définir la taille de l'étiquette avec précision. Les paramètres ne sont significatifs que lorsque la marque et le type des étiquettes ne sont pas connus. Notez que pour imprimer des étiquettes de 7,00 cm de large, vous avez besoin d'une largeur de page un peu plus grande que 3 * 7,00 cm = 21,00 cm. Ce n'est qu'alors que trois étiquettes seront imprimées d'affilée sur la page.

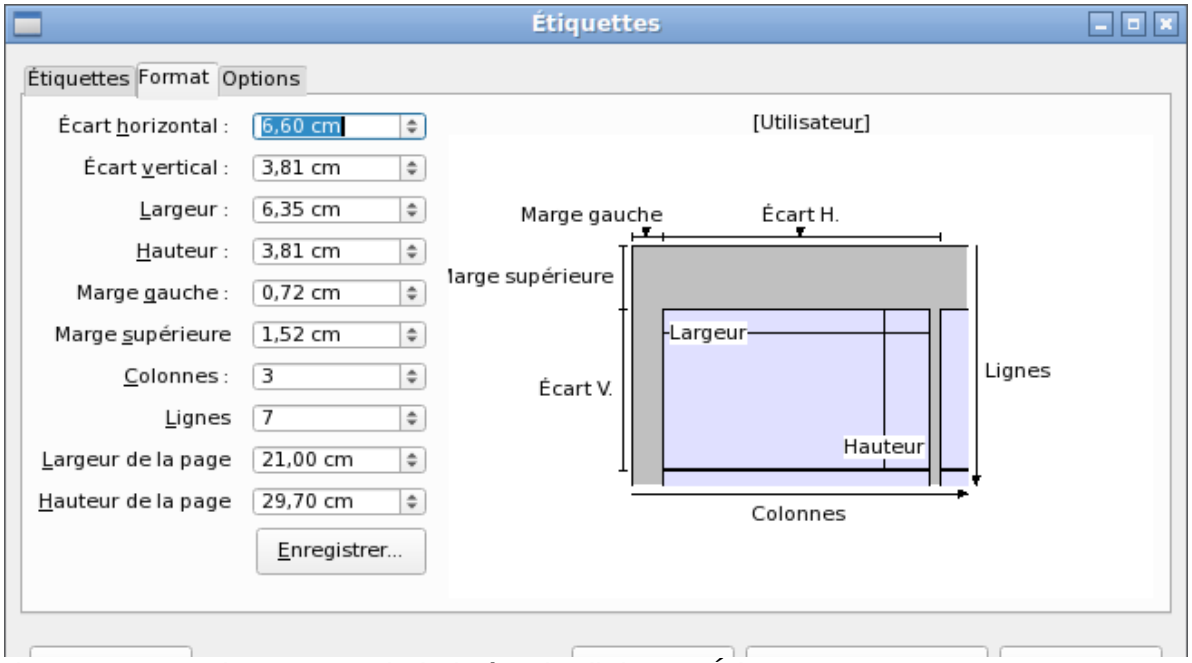


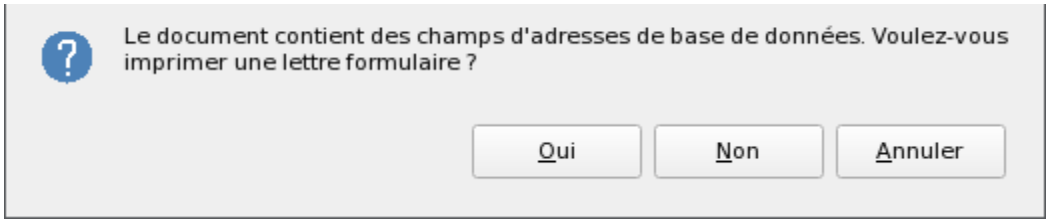
Figure 111: Onglet Format de la boîte de dialogue Étiquettes

Dans l'onglet Options, vous pouvez spécifier si une seule étiquette ou une page entière d'étiquettes sera produite. La page sera alors remplie avec les données des enregistrements successifs de la base de données, en commençant par le premier enregistrement. S'il y a plus d'enregistrements qu'il n'en peut tenir sur la page, la page suivante sera automatiquement remplie avec l'ensemble d'enregistrements suivant.

La case à cocher Synchroniser le contenu relie toutes les étiquettes ensemble afin que les modifications ultérieures de la mise en page de n'importe quelle étiquette soient appliquées à toutes les autres. Pour transférer le contenu modifié, utilisez le bouton Synchroniser, qui apparaît pendant la production de l'étiquette si vous avez coché cette case.

Utilisez le bouton **Nouveau document** pour créer un document contenant les champs sélectionnés.

Lorsque vous lancez le processus d'impression, la question suivante apparaît (comme pour les lettres types) :



Choisissez Oui pour remplir les champs de la base de données d'adresses avec le contenu correspondant.

La source des données pour l'impression d'étiquettes n'est pas trouvée automatiquement ; seule la base de données est présélectionnée. La requête réelle doit être spécifiée par l'utilisateur, car dans ce cas il ne s'agit pas d'une table.

Lorsque la requête est sélectionnée et les enregistrements correspondants choisis (dans ce cas Tous), l'impression peut commencer. Il est conseillé, en particulier pour les premiers tests, de choisir Sortie dans un fichier, dans la boîte de dialogue Publipostage qui s'ouvre ensuite (figure 109), qui enregistrera les étiquettes sous forme de document. L'option de sauvegarde dans plusieurs documents n'est pas appropriée pour l'impression d'étiquettes mais plutôt pour des lettres à différents destinataires qui peuvent ensuite être travaillées par la suite.

Création directe de documents de publipostage et d'étiquettes

Au lieu d'utiliser l'Assistant, vous pouvez produire directement des documents de publipostage et d'étiquettes.

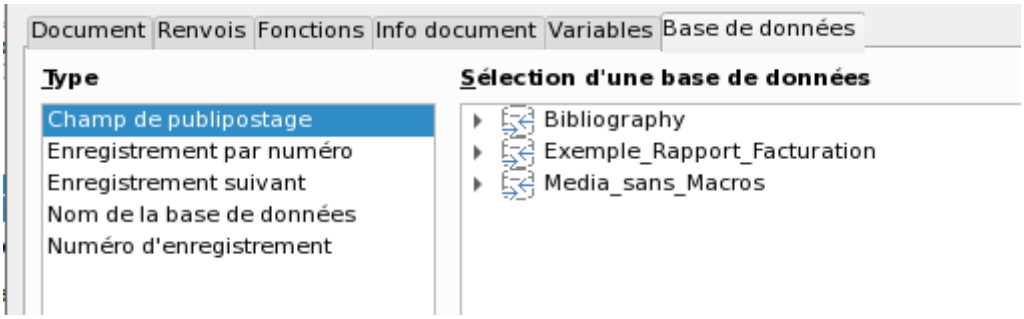
Fusion et publipostage à l'aide de la souris

Les champs de publipostage peuvent être extraits du navigateur de base de données à l'aide de la souris.

Sélectionnez l'en-tête de la table avec le bouton gauche de la souris. Maintenez le bouton enfoncé et faites glisser le curseur dans le document texte. Le curseur change sa forme en symbole d'insertion. Le champ de publipostage est inséré dans le document texte, ici montré dans la description complète qui est rendue visible en utilisant **Affichage> Noms des champs**.

Créer des lettres types en sélectionnant des champs

Les champs de publipostage peuvent être insérés en utilisant **Insertion> Champ> Autres champs> Base de données.**



Ici, toutes les tables et requêtes de la base de données sélectionnée sont disponibles. En utilisant le bouton Insérer, vous pouvez insérer les différents champs l'un après l'autre directement dans le texte à la position actuelle du curseur.

Si vous souhaitez créer une salutation, qui est habituelle dans les lettres types, vous pouvez utiliser un paragraphe masqué ou un texte masqué : **Insertion> Champs> Autres champs> Fonctions> Paragraphe masqué**. Pour les deux variantes, veillez à ce que la condition que vous formulez ne soit pas remplie, car vous voulez que le paragraphe soit visible.

Pour que la formule Chère Madame <Nom de famille> n'apparaisse que lorsque la personne est une femme, une condition suffisante est :

```
[Media.Adresses_Lecteurs.Civilite] ! = "Mme"
```

Maintenant, le seul problème qui reste est qu'il n'y a peut-être pas de Civilite. Dans ces circonstances, "Cher Monsieur / Madame" devrait apparaître, c'est donc la condition que vous devez insérer. L'état général est :

```
[Media.Adresses_Lecteurs.Civilite] ! "Mme" OR NOT
[Media.Adresses_Lecteurs.Civilite]
```

Cela exclut la possibilité que ce paragraphe apparaisse lorsque la personne n'est pas une femme ou qu'il n'y a pas de CiviliteSaisie.

De la même manière, vous pouvez créer des entrées pour le sexe masculin et pour les entrées manquantes pour les deux types de salutation restants.

Naturellement, vous pouvez créer une salutation dans le champ d'adresse exactement de la même manière, quel que soit le sexe.

De plus amples informations sont données dans l'aide de LibreOffice sous Masquage du texte et du texte conditionnel.

Bien sûr, ce serait encore plus simple si le concepteur de la bases de données mettait toute la salutation directement dans la requête. Cela peut être fait en utilisant une sous-requête corrélée (voir le chapitre 5, Requêtes, dans ce livre).

Le type de champ **Enregistrement suivant** (de l'onglet Base de données) est particulièrement intéressant pour les étiquettes. Si ce type de champ est choisi à la fin d'une étiquette, l'étiquette suivante sera remplie avec les données de l'enregistrement suivant. Les étiquettes typiques pour l'impression d'étiquettes séquentielle ressemblent à la figure suivante lorsque vous utilisez **Affichage> Noms de champ** pour rendre visibles les désignations de champ correspondantes :

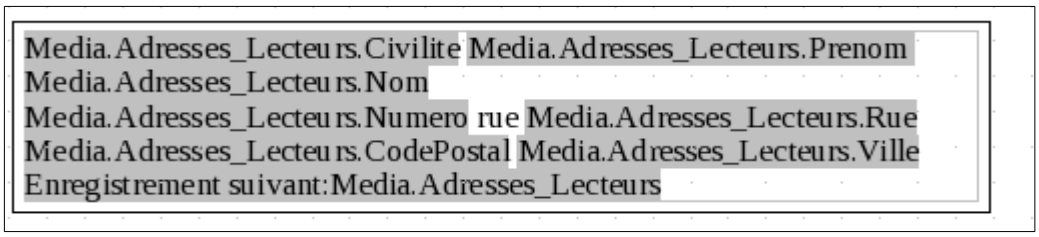


Figure 112: Sélection de champ pour les étiquettes à contenu séquentiel

Pour la dernière étiquette de la page, vous devez tenir compte du fait que l'enregistrement suivant est automatiquement appelé après un saut de page. Ici, le type de champ Enregistrement suivant ne doit pas se produire. Sinon, un enregistrement sera manqué car un double saut d'enregistrement se produit.

Conseil

La création de lettres de publipostage est également possible directement à partir d'un formulaire de base de données. La seule exigence est que la base de données soit enregistrée dans LibreOffice.

Lorsqu'un publipostage est effectué, veillez à choisir Affichage> Normal. Cela garantit que les éléments sont correctement positionnés sur la page. Si un formulaire est ensuite imprimé, la requête de fusion et publipostage habituelle apparaît.

Ce type de fusion et publipostage présente l'avantage que vous n'avez pas besoin de fichiers autres que le fichier *.odb pour imprimer.

Formulaires externes

Si les propriétés de formulaire simples disponibles dans LibreOffice doivent être utilisées dans d'autres composants tels que Writer et Calc, il vous suffit d'afficher la barre d'outils Ébauche de formulaire, en utilisant **Affichage> Barres d'outils> Ébauche de formulaire**, puis ouvrez le Navigateur de formulaire. Vous pouvez créer un formulaire ou, comme décrit dans le chapitre 4, Formulaires, créer un champ de formulaire. L'onglet Données de la boîte de dialogue Propriétés du formulaire est légèrement différent de celui que vous voyez lorsque les formulaires sont créés directement dans un fichier de base de données ODB.

Figure 1: Formulaire avec une source de données externe

Figure 2: Formulaire avec une source de données Iinterne

La source de données doit être sélectionnée séparément lors de l'utilisation d'un formulaire externe. La zone de liste *Source de données*, présente toutes les bases de données déjà enregistrées dans LibreOffice sous leurs noms enregistrés.

Tout fichier ODB peut être sélectionné. Utilisez le bouton à droite de la zone de liste des sources de données pour ouvrir le navigateur de fichiers. Dans ce cas, le champ de la source de données est de la forme URL, commençant par file:///.

Les formulaires sont créés exactement de la même manière que dans Base lui-même.

Les formulaires ainsi produits sont affichés par défaut en mode édition à chaque ouverture du fichier, non protégés en écriture comme dans Base. Pour éviter toute modification accidentelle du formulaire, vous pouvez utiliser **Fichier> Propriétés> Sécurité** pour ouvrir le fichier en lecture seule. Vous pouvez même protéger le fichier contre toute modification à l'aide d'un mot de passe. Dans les systèmes bureautiques, il est également possible de déclarer l'intégralité du fichier comme protégé en écriture. Cela permet toujours la saisie dans les champs du formulaire, mais pas le déplacement des champs ou la saisie de texte entre eux.

Conseil

Les formulaires peuvent également être créés rapidement en utilisant le glisser-déposer. Pour ce faire, ouvrez la base de données, recherchez la table ou la requête appropriée et sélectionnez les en-têtes de table.

Dans Writer, sélectionnez les en-têtes de champ appropriés avec le bouton gauche de la souris, maintenez les touches Maj et Ctrl enfoncées et le curseur de la souris se transforme en symbole de lien. Faites ensuite glisser les en-têtes dans le document Writer.

Vous pouvez faire glisser les champs dans des fichiers Calc sans utiliser les touches supplémentaires. Le symbole de copie apparaît sous la forme d'un curseur de souris.

Dans les deux cas, un champ de saisie est créé avec le libellé associé. Le lien vers la source de données est créé avec la première entrée réelle de données. Ainsi, l'entrée de données dans un tel formulaire peut commencer immédiatement après l'opération de glisser-déposer.

Avantages des formulaires externes

La base n'a pas besoin d'être ouverte en premier pour fonctionner avec la base de données. Par conséquent, vous n'avez pas besoin d'une fenêtre ouverte supplémentaire en arrière-plan.

Dans une base de données déjà complète, les utilisateurs de base de données existants peuvent ensuite recevoir sans problème le formulaire amélioré. Ils peuvent continuer à

utiliser la base de données pendant le développement d'autres formulaires et n'ont pas besoin de copier des formulaires externes complexes d'une base de données dans une autre.

Les formulaires d'une base de données peuvent être modifiés en fonction de l'utilisateur. Les utilisateurs qui ne sont pas autorisés à corriger des données ou à faire de nouvelles entrées peuvent recevoir un ensemble de données actuel par d'autres utilisateurs, et simplement remplacer leur fichier *.odb pour avoir une vue à jour. Cela pourrait, par exemple, être utile pour une base de données pour une organisation où tous les membres du comité obtiennent la base de données mais une seule personne peut éditer les données ; les autres peuvent toujours consulter les adresses de leurs services respectifs.

Inconvénients des formulaires externes

Les utilisateurs doivent toujours installer les formulaires et Base avec la même structure de répertoires. C'est la seule façon dont l'accès à la base de données peut être exempt d'erreurs. Les liens étant stockés par rapport au formulaire, il suffit de stocker la base de données et ses formulaires dans un répertoire commun.

Seuls les formulaires peuvent être créés en externe, pas les requêtes ou les rapports. Un simple coup d'œil sur une requête doit donc passer par un formulaire. Un rapport nécessite en revanche l'ouverture de la base de données. Il est également possible de le créer, au moins partiellement, à l'aide du publipostage.

Utilisation d'une base de données dans Calc

Les données peuvent être utilisées dans Calc à des fins de calcul. Pour cela, il est d'abord nécessaire de rendre les données accessibles dans une feuille de calcul Calc.

Saisie des données dans Calc

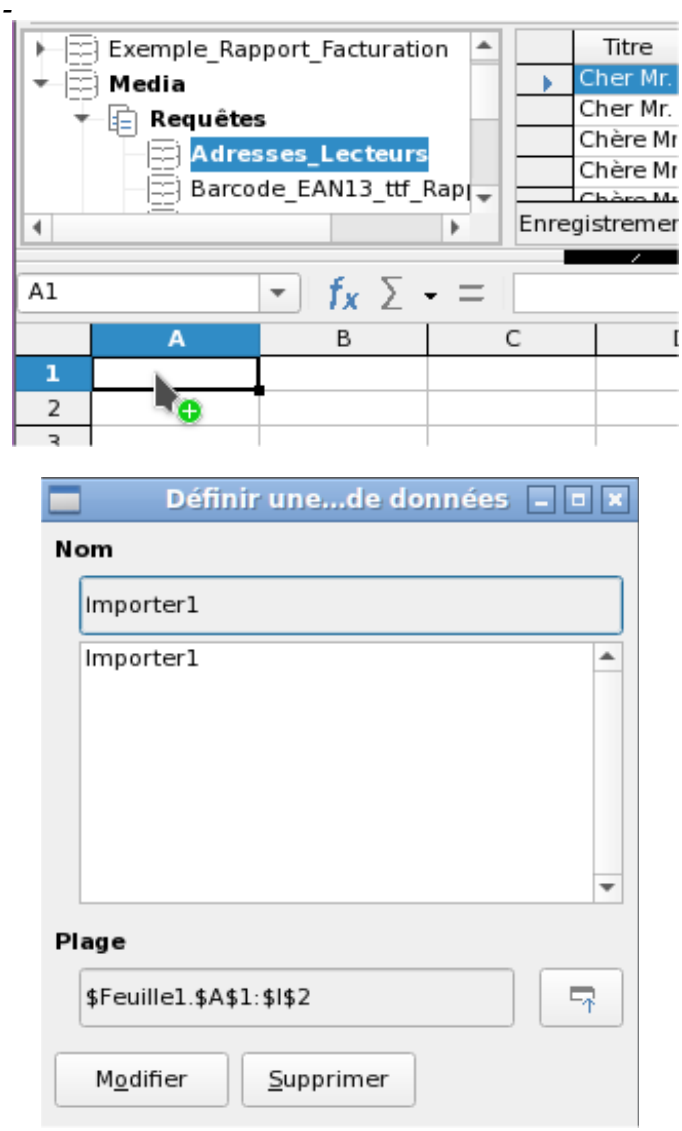
Il existe différentes manières d'entrer des données dans Calc.

Sélectionnez une table avec le bouton gauche de la souris et faites-le glisser dans une feuille de calcul Calc. Le curseur est amené dans le coin supérieur gauche du tableau de la feuille. Le tableau est créée avec les noms de champs. Dans ce cas, le navigateur de source de données n'offre pas les options *Données dans le texte* ou *Données dans les champs*.

Les données glissées dans Calc de cette manière montrent les propriétés suivantes :

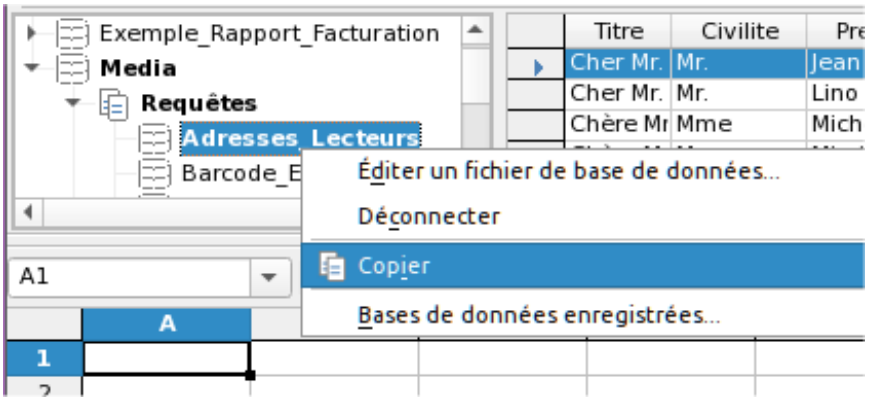
Non seulement les données sont importées, mais les propriétés du champ sont également lues et traitées lors de l'importation. Les champs tels que les numéros de maison, qui ont été déclarés comme champs de texte, sont mis en forme sous forme de texte après insertion dans Calc.

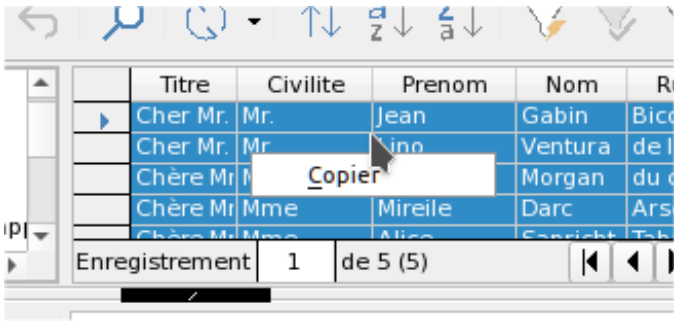
L'importation devient une plage Calc, qui par défaut reçoit le nom Importer1. Les données peuvent être consultées ultérieurement en utilisant cette plage. **Plage de données> Actualiser** permet à la plage, le cas échéant, de recevoir de nouvelles données de la base de données.



Les données importées ne sont pas formatées sauf si les propriétés des champs de la base de données l'exigent.

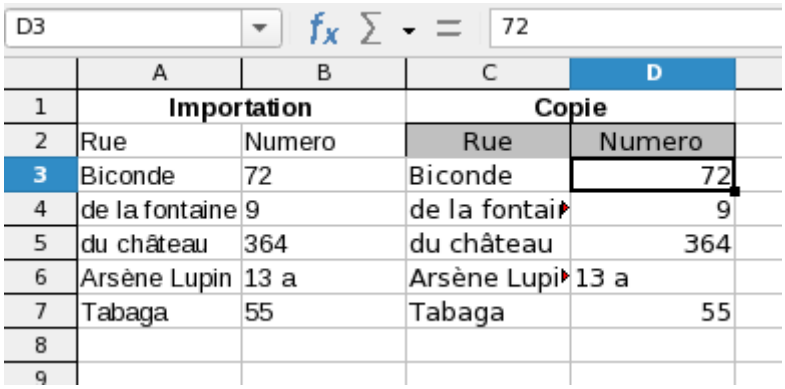
Vous pouvez également utiliser le menu contextuel d'une table pour faire une copie des données. Dans ce cas, cependant, il n'y a pas d'importation mais simplement une copie. Les propriétés des champs de données ne sont pas lues avec eux mais sont déterminées par Calc. De plus, les noms de champ sont formatés comme des en-têtes de tableau.





Vous voyez la différence, en particulier dans les champs de base de données formatés sous forme de texte. Lors de l'importation, Calc les transforme en champs de texte, alignés à gauche comme tout autre texte. Ces nombres ne peuvent alors plus être utilisés dans les calculs.

Si vous les exportez à nouveau, les données restent telles quelles.



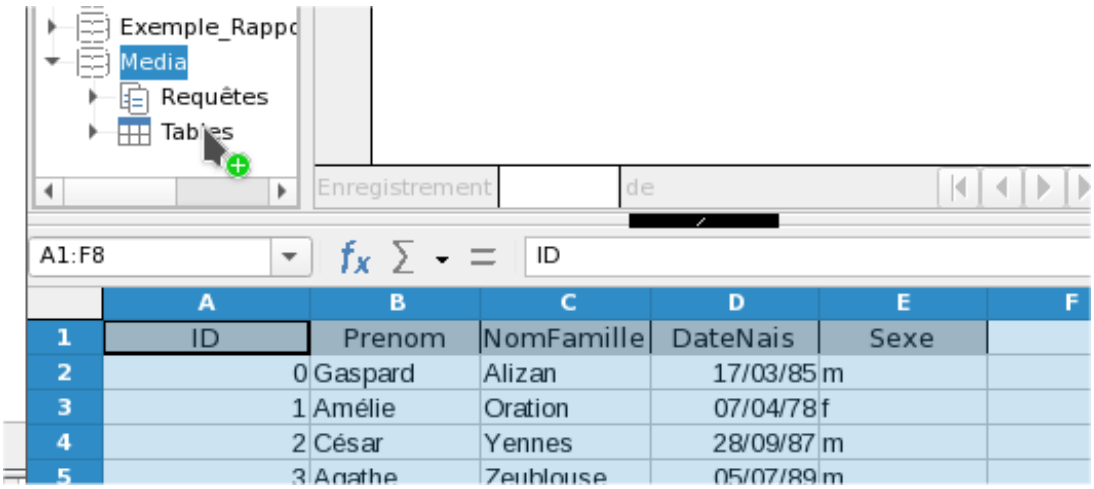
Conseil

L'importation de données dans Calc écrase le contenu précédent ainsi que tout formatage précédent. Si les données doivent être exportées de manière cohérente dans la même table, vous devez utiliser une feuille distincte pour l'importation des données. Les données sont ensuite lues dans l'autre feuille en utilisant le terme Nom_Table. Nom_Champ. Les champs de cette feuille peuvent être formatés de manière appropriée sans risque que le formatage soit écrasé.

Les enregistrements peuvent également être copiés directement depuis la base de données à l'aide du presse-papiers, ou par glisser-déposer à l'aide de la souris. Si une table ou une requête est glissée dans une feuille de calcul Calc, l'ensemble du contenu est inséré. Si la table ou la requête est ouverte et qu'un ou plusieurs enregistrements sont sélectionnés, seuls ces enregistrements avec les noms de champ sont copiés lorsque vous les faites glisser.

Exporter des données de Calc dans une base de données

Sélectionnez les données dans la feuille de calcul Calc. Maintenez le bouton gauche de la souris enfoncé et faites glisser les données que vous souhaitez transformer en base de données dans la zone de table du navigateur de base de données.



Le curseur change d'apparence, montrant que quelque chose peut être inséré. La première fenêtre de l'assistant d'importation s'ouvre. Les étapes suivantes de l'assistant sont décrites dans le chapitre 3, Tables, dans la section "Importation de données à partir d'autres sources".

Conversion de données d'une base de données à une autre

Dans l'explorateur du navigateur de sources de données, les tables peuvent être copiées d'une base de données à une autre en sélectionnant la table source avec le bouton gauche de la souris, en maintenant le bouton enfoncé et en la faisant glisser dans la base de données cible dans le conteneur de table. Cela provoque l'affichage de la boîte de dialogue de copie des tables.

De cette manière, par exemple, des bases de données en lecture seule (sources de données telles que des carnets d'adresses d'un programme de messagerie électronique ou d'un tableur) peuvent être utilisées comme base pour une base de données dans laquelle les données deviennent modifiables. Les données peuvent également être directement copiées lors du passage à un autre programme de base de données (par exemple en passant de PostgreSQL à MySQL).

Si vous souhaitez que la nouvelle base de données ait des relations différentes de l'originale, vous pouvez organiser cela en utilisant les requêtes appropriées. Ceux qui ne sont pas suffisamment experts peuvent à la place utiliser Calc. Faites simplement glisser les données dans une feuille de calcul et préparez-les pour l'importation dans la base de données cible à l'aide des fonctionnalités fournies par Calc.

Pour une importation la plus propre possible dans une nouvelle base de données, les tables doivent être préparées à l'avance. Cela permet de reconnaître longtemps à l'avance les problèmes de formatage et ceux impliquant la création de clés primaires.

Importation d'enregistrements dans une table à l'aide du presse-papiers

Si les enregistrements sont disponibles sous forme de tableau, ils peuvent être insérés dans Base à l'aide du presse-papiers et de l'assistant.

Dans Base, un clic droit sur la table de destination démarre l'importation. Dans le menu contextuel sous **Copier** se trouvent les commandes **Importer** et **Importer du contenu**. Si vous choisissez

Coller, l'assistant d'importation aura déjà sélectionné la table et *Ajouter les données*. **Collage spécial** donne uniquement une requête pour un filtre d'importation. Les options disponibles sont HTML et RTF.

Si à la place vous cliquez avec le bouton droit dans le conteneur de table, l'assistant d'importation vous donne uniquement le choix de créer une nouvelle table.

Importation d'enregistrements PDF

Si vous souhaitez importer des données à partir de diverses sources externes, il est préférable de choisir un format qui empêche votre formulaire d'être modifié lors de la saisie des données. À l'aide de Writer, vous pouvez créer des formulaires au format PDF, les mettre en ligne et vous faire renvoyer les formulaires remplis, par exemple sous forme de pièces jointes à un courriel. Tout ce qui manque, c'est la saisie la plus simple possible des données dans Base. La base Exemple_Import_Formulaire_PDF illustre un tel moyen d'importation.

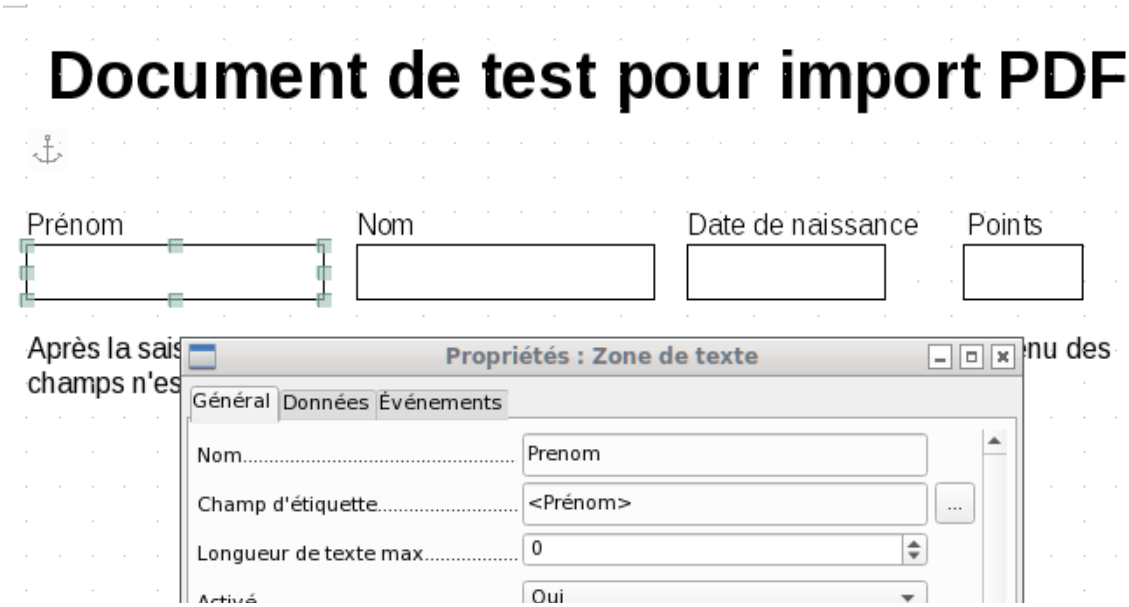
Créer un formulaire PDF

Un formulaire PDF est créé en tant que formulaire externe sans lien vers la base de données. À l'aide de **Affichage> Barres d'outils> Contrôles de formulaire**, les éléments nécessaires au formulaire sont affichés et peuvent être insérés si nécessaire.

Malheureusement, le format PDF ne fait aucune distinction entre les champs numériques, les champs de date et les champs de texte. Pour l'exemple fourni ici, il suffit d'utiliser des champs de texte pour toutes les entrées. Les autres formats de champ du formulaire Writer seront inévitablement perdus lors de l'exportation PDF.

Fondamentalement, les formulaires PDF peuvent contenir les champs suivants :

- Boutons
- Zone de texte
- Case à cocher
- Boîte combinée
- Zone de liste



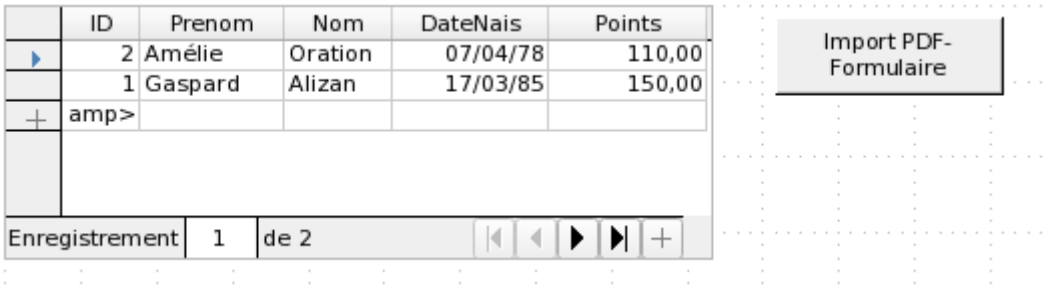
Le formulaire de test contient un total de 4 champs de texte. Dans **Propriétés : Zone de texte> Général> Nom**, vous devez toujours choisir le nom de champ utilisé dans la table de base de données lors de l'utilisation de la méthode d'importation suivante afin d'éviter des problèmes avec les noms de champ et le contenu des champs.

Les messages d'aide s'affichent lorsque les enregistrements sont lus, mais n'apparaissent pas dans toutes les visionneuse PDF.

Pour garantir que le formulaire contient réellement les enregistrements, il doit être enregistré dans la visionneuse PDF, après la saisie des données, à l'aide de l'option de menu **Fichier > Enregistrer sous**. La commande réelle pour ce faire peut varier selon les visualiseurs. Sans cette procédure, le visualiseur affichera les enregistrements une fois le formulaire ouvert sur votre propre ordinateur, mais il les lit en fait à partir du fichier de stockage temporaire du visualiseur et non directement à partir du fichier PDF. Si le formulaire est ensuite transféré sur un autre ordinateur, il sera vide.

Lire les enregistrements du formulaire PDF

Le formulaire de la base de données de base est très simple en apparence. Il est lié à la table et affiche les enregistrements qui viennent d'être lus. Les entrées les plus récentes sont affichées dans le champ de table ci-dessus.



La macro de lecture des enregistrements est saisie sous **Propriétés : Bouton> Evénements> Exécuter l'action**.

Note

Pour lire les enregistrements, nous utilisons le programme open source pdftk. Le programme est disponible gratuitement pour Windows et Linux. Les distributions Linux l'ont principalement sous forme de package dans leurs référentiels. Les utilisateurs de Windows le trouveront à https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/

Les enregistrements lus à l'aide de pdftk sont écrits dans un fichier texte, qui ressemble à ceci :

```
---
FieldType: Text
FieldName: Prenom
FieldFlags: 0
FieldValue: Gaspard
FieldJustification: Left
---
FieldType: Text
FieldName: Nom
FieldFlags: 0
FieldValue: Alizan
FieldJustification: Left
---
FieldType: Text
FieldName: DateNais
FieldNameAlt: Date, year minimum 2 places
FieldFlags: 0
FieldValue: 17/03/85
FieldJustification: Left
---
FieldType: Text
FieldName: Points
FieldNameAlt: Decimal value, 2 decimal places
FieldFlags: 0
FieldValue: 150,00
FieldJustification: Left
```

Chaque champ est représenté par cinq à six lignes dans le fichier. Pour la macro, les lignes importantes sont **FieldName** (doit être identique à **Nom de Champ** dans la table de destination), **FieldValue** (contenu du champ après avoir enregistré le fichier PDF) et **FieldJustification** (dernière ligne de l'entrée).

L'ensemble du processus d'importation est contrôlé par des macros. Le formulaire PDF doit être sur le même chemin que la base de données. Les enregistrements sont lus dans le fichier texte, puis écrits dans la base de données à partir de ce fichier texte. Cela continue pour tous les fichiers PDF du dossier. Les anciens enregistrements doivent donc être supprimés du dossier dans la mesure du possible, car la fonction ne vérifie pas la duplication.

```
SUB Import_Formulaire_PDF(oEvent AS OBJECT)
	DIM inNombre AS INTEGER
	DIM stLigne AS STRING
	DIM i AS INTEGER
	DIM k AS INTEGER
	DIM oSourceDonnees AS OBJECT
	DIM oConnexion AS OBJECT
	DIM oSQL_Commande AS OBJECT
	DIM oResult AS OBJECT
	DIM stSql AS STRING
	DIM oDB AS OBJECT
	DIM oAccesFichier AS OBJECT
	DIM inChamps AS INTEGER
	DIM stTable AS STRING
	DIM stNomChamp AS STRING
	DIM stValeurChamp AS STRING
	DIM stDir AS STRING
	DIM stDir2 AS STRING
	DIM stFormulairePDF AS STRING
	DIM stFichier AS STRING
	DIM inNull AS INTEGER
	DIM aFichiers()
```

```
        DIM aNull()
        DIM stCommande AS STRING
        DIM stParametre AS STRING
        DIM oShell AS OBJECT
```

Une fois les variables déclarées, le nombre de champs du formulaire PDF est indiqué. Le décompte commence à 0, donc une valeur de 3 signifie en fait un total de quatre champs. En utilisant ce décompte, il peut être déterminé si toutes les données d'un enregistrement ont été lues, de sorte qu'il est prêt à être transféré dans la table.

```
    inChamps = 3 'Nombre de champs dans le pdf. Il y a donc 4 champs.
        stTable = "Noms" 'Table qui doit enregistrer les données du formulaire
        REM La connexion à la base de données doit être vérifiée. Si non connecté:
conneion.
        oSourceDonnees = ThisComponent.Parent.CurrentController
        If NOT (oSourceDonnees.isConnected()) THEN
           oSourceDonnees.connect()
        END IF
        REM SQL-l'entrée doit être préparée
        oConnexion = oSourceDonnees.ActiveConnection()
        oSQL_Commande = oConnexion.createStatement()
```

La connexion à la base de données est établie. Le chemin d'accès au fichier de base de données dans le système de fichiers est lu. En utilisant ce chemin, le contenu du dossier est lu dans le tableau aFiles. Une boucle vérifie chaque nom de fichier du tableau pour voir s'il se termine par.pdf. Les majuscules et les minuscules ne sont pas distinguées, car les résultats de la recherche sont tous convertis en minuscules à l'aide de Lcase.

```
    oDB = ThisComponent.Parent
    stDir = Left(oDB.Location,Len(oDB.Location)-Len(oDB.Title))
    oAccesFichier = createUnoService("com.sun.star.ucb.SimpleFileAccess")
    aFichiers = oAccesFichier.getFolderContents(stDir, False)
    FOR k = 0 TO uBound(aFichiers())
          IF LCase(Right(aFichiers(k),4)) = ".pdf" THEN
                stDir2 = ConvertFromUrl(stDir)
                stFormulairePDF = ConvertFromUrl(aFichiers(k))
```

Pour déterminer la commande de lecture des données, il est nécessaire de comprendre les conventions d'adresse de fichier du système d'exploitation. Par conséquent, l'URL d'origine commençant par file : // doit être adaptée au système actuel. La commande de démarrage du programme pdftk dépend du système d'exploitation. Il peut porter le suffixe.exe ou peut-être un chemin complet vers le programme comme `C:\Program Files (x86)\pdftk\pdftk.exe` ou le suffixe peuvent ne pas être du tout requis. GetGuiType est utilisé pour déterminer le type de système utilisé : 1 signifie Windows, 3 pour macOS et 4 pour Linux. Les étapes suivantes distinguent uniquement Windows des autres.

Après cela, la fonction Shell () est utilisée pour passer la commande de lancement appropriée pour pdftk à la console. L'argument True garantit que LibreOffice attendra la fin du processus shell.

```
                    IF GetGuiType = 1 THEN '(1: Windows, 3: MAC, 4: Unix/Linux, -1:
indéfini, non pris en charge)
                          stCommande = "pdftk.exe" 'Si le programme n'est pas installé
via la configuration du système Windows, il peut s'agir de: "C:\Program Files(x86)\pdftk\
pdftk.exe"
                    ELSE
                          stCommande = "pdftk"
                    END IF
                    stParametre = stFormulairePDF & " dump_data_fields_utf8 output "&
stDir2 &"PDF_Form_Data.txt"
                    Shell(stCommande,0,stParametre,True)
                    stFichier = stDir & "PDF_Form_Data.txt"
                    i = -1
```

```
                inNombre = FreeFile 'Directement après avoir recherché le canal de
données libre du système pour le fichier, le fichier doit être ouvert.
```

La fonction FreeFile détermine quel est le prochain canal de données libre disponible dans le système d'exploitation. Ce canal est lu sous forme de nombre entier et utilisé pour se connecter directement au fichier de données PDF qui vient d'être créé. L'instruction INPUT est utilisée pour lire le fichier. Cela a lieu en dehors de LibreOffice. Les enregistrements externes sont ensuite lus dans LibreOffice.

```
            OPEN stFichier FOR INPUT AS inNombre
                DO WHILE NOT Eof(inNombre)
                    LINE INPUT #inNombre, stLigne    'le canal dedonnées doit être
écrit avec #
```

Le fichier de données PDF est maintenant lu ligne par ligne. Chaque fois que le terme **FieldName** apparaît, le contenu restant de la ligne est considéré comme le nom du champ dans le formulaire PDF et aussi, en raison de la façon dont le formulaire a été défini, le nom du champ de base de données dans lequel les données doivent être écrites.

Tous les noms de champ sont combinés directement pour être utilisés dans les commandes SQL ultérieures. En pratique, cela signifie que les noms de champs sont entre guillemets et séparés par des virgules.

De plus, pour chaque nom de champ, une requête détermine le type de champ dans la table. La date et les valeurs décimales doivent être transférées d'une manière différente du texte.

```
                IF instr(stLigne, "FieldName: ") THEN
                    IF stNomChamp = "" THEN
                        stNomChamp = """"+mid(stLigne,12)+""""
                    ELSE
                        stNomChamp = stNomChamp & ","""" + mid(stLigne,12)+""""
                    END IF
                    stSql = "SELECT TYPE_NAME FROM INFORMATION_SCHEMA.SYSTEM_COLUMNS
WHERE "
                    stSql = stSql + "TABLE_NAME = '" + stTable + "' AND COLUMN_NAME =
'"+mid(stLigne,12)+"'"
                    oResult = oSQL_Commande.executeQuery(stSql)
                    WHILE oResult.next
                        stFieldType = oResult.getString(1)'Premier champ de données
                    WEND ' Ligne suiivante
                END IF
```

Quant aux noms de champs, il en va de même pour les valeurs des champs. Cependant, ceux-ci ne doivent pas être entre guillemets mais doivent être préparés conformément aux exigences du code SQL. Cela signifie que le texte doit être entre guillemets simples, les dates doivent être converties pour se conformer aux conventions SQL, etc. Ceci est fait par la fonction SQL_Value externe supplémentaire.

```
                IF instr(stLigne, "FieldValue: ") THEN
                       IF stValeurChamp = "" THEN
                          stValeurChamp = SQL_Value(mid(stLigne,13), stFieldType)
                       ELSE
                          stValeurChamp = stValeurChamp & "," &
SQL_Value(mid(stLigne,13), stFieldType)
                       END IF
                    END IF
```

Si le terme **FieldJustification** est trouvé, cela marque la fin du bloc combiné du nom de champ et des propriétés. Le compteur i, qui sera ensuite comparé au compteur de champ précédemment déclaré dans Fields, est donc incrémenté de 1.

Lorsque i et inFields deviennent égaux, la commande SQL peut être combinée. Cependant, vous devez vous assurer que les enregistrements vides ne sont pas créés à partir de formulaires vides. Par conséquent, il y a une vérification précédente pour toutes les valeurs de champ étant NULL. Dans de tels cas, la commande SQL est lancée immédiatement. Sinon, l'enregistrement est inclus pour être inséré dans la table Nom. Après cela, les variables sont restaurées à leurs valeurs par défaut et le formulaire PDF suivant peut être lu.

```
                    IF instr(stLigne, "FieldJustification:") THEN
                        i = i + 1
                    END IF
                    IF i = inChamps THEN
                        aNull = Split(stValeurChamp,",")
                        FOR n = 0 TO Ubound(aNull())
                            IF aNull(n) = "NULL" THEN inNull = inNull + 1
                        NEXT
                        IF inNull < inChamps THEN
                            stSql ="INSERT INTO """ + stTable + """ (" + stNomChamp +
") "
                            stSql = stSql + "VALUES (" + stValeurChamp + ")"
                            oSQL_Commande.executeUpdate(stSql)
                        END IF
                        stNomChamp = ""
                        stValeurChamp = ""
                        stFieldType = ""
                        i = -1
                        inNull = 0
                    END IF
                LOOP
            CLOSE inNombre
```

À la fin de la procédure, il reste un fichier `PDF_Form_Data.txt` qui est supprimé. Ensuite, le formulaire de la base est rechargé afin que les enregistrements lus puissent être affichés.

```
            Kill(stFichier)
        END IF
    NEXT
    oEvent.Source.Model.Parent.reload()

END SUB
```

Si le texte contient un `"'"`, cela sera vu comme un marqueur de fin de texte lors de l'insertion par SQL. Le code SQL de la commande d'insertion échoue si plus de texte suit sans être placé entre guillemets simples. Pour éviter cela, chaque guillemet simple dans le texte doit être masqué par un autre guillemet simple. C'est le travail de la fonction String_to_SQL.

```
FUNCTION String_to_SQL(st AS STRING) AS STRING
    IF InStr(st,"'") THEN
        st = Join(Split(st,"'"),"''")
    END IF
    String_to_SQL = st
END FUNCTION
```

Les dates du fichier PDF sont lues sous forme de texte. Elles ne peuvent pas être vérifiées à l'avance pour une entrée correcte.

Lorsque les dates sont écrites en anglais, le jour, le mois et l'année sont séparés par des points ou, plus souvent, des tirets. Le jour et le mois peuvent avoir un ou deux chiffres. L'année peut avoir deux ou quatre chiffres.

Dans le code SQL, les dates doivent commencer par une année à quatre chiffres et être écrites AAAA-MM-JJ. Les dates saisies doivent donc passer par un processus de conversion.

La date saisie est divisée en parties jour, mois et année. Le jour et le mois reçoivent un zéro non significatif, puis tronqués à droite à deux chiffres. Cela garantit un nombre à deux chiffres dans tous les cas.

Si la partie année comporte déjà quatre chiffres (supérieurs à 1000), la valeur n'est pas modifiée. Sinon, si l'année est supérieure à 30, la date est supposée appartenir au siècle dernier et doit avoir 1900 ajouté. Toutes les autres dates sont attribuées au siècle actuel.

```
FUNCTION Date_to_SQLDate(st AS STRING) AS STRING
       DIM stDay AS STRING
       DIM stMonth AS STRING
       DIM stDate AS STRING
       DIM inYear AS INTEGER
       stDay = Right("0" & Day(CDate(st)), 2)
       stMonth = Right("0" & Month(CDate(st)), 2)
       inYear = Year(CDate(st))
       IF inYear = 0 THEN
          inYear = Year(Now())
       END IF
       IF inYear > 1000 THEN
       ELSEIF inYear > 30 THEN
          inYear = 1900 + inYear
       ELSE
          inYear = 2000 + inYear
       END IF
       stDate = inYear & "-" & stMonth & "-" & stDay
       Date_to_SQLDate = stDate
END FUNCTION
```

La fonction SQL_Value combine cette fonction avec les paramètres NULL indiqués ci-dessous et donne ainsi des valeurs correctement formatées à saisir dans la base de données pour sa fonction appelante.

Les champs vides donnent une valeur NULL. Le champ correspondant dans le tableau sera également vide.

```
FUNCTION SQL_Value(st AS STRING, stType AS STRING) AS STRING
       DIM stValue AS STRING
       IF st = "" THEN
          SQL_Value = “NULL”
```

S'il s'agit d'un champ de date et que le contenu doit être reconnaissable en tant que date, son contenu doit être converti au format de date SQL. S'il n'est pas reconnaissable en tant que date, le champ doit rester vide.

```
       ELSEIF stType = “DATE” THEN
          IF isDate(st) THEN
             SQL_Value = "'" & Date_to_SQLDate(st) & "'"
          ELSE
             SQL_Value = “NULL”
          END IF
```

Un champ décimal peut contenir des virgules au lieu de points décimaux, suivis de décimales. Dans Basic et SQL, le séparateur décimal est toujours un point. Par conséquent, les nombres contenant une virgule doivent être convertis. Le champ doit contenir un nombre, donc les autres caractères tels que les unités doivent être supprimés. Ceci est réalisé par la fonction **Val ()**.

```
        ELSEIF stType = “DECIMAL” THEN
            stValue = Str(Val(Join(Split(st,","),".")))
```

Tout autre contenu est traité comme du texte. Les guillemets simples sont masqués par un autre guillemet simple et le terme entier est à nouveau entre guillemets simples.

```
        ELSE
            SQL_Value = "'" & String_to_SQL(st) & "'"
        END IF
    END FUNCTION
```

Pour plus de détails sur la construction de macros, voir le chapitre 9 de ce manuel. Cet exemple montre simplement qu’il est possible de transférer des données de formulaires PDF vers Base sans avoir à copier les valeurs champ par champ à l’aide du presse-papiers. La construction de la procédure ci-dessus a été délibérément gardée très générale et devrait pouvoir être adaptée à des situations particulières.

Guide Base

Chapitre 8
Trucs et astuces

Informations générales sur les tâches de base de données

Ce chapitre décrit quelques solutions aux problèmes rencontrés par de nombreux utilisateurs de Base.

Filtrage des données

Le filtrage des données à l'aide de l'interface graphique est décrit au chapitre 3, Tables. Nous décrivons ici une solution à un problème soulevé par de nombreux utilisateurs : comment utiliser les zones de liste pour rechercher le contenu des champs dans les tables, qui apparaissent ensuite filtrés dans la section de formulaire sous-jacente et peuvent être modifiés.

La base de ce filtrage est une requête modifiable (voir Chapitre 5, Requêtes) et une table supplémentaire, dans laquelle les données à filtrer sont stockées. La requête affiche à partir de sa table sous-jacente uniquement les enregistrements qui correspondent aux valeurs de filtre. Si aucune valeur de filtre n'est donnée, la requête affiche tous les enregistrements.

L'exemple suivant est basé sur une table « **Media** » qui inclut, entre autres, les champs suivants : **« ID »** (clé primaire), **« Titre »**, **« Catégorie »**. Les types de champ sont respectivement INTEGER, VARCHAR et VARCHAR.

Nous avons d'abord besoin d'une table **« Filtre »**. Elle contient une clé primaire et deux champs de filtre (bien sûr, vous pouvez en avoir plus si vous le souhaitez) : **« ID »** (clé primaire), **« Filtre_1 »**, **« Filtre_2 »**. Comme les champs de la table **« Media »**, à filtrer, sont de type **VARCHAR**, les champs **« Filtre_1 »** et **« Filtre_2 »** sont également de ce type. **« ID »** peut être un champ **oui/non**, car la table **« Filtre »** ne contiendra jamais plus d'un enregistrement.

Attention

Si la base de données utilisée n'est pas de type mono-utilisateur, une telle table de filtrage générerait également le filtrage pour les autres utilisateurs. Si nécessaire, vous pouvez travailler avec des tables temporaires qui ne sont visibles que par l'utilisateur qui les a créées. Ou vous pouvez simplement utiliser le nom d'utilisateur directement comme clé primaire de la table de filtrage. L'ID du champ ne serait alors pas un champ oui/non, mais un champ VARCHAR.

Vous pouvez également filtrer les champs qui apparaissent dans la table **« Media »** uniquement en tant que clés étrangères. Dans ce cas, vous devez donner aux champs correspondants de la table **« Filtre »** le type approprié pour les clés étrangères, généralement INTEGER.

La requête suivante doit être améliorée :

```
SELECT * FROM "Media"
```

Tous les enregistrements de la table **Media** sont affichés, incluant la clé primaire.

```
SELECT * FROM "Media"
WHERE "Titre" = IFNULL((SELECT "Filtre_1" FROM "Filtre"), "Titre")
```

Si le champ **Filtre_1** n'est pas **NULL**, les enregistrements dont le **Titre** est le même que **Filtre_1** sont affichés. Si le champ **Filtre_1** est **NULL**, la valeur du champ **Titre** est utilisée à la place. Il ressort évidemment que tous les enregistrements seront affichés, puisque **Titre** (de IFNULL) est

identique au **Titre** (de WHERE **"Titre"** =). Sauf, si le champ **Titre** d'un enregistrement est vide (contient **NULL**), alors cet enregistrement ne sera pas affiché. Par conséquent, nous devons améliorer la requête :

```
SELECT *, IFNULL("Titre", '') AS "T" FROM "Media"
WHERE "T" = IFNULL((SELECT "Filtre_1" FROM "Filtre"), "T")
```

Conseil

IFNULL (expression, valeur) nécessite que l'expression ait le même type de champ que la valeur.

Si l'expression a le type de champ VARCHAR, utilisez deux guillemets simples "" comme valeur.

S'il a DATE comme type de champ, entrez une date comme valeur qui n'est pas contenue dans le champ de la table à filtrer. Utilisez ce format : {D 'AAAA-MM-JJ'}.

S'il s'agit de l'un des types de champ numérique, utilisez le type de champ NUMÉRIQUE pour la valeur. Saisissez un nombre qui n'apparaît pas dans le champ du tableau à filtrer.

Cette variante mènera au but recherché. Au lieu d'utiliser le champ **"Titre"** directement dans la clause WHERE on introduit dans le **SELECT** un alias de ce champ (nommé **"T"**), qui sera vide ('') si **"Titre"** est NULL. Dans ces conditions, seul le champ **T** est considéré. Tous les enregistrements sont donc affichés même si **Titre** est **NULL.**

Malheureusement, vous ne pouvez pas faire cela en utilisant l'interface graphique. La commande ne peut être exécutée que directement avec SQL. Pour le rendre modifiable dans l'interface graphique, d'autres ajustements sont nécessaires :

```
SELECT "Media".*, IFNULL("Media"."Titre", '') AS "T"
FROM "Media"
WHERE "T" = IFNULL((SELECT "Filtre_1" FROM "Filtre"), "T")
```

Si la relation de la table avec les champs est maintenant établie, la requête devient modifiable dans l'interface graphique. À titre de test, vous pouvez mettre un titre dans **"Filtre"."Filtre_1"**.

Comme **"Filtre"."ID"** définit la valeur "0", l'enregistrement est sauvegardé et le filtrage peut être compris. Si **"Filtre"."Filtre_1"** est vidé, l'interface graphique traite cela comme **NULL**. Un nouveau test donne un affichage de tous les médias. Dans tous les cas, avant qu'un formulaire ne soit créé et testé, un seul enregistrement avec une clé primaire doit être entré dans la table Filtre. Il ne doit s'agir que d'un seul enregistrement, car les sous-requêtes illustrées ci-dessus ne peuvent transmettre qu'une seule valeur.

La requête peut maintenant être dévelloppée pour filtrer également un second champ :

```
SELECT "Media".*, IFNULL("Media"."Titre", '') AS "T",
                        IFNULL("Media"."Categorie", '') AS "K"
FROM "Media"
WHERE "T" = IFNULL((SELECT "Filtre_1" FROM "Filtre"), "T")
      AND "K" = IFNULL((SELECT "Filtre_2" FROM "Filtre"), "K")
```

Ceci conclut la création de la requête modifiable.

Maintenant, pour la requête de base pour les deux zones de liste :

```
SELECT DISTINCT "Titre", "Titre"
FROM "Media" ORDER BY "Titre" ASC
```

La zone de liste doit afficher le **Titre**, puis transmettre également ce **Titre** au champ **Filtre_1** de la table **Filtre** qui sous-tend le formulaire. De plus, aucune valeur en double ne doit être affichée (condition **DISTINCT**). Et le tout doit bien sûr être trié dans le bon ordre.

Une requête correspondante est ensuite créée pour le champ **Categorie**, qui consiste à écrire ses données dans le champ **Filtre_2** de la table **Filtre**.

Si l'un de ces champs contient une clé étrangère, la requête est adaptée afin que la clé étrangère soit transmise à la table **Filtre** sous-jacente.

Le formulaire se compose de deux parties. Le **Formulaire 1** est le formulaire basé sur la table **Filtre**. Le **Formulaire 2** est le formulaire basé sur la requête. Le **Formulaire 1** n'a pas de barre de navigation **Propriétés> Données> Barre de navigation** (Non) et le cycle est défini sur **Enregistrement actif**. En outre, la propriété **Autoriser les ajouts** est définie sur Non. Le premier et le seul enregistrement de ce formulaire existe déjà.

Le formulaire 1 contient deux zones de liste avec les étiquettes appropriées. La **zone de liste 1** renvoie les valeurs de **Filtre_1** et est liée à la requête du champ **Titre**.

ZoneListe2 renvoie des valeurs pour **Filtre_2** et se rapporte à la requête pour le champ **Categorie**.

Le formulaire 2 contient un champ de contrôle de table, dans lequel tous les champs de la requête peuvent être répertoriés à l'exception des champs T et K. Le formulaire fonctionnerait toujours si ces champs étaient présents ; ils sont omis pour éviter une duplication déroutante du contenu des champs. De plus, le formulaire 2 contient un bouton, lié à la fonction de mise à jour du formulaire. Une barre de navigation supplémentaire peut être intégrée pour empêcher le scintillement de l'écran à chaque fois que le formulaire change, car la barre de navigation est présente dans un formulaire et pas dans l'autre.

Une fois le formulaire terminé, la phase de test commence. Lorsqu'une zone de liste est modifiée, le bouton du formulaire 2 est utilisé pour stocker cette valeur et mettre à jour le formulaire 2. Cela concerne désormais la valeur fournie par la zone de liste. Le filtrage peut être rétrospectif en choisissant un champ vide dans la zone de liste.

Ces modèles de requêtes, ici voulus simples, sont repris sous des formes plus élaborées dans la base d'exemple « **Exemple_Cherche_et_Filtre.odb** » dans la requête « **RequeteFiltre** ».

Recherche de données

La principale différence entre la recherche de données et le filtrage des données réside dans la technique de requête. L'objectif est de fournir, en réponse à des termes de recherche en langage courant, une liste résultante d'enregistrements qui peuvent ne contenir que partiellement ces termes réels.

Rechercher avec LIKE

La table dans laquelle la recherche est effectuée peut être la même que celle qui contient déjà les valeurs de filtre. La table **Filtre** est simplement complété par un champ nommé **TexteCherche**

(**Varchar**). Cela signifie que la même table peut être consultée, filtrée et recherchée dans les formulaires en même temps.

Le formulaire est construit comme pour le filtrage. Au lieu d'une zone de liste, nous avons besoin d'un contrôle de saisie de texte, pour le terme de recherche, et peut être aussi d'un contrôle étiquette avec le libellé Recherche. Le contrôle « Zone de texte » du terme de recherche peut être autonome dans le formulaire ou avec les champs de filtrage, si les deux fonctions sont souhaitées.

Alors que le filtrage utilise un terme qui apparaît déjà dans la table sous-jacente, la recherche utilise des entrées arbitraires.

```
SELECT * FROM "Media"
WHERE "Titre" = (SELECT "TexteCherche" FROM "Filtre")
```

Cette requête conduit normalement à une liste vide de résultat pour les raisons suivantes :

- Il est rare que quelqu'un connaisse le titre complet par cœur lors de la saisie du terme de recherche. Cela n'afficherait pas le titre. Pour trouver le livre « Le guide de l'auto-stoppeur de la galaxie », il devrait suffire de saisir « Guide de l'auto-stoppeur », « auto-stop », et même simplement « auto » dans le champ de recherche.
- Si le champ TexteCherche est vide, aucun enregistrement de données ne sera affiché. La requête au dessus reverrait **NULL**, qui ne peut apparaître que dans une condition utilisant **IS NULL**.
- Même si cela était ignoré, la requête entraînerait l'affichage de tous les enregistrements qui n'ont pas d'entrée dans le champ **Titre**.

La dernière condition peut être supprimée si la condition de filtrage est :

```
SELECT * FROM "Media"
WHERE "Titre" = IFNULL((SELECT "TexteCherche" FROM "Filtre"), "Titre")
```

Avec ce raffinement du filtrage (que se passe-t-il si le titre est NULL?). On obtient un résultat plus conforme aux attentes. Mais la première condition n'est toujours pas remplie. La recherche devrait bien fonctionner lorsque seules des connaissances fragmentaires sont disponibles. La technique de requête doit donc utiliser la condition **LIKE** :

```
SELECT * FROM "Media"
WHERE "Titre" LIKE (SELECT '%' || "TexteCherche" ||'%' FROM "Filtre")
```

ou mieux encore :

```
SELECT * FROM "Media"
      WHERE "Titre" LIKE IFNULL((SELECT '%' || "TexteCherche" ||'%'
FROM "Filtre"), "Titre")
```

LIKE, associé à %, signifie que tous les enregistrements contenant n'importe où le terme de recherche sont affichés. % est un caractère générique pour n'importe quel nombre de caractères avant ou après le terme de recherche. Diverses questions subsistent après la création de cette version de la requête :

- Il est courant d'utiliser des lettres minuscules pour les termes de recherche. Alors, quel résultat j'obtiens si je tape "auto" au lieu de "Auto"?
- Quelles autres conventions écrites doivent être prises en compte ?
- Qu'en est-il des champs qui ne sont pas formatés en tant que champs de texte ? Pouvez-vous rechercher des dates ou des nombres avec le même champ de recherche ?
- Et si, comme dans le cas du filtre, vous voulez empêcher les valeurs NULL dans le champ de provoquer l'affichage de tous les enregistrements ?

La variante suivante couvre une ou deux de ces possibilités :

```
SELECT * FROM "Media"
      WHERE LOWER("Titre")
      LIKE IFNULL((SELECT '%' ||LOWER("TexteCherche") ||'%' FROM "Filtre"),
      LOWER("Titre"))
```

La condition modifie le terme de recherche et le contenu du champ en minuscules. Cela permet également de comparer des phrases entières.

```
SELECT * FROM "Media"
      WHERE LOWER("Titre")
      LIKE IFNULL((SELECT '%' || LOWER("TexteCherche") ||'%' FROM "Filtre"),
            LOWER("Title")) OR LOWER("Categorie")
      LIKE (SELECT '%' || LOWER("TexteCherche") ||'%' FROM "Filtre")
```

La fonction IFNULL doit se produire une seule fois, de sorte que lorsque le TexteCherche est NULL, LOWER ("Titre") LIKE LOWER ("Titre") est interrogé. Et comme le titre doit être un champ qui ne peut pas être NULL, dans ce cas, tous les enregistrements sont affichés. Bien sûr, pour plusieurs recherches de champ, ce code devient d'autant plus long. Dans de tels cas, il est préférable d'utiliser une macro, pour permettre au code de couvrir tous les champs en une seule fois.

Mais le code fonctionne-t-il toujours avec des champs qui ne sont pas du texte ? Bien que la condition LIKE soit vraiment adaptée au texte, elle fonctionne également pour les nombres, les dates et les heures sans nécessiter de modifications. Donc, en fait, la conversion de texte n'a pas besoin d'avoir lieu. Cependant, un champ de temps qui est un mélange de texte et de nombres ne peut pas interagir avec les résultats de la recherche, sauf si la requête est élargie, de sorte qu'un seul terme de recherche soit subdivisé sur tous les espaces entre le texte et les nombres. Ceci, cependant, gonflera considérablement la requête.

Conseil

Les requêtes utilisées pour filtrer et rechercher des enregistrements peuvent être directement incorporées dans le formulaire.

Toutes les conditions ci-dessus peuvent être saisies dans les propriétés du formulaire à la ligne Filtre.

```
SELECT * FROM "Media" WHERE "Titre" = IFNULL((SELECT "TexteCherche" FROM
"Filtre"), "Titre")
```

devient alors un formulaire qui utilise le contenu de la table Media.

Sous "Filtre", nous avons

```
("Media"."Titre" =
      IFNULL((SELECT "TexteCherche" FROM "Filtre"), "Media"."Titre"))
```

Dans l'entrée de filtre, veillez à ce que la condition soit mise entre parenthèses et fonctionne avec le terme "Table". "Champ".

L'avantage de cette variante est que le filtre peut être activé et désactivé lorsque le formulaire est ouvert.

Rechercher avec LOCATE

La recherche avec LIKE est généralement satisfaisante pour les bases de données avec des champs contenant une quantité de texte qui peut être facilement visualisée. Mais qu'en est-il des champs Mémo, qui peuvent contenir plusieurs pages de texte ? Dans ce cas, la recherche doit déterminer où se trouve le texte spécifié.

Pour localiser exactement le texte, **HSQLDB** a la fonction **LOCATE**. Elle prend un terme de recherche (le texte que vous souhaitez rechercher) comme argument. Vous pouvez également ajouter une position à rechercher. En bref : **LOCATE** (**terme de recherche**, **champ de texte de la base de données**, **position**).

Dans **FIREBIRD**, il faut utiliser **POSITION** à la place de LOCATE, pour cela consulter ce Lien.

La fonction **`SUBSTRING`** utilisée dans ce qui `suit` doit également être écrite dans une syntaxe différente pour Firebird. Au lieu de **`SUBSTRING (texte, position de départ [, longueur])`**, utilisez **`SUBSTRING (texte FROM position de départ [FOR longueur])`**.

L'explication suivante utilise une table appelée Table. La clé primaire s'appelle ID et doit être unique. Il existe également un champ appelé **Memo** qui a été créé comme un champ de type Memo (LONGVARCHAR). Ce champ Mémo contient quelques phrases de ce manuel.

Les captures d'écran de ce chapitre sont tirées de la base de données « **Exemple_Autotexte_Recherche_Orthographe.odb** », qui est jointe à ce manuel.

Les exemples de requêtes sont présentés sous forme de requêtes paramétrées. Le texte de recherche à saisir est toujours "office".

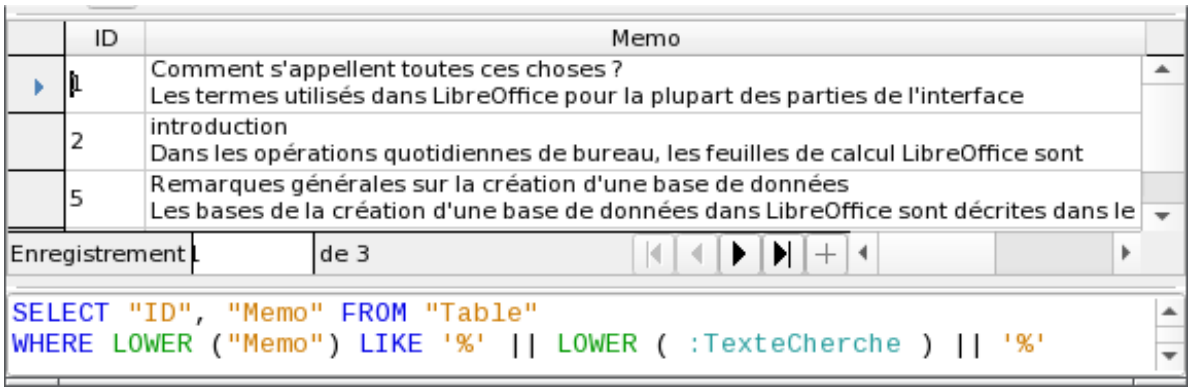
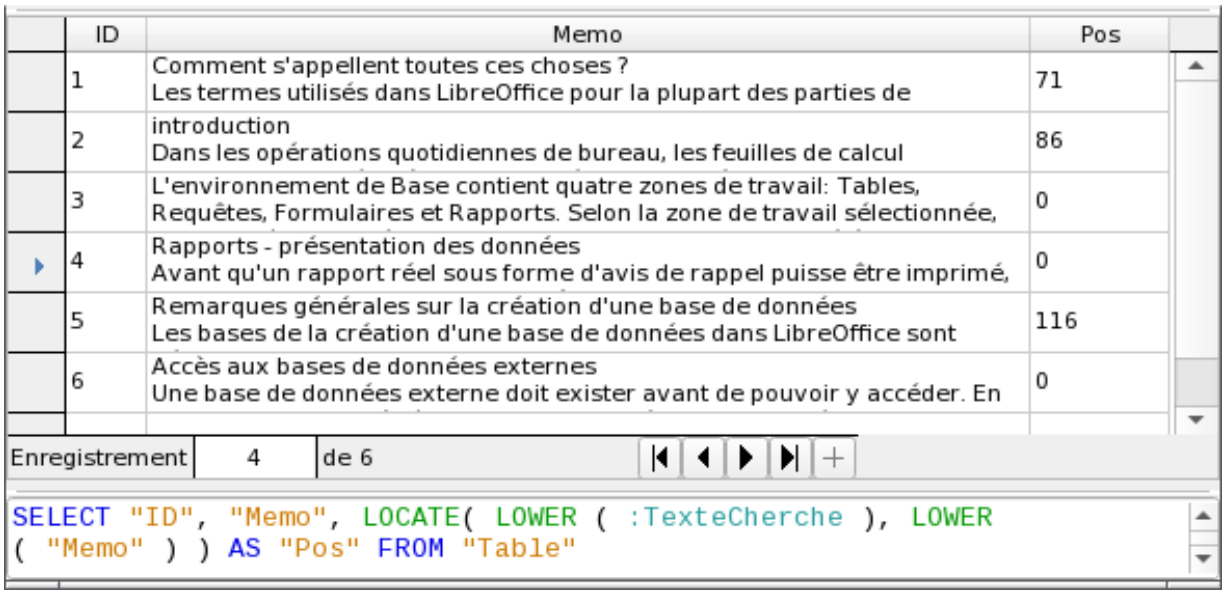


Figure 113: Recherche avec LIKE

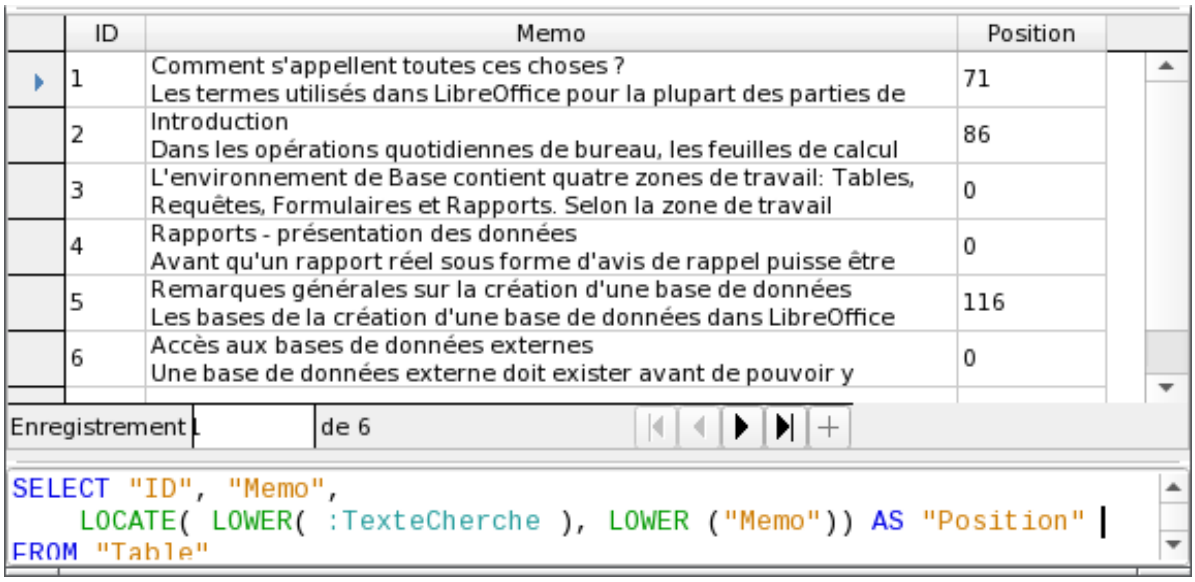
Nous utilisons d'abord **LIKE**, qui ne peut être utilisé que dans certaines conditions. Si le texte recherché se trouve n'importe où, l'enregistrement correspondant s'affiche. La comparaison est entre une version minuscule du contenu du champ, utilisant **LOWER ("Memo")** et une version minuscule du texte de recherche utilisant **LOWER (:TexteCherche)**, pour rendre la recherche insensible à la casse. Plus le texte du champ Mémo est long, plus il devient difficile de voir le terme dans le texte récupéré. (**Cf. Requête_Like dans la base exemple**).



	ID	Memo	Pos
	1	Comment s'appellent toutes ces choses ? Les termes utilisés dans LibreOffice pour la plupart des parties de	71
	2	introduction Dans les opérations quotidiennes de bureau, les feuilles de calcul	86
	3	L'environnement de Base contient quatre zones de travail: Tables, Requêtes, Formulaires et Rapports. Selon la zone de travail sélectionnée,	0
▸	4	Rapports - présentation des données Avant qu'un rapport réel sous forme d'avis de rappel puisse être imprimé,	0
	5	Remarques générales sur la création d'une base de données Les bases de la création d'une base de données dans LibreOffice sont	116
	6	Accès aux bases de données externes Une base de données externe doit exister avant de pouvoir y accéder. En	0

Figure 114: Recherche avec LOCATE

LOCATE vous montre plus précisément où se trouve votre terme de recherche. Dans les enregistrements 3, 4, 6, le terme n'apparaît pas. Dans ce cas, **LOCATE** donne la position "0". Il est assez facile de confirmer le nombre donné pour l'enregistrement 1, la chaîne "Office" est présente à la seconde ligne. Naturellement, il serait également possible d'afficher les résultats de LOCATE de la même manière que pour LIKE. (**Cf. Requête_Locate dans la base exemple**).



	ID	Memo	Position
▸	1	Comment s'appellent toutes ces choses ? Les termes utilisés dans LibreOffice pour la plupart des parties de	71
	2	Introduction Dans les opérations quotidiennes de bureau, les feuilles de calcul	86
	3	L'environnement de Base contient quatre zones de travail: Tables, Requêtes, Formulaires et Rapports. Selon la zone de travail	0
	4	Rapports - présentation des données Avant qu'un rapport réel sous forme d'avis de rappel puisse être	0
	5	Remarques générales sur la création d'une base de données Les bases de la création d'une base de données dans LibreOffice	116
	6	Accès aux bases de données externes Une base de données externe doit exister avant de pouvoir y	0

Figure 115: Recherche avec LOCATE

Dans la colonne Cible (Figure 116, les résultats de la recherche sont affichés avec plus de précision. La requête précédente a été utilisée comme base pour celle-ci. Cela permet d'utiliser le mot "Position" dans la requête externe au lieu d'avoir à répéter **LOCATE (LOWER (:TexteCherche), LOWER ("Memo"))** à chaque fois. En principe, ce n'est pas différent d'enregistrer la requête précédente et de l'utiliser comme source pour celle-ci.

`"Position" = 0` signifie qu'il n'y a pas de résultat. Dans ce cas, « `**non trouvé**` » est affiché.

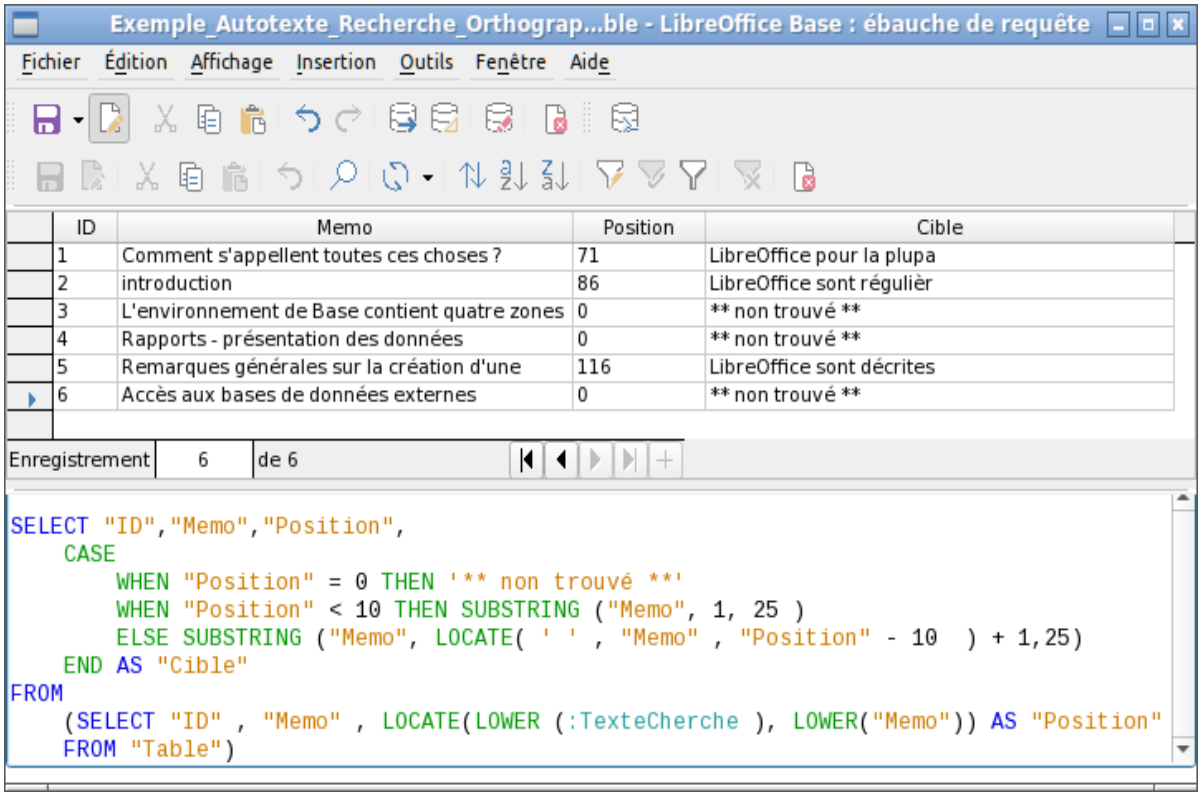


Figure 116: Cible extraite de la recherche

`"Position" < 10` signifie que le terme de recherche apparaît juste au début du texte. 10 caractères peuvent être facilement scannés à l'œil nu. Par conséquent, le texte entier est affiché. Ici au lieu de **SUBSTRING ("Memo", 1)**, nous aurions pu utiliser simplement **"Memo"**.

Pour tous les autres résultats, un espace est recherché **' '** jusqu'à 10 caractères avant le terme de recherche. Le texte affiché ne commence pas au milieu d'un mot mais après un espace. **`SUBSTRING("Memo", LOCATE(' ',"Memo","Position"-10)+1)`** garantit que le texte commence au début d'un mot qui se trouve au plus 10 caractères avant le terme de recherche.

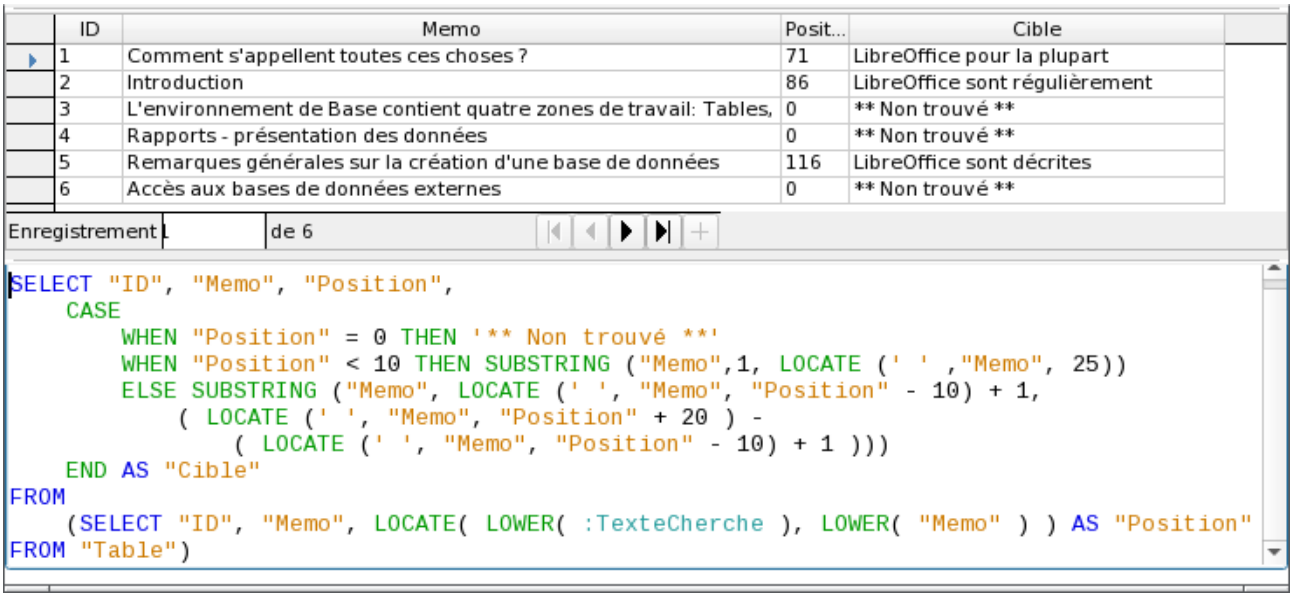
En pratique, nous voudrions utiliser plus de caractères, car il y a beaucoup de mots plus longs que cela, et le terme de recherche peut se trouver dans un autre mot avec plus de 10 caractères devant lui. LibreOffice contient le terme de recherche "office" avec le "O" comme sixième caractère. Dans d'autres cas, le résultat pourrait être faux

La technique de requête est la même que pour la requête précédente. Seule la longueur de la cible à afficher a été réduite à 25 caractères. La fonction **SUBSTRING** requiert comme arguments le **texte à rechercher**, puis la **position de départ** du résultat, et comme troisième argument facultatif, la **longueur** de la chaîne de texte à afficher. Ici, il a été défini assez court à des fins de démonstration. Un avantage de le raccourcir est que les exigences de stockage pour un grand nombre d'enregistrements sont réduites et que l'emplacement est facilement visible. Un inconvénient visible de ce type de raccourcissement de chaîne est que la coupe est faite en stricte conformité avec la limite de 25 caractères, sans tenir compte de l'endroit où les mots commencent.

Ici, nous recherchons du 25e caractère dans "Memo" au caractère espace suivant. Le contenu à afficher se situe entre ces deux positions.

C'est beaucoup plus simple si la correspondance apparaît au début du champ. Ici **`LOCATE(' ',"Memo",25)`** donne la position exacte où commence le texte. Puisque nous

voulons que le texte soit affiché depuis le début, cela correspond exactement à la longueur du terme affichable.



ID	Memo	Posit...	Cible
1	Comment s'appellent toutes ces choses ?	71	LibreOffice pour la plupart
2	Introduction	86	LibreOffice sont régulièrement
3	L'environnement de Base contient quatre zones de travail: Tables,	0	** Non trouvé **
4	Rapports - présentation des données	0	** Non trouvé **
5	Remarques générales sur la création d'une base de données	116	LibreOffice sont décrites
6	Accès aux bases de données externes	0	** Non trouvé **

Enregistrement 1 de 6

```
SELECT "ID", "Memo", "Position",
    CASE
        WHEN "Position" = 0 THEN '** Non trouvé **'
        WHEN "Position" < 10 THEN SUBSTRING ("Memo",1, LOCATE (' ' ,"Memo", 25))
        ELSE SUBSTRING ("Memo", LOCATE (' ', "Memo", "Position" - 10) + 1,
            ( LOCATE (' ', "Memo", "Position" + 20 ) -
                ( LOCATE (' ', "Memo", "Position" - 10) + 1 )))
    END AS "Cible"
FROM
    (SELECT "ID", "Memo", LOCATE( LOWER( :TexteCherche ), LOWER( "Memo" ) ) AS "Position"
FROM "Table")
```

Figure 117: Cible sans coupure de mot.

La recherche de l'espace suivant le terme recherché n'est pas plus compliquée si le terme se trouve plus loin dans le champ. La recherche commence simplement là où se trouve la correspondance. Ensuite, 20 caractères supplémentaires sont comptés, qui doivent suivre en toutes circonstances. L'espace suivant est localisé à l'aide de **`LOCATE(' ","Memo","Position"+20)`**. Cela donne uniquement l'emplacement dans le champ dans son ensemble, pas la longueur de la chaîne à afficher. Pour cela, nous devons soustraire la position à laquelle l'affichage du texte correspondant doit commencer. Cela a déjà été défini dans la requête par **`LOCATE(' ","Memo","Position"-10)+1`**. De cette manière, la longueur correcte du texte peut être trouvée.

La même technique peut être utilisée pour enchaîner les requêtes. La requête précédente devient désormais la source de données de la nouvelle. Elle a été insérée, entre parenthèses, sous le terme FROM. Seuls les champs sont renommés dans une certaine mesure, car il y a maintenant plusieurs positions et correspondances. De plus, la position suivante reçoit une référence en utilisant **`LOCATE(LOWER(: TexteCherche), LOWER("Memo"),"Position01"+1)`**. Cela signifie que la recherche recommence à la position après le texte correspondant précédent.

La requête la plus externe définit les champs correspondants pour les deux autres requêtes et fournit également "Cible02" en utilisant la même méthode que celle utilisée précédemment pour "Cible01". En outre, cette requête la plus externe détermine s'il existe d'autres correspondances.

ID	Memo	Position01	Cible01	Position02	Cible02	Position03
1	Comment s'appellent	71	LibreOffice pour la	0	** Non trouvé **	0
2	Introduction	86	LibreOffice sont	0	** Non trouvé **	0
3	L'environnement de	0	** Non trouvé **	0	** Non trouvé **	0
4	Rapports - présentation	0	** Non trouvé **	0	** Non trouvé **	0
5	Remarques générales	116	LibreOffice sont	243	de LibreOffice,	353
6	Accès aux bases de	0	** Non trouvé **	0	** Non trouvé **	0

Enregistrement 5 de 6

```
SELECT "ID", "Memo", "Position01", "Cible01", "Position02",
    CASE
        WHEN "Position02" = 0 THEN '** Non trouvé **'
        WHEN "Position02" < 10 THEN SUBSTRING ("Memo",1, LOCATE (' ' ,"Memo", 25))
        ELSE SUBSTRING ("Memo", LOCATE (' ', "Memo", "Position02" - 10) + 1,
            ( LOCATE (' ', "Memo", "Position02" + 20 ) -
                ( LOCATE (' ', "Memo", "Position02" - 10) + 1 ))
            )
    END AS "Cible02",
    CASE
        WHEN "Position02" = 0 THEN 0
        ELSE LOCATE( LOWER( :TexteCherche ), LOWER( "Memo" ) , "Position02" + 1)
    END AS "Position03"
FROM
    (SELECT "ID", "Memo", "Position01",
    CASE
        WHEN "Position01" = 0 THEN '** Non trouvé **'
        WHEN "Position01" < 10 THEN SUBSTRING ("Memo",1, LOCATE (' ' ,"Memo", 25))
        ELSE SUBSTRING ("Memo", LOCATE (' ', "Memo", "Position01" - 10) + 1,
            ( LOCATE (' ', "Memo", "Position01" + 20 ) -
                ( LOCATE (' ', "Memo", "Position01" - 10) + 1 ))
            )
    END AS "Cible01",
    CASE
        WHEN "Position01" = 0 THEN 0
        ELSE LOCATE( LOWER( :TexteCherche ), LOWER( "Memo" ) ,  "Position01" + 1)
    END AS "Position02"
    FROM
    (SELECT "ID", "Memo", LOCATE( LOWER( :TexteCherche ), LOWER("Memo")) AS "Position01"
    FROM "Table")
    )
```

Figure 118: Recherche de cibles multiples

La position correspondante est donnée par "**Position03**". Seul l'enregistrement 5 a d'autres correspondances, et celles-ci peuvent être trouvées dans une autre sous-requête.

L'empilement des requêtes présentées ici pourrait être poursuivi si vous le souhaitez. Cependant, l'ajout de chaque nouvelle requête externe impose une charge supplémentaire sur le système. Il serait nécessaire de faire quelques tests pour déterminer jusqu'où il est utile et réaliste d'aller. Le chapitre 9, Macros, montre comment les macros peuvent être utilisées pour rechercher toutes les chaînes de texte correspondantes dans un champ grâce à l'utilisation d'un formulaire.

Note

Toutes ces requêtes sont visibles dans la base Exemple_Autotexte_Recherche_Orthographe.odb associée à ce manuel, mais également dans l' Annexe à la fin de ce document.

Gestion des images et des documents dans Base

Les formulaires Base utilisent des contrôles graphiques pour gérer les images. Si vous utilisez une base de données HSQLDB interne, les contrôles graphiques sont le seul moyen de lire des images hors de la base de données sans utiliser de macros. Ils peuvent également être utilisés comme liens vers des images en dehors du fichier de base de données.

Lecture d’images dans la base de données

La base de données nécessite une table qui remplit au moins les conditions suivantes :

Nom de champ	***Type de champ***	***Description***
ID	Integer	ID est la clé primaire de cette table.
Image	Contrôle picto	Contient l’image sous forme de données binaires.

Une clé primaire doit être présente, mais il n’est pas nécessaire qu’elle soit un entier. D’autres champs qui ajoutent des informations sur l’image doivent être ajoutés.

Les données qui seront lues dans le champ image ne sont pas visibles dans une table. Au lieu de cela, vous voyez le mot <OBJECT>. De la même manière, les images ne peuvent pas être saisies directement dans une table. Vous devez utiliser un formulaire qui contient un contrôle graphique. Le contrôle graphique s’ouvre lorsque vous cliquez dessus pour afficher une boîte de dialogue de sélection de fichiers. Ensuite, il affiche l’image qui a été lue à partir du fichier sélectionné.

Les images qui doivent être insérées directement dans la base de données doivent être aussi petites que possible. Comme Base ne fournit aucun moyen (sauf en utilisant des macros) d’exporter des images dans leur taille d’origine, il est logique de n’utiliser que la taille nécessaire, par exemple pour l’impression dans un rapport. Les images originales dans la gamme mégapixel sont complètement inutiles et gonflent la base de données. Après avoir ajouté seulement quelques images, le HSQLDB interne donne une erreur : **Java. NullPointerException** et ne peut plus stocker l’enregistrement. Même si les images ne sont pas aussi grandes, il peut arriver que la base de données devienne inutilisable. Une bonne méthode est de réduire le nombre de couleurs (256 est suffisant).

De plus, les images ne doivent pas être intégrées dans des tables conçues pour y faire des recherches. Si, par exemple, vous disposez d’une base de données du personnel et que des images à utiliser dans les passes doivent être incluses, il est préférable de les stocker dans une table séparée avec une clé étrangère dans la table principale. Cela signifie que la table principale peut être scannée beaucoup plus rapidement, car elle ne nécessite pas autant de mémoire.

Liens vers des images et des documents

Avec une structure de dossiers soigneusement conçue, il est plus pratique d’accéder directement aux fichiers externes. Les fichiers en dehors de la base de données peuvent être aussi volumineux que nécessaire, sans avoir aucun effet sur le fonctionnement de la base de données elle-même. Malheureusement, cela signifie également que renommer un dossier sur votre propre ordinateur ou sur Internet peut vous faire perdre l’accès au fichier image, s’il n’est pas dans un sous dossier de la base.

Si vous ne souhaitez pas lire les images directement dans la base de données mais uniquement les lier, vous devez apporter une petite modification au tableau précédent :

Nom de champ	***Type de champ***	***Description***
ID	Integer	ID est la clé primaire de cette table.

Image	Zone Texte	Contient le chemin d'accès à l'image.

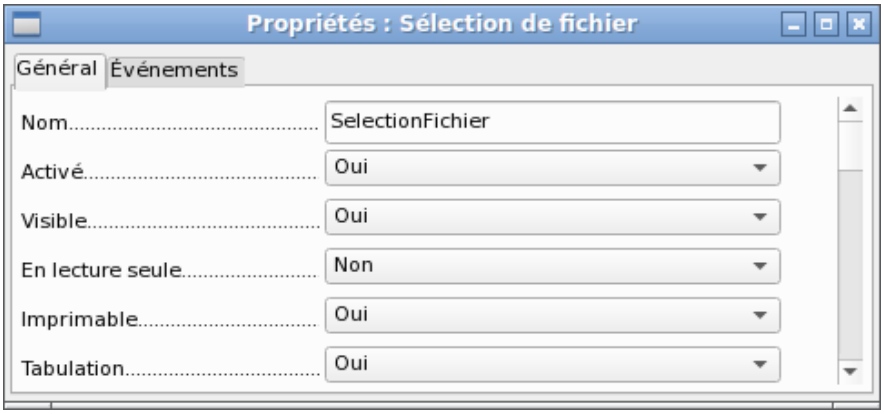
Si le type de champ est défini sur texte, le contrôle graphique du formulaire transmettra le chemin d'accès au fichier. L'image est toujours accessible par le contrôle graphique exactement comme une image interne.

Malheureusement, vous ne pouvez pas faire la même chose avec un document. Il n'est même pas possible de lire le chemin, car les commandes graphiques sont conçues pour les images graphiques et la boîte de dialogue du sélecteur de fichiers n'affiche que les fichiers au format graphique.

Avec une image, le contenu peut au moins être vu dans le contrôle graphique, en utilisant le chemin d'accès au fichier. Avec un document, il ne peut y avoir aucun affichage même si le chemin est stocké dans une table (sauf peut-être pour un document PDF, si le système possède un visualiseur). Nous devons d'abord agrandir un peu le tableau afin qu'au moins une petite quantité d'informations sur le document puisse être rendue visible.

Nom du champ	***Type du champ***	***Description***
ID	Integer	ID est la clé primaire de cette table.
Description	Zone Texte	Description du document, termes de recherche
Chemin	Zone Texte	Contient le chemin d'accès au document.

Pour rendre le chemin d'accès au document visible, nous devons créer un champ de sélection de fichier dans le formulaire.



Un champ de sélection de fichier n'a pas d'onglet pour les données dans sa boîte de dialogue de propriétés. Il n'est donc lié à aucun champ de la table sous-jacente.

Lier des documents avec un chemin absolu

En utilisant le champ de sélection de fichier, le chemin peut être affiché mais pas stocké. Pour cela, une procédure spéciale est nécessaire, liée à **Evénements> Texte modifié** :

```
SUB LireChemin(oEvent AS OBJECT)
	DIM oFormulaire AS OBJECT
	DIM oChamp AS OBJECT
	DIM oChamp2 AS OBJECT
```

```
        DIM stUrl AS STRING
        oChamp = oEvent.Source.Model
        oFormulaire = oChamp.Parent
        oChamp2 = oFormulaire.getByName("imgImage")
        IF oChamp.Text <> "" THEN
            stUrl = ConvertToUrl(oChamp.Text)
            oChamp2.BoundField.updateString(stUrl)
        END IF
END SUB
```

L'événement qui déclenche la procédure lui est transmis et aide à trouver le formulaire et le champ dans lesquels le chemin doit être stocké. Utiliser **`oEvent AS OBJECT`** rend l'accès plus simple lorsqu'un autre utilisateur souhaite utiliser une macro du même nom dans un sous-formulaire. Il rend le champ de sélection de fichier accessible via **`oEvent.Source.Model`**. Le formulaire est accessible en tant que parent du champ de sélection de fichier. Le nom du formulaire est donc sans importance. Depuis le formulaire, le champ appelé "*imgImage*" est désormais accessible. Ce champ est normalement utilisé pour stocker les chemins vers les fichiers image. Dans ce cas, l'URL du fichier sélectionné y est écrite. Pour garantir que l'URL fonctionne avec les conventions du système d'exploitation, le texte du champ de sélection de fichier est converti dans une forme généralement valide à l'aide de **ConvertToUrl**.

La table de base de données contient désormais un chemin au format absolu : `file:///…`

Si les entrées de chemin sont lues à l'aide d'un contrôle graphique, cela donnera un chemin relatif. Pour rendre cela utilisable, il faut l'améliorer. La procédure pour ce faire est beaucoup plus longue, car elle implique une comparaison entre le chemin d'entrée et le chemin réel.

Lier des documents avec un chemin relatif

La macro suivante est liée à la propriété "*Texte modifié*" du champ de sélection de fichier.

```
SUB LireChemin
        DIM oDoc AS OBJECT
        DIM oDrawpage AS OBJECT
        DIM oFormulaire AS OBJECT
        DIM oChamp AS OBJECT
        DIM oChamp2 AS OBJECT
        DIM arDebut_Url()
        DIM ar()
        DIM ar1()
        DIM ar2()
        DIM stTexte AS STRING
        DIM stUrl_complete AS STRING
        DIM stUrl_Texte AS STRING
        DIM stUrl AS STRING
        DIM stUrl_Coupe AS STRING
        DIM ink AS INTEGER
        DIM i AS INTEGER
        oDoc = thisComponent
        oDrawpage = oDoc.Drawpage
        oFormulaire = oEvent.Source.Model.Parent
        oChamp = oFormulaire.getByName("imgImage")
        oChamp2 = oFormulaire.getByName("SelectionFichier")
```

Tout d'abord, comme dans toutes les procédures, les variables sont déclarées. Ensuite, les champs importants pour la saisie des chemins sont recherchés. L'ensemble du code suivant n'est

alors exécuté que s'il y a effectivement quelque chose dans le champ de sélection de fichier, c'est-à-dire qu'il n'a pas été vidé par un changement d'enregistrement.

```
IF oChamp2.Text <> "" THEN
    arDebut_Url = split(oDoc.Parent.Url, oDoc.Parent.Title)
    ar = split(ConvertToUrl(oFeld2.Text),"/")
    stTexte = ""
```

Le chemin d'accès au fichier de base de données est lu. Ceci est effectué, comme indiqué ci-dessus, d'abord en lisant l'URL entière, puis en la divisant en un tableau de sorte que le premier élément du tableau contienne le chemin direct.

Ensuite, tous les éléments du chemin trouvés dans le champ de sélection de fichier sont lus dans le tableau **ar**. Le séparateur est */*. Cela peut être fait directement sous Linux. Sous Windows, le contenu de **oChamp2** doit être converti en une URL, qui utilisera une barre oblique (pas une barre oblique inverse) comme délimiteur de chemin.

Le but du fractionnement est d'obtenir le chemin d'accès au fichier en coupant simplement le nom du fichier à la fin. Par conséquent, à l'étape suivante, le chemin d'accès au fichier est reconstitué et placé dans la variable **stTexte**. La boucle ne se termine pas avec le dernier élément du tableau **ar** mais avec l'élément précédent.

```
FOR i = LBound(ar()) TO UBound(ar()) - 1
   stTexte = stTexte & ar(i) & "/"
NEXT
stTexte = Left(stTexte, Len(stTexte)-1)
arDebut_Url(0) = Left(arDebut_Url(0), Len(arDebut_Url(0))-1)
```

Le / final est à nouveau supprimé, sinon une valeur de tableau vide apparaîtrait dans le tableau suivant, ce qui interférerait avec la comparaison de chemin. Pour une comparaison correcte, le texte doit être converti en une URL correcte commençant par `file:///`. Enfin, le chemin d'accès au fichier de la base de données est comparé au chemin qui a été créé.

```
stUrl_Texte = ConvertToUrl(stTexte)
ar1 = split(stUrl_Texte,"/")
ar2 = split(arDebut_Url(0),"/")
stUrl = ""
ink = 0
stUrl_Coupe = ""
```

les tableaux **`ar1`** et **`ar2`** sont comparés étape par étape dans une boucle.

```
FOR i = LBound(ar2()) TO UBound(ar2())
   IF i <= UBound(ar1()) THEN
```

Le code suivant n'est exécuté que si le nombre **i** n'est pas supérieur au nombre d'éléments dans **ar1**. Si la valeur dans **ar2** est la même que la valeur correspondante dans **ar1**, et qu'aucune valeur incompatible n'a été trouvée jusqu'à ce point, le contenu commun est stocké dans une variable qui peut finalement être coupée de la valeur du chemin.

```
IF ar2(i) = ar1(i) AND ink = 0 THEN
   stUrl_Coupe = stUrl_Coupe & ar1(i) & "/"
ELSE
```

S'il y a une différence à tout moment entre les deux tableaux, alors pour chaque valeur différente, le signe pour remonter d'un répertoire sera ajouté à la variable **stUrl**.

```
   stUrl = stUrl & "../"
   ink = 1
END IF
```

Dès que l'index stocké dans **i** est supérieur au nombre d'éléments dans **ar1**, chaque valeur supplémentaire dans **ar2** entraînera le stockage d'un../ supplémentaire dans la variable **stUrl**.

```
            ELSE
                stUrl = stUrl & "../"
            END IF
            NEXT
            stUrl_complete = ConvertToUrl(oFeld2. Text)
            Champ.boundField. UpdateString(stUrl & Right(stUrl_complete,
Len(stUrl_complete)-Len(stUrl_Coupe)))
        END IF
END SUB
```

Lorsque la boucle via **ar2** est terminée, nous avons établi si, et de combien, le fichier auquel accéder est plus haut dans l'arborescence que le fichier de base de données. Maintenant, **stUrl_complete** peut être créé à partir du texte dans le champ de sélection de fichier. Celui-ci contient également le nom du fichier. Enfin, la valeur est transférée dans le contrôle graphique. La valeur de l'URL commence par stUrl, qui contient le nombre de points nécessaire (../). Ensuite, le début de **stUrl_complete**, la partie qui s'est avérée être la même pour la base de données et le fichier externe, est coupé. La manière de couper la chaîne est stockée dans **stUrl_Coupe**.

Affichage d'images et de documents liés

Les images liées peuvent être affichées directement dans un contrôle graphique. Mais affichage plus grand serait plus approprié pour montrer les détails.

Les documents ne sont normalement pas visibles dans Base, sauf PDF dans de bonnes conditions.

Pour rendre ce type d'affichage possible, nous devons à nouveau utiliser des macros. Cette macro est lancée à l'aide d'un bouton sur le formulaire qui contient le contrôle graphique.

```
SUB Voir(oEvent AS OBJECT)
        DIM oDoc AS OBJECT
        DIM oDrawpage AS OBJECT
        DIM oFormulaire AS OBJECT
        DIM oChamp AS OBJECT
        DIM oShell AS OBJECT
        DIM stUrl AS STRING
        DIM stChamp AS STRING
        DIM arDebut_Url()
        oDoc = thisComponent
        oDrawpage = oDoc.Drawpage
        oFormulaire = oDrawpage.Forms.getByName("Formulaire")
        oChamp = oFormulaire.getByName("imgCheminImage")
        stUrl = oChamp.BoundField.getString
```

Le contrôle graphique dans le formulaire est localisé. Comme la table ne contient pas l'image elle-même mais seulement un chemin d'accès stocké sous forme de chaîne de texte, ce texte est récupéré en utilisant **`getString`**.

Ensuite, le chemin d'accès au fichier de base de données est déterminé. Le fichier odb, le conteneur des formulaires, est accessible à l'aide de **oDoc.Parent**. L'URL entière, y compris le nom de fichier, est lue à l'aide de **oDoc.Parent.Url**. Le nom de fichier est également stocké dans **oDoc.Parent.Title**. Le texte est séparé à l'aide de la fonction de fractionnement avec le nom de

fichier comme séparateur. Cela donne le chemin d'accès au fichier de base de données comme premier et unique élément du tableau.

```
        arDebut_Url = split(oDoc.Parent.Url, oDoc.Parent.Title)
        oShell = createUnoService("com.sun.star.system. SystemShellExecute")
        stChamp = convertToUrl(arDebut_Url(0) + stUrl)
        oShell.execute(stChamp,,0)
END SUB
```

Les programmes externes peuvent être lancés à l'aide de la structure **com.sun.star.system. SystemShellExecute**. Le chemin d'accès absolu au fichier, composé du chemin d'accès au fichier de base de données et du chemin relatif stocké en interne à partir du fichier de base de données, est transmis au programme externe.-Le système d'exploitation détermine quel programme est appelé pour ouvrir le fichier.

La commande **oShell.execute** prend trois arguments. Le premier est un fichier **exécutable ou** le **chemin d'accès** à un fichier de données lié à un programme par le système. Le second est une **liste d'arguments** pour le programme. Le troisième est un **nombre** qui détermine comment les erreurs doivent être signalées. Les possibilités sont **0** (message d'erreur par défaut), **1** (pas de message) et **2** (autoriser uniquement l'ouverture d'URL absolues).

Lire des documents dans la base de données

Lors de la lecture des documents, les conditions suivantes doivent toujours être respectées :

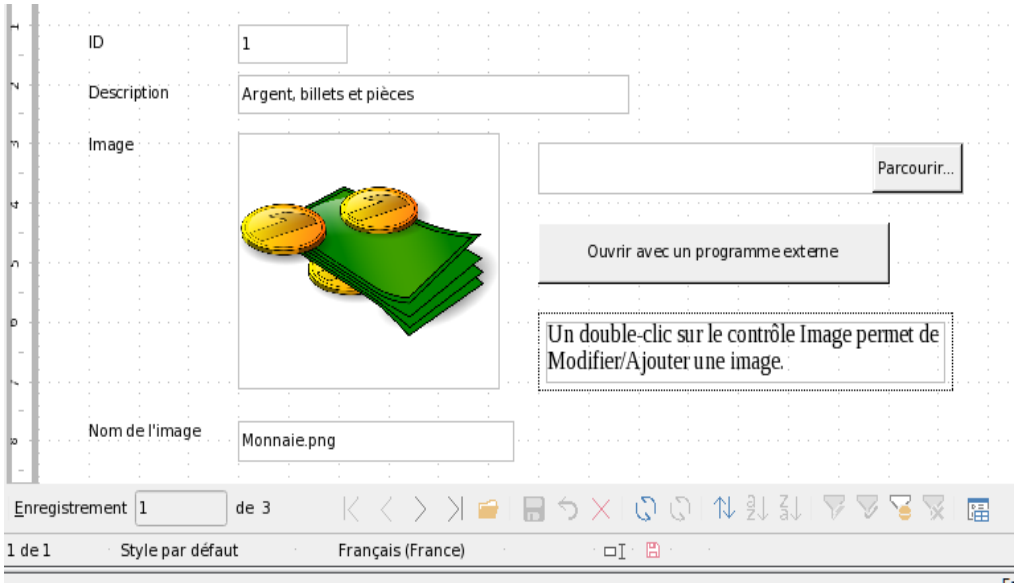
- Plus les documents sont volumineux, plus la base de données devient lourde. Par conséquent, pour les documents volumineux, une base de données externe est préférable à une base de données interne.
- Tout comme les images, les documents ne sont pas consultables. Ils sont stockés sous forme de données binaires et peuvent donc être placés dans un champ image.
- Les documents lus dans la base de données interne HSQLDB ne peuvent être lus qu'à l'aide de macros. Vous ne pouvez pas le faire avec des requêtes SQL.

Les macros suivantes pour la lecture d'entrée et de sortie dépendent d'une table qui comprend une description des données et le nom de fichier d'origine, ainsi qu'une version binaire du fichier. Le nom de fichier n'est pas automatiquement stocké avec le fichier, mais il peut fournir des informations utiles sur le type de données stockées dans un fichier qui doit être lu. Ce n'est qu'alors que le fichier peut être lu en toute sécurité par d'autres programmes.

La table contient les champs suivants :

Nom du champ	***Type du champ***	***Description***
ID	Integer	ID est la clé primaire de cette table.
Description	Zone Texte	Description du document, termes de recherche, etc.
Image	Contrôle picto	L'image ou le fichier sous forme binaire.
NomImage	Zone Texte	Le nom du fichier, y compris le suffixe du fichier. Important pour la lecture ultérieure.

Le formulaire de lecture des fichiers entrants et sortants ressemble à ceci :



Si des fichiers image sont présents dans la base de données, ils peuvent être visualisés dans le contrôle graphique du formulaire. Tous les autres types de fichiers sont masqués.

La macro suivante, pour lire un fichier, est déclenchée par **Propriétés> Sélection de fichier > Evénements> Texte modifié.**

```
SUB EntreeFichier_AvecNom(oEvent AS OBJECT)
	DIM oFormulaire AS OBJECT
	DIM oChamp AS OBJECT
	DIM oChamp2 AS OBJECT
	DIM oChamp3 AS OBJECT
	DIM oStream AS OBJECT
	DIM oSimpleFileAccess AS OBJECT
	DIM stUrl AS STRING
	DIM stNom AS STRING
	oChamp = oEvent.Source.Model
	oFormulaire = oChamp.Parent
	oChamp2 = oFormulaire.getByName("txt_NomFichier")
	oChamp3 = oFormulaire.getByName("imgImage")
	IF oChamp.Text <> "" THEN
	    stUrl = ConvertToUrl(oChamp. Text)
	    ar = split(stUrl,"/")
	    stNom = ar(UBound(ar))
	    oChamp2. BoundField.updateString(stNom)
	    oSimpleFileAccess = createUnoService("com.sun.star.ucb. SimpleFileAccess")
	    oStream = oSimpleFileAccess.openFileRead(stUrl)
	    oChamp3.BoundField.updateBinaryStream(oStream, oStream.getLength())
	END IF
END SUB
```

Comme l'événement déclencheur de la macro fournit le nom d'un autre champ de formulaire, il n'est pas nécessaire de vérifier si les champs se trouvent dans le formulaire principal ou un sous-formulaire. Il suffit que tous les champs soient de la même forme.

Le champ "**txt_NomFichier**" stocke le nom du fichier à rechercher. Dans le cas des images, ce nom doit être saisi à la main sans utiliser de macro. Ici, à la place, le nom de fichier est déterminé via une URL et saisi automatiquement lorsque les données sont lues.

Le champ "**imgImage**" stocke les données réelles à la fois pour les images et pour les autres fichiers.

Le chemin complet, y compris le nom de fichier, est lu à partir du champ de sélection de fichier à l'aide de **oChamp.Text**. Pour garantir que l'URL n'est pas affectée par les conditions spécifiques au système d'exploitation, le texte qui a été lu est converti au format d'URL standard à l'aide de ***ConvertToUrl***. Cette URL universellement valide est divisée dans un tableau. Le séparateur est /. Le dernier élément du chemin est le nom du fichier. **Ubound(ar)** donne l'index de ce dernier élément. Le nom de fichier réel peut ensuite être lu en utilisant **ar(Ubound(ar))** et transféré dans le champ sous forme de chaîne.

La lecture du fichier lui-même nécessite le **Service UNO com.sun.star.ucb.SimpleFileAccess**. Ce service peut lire le contenu du fichier sous forme de flux de données. Celui-ci est stocké temporairement dans l'objet **oStream** puis inséré en tant que flux de données dans le champ lié au champ "**imgImage**" de la table. Cela nécessite la longueur du flux de données à fournir ainsi que l'objet **oStream**.

Les données sont maintenant à l'intérieur du champ de formulaire comme avec une entrée normale. Cependant, si le formulaire est simplement fermé à ce stade, les données ne sont pas stockées. Le stockage nécessite d'appuyer sur le bouton Enregistrer l'enregistrement de la barre de navigation, cela se produit également automatiquement lors du passage à l'enregistrement suivant.

Déterminer les noms des fichiers image

Dans la méthode ci-dessus, il a été brièvement mentionné que le nom du fichier utilisé pour l'entrée dans un contrôle graphique ne peut pas être déterminé directement. Voici une macro pour déterminer ce nom de fichier, qui correspond au formulaire ci-dessus. Le nom de fichier ne peut pas être déterminé avec certitude par un événement directement lié au contrôle graphique. Par conséquent, la macro est lancée en utilisant **Propriétés du formulaire> Événements> Avant l'action d'enregistrement.**

```
SUB LireNomImage(oEvent AS OBJECT)
       oFormulaire = oEvent.Source
       IF InStr(oFormulaire.ImplementationName, "ODatabaseForm") THEN
          oChamp = oFormulaire.getByName("imgImage")
          oChamp2 = oFormulaire.getByName("txtNomFichier")
          IF oChamp.ImageUrl <> "" THEN
             stUrl = ConvertToUrl(oChamp.ImageUrl)
             ar = split(stUrl,"/")
             stNom = ar(UBound(ar))
             oChamp2.BoundField.updateString(stNom)
          END IF
       END IF
END SUB
```

Avant l'action d'enregistrement, deux implémentations avec des noms d'implémentation différents sont exécutées. Le formulaire est le plus facilement accessible à l'aide de l'implémentation **ODatabaseForm**.

Dans le contrôle graphique, l'URL de la source de données est accessible à l'aide de **ImageUrl**. Cette URL est lue, le nom de fichier est déterminé à l'aide de la procédure précédente *EntreeFichier_AvecNom*, et est transféré dans le champ *txtNomFichier*.

Suppression des noms de fichiers d'image de la mémoire

Si après l'exécution de la macro ci-dessus, vous passez à l'enregistrement suivant, le chemin d'accès à l'image d'origine est toujours disponible. Si un fichier non-image est maintenant lu à l'aide du champ de sélection de fichier, le nom de fichier de l'image écrasera le nom de ce fichier, sauf si vous utilisez la macro suivante.

Malheureusement, le chemin d'accès ne peut pas être supprimé par la macro précédente, car le fichier image n'est lu que lorsque l'enregistrement est enregistré. Supprimer le chemin à ce stade supprimerait l'image.

La macro est lancée à l'aide de **Propriétés du formulaire> Événements> Après l'action d'enregistrement.**

```
SUB ReinitialiseNomImage(oEvent AS OBJECT)
	oFormulaire = oEvent.Source
	IF InStr(oFormulaire.ImplementationName, "ODatabaseForm") THEN
	   oChamp = oFormulaire.getByName("imgImage")
	   IF oChamp.ImageUrl <> "" THEN
	      oChamp.ImageUrl = ""
	   END IF
	END IF
END SUB
```

Comme dans la procédure "*LireImages*", le contrôle graphique est accessible. S'il existe une entrée dans ImageUrl, elle est supprimée.

Lire et afficher des images et des documents

Pour les fichiers non graphiques et les images de taille d'origine, le bouton **Ouvrir un fichier avec un programme externe** doit être enfoncé. Ensuite, les fichiers du dossier temporaire peuvent être lus et affichés à l'aide du programme lié au suffixe de fichier dans le système d'exploitation.

La macro est lancée en utilisant **Propriétés : Bouton> Evénements> Exécuter l'action.**

```
SUB LireImage_avecNom(oEvent AS OBJECT)
	DIM oDoc AS OBJECT
	DIM oDrawpage AS OBJECT
	DIM oFormulaire AS OBJECT
	DIM oChamp AS OBJECT
	DIM oChamp2 AS OBJECT
	DIM oStream AS OBJECT
	DIM oShell AS OBJECT
	DIM oChemin AS OBJECT
	DIM oSimpleFileAccess AS OBJECT
	DIM stNom AS STRING
	DIM stChemin AS STRING
	DIM stChamp AS STRING
	oFormulaire = oEvent.Source.Model.Parent
	oChamp = oFormulaire.getByName("imgImage")
	oChamp2 = oFormulaire.getByName("txtNomImage")
	stNom = oChamp2.Text
	IF stNom = "" THEN
	   stNom = "DbImage"
	END IF
	oStream = oChamp.BoundField.getBinaryStream
	oChemin = createUnoService("com.sun.star.util. PathSettings")
```

```
        stChemin = oChemin.Temp & "/" & stNom
        oSimpleFileAccess = createUnoService("com.sun.star.ucb.SimpleFileAccess")
        oSimpleFileAccess.writeFile(stChemin, oStream)
        oShell = createUnoService("com.sun.star.system.SystemShellExecute")
        stChamp = convertToUrl(stChemin)
        oShell.execute(stChamp,,0)
END SUB
```

La position des autres champs concernés dans le formulaire est donnée par le bouton. S'il manque un nom de fichier, le fichier reçoit simplement le nom "Fichier".
Le contenu du champ de formulaire "**imgImage**" correspond à celui du champ Image de la table. Il est lu comme un flux de données. Le chemin d'accès au dossier temporaire est utilisé comme chemin pour ces données, il peut être défini en utilisant **Outils> Options> LibreOffice> Chemins**. Si les données doivent ensuite être utilisées à d'autres fins, et pas seulement affichées, elles peuvent être copiées à partir de ce chemin. Dans la macro, le fichier est ouvert directement après une lecture réussie, en utilisant le programme qui a été lié au suffixe de fichier par le système d'exploitation.

Toutes ces procédures font partie du fichier Exemple_Connexions_Images.odb dans la base d'exemples disponible avec ce manuel.

Extraits de code

Ces extraits de code proviennent de demandes faites dans des listes de diffusion. Des problèmes particuliers surgissent qui pourraient peut-être être utiles comme solutions pour vos propres expériences de base de données.

Obtenir l'âge actuel de quelqu'un

Une requête doit calculer l'âge réel d'une personne à partir d'une date de naissance. Voir également les fonctions dans l'annexe à ce Guide de base.

```
SELECT DATEDIFF('yy',"DateNais", CURDATE()) AS "Age" FROM "Individus"
```

Cette requête donne l'âge comme une différence en années. Mais, l'âge d'un enfant né le 31 décembre 2011 serait donné à 1 an au 1er janvier 2012, car il s'agit d'une année bissextile. Nous devons donc également tenir compte de la position de la journée dans l'année. Ceci est accessible en utilisant la fonction DAYOFYEAR(). Une autre fonction effectuera la comparaison.

```
SELECT CASEWHEN (DAYOFYEAR("Datenais") > DAYOFYEAR(CURDATE()),
            DATEDIFF ('yy',"Datenais", CURDATE())-1,
            DATEDIFF ('yy',"Datenais", CURDATE()))
AS "Age" FROM "Individus"
```

Maintenant, nous obtenons l'âge actuel correct en années.

CASEWHEN peut également être utilisé pour faire apparaître le texte DateNais aujourd'hui dans un autre champ, si `DAYOFYEAR("DateNais") = DAYOFYEAR(CURDATE())`.

Une subtile objection pourrait alors surgir : "Et les années bissextiles?". Pour les personnes nées après le 28 février, il y aura une erreur d'un jour. Pas un problème sérieux dans l'usage quotidien, mais ne devrions-nous pas rechercher la précision ?

```
CASEWHEN (
      (MONTH("DateNais") > MONTH(CURDATE())) OR
```

```
      ((MONTH("DateNais") = MONTH(CURDATE())) AND
      (DAY("DateNais") >      DAY(CURDATE()))),
DATEDIFF('yy',"DateNais", CURDATE())-1,
DATEDIFF('yy',"DateNais", CURDATE()))
```

Le code ci-dessus atteint cet objectif. Tant que le mois de la date de naissance est supérieur au mois en cours, la fonction de différence d'année soustrait un an. De même, un an sera soustrait lorsque les deux mois sont identiques, mais le jour du mois de la date de naissance est supérieur au jour de la date actuelle. Malheureusement, cette formule n'est pas compréhensible par l'interface graphique. Seule une *commande SQL directe* gérera cette requête avec succès et cela empêcherait notre requête d'être modifiée. Mais la requête doit être modifiable, voici donc comment tromper l'interface graphique :

```
CASE
      WHEN MONTH("DateNais") > MONTH(CURDATE())
            THEN DATEDIFF('yy',"DateNais", CURDATE())-1
      WHEN (MONTH("DateNais") = MONTH(CURDATE()) AND
            DAY("DateNais") > DAY(CURDATE()))
            THEN DATEDIFF('yy',"DateNais", CURDATE())-1
      ELSE DATEDIFF('yy',"DateNais", CURDATE())
END
```

Avec cette formulation, l'interface graphique ne réagit plus avec un message d'erreur. L'âge est maintenant donné avec précision même pendant les années bissextiles et la requête reste modifiable.

Affichage des anniversaires qui auront lieu dans les prochains jours

À l'aide d'un petit extrait de calcul, nous pouvons déterminer à partir de la table qui fêtera son anniversaire dans les huit prochains jours.

```
SELECT *
FROM "Table"
WHERE
      DAYOFYEAR("Date") BETWEEN DAYOFYEAR(CURDATE()) AND
            DAYOFYEAR(CURDATE()) + 7
      OR DAYOFYEAR("Date") < 7 –
            DAYOFYEAR(CAST(YEAR(CURDATE())||'-12-31' AS DATE)) +
            DAYOFYEAR(CURDATE())
```

La requête affiche tous les enregistrements dont la date d'entrée se situe entre le jour actuel de l'année et les 7 jours suivants.

Pour afficher 8 jours même à la fin d'une année, le jour du début de l'année doit être soigneusement vérifié. Cette vérification se produit uniquement pour les numéros de jour qui sont au plus 7 jours après le dernier numéro de jour de l'année en cours (généralement 365) plus le numéro de jour de la date actuelle. Si la date actuelle est à plus de 7 jours de la fin de l'année, le total est <1. Aucun enregistrement dans la table n'a une date comme celle-là, donc dans de tels cas, cette condition partielle n'est pas remplie.

Dans la formule ci-dessus, les années bissextiles donneront un résultat erroné, car leurs dates sont déplacées par l'occurrence du 29 février. Le code doit être plus complet pour éviter cette erreur :

```
SELECT *
FROM "Table"
WHERE
      CASE
            WHEN
                  DAYOFYEAR(CAST(YEAR("Date")||'-12-31' AS DATE)) = 366
                  AND DAYOFYEAR("Date") > 60 THEN DAYOFYEAR("Date") – 1
            ELSE
                  DAYOFYEAR("Date")
      END
BETWEEN
      CASE
            WHEN
                  DAYOFYEAR(CAST(YEAR(CURDATE())||'-12-31' AS DATE)) = 366
                  AND DAYOFYEAR(CURDATE()) > 60 THEN DAYOFYEAR(CURDATE()) – 1
            ELSE
                  DAYOFYEAR(CURDATE())
      END
AND
      CASE
            WHEN
                  DAYOFYEAR(CAST(YEAR(CURDATE())||'-12-31' AS DATE)) = 366
                  AND DAYOFYEAR(CURDATE()) > 60 THEN DAYOFYEAR(CURDATE()) + 6
            ELSE
                  DAYOFYEAR(CURDATE()) + 7
      END
OR DAYOFYEAR("DateNais") < 7 –
      DAYOFYEAR(CAST(YEAR(CURDATE())||'-12-31' AS DATE)) +
      DAYOFYEAR(CURDATE())
```

Les années bissextiles peuvent être reconnues en ayant 366 comme nombre total de jours plutôt que 365. Ceci est utilisé pour la détermination correspondante.

D'une part, chaque valeur de date doit être testée pour voir si elle se situe dans une année bissextile, et également pour le décompte correct pour le 60e jour (31 jours en janvier et 29 en février). Dans ce cas, toutes les valeurs `DAYOFYEAR` suivantes pour la date doivent être augmentées de 1. Ensuite, le 1er mars d'une année bissextile correspondra exactement au 1er mars d'une année normale.

D'autre part, l'année en cours (`CURDATE()`) doit être testée pour voir s'il s'agit en fait d'une année bissextile. Ici aussi, le nombre de jours doit être augmenté de 1.

L'affichage de la valeur finale pour les 8 prochains jours n'est pas non plus si simple, car l'année n'est toujours pas incluse dans la requête. Cependant, ce serait une condition facile à ajouter : `YEAR("Date") = YEAR(CURDATE())` pour l'année courante ou `YEAR("Date") = YEAR(CURDATE()) + 1` pour l'année en cours.

Ajout de jours à la valeur de la date

Lors du prêt de supports, la bibliothèque peut souhaiter connaître le jour exact où le support doit être retourné. Malheureusement, le HSQLDB interne ne fournit pas la fonction `DATEADD()` qui est disponible dans de nombreuses bases de données externes et également dans Firebird interne. Voici une manière détournée d'y parvenir pour une durée limitée.

Tout d'abord, un tableau est créé contenant une séquence de dates couvrant la période souhaitée. Pour cela, Calc est ouvert et le nom "ID" est placé dans le champ A1 et "Date" dans le champ B1. Dans le champ A2, nous saisissons 1 et dans le champ B2 la date de début, par exemple 15/01/2015. Sélectionnez A2 et B2 et faites-les glisser vers le bas. Cela créera une séquence de nombres dans la colonne A et une séquence de dates dans la colonne B.

Ensuite, tout ce tableau, y compris les en-têtes de colonnes, est sélectionné et importé dans Base : **clic droit> Coller> Nom de la table> Dates**. **Sous Options, Définition et données et Utiliser la première ligne comme noms de colonne** sont validés par un clic. Toutes les colonnes sont transférées. Après cela, assurez-vous que le champ ID a le type Integer [INTEGER] et le champ Date le type Date [DATE]. Une clé primaire n'est pas nécessaire, car les enregistrements ne seront pas modifiés ultérieurement. Puisqu'une clé primaire n'a pas été définie, la table est protégée en écriture.

Conseil

Vous pouvez également utiliser une technique de requête pour créer une telle vue. Si vous utilisez une table de filtrage, vous pouvez même contrôler la date de début et la plage de valeurs de date.

```
SELECT DISTINCT CAST
      ("Y"."No" +
         (SELECT "Annee" FROM "Filter" WHERE "ID" = True) - 1 || '-' ||
      CASEWHEN("M"."No" < 10, '0' || "M"."No", '' || "M"."No") || '-' ||
      CASEWHEN("D"."No" < 10, '0' || "D"."No", '' || "D"."No")
      AS DATE) AS "Date"
FROM "Noto31" AS "D", "Noto31" AS "M", "Noto31" AS "Y"
WHERE "Y"."No" <= (SELECT "Annee" FROM "Filter" WHERE "ID" = True) AND
"M"."No" <= 12 AND "D"."No" <= 31
```

Cette vue accède à une table **"Noto31"** qui contient uniquement les nombres de 1 à 31 et est protégée en écriture. Un autre tableau de filtre contient l'année de départ et la plage d'année que la vue doit couvrir. La date est assemblée à partir de ceux-ci, créant une expression de date (année, mois, jour) dans le texte, qui peut ensuite être convertie en date. HSQLDB accepte tous les jours jusqu'à 31 par mois et des chaînes comme le 31/02/2015. Cependant, le 31/02/2015 est transmis en tant que 3/03/2015. Par conséquent, lors de la préparation de la vue, vous devez utiliser DISTINCT pour exclure les valeurs de date en double.
Ici, la vue suivante est effective :

```
SELECT "a"."Date",
      (SELECT COUNT(*) FROM "Vue_Date" WHERE "Date" ≤ "a"."Date")
         AS "lfdNo"
FROM "Vue_Date" AS "a"
```

Grâce à la numérotation des lignes, la valeur de la date est convertie en un nombre. Comme vous ne pouvez pas supprimer des données dans une vue, aucune protection supplémentaire en écriture n'est nécessaire.
En utilisant une requête, nous pouvons maintenant déterminer une date spécifique, par exemple la date dans 14 jours :

```
SELECT "a"."Date_Pret",
      (SELECT "Date" FROM "Date" WHERE "ID" =
         (SELECT "ID" FROM "Date" WHERE "Date" = "a"."Date_Pret")+14)
         AS "Date_Retour"
```

```
FROM "Prets" AS "a"
```

La première colonne indique la date du prêt. Cette colonne est accessible par une sous-requête de corrélation qui est à nouveau divisée en deux requêtes. `SELECT "ID" FROM "Date"` donne la valeur du champ ID, correspondant à la date d'émission. 14 jours sont ajoutés à la valeur. Le résultat est affecté au champ ID par la sous-requête externe. Ce nouvel identifiant détermine ensuite la date qui entre dans le champ de date.

Malheureusement dans l'affichage de cette requête, le type de date n'est pas automatiquement reconnu, de sorte qu'il devient nécessaire d'utiliser le formatage. Dans un formulaire, l'affichage correspondant peut être stocké, de sorte que chaque requête produira une valeur de date.

Une variante directe pour déterminer la valeur de la date est possible en utilisant un moyen plus court :

```
SELECT "Date_Pret",
      DATEDIFF('dd', '1899-12-30', "Date_Pret") + 14
         AS "Date_Retour"
FROM "Prets"
```

La valeur numérique renvoyée peut être mise en forme dans un formulaire en tant que date, à l'aide d'un champ formaté. Cependant, il faut beaucoup de travail pour le rendre disponible pour un traitement SQL ultérieur dans une requête.

Ajout d'une heure à un horodatage

MySQL a une fonction appelée TIMESTAMPADD(). Une fonction similaire n'existe pas dans HSQLDB. Mais la valeur numérique interne de l'horodatage peut être utilisée pour faire l'addition ou la soustraction, en utilisant un champ formaté dans un formulaire.

Contrairement à l'ajout de jours à une date, les heures posent un problème qui peut ne pas être évident au début.

```
SELECT "HoroDatage"
      DATEDIFF('ss', '1899-12-30', "HoroDatage") / 86400.0000000000 +
         36/24 AS "HoroDatage+36 heures"
FROM "Table"
```

La nouvelle heure calculée est basée sur la différence avec l'heure zéro du système. Comme dans les calculs de date, il s'agit de la date du 30/12/1899.

Note

La date zéro du 30/12/1899 est censée avoir été choisie, car l'année 1900, contrairement à la plupart des années divisibles par 4, n'était pas une année bissextile. La balise "1" du calcul interne a donc été replacée au 31/12/1899 et non au 01/01/1900.

La différence est exprimée en secondes, mais le nombre interne compte les jours sous forme de nombres avant la virgule décimale et les heures, minutes et secondes sous forme de décimales. Puisqu'un jour contient 60 * 60 * 24 secondes, le deuxième décompte doit être divisé par 86400 pour pouvoir calculer correctement les jours et les fractions de jours. Si la HSQLDB interne doit donner des décimales, elles doivent être incluses dans le calcul, donc au lieu de 86400, nous devons diviser par 86400.0000000000. Les décimales dans une requête doivent utiliser un point

décimal comme séparateur, quelles que soient les conventions locales. Le résultat aura 10 décimales après le point.

À ce résultat, il faut ajouter le nombre total d'heures sous forme de fraction de journée. Le chiffre calculé, correctement formaté, peut être créé dans la requête. Malheureusement, le formatage n'est pas enregistré, mais il peut être transféré avec le format correct à l'aide d'un champ formaté dans un formulaire ou un état.

Si des minutes ou des secondes doivent être ajoutées, veillez à ce qu'elles soient fournies sous forme de fractions de jour.

Si la date tombe entre novembre, décembre, janvier, etc., le calcul ne pose aucun problème. Ils semblent assez précis : ajouter 36 heures à un horodatage du 20/01/2015 13:00:00 donne 22/01/2015 00:00:00. Mais les choses sont différentes pour le 20/04/2015 13:00:00. Le résultat est le 22/04/2015 00:00:00. Le calcul tourne mal à cause de l'heure d'été. L'heure "perdue" ou "gagnée" par le changement d'heure n'est pas prise en compte. Dans un même fuseau horaire, il existe différentes manières d'obtenir un résultat "correct". Voici une variation simple :

```
SELECT "HoroDatage"
      DATEDIFF('dd', '1899-12-30', "HoroDatage") +
         HOUR("HoroDatage") / 24.0000000000 +
         MINUTE("HoroDatage") / 1440.0000000000 +
         SECOND("HoroDatage") / 86400.0000000000 +
         36/24
            AS "HoroDatage+36hours"
FROM "Table"
```

Au lieu de compter les heures, les minutes et les secondes depuis l'origine de la date, elles sont comptées à partir de la date actuelle. Le 20/05/2015, l'heure est 13:00 mais sans l'heure d'été, elle sera indiquée comme 12:00. La fonction HEURE prend en compte l'heure d'été et donne 13 heures comme partie horaire de l'heure. Cela peut ensuite être ajouté correctement à la partie quotidienne. Les minutes et les secondes sont traitées exactement de la même manière. Enfin, les heures supplémentaires sont ajoutées sous forme de partie fractionnaire d'une journée et le tout est affiché sous forme d'horodatage calculé à l'aide du formatage de cellule.

Deux choses doivent être gardées en vue dans ce calcul :

- Lors du passage de l'heure d'hiver à l'heure d'été, les valeurs horaires ne s'affichent pas correctement. Cela peut être corrigé à l'aide d'un tableau auxiliaire, qui prend les dates de début et de fin de l'heure d'été et corrige le décompte horaire. Une affaire un peu compliquée.
- L'affichage des heures n'est possible qu'avec des champs formatés. Le résultat est un nombre décimal, pas un horodatage qui pourrait être stocké directement en tant que tel dans la base de données. Soit il doit être copié dans le formulaire, soit converti d'un nombre décimal en un horodatage à l'aide d'une requête complexe. Le point de rupture dans la conversion est la valeur de la date, car des années bissextiles ou des mois avec différents nombres de jours peuvent être impliqués.

Obtenir un solde courant par catégories

Au lieu d'utiliser un livre de ménage, une base de données sur un PC peut simplifier la tâche fastidieuse consistant à additionner les dépenses de nourriture, de vêtements, de transport, etc. Nous voulons que la plupart de ces détails soient immédiatement visibles dans la base de données, notre exemple suppose donc que les revenus et les dépenses seront stockés sous forme

de valeurs signées dans un champ appelé Montant. En principe, le tout peut être étendu pour couvrir des champs séparés et une sommation appropriée pour chacun.

```
SELECT "ID", "Montant", (SELECT SUM("Montant") FROM "Especes"
            WHERE "ID" <= "a"."ID") AS "Balance"
      FROM "Especes" AS "a" ORDER BY "ID" ASC
```

Cette requête entraîne pour chaque nouvel enregistrement un calcul direct du solde courant du compte. Dans le même temps, la requête reste modifiable, car le champ Balance est créé via une sous-requête de corrélation. La requête dépend de l'ID de clé primaire créé automatiquement pour calculer l'état du compte. Cependant, les soldes sont généralement calculés sur une base quotidienne. Nous avons donc besoin d'une requête de date.

```
SELECT "ID", "Date", "Montant", (SELECT SUM("Montant") FROM "Especes"
WHERE "Date" <= "a"."Date") AS "Balance"
FROM "Especes" AS "a" ORDER BY "Date", "ID" ASC
```

Les dépenses apparaissent maintenant triées et additionnées par date. Reste la question de la catégorie, puisque nous voulons des soldes correspondants pour les différentes catégories de dépenses.

```
SELECT "ID", "Date", "Montant", "ID_Compte",
(SELECT "Compte" FROM "Compte" WHERE "ID" = "a"."ID_Compte") AS "Compte_Nom",
(SELECT SUM("Montant") FROM "Especes" WHERE "Date" <= "a"."Date" AND
"ID_Compte" = "a"."ID_Compte") AS "Balance",
(SELECT SUM("Montant") FROM "Especes" WHERE "Date" <= "a"."Date") AS
"Total_balance"
FROM "Especes" AS "a" ORDER BY "Date", "ID" ASC
```

Cela crée une requête modifiable dans laquelle, en plus des champs de saisie (Date, Montant, ID_Compte), le nom du compte, le solde pertinent et le solde total apparaissent ensemble. Comme les sous-requêtes corrélées sont partiellement basées sur les entrées précédentes ("Date" <= "a". "Date"), seules les nouvelles entrées seront traitées sans problème. Les modifications apportées à un enregistrement précédent ne sont initialement détectables que dans cet enregistrement. La requête doit être mise à jour si des calculs ultérieurs qui en dépendent doivent être effectués.

Numérotation des lignes

Les champs incrémentés automatiquement sont très utiles. Cependant, ils ne vous indiquent pas avec certitude combien d'enregistrements sont présents dans la base de données ou sont réellement disponibles pour être interrogés. Les enregistrements sont souvent supprimés et de nombreux utilisateurs essaient en vain de déterminer quels numéros ne sont plus présents afin de faire correspondre le numéro courant.

```
SELECT "ID", (SELECT COUNT("ID") FROM "Table" WHERE "ID" <= "a"."ID") AS
"No." FROM "Table" AS "a"
```

Le champ ID est lu et le second champ est déterminé par une sous-requête corrélée, qui cherche à déterminer combien de valeurs de champ dans ID sont inférieures ou égales à la valeur de champ actuelle. À partir de là, un numéro de ligne courant est créé.

Chaque enregistrement auquel vous souhaitez appliquer cette requête contient des champs. Pour appliquer cette requête aux enregistrements, vous devez d'abord ajouter ces champs à la requête. Vous pouvez les placer dans l'ordre de votre choix dans la clause SELECT. Si vous disposez des

enregistrements dans un formulaire, vous devez modifier le formulaire afin que les données du formulaire proviennent de cette requête.

Par exemple, l'enregistrement contient champ1, champ2 et champ3. La requête complète serait :

```
SELECT "ID", "Champ1", "Champ2", "Champ3", (SELECT COUNT("ID") FROM "Table"
WHERE "ID" <= "a"."ID") AS "No." FROM "Table" AS "a"
```

Une numérotation pour un groupement correspondant est également possible :

```
SELECT "ID", "Calcul", (SELECT COUNT("ID") FROM "Table" WHERE "ID" <=
"a"."ID" AND "Calcul" = "a"."Calcul") AS "No." FROM "Table" AS "a" ORDER BY
"ID" ASC, "No." ASC
```

Ici, une table contient différents nombres calculés. ("Calcul"). Pour chaque nombre calculé, "No." est exprimé séparément dans l'ordre croissant après le tri sur le champ ID. Cela produit une numérotation à partir de 1.

Si l'ordre de tri réel dans la requête correspond aux numéros de ligne, un type de tri approprié doit être mappé. Pour cela, le champ de tri doit avoir une valeur unique dans tous les enregistrements. Sinon, deux emplacements auront la même valeur. Cela peut en fait être utile si, par exemple, l'ordre de passage dans un concours doit être représenté, car des résultats identiques conduiront alors à une position commune. Pour que l'ordre de place soit exprimé de telle manière que, en cas de positions conjointes, la valeur suivante soit omise, la requête doit être construite différemment :

```
SELECT "ID", (SELECT COUNT("ID") + 1 FROM "Table" WHERE "Temps" <
"a"."Temps") AS "Place" FROM "Table" AS "a"
```

Toutes les entrées sont évaluées pour lesquelles le champ Heure a une valeur plus petite. Cela couvre tous les athlètes qui ont atteint le poste de vainqueur avant l'athlète actuel. A cette valeur s'ajoute le chiffre 1. Celui-ci détermine la place de l'athlète actuel. Si le temps est identique à celui d'un autre athlète, ils sont placés conjointement. Cela permet de passer des commandes telles que la 1ère place, la 2e place, la 2e place, la 4e place.

Ce serait plus problématique si les numéros de ligne étaient nécessaires ainsi qu'un numéro d'emplacement. Cela peut être utile si plusieurs enregistrements doivent être combinés sur une seule ligne.

```
SELECT "ID", (SELECT COUNT("ID") + 1 FROM "Table" WHERE "Temps" <
"a"."Temps") AS "Place",
CASE WHEN
(SELECT COUNT("ID") + 1 FROM "Table" WHERE "Temps" = "a"."Temps") = 1
THEN (SELECT COUNT("ID") + 1 FROM "Table" WHERE "Temps" < "a"."Temps")
ELSE (SELECT (SELECT COUNT("ID") + 1 FROM "Table" WHERE "Temps" <
"a"."Temps") + COUNT("ID") FROM "Table" WHERE "Temps" = "a"."Temps" "ID" <
"a"."ID"
END
AS "LineNumber" FROM "Table" AS "a"
```

La deuxième colonne donne toujours l'ordre de la place. La troisième colonne vérifie d'abord si une seule personne a franchi la ligne avec ce temps. Si tel est le cas, la commande passée est directement convertie en un numéro de ligne. Sinon, une valeur supplémentaire est ajoutée à la commande. Pour le même temps ("Heure" = "a"."Heure") au moins 1 est ajouté, s'il y a une autre personne avec l'ID de clé primaire, dont la clé primaire est plus petite que la clé primaire dans l'enregistrement actuel (" ID "<" a "." ID "). Cette requête donne donc des valeurs identiques pour la commande à condition qu'aucune deuxième personne avec la même heure n'existe. Si une

deuxième personne avec la même heure existe, l'ID détermine quelle personne a le numéro de ligne le plus petit.

Incidemment, ce tri par numéro de ligne peut servir quel que soit le but souhaité par les utilisateurs de la base de données. Par exemple, si une série d'enregistrements est triée par nom, les enregistrements portant le même nom ne sont pas triés au hasard mais en fonction de leur clé primaire, qui est bien entendu unique. De cette manière également, la numérotation peut conduire à un tri des enregistrements.

La numérotation des lignes est également un bon prélude à la combinaison d'enregistrements individuels en un seul enregistrement. Si une requête de numérotation de ligne est créée en tant que vue, une autre requête peut lui être appliquée sans créer de problème. À titre d'exemple simple, voici à nouveau la première requête de numérotation avec un champ supplémentaire :

```
SELECT "ID", "Nom", (SELECT COUNT("ID") FROM "Table" WHERE "ID" <= "a"."ID")
AS "No." FROM "Table" AS "a"
```

Cette requête est transformée en vue Vue1. La requête peut être utilisée, par exemple, pour rassembler les trois premiers noms sur une seule ligne :

```
SELECT "Nom" AS "Nom_1", (SELECT "Nom" FROM "Vue1" WHERE "No." = 2) AS
"Nom_2", (SELECT "Nom" FROM "Vue1" WHERE "No." = 3) AS "Nom_3" FROM "Vue1"
WHERE "No." = 1
```

De cette manière, plusieurs enregistrements peuvent être convertis en champs adjacents. Cette numérotation va simplement du premier au dernier enregistrement.

Si tous ces individus doivent se voir attribuer le même nom de famille, cela peut être effectué comme suit :

```
SELECT "ID", "Nom", "Nom", (SELECT COUNT("ID") FROM "Table" WHERE "ID" <=
"a"."ID" AND "Nom" = "a"."Nom") AS "No." FROM "Table" AS "a"
```

Maintenant que la vue a été créée, la famille peut être assemblée.

```
SELECT "Nom", "Nom" AS "Nom_1", (SELECT "Nom" FROM "Vue1" WHERE "No." = 2 AND
"Nom" = "a"."Nom") AS "Nom_2", (SELECT "Nom" FROM "Vue1" WHERE "No." = 3 AND
"Nom" = "a"."Nom") AS "Nom_3" FROM "Vue1" AS "a" WHERE "No." = 1
```

De cette manière, dans un carnet d'adresses, tous les membres d'une même famille ("Nom") peuvent être rassemblés afin que chaque adresse ne soit prise en compte qu'une seule fois lors de l'envoi d'une lettre, mais tous ceux qui devraient recevoir la lettre sont répertoriés.

Nous devons être prudents ici, car nous ne voulons pas d'une fonction en boucle sans fin. La requête de l'exemple ci-dessus limite à 3 les enregistrements parallèles qui doivent être convertis en champs. Cette limite a été choisie délibérément. Aucun autre nom n'apparaîtra même si la valeur de "No." est supérieur à 3.

Dans quelques cas, une telle limite est clairement compréhensible. Par exemple, si nous créons un calendrier, les lignes peuvent représenter les semaines de l'année et les colonnes les jours de la semaine. Comme dans le calendrier d'origine, seule la date détermine le contenu du champ, la numérotation des lignes est utilisée pour numéroter les jours de chaque semaine en continu, puis les semaines de l'année deviennent les enregistrements. Cela nécessite que la table contienne un champ de date avec des dates continues et un champ pour les événements. De plus, la date la plus ancienne créera toujours un "No." = 1. Donc, si vous voulez que le calendrier commence le lundi, la première date doit être le lundi. La colonne 1 est alors lundi, la colonne 2 mardi et ainsi de

suite. La sous-requête se termine alors par "No." = 7. De cette manière, les sept jours de la semaine peuvent être affichés côte à côte et une vue de calendrier correspondante créée.

Obtenir un saut de ligne dans une requête

Parfois, il est utile d'assembler plusieurs champs à l'aide d'une requête et de les séparer par des sauts de ligne, par exemple lors de la lecture d'une adresse complète dans un rapport.

Le saut de ligne dans la requête est représenté par `Char (13)`. Exemple :

```
SELECT "Prenom"||' '||"Nom"||Char(13)||"Rue"||Char(13)||"Ville" FROM "Table"
```

Cela donne :

Prenom Nom
Rue
Ville

Une telle requête, avec une numérotation de ligne jusqu'à 3, vous permet d'imprimer des étiquettes d'adresse sur trois colonnes en créant un rapport. La numérotation est nécessaire à cet égard pour que trois adresses puissent être placées l'une à côté de l'autre dans un enregistrement. C'est la seule façon dont ils resteront côte à côte lorsqu'ils seront lus dans le rapport.

Dans certains systèmes d'exploitation (Windows, notamment), il est nécessaire d'inclure char (10) à côté de char (13) dans le code.

```
SELECT "Prenom"||' '||"Nom"||Char(13)||Char(10)||"Rue"||Char(13)||
Char(10)||"Ville" FROM "Table"
```

Regrouper et résumer

Pour les autres bases de données et pour les versions plus récentes de HSQLDB, la commande **Group_Concat()** est disponible. Elle peut être utilisée pour regrouper des champs individuels dans un enregistrement en un seul champ. Ainsi, par exemple, il est possible de stocker les prénoms et noms de famille dans une table, puis de présenter les données de telle sorte qu'un champ affiche les noms de famille comme noms de famille tandis qu'un deuxième champ contient tous les prénoms pertinents de manière séquentielle, séparés par des virgules.

Cet exemple est similaire à bien des égards à la numérotation des lignes. Le regroupement dans un champ commun est une sorte de complément à cela.

NomDeFamille	*Prenom*
Titegoutte	Anne
Térieur	Alain
Titegoutte	Emma
Térieur	Alex
Titegoutte	Justine
Titegoutte	Corinne
Titegoutte	Germaine

est converti par la requête en :

NomDeFamille	*Prenom*
Titegoutte	Anne, Emma, Justine, Corinne, Germaine
Térieur	Alain, Alex

Cette procédure peut, dans certaines limites, être exprimée en HSQLDB. L'exemple suivant fait référence à une table appelée Noms avec les champs ID, Prénom et Nom. La requête suivante est d'abord exécutée sur la table et enregistrée en tant que vue appelée Vue_Groupe.

```
SELECT "Nom", "Prenom", (SELECT COUNT("ID") FROM "Nom" WHERE "ID" <= "a"."ID"
AND "Nom" = "a"."Nom") AS "GroupNo" FROM "Nom" AS "a"
```

Vous pouvez lire dans le chapitre Requêtes comment cette requête accède au contenu du champ dans la même ligne de requête. Il produit une séquence numérotée ascendante, regroupée par nom. Cette numérotation est nécessaire pour la requête suivante, de sorte que dans l'exemple, un maximum de 5 prénoms soit répertorié.

```
SELECT "Nom",
(SELECT "Prenom" FROM "Vue_Groupe" WHERE "Nom" = "a"."Nom" AND "GroupNo" = 1)
||
IFNULL((SELECT ', ' || "Prenom" FROM "Vue_Groupe" WHERE "Nom" = "a"."Nom" AND
"GroupNo" = 2), '') ||
IFNULL((SELECT ', ' || "Prenom" FROM "Vue_Groupe" WHERE "Nom" = "a"."Nom" AND
"GroupNo" = 3), '') ||
IFNULL((SELECT ', ' || "Prenom" FROM "Vue_Groupe" WHERE "Nom" = "a"."Nom" AND
"GroupNo" = 4), '') ||
IFNULL((SELECT ', ' || "Prenom" FROM "Vue_Groupe" WHERE "Nom" = "a"."Nom" AND
"GroupNo" = 5), '')
AS "Prenoms"
FROM "Vue_Groupe" AS "a"
```

À l'aide de sous-requêtes, les prénoms des membres du groupe sont recherchés l'un après l'autre et combinés. À partir de la deuxième sous-requête, vous devez vous assurer que les valeurs "NULL" ne définissent pas la combinaison entière sur "NULL". C'est pourquoi un résultat chaîne vide «» plutôt que "NULL" est affiché.

Annexe

Texte des requêtes SQL

Ces requêtes sont celles présentées dans les copies d'écran du chapitre Recherche de données.

Requête LOCATE – page 377

```
SELECT "ID", "Memo",
      LOCATE(LOWER(: TexteCherche), LOWER ("Memo")) AS "Position"
FROM "Table"
```

Requête cible - page 378

```
SELECT "ID", "Memo", "Position",
      CASE
         WHEN "Position" = 0 THEN '** Non trouvé **'
         WHEN "Position" < 10 THEN SUBSTRING ("Memo", 1, 25)
         ELSE SUBSTRING ("Memo", LOCATE(' ', "Memo", "Position" – 10) + 1,
25)
END AS "Cible"
FROM
      (SELECT "ID", "Memo", LOCATE(LOWER(: TexteCherche), LOWER ("Memo")) AS
"Position"
FROM "Table")
```

Cible sans coupures – page 380

```
SELECT "ID", "Memo", "Position",
      CASE
            WHEN "Position" = 0 THEN '** Non trouvé **'
            WHEN "Position" < 10 THEN SUBSTRING ("Memo",1, LOCATE ('
',"Memo", 25))
            ELSE SUBSTRING ("Memo", LOCATE (' ', "Memo", "Position" – 10) +
1,
                  (LOCATE (' ', "Memo", "Position" + 20) -
                        (LOCATE (' ', "Memo", "Position" – 10) + 1)))
      END AS "Cible"
FROM
      (SELECT "ID", "Memo", LOCATE(LOWER(: TexteCherche), LOWER("Memo")) AS
"Position"
FROM "Table")
```

Cibles multiples

```
SELECT "ID", "Memo", "Position01", "Cible01", "Position02",
      CASE
         WHEN "Position02" = 0 THEN '** Non trouvé **'
         WHEN "Position02" < 10 THEN SUBSTRING ("Memo",1, LOCATE (' ','Memo",
25))
         ELSE SUBSTRING ("Memo", LOCATE (' ', "Memo", "Position02" – 10) + 1,
            (LOCATE (' ', "Memo", "Position02" + 20) -
               (LOCATE (' ', "Memo", "Position02" – 10) + 1))
            )
      END AS "Cible02",
      CASE
```

```
         WHEN "Position02" = 0 THEN 0
         ELSE LOCATE(LOWER(: TexteCherche), LOWER("Memo"), "Position02" + 1)
      END AS "Position03"
FROM
      (SELECT "ID", "Memo", "Position01",
      CASE
         WHEN "Position01" = 0 THEN '** Non trouvé **'
         WHEN "Position01" < 10 THEN SUBSTRING ("Memo",1, LOCATE (' ',"Memo",
25))
         ELSE SUBSTRING ("Memo", LOCATE (' ', "Memo", "Position01" - 10) + 1,
            (LOCATE (' ', "Memo", "Position01" + 20) -
               (LOCATE (' ', "Memo", "Position01" - 10) + 1))
            )
      END AS "Cible01",
      CASE
         WHEN "Position01" = 0 THEN 0
         ELSE LOCATE(LOWER(: TexteCherche), LOWER("Memo"), "Position01" + 1)
      END AS "Position02"
      FROM
      (SELECT "ID", "Memo", LOCATE(LOWER(: TexteCherche), LOWER("Memo")) AS
"Position01"
      FROM "Table")
      )
```

Guide Base

Chapitre 9
Macros

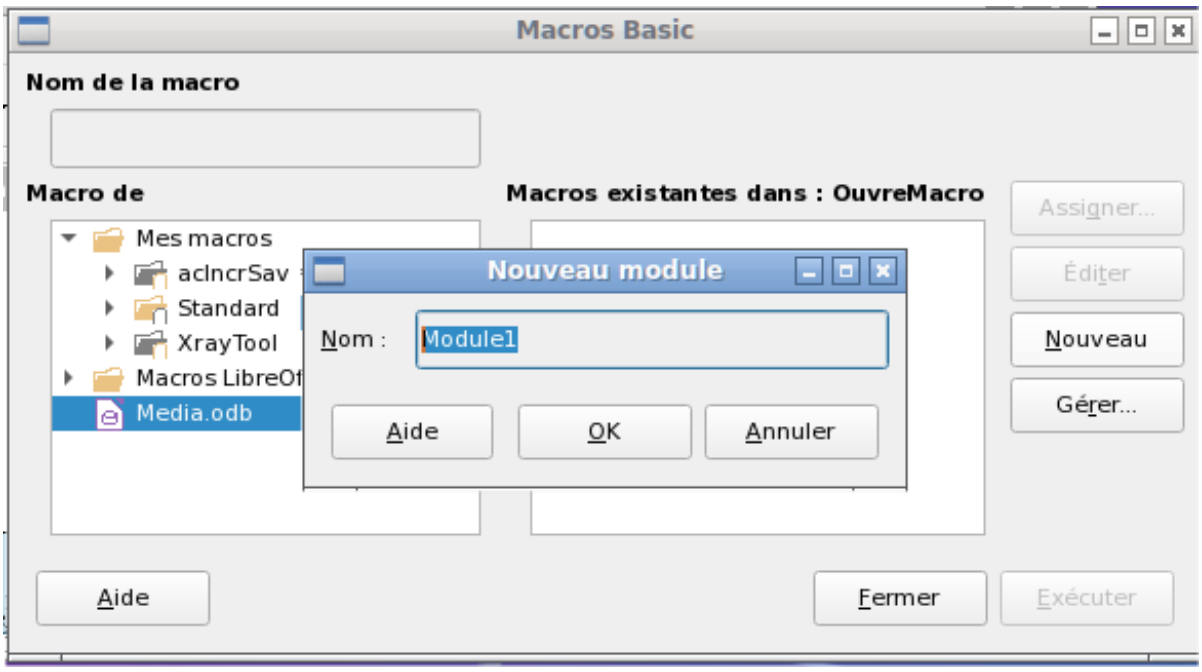
Remarques générales sur les macros

En principe, une base de données dans Base peut être gérée sans macros. Parfois, cependant, elles peuvent devenir nécessaires pour :

- Prévenir plus efficacement les erreurs de saisie ;
- Simplifier certaines tâches de traitement (passage d'un formulaire à un autre, mise à jour des données après saisie dans un formulaire, etc.) ;
- Permettre à certaines commandes SQL d'être appelées plus facilement qu'avec l'éditeur SQL séparé.

Vous devez décider vous-même de l'intensité avec laquelle vous souhaitez utiliser les macros dans Base. Les macros peuvent améliorer la convivialité mais sont toujours associées à de petites réductions de la vitesse du programme, et parfois à des plus grandes (lorsqu'elles sont mal codées). Il est toujours préférable de commencer par utiliser pleinement les possibilités de la base de données et les dispositions de configuration des formulaires avant d'essayer de fournir des fonctionnalités supplémentaires avec des macros. Les macros doivent toujours être testées sur des bases de données plus volumineuses pour déterminer leur effet sur les performances.

Les macros sont créées en utilisant **Outils> Macros> Gérer les macros> Basic**. Une fenêtre apparaît qui donne accès à toutes les macros. Pour Base, la zone importante correspond au nom de fichier du fichier Base.



Le bouton **Nouveau** de la boîte de dialogue Macros Basic ouvre la boîte de dialogue **Nouveau module**, qui demande le nom du module (le dossier dans lequel la macro sera classée). Le nom peut être modifié ultérieurement si vous le souhaitez.

Dès que cela est fait, l'éditeur de macro apparaît. Sa zone de saisie contient déjà le début et la fin d'un sous-programme :

```
REM ***** BASIC *****

Sub Main
```

```
End Sub
```

Si des macros doivent être utilisées, les étapes suivantes sont nécessaires :

- Sous **Outils> Options>LibreOffice > Sécurité> Sécurité des macros**, le niveau de sécurité doit être réduit à Moyen. La sécurité pourra être augmentée avec l'expérience. Si nécessaire, vous pouvez en outre utiliser l'onglet *Sources de confiance* pour définir le chemin d'accès à vos propres fichiers contenant des macros afin d'éviter les demandes d'activation ultérieures sur l'activation des macros.
- Les macros sont enregistrées au niveau où elles sont créées (Mes macros, Macros LibreOffice, fichier *.odb). Les macros adaptées à la gestion d'une base seront créées dans le fichier « .odb »de cette base

Quelques principes de base pour l'utilisation du code Basic dans LibreOffice :

- Les lignes n'ont pas de numéros de ligne, par défaut (il existe une option pour activer la numérotation : **Affichage > Numéro de ligne**) et doivent se terminer par un retour forcé. Elles peuvent également se terminer par un espace suivit du caractère « _ » et un retour forcé, soit parce que la ligne de code est trop longue, soit pour apporter plus de lisibilité au code.
- Les fonctions, expressions réservées et éléments similaires ne sont pas sensibles à la casse. Ainsi, « String » est identique à « STRING » ou « string » ou à toute autre combinaison de majuscules et minuscules. La casse ne doit être utilisée que pour améliorer la lisibilité. Cependant, les noms des constantes et des énumérations sont sensibles à la casse la première fois qu'ils sont vus par le compilateur de macros, il est donc préférable de toujours les écrire en utilisant la casse appropriée.
- On peut distinguer les procédures (commençant par Sub) et les fonctions (commençant par Function). Les procédures étaient à l'origine des segments de programme sans valeur de retour, tandis que les fonctions renvoient des valeurs qui peuvent être traitées ultérieurement. Mais cette distinction devient de moins en moins pertinente. Les gens utilisent aujourd'hui des termes tels que « méthode » ou « routine », qu'il y ait une valeur de retour ou non. Une procédure peut également avoir une valeur de retour (en dehors de « Variant »).

```
Sub maProcedure As Integer
      End Sub
```

Pour plus de détails, reportez-vous au chapitre 13, Prise en main des macros, dans le Guide de mise en route.

Note

Les macros des versions PDF et ODT de ce chapitre sont colorées selon les règles de l'éditeur de macro LibreOffice :

```
Macro Identificateur
Macro commentaire
Macro operateur
Macro Mot clé
Macro nombre
Macro chaîne de caractères
```

Macros dans Base

Utiliser des macros

La « voie directe », en utilisant **Outils> Macros> Exécuter la macro** est possible, mais pas habituelle pour les macros pour Base. Une macro est normalement affectée à un événement et lancée lorsque cet événement se produit. Les macros sont utilisées pour :

- Gestion des événements dans les formulaires
- Modifier une source de données dans un formulaire
- Basculer entre les contrôles de formulaire
- Réagir à ce que fait l'utilisateur à l'intérieur d'un contrôle

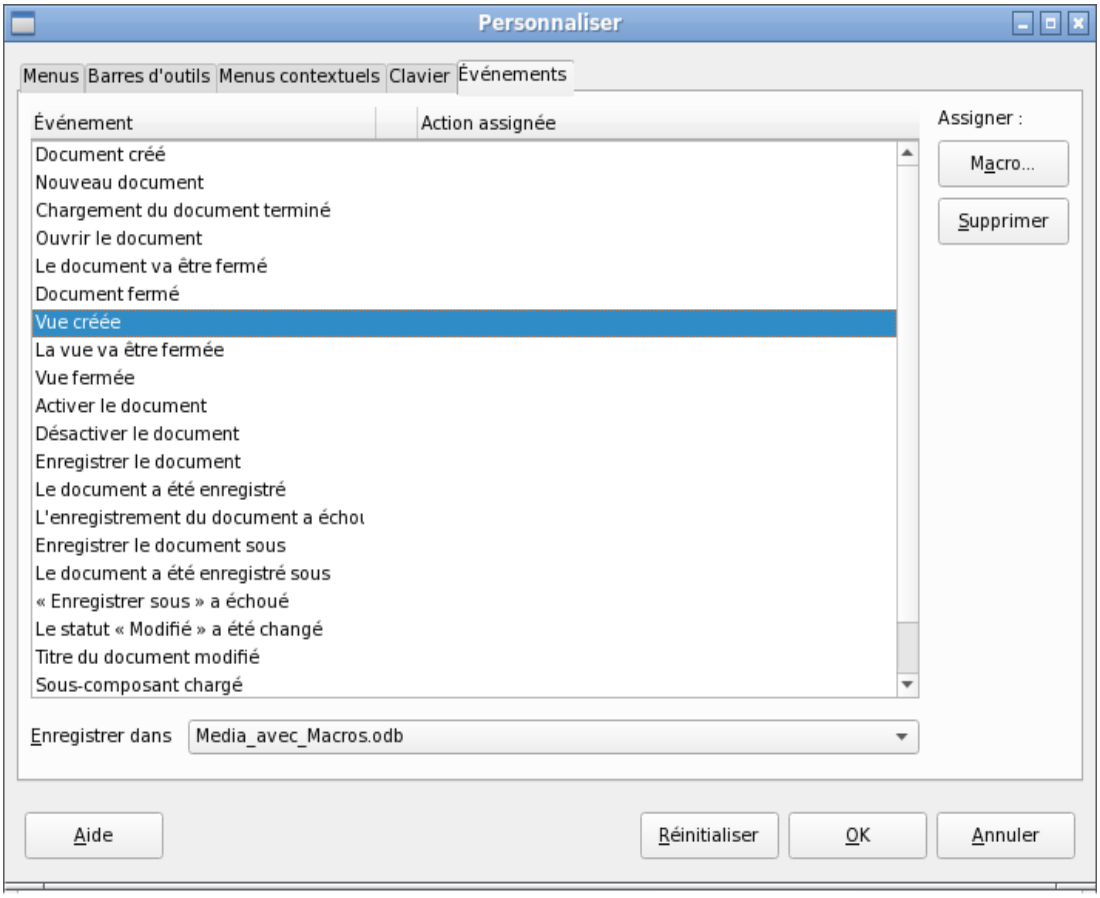
La "voie directe" n'est pas utilisable – pas même pour les tests – lorsque l'un des objets **thisComponent** (voir « Accéder aux formulaires » à la page 411) ou **oEvent** (voir « Accéder aux éléments de formulaire » à la page 412) doit être utilisé.

Assigner des macros

Si une macro doit être lancée par un événement, elle doit d'abord être définie. Ensuite, elle peut être affectée à un événement. Ces événements sont accessibles via deux emplacements.

Événements produits dans un formulaire lorsque la fenêtre est ouverte ou fermée

Les actions qui ont lieu lors de l'ouverture ou de la fermeture d'un formulaire sont enregistrées comme suit :



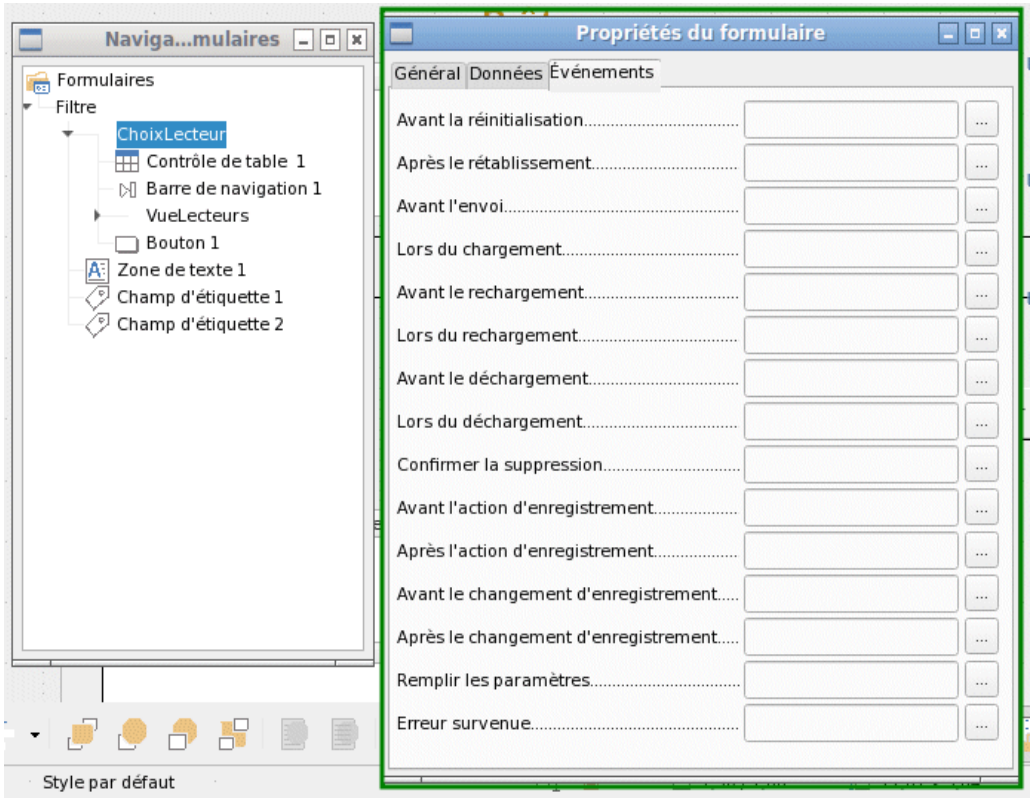
1. Lors de la conception du formulaire, ouvrez l'onglet Événements dans **Outils> Personnaliser**.
2. Choisissez l'événement approprié. Certaines macros ne peuvent être lancées que lorsque l'événement Vue créée est choisi. D'autres macros, par exemple pour créer un formulaire plein écran, doivent être lancées par Ouvrir le document.
3. Utilisez le bouton **Macro** pour trouver la macro souhaitée et confirmez votre choix.
4. En face de Enregistrer dans, choisissez le nom du formulaire.
5. Confirmez avec **OK**.

Événements dans un formulaire dans une fenêtre ouverte

Une fois la fenêtre ouverte pour afficher le contenu global du formulaire, des éléments individuels du formulaire sont accessibles. Cela inclut les éléments que vous avez attribués au formulaire.

Les éléments de formulaire sont accessibles à l'aide du navigateur de formulaires, comme indiqué dans l'illustration ci-dessous. Ils sont également accessibles en utilisant les menus contextuels des contrôles individuels dans l'interface du formulaire (clic droit sur le contrôle / Propriétés du formulaire).

Les événements répertoriés sous **Propriétés du formulaire > Événements** ont tous lieu lorsque la fenêtre du formulaire est ouverte. Ils peuvent être définis séparément pour chaque formulaire ou sous-formulaire dans la fenêtre de formulaire.



Note

Malheureusement, Base utilise le mot « formulaire » à la fois pour une fenêtre ouverte pour la saisie de données et pour les éléments de cette fenêtre liés à une source de données spécifique (table ou requête).

Une seule fenêtre de formulaire peut contenir plusieurs formulaires avec différentes sources de données. Dans le navigateur de formulaires, vous voyez toujours d'abord le terme Formulaires, qui, dans le cas d'un formulaire simple, ne contient qu'une seule entrée subordonnée.

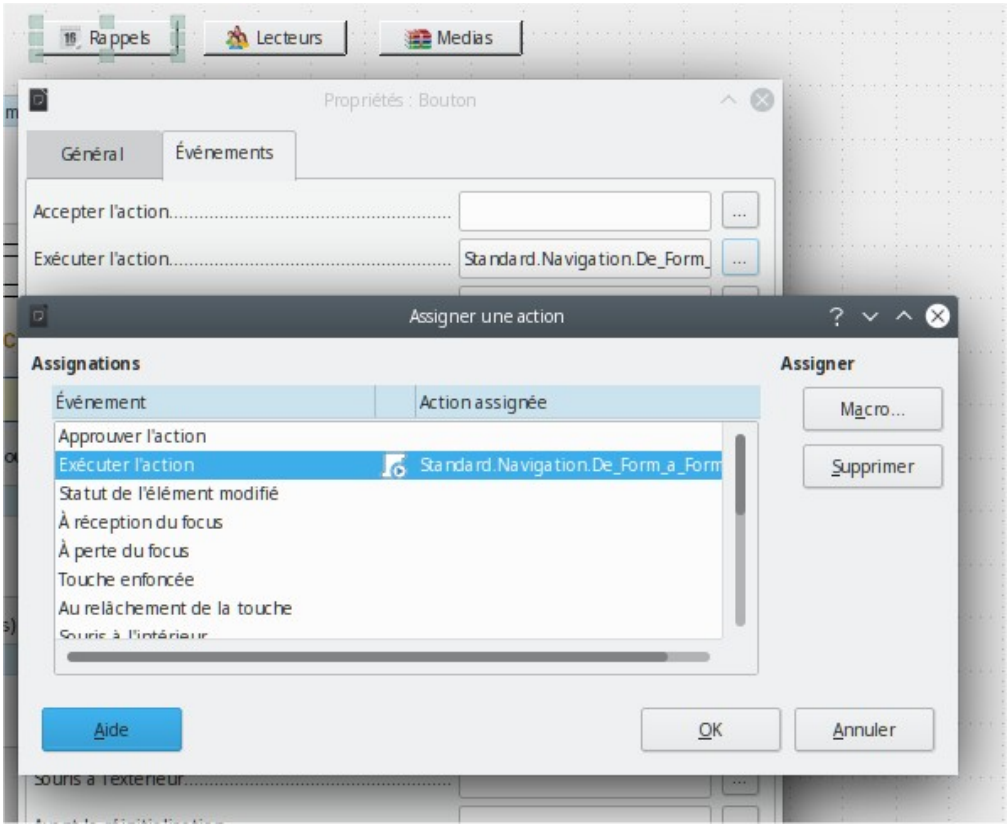
Événements dans un formulaire

Toutes les autres macros sont connectées à l'aide des propriétés des sous-formulaires et des contrôles via l'onglet Événements.

1. Ouvrez la fenêtre des propriétés du champ/contrôle (si vous ne l'avez pas déjà fait).
2. Choisissez un événement approprié dans l'onglet Événements.
3. Pour modifier la source de données, utilisez des événements faisant référence à *Enregistrer* ou *Actualiser* ou *Réinitialiser*.

   Pour les boutons ou les choix dans les champs de liste ou d'option, l'action *Exécuter l'action* sera le premier point d'entrée.

   Tous les autres événements dépendent du type de contrôle et de l'action souhaitée.
4. Cliquez sur le bouton ... à droite pour ouvrir la boîte de dialogue Assigner une action.
5. Cliquez sur le bouton Macro... pour choisir la macro définie pour l'action.
6. Cliquez sur **OK** pour confirmer l'affectation.



Composants des macros

Cette section explique certains des langages de macros couramment utilisés dans Base, en particulier dans les formulaires. Dans la mesure du possible (et du raisonnable), des exemples sont donnés dans toutes les sections suivantes.

Le « cadre » d'une macro

La définition d'une macro commence par son type – **Sub** ou **Function** – et se termine par **End Sub** ou **End Function**. Une macro affectée à un événement peut recevoir des arguments (valeurs), le seul utile est l'argument **oEvent**. Toutes les autres routines qui pourraient être appelées par une telle macro peuvent être définies avec ou sans valeur de retour, selon leur objectif, et fournies avec des arguments si nécessaire.

```
Sub Miseajour_pret
End Sub

Sub de_Formulaire_a_Formulaire(oEvent As Object)
End Sub

Function confirme_efface(oEvent As Object) As Boolean
       confirme_efface = False
End Function
```

Il est utile d'écrire ce cadre immédiatement et de rédiger le contenu par la suite. N'oubliez pas d'ajouter des commentaires pour expliquer la macro, en vous rappelant la règle "Autant que nécessaire, aussi peu que possible" en gardant à l'esprit qu'un code bien documenté est plus facile à lire. De plus, Basic ne fait pas la distinction entre les majuscules et les minuscules. Les termes habituellement fixes comme **SUB** sont écrits de préférence avec une majuscule, les autres concepts en casse mixte.

Définition des variables

Dans l'étape suivante, au début de la routine, la commande **Dim** est utilisée pour définir les variables qui seront employées dans la routine, chacune avec son type de données approprié. Basic lui-même ne l'exige pas ; il accepte toutes les nouvelles variables utilisées dans le programme sans avoir été définies (déclarées) au préalable. Cependant, le code du programme est « plus sûr » si les variables, en particulier leurs types de données, sont déclarées. De nombreux programmeurs en font une exigence, en utilisant l'instruction **Option Explicit** de Basic lorsqu'ils commencent à écrire un module. Cela signifie : Ne reconnaissez aucune variable, hormis celles que j'ai déclarées. Cela est plus contraignant, mais peut éviter des "plantages catastrophiques" et permet une recherche d'erreurs plus aisée.

```
Dim oDoc As Object
Dim oDrawpage As Object
Dim oFormulaire As Object
Dim sNom As String
Dim bOKActive As Boolean
Dim iCompteur As Integer
Dim dDateNais As Date
```

Seuls les caractères alphabétiques (A-Z ou a-z), les nombres et le caractère de soulignement «_» peuvent être utilisés dans les noms de variables. Aucun caractère spécial n'est autorisé. Les espaces sont autorisés sous certaines conditions, mais il faut les éviter, pour ne pas créer de confusions. Le premier caractère doit être alphabétique.

Il est courant de spécifier le type de données dans le premier caractère[8]. Ensuite, il peut être reconnu partout où la variable apparaît dans le code. Les « noms expressifs » sont également recommandés, de sorte que la signification de la variable soit évidente à partir de son nom.

Une liste des types de données possibles dans LibreOffice Basic se trouve dans l'annexe A de ce livre. Ils diffèrent à divers endroits des types de la base de données et de l'API LibreOffice. Ces changements sont clairement indiqués dans les exemples.

Définition de tableaux (array)

Pour les bases de données en particulier, l'assemblage de plusieurs variables dans un enregistrement est important. Si plusieurs variables sont stockées ensemble dans un seul emplacement commun, cela s'appelle un tableau. Un tableau doit être défini avant de pouvoir y écrire des données.

```
Dim arDonnees()
```

crée un tableau vide.

```
arDonnees = Array("Michèle","Morgan")
```

crée un tableau d'une taille spécifique (2 éléments) et lui fournit des valeurs.

En utilisant :

```
Print arDonnees(0), arDonnees(1)
```

on provoque l'affichage à l'écran des deux éléments définis. La numérotetion des éléments commence à 0.

```
Dim arDonnees(2)
arDonnees(0) = "Michèle"
arDonnees(1) = "Morgan"
arDonnees(2) = "Paris"
```

Cela crée un tableau dans lequel trois éléments de n'importe quel type peuvent être stockés, par exemple un enregistrement pour : Michèle, Morgan, Paris. Vous ne pouvez pas mettre plus de trois éléments dans ce tableau. Si vous souhaitez stocker plus d'éléments, vous devez agrandir le tableau. Cependant, si la taille d'un tableau est redéfinie pendant l'exécution d'une macro, le tableau est réinitiallisé à vide, tout comme un nouveau tableau.

```
ReDim Preserve arDonnees(3)
arDonnees(3) = "29/02/1920"
```

L'ajout de **Preserve** conserve les données précédentes pour que le tableau soit véritablement étendu par l'entrée de la date (ici sous forme de texte).

Le tableau ci-dessus ne peut stocker qu'un seul enregistrement. Si vous souhaitez stocker plusieurs enregistrements, comme le fait une table dans une base de données, vous devez définir un tableau à deux dimensions.

```
Dim arDonnees(2,1)
arDonnees(0,0) = "Michèle"
arDonnees(1,0) = "Morgan"
arDonnees(2,0) = "Paris"
arDonnees(0,1) = "Jean"
```

8 Vous devriez rendre cela plus précis si nécessaire, car une seule lettre ne vous permet pas toujours de faire la distinction entre les types de données "Double" et "Date" ou "Simple" et "String".

```
arDonnees(1,1) = "Gabin"
arDonnees(2,1) = "Neuilly"
```

Ici aussi, il est possible d'étendre le tableau précédemment défini et de conserver le contenu existant en utilisant **Preserve**.

Accéder aux formulaires

Le formulaire se trouve dans le document actuellement actif. La région représentée ici est appelée **drawpage**. Le conteneur dans lequel tous les formulaires sont conservés est appelé **forms** ; dans le navigateur de formulaires, cela apparaît comme l'en-tête principal (*Formulaires*) avec tous les formulaires individuels attachés. Les variables nommées ci-dessus reçoivent leurs valeurs comme ceci :

```
oDoc = thisComponent
oDrawpage = oDoc.drawpage
oFormulaire = oDrawpage.forms.getByName("Filtre")
```

Le formulaire auquel accéder est appelé *Filtre*. C'est le nom qui est visible dans le niveau supérieur du navigateur de formulaires (par défaut, le premier formulaire est appelé *Formulaire*). Les sous-formulaires sont classés par ordre hiérarchique dans le formulaire principal et peuvent être atteints étape par étape :

```
Dim oSousFormulaire As Object
Dim oSousSousFormulaire As Object
oSousFormulaire = oFormulaire.getByName("SelectionLecteur")
oSousSousFormulaire = oSousFormulaire.getByName("AfficheLecteur")
```

Au lieu d'utiliser des variables intermédiaires, vous pouvez accéder directement à un formulaire particulier. Un objet intermédiaire, qui peut être utilisé plusieurs fois, doit être déclaré et se voir attribué une valeur distincte. Dans l'exemple suivant, oSousFormulaire n'est plus utilisé.

```
oFormulaire = thisComponent.drawpage.forms.getByName("Filtre")
oSousSousFormulaire = _
oFormulaire.getByName("SelectionLecteur").getByName("AfficheLecteur")
```

Note

Si un nom se compose uniquement de lettres ASCII et de chiffres sans espaces ni caractères spéciaux, le nom peut être utilisé directement dans une instruction d'affectation.

```
oFormulaire = thisComponent.drawpage.forms.Filtre
oSubSubForm = oFormulaire.SelectionLecteur.AficheLecteur
```

Contrairement à l'usage habituel en Basic, ces noms doivent être écrits avec la casse correcte.

Un autre mode d'accès au formulaire est fourni par l'événement qui déclenche la macro.

Si une macro est lancée à partir d'un événement de formulaire tel que **Propriétés du formulaire> Avant l'action d'enregistrement**, le formulaire lui-même peut être atteint comme suit :

```
Sub Calculer(oEvent As Object)
	oFormulaire = oEvent.Source
	…
End Sub
```

Si la macro est lancée à partir d'un événement dans un contrôle de formulaire, tel que **Zone de texte> A la perte de focus**, le formulaire et le champ deviennent accessibles :

```
Sub Calculer(oEvent As Object)
	oChamp = oEvent.Source.Model ' récupère la source de l'événement
	oFormulaire = oChamp.Parent  ' récupère le formulaire contenant la source
	…
End Sub
```

L'accès aux événements présente l'avantage que vous n'avez pas à vous soucier de savoir si vous avez affaire à un formulaire principal ou à un sous-formulaire. De plus, le nom du formulaire n'a aucune importance pour le fonctionnement de la macro.

Accéder aux éléments de formulaire

Les éléments dans les formulaires sont accessibles de la même manière : déclarez une variable appropriée en tant qu'**object** et recherchez le contrôle approprié dans le formulaire :

```
Dim btnOK As Object ' Bouton OK
btnOK = oSousSousFormulaire.getByName("Bouton 1") ' depuis le formulaire VueLecteur
```

Cette méthode fonctionne toujours lorsque vous savez avec quel élément la macro est censée fonctionner. Cependant, lorsque la première étape consiste à déterminer quel événement a lancé la macro, la variable **oEvent** présentée ci-dessus devient utile. La variable est déclarée dans le « cadre » de la macro et se voit attribuer une valeur lorsque la macro est lancée. La propriété **Source** renvoie toujours l'élément qui a lancé la macro, tandis que la propriété **Model** décrit le contrôle en détail :

```
Sub Confirme_Choix(oEvent As Object)
	Dim btnOK As Object
	btnOK = oEvent.Source.Model
End Sub
```

Si vous le souhaitez, d'autres actions peuvent être effectuées avec l'objet obtenu par cette méthode.

Veuillez noter que les sous-formulaires comptent comme composants d'un formulaire.

Accès à la base de données

Normalement, l'accès à la base de données est contrôlé par des formulaires, des requêtes, des rapports ou la fonction publipostage, comme décrit dans les chapitres précédents. Si ces possibilités s'avèrent insuffisantes, une macro peut spécifiquement accéder à la base de données de plusieurs manières.

Connexion à la base de données

La méthode la plus simple utilise la même connexion que le formulaire. oFormulaire est déterminé comme indiqué ci-dessus.

```
Dim oConnexion As Object
oConnexion = oFormulaire.activeConnection()
```

Ou vous pouvez récupérer la source de données (c'est-à-dire la base de données) via le document et utiliser sa connexion existante pour la macro :

```
Dim oSourceDonnees As Object
Dim oConnexion As Object
oSourceDonnees = thisComponent.Parent.dataSource
oConnexion = oSourceDonnees.getConnection("","") ‘sans mot de passe
```

Un autre moyen permet de créer la connexion à la base de données à la volée :

```
Dim oSourceDonnees As Object
Dim oConnexion As Object
oSourceDonnees = thisComponent.Parent.CurrentController
If Not (oSourceDonnees.isConnected()) Then oSourceDonnees.connect()
oConnexion = oSourceDonnees.ActiveConnection()
```

La condition **If** ne contrôle qu’une seule ligne, donc **End If** n’est pas obligatoire.Mais cela peut conduire à des erreurs lorsqu'il s'avère nécessaire d'ajouter un **Else** ou d'autres instructions dans le **Then** par la suite .

Si la macro doit être lancée via l’interface utilisateur et non à partir d’un événement dans un formulaire, la variante suivante convient :

```
Dim oSourceDonnees As Object
Dim oConnexion As Object
oSourceDonnees = thisDatabaseDocument.CurrentController
If Not (oSourceDonnees.isConnected()) Then oSourceDonnees.connect()
          oConnexion = oSourceDonnees.ActiveConnection()
```

L’accès aux bases de données en dehors de la base de données actuelle est possible comme suit :

```
Dim oContexteBase As Object
Dim oSourceDonnees As Object
Dim oConnexion As Object
oContexteBase = createUnoService("com.sun.star.sdb. DatabaseContext")
oSourceDonnees = oContexteBase.getByName("nom de la base enregistrée dans LibO")
oConnexion = oSourceDonnees.GetConnection("","")
```

Des connexions à des bases de données non enregistrées avec LibreOffice sont également possibles. Dans de tels cas, au lieu du nom enregistré, le chemin d’accès à la base de données doit être indiqué sous forme de **file: ///……/NomdelaBase.odb**.

Des instructions détaillées sur les connexions à la base de données sont données dans « Établir une connexion à une base de données » (page 483).

Commandes SQL

Vous travaillez avec la base de données à l’aide de commandes SQL. Celles-ci doivent être créées et envoyés à la base de données. Le résultat est déterminé en fonction du type de commande et les résultats peuvent être traités ultérieurement. La directive **createStatement** crée un objet approprié à cet effet.

```
Dim oInstruction_SQL As Object ' l’objet qui contiendra la commande SQL-
Dim stSql As String ' Texte de la commande SQL actuelle
Dim oResultat As Object ' résultat de la fonction executeQuery
Dim iResultat As Integer ' résultat de la fonction executeUpdate
oInstruction_SQL = oConnexion.createStatement()
```

Pour *interroger des données*, vous appelez la méthode **executeQuery** ; le résultat est ensuite évalué. Les noms de table et de champ sont généralement entre guillemets. La macro doit les

masquer avec des guillemets doubles supplémentaires pour s'assurer qu'elles apparaissent dans la commande.

```
stSql = "SELECT * FROM ""Table1"""
oResultat = oInstruction_SQL.executeQuery(stSql)
```

Pour *modifier les données* – c'est-à-dire **INSERT**, **UPDATE** ou **DELETE** – ou pour agir sur la *structure de la base de données*, vous appelez la méthode **executeUpdate**. Selon la commande et la base de données, cela ne donne rien d'utile (un zéro) ou bien le nombre d'enregistrements modifiés.

```
stSql = "DROP TABLE ""temporaire"" IF EXISTS"
iResultat = oInstruction_SQL.executeUpdate(stSql)
```

Par souci d'exhaustivité, il y a un autre cas particulier à mentionner : si **oInstruction_SQL** doit être utilisé de différentes manières pour **SELECT** ou à d'autres fins, il existe une autre méthode disponible, à savoir **execute**. Nous ne l'utiliserons pas ici. Pour plus d'informations, consultez la référence API (Xstatement, XPreparedStatement).

Commandes SQL pré-préparées avec paramètres

Dans tous les cas où la saisie manuelle par un utilisateur doit être transférée dans une instruction SQL, il est plus facile et plus sûr de ne pas créer la commande sous la forme d'une longue chaîne de caractères mais de la concevoir à l'avance et de l'utiliser avec des paramètres. Cela facilite le formatage des nombres, des dates et des chaînes (les guillemets doubles constants disparaissent) et empêche les entrées malveillantes (injections SQL) de provoquer une perte de données.

Pour utiliser cette méthode, l'objet d'une commande SQL particulière est créé et préparé :

```
Dim oInstruction_SQL As Object ' l'objet, qui exécute la commande SQL
Dim stSql As String ' Texte de la commande SQL.
stSql = "UPDATE Auteurs " _
& "SET Nom = ?, Prenom = ?" _
& "WHERE ID = ?"
oInstruction_SQL = oConnexion.prepareStatement(stSql)
```

L'objet est créé avec **prepareStatement** afin que la commande SQL soit connue à l'avance. Chaque point d'interrogation indique une position qui plus tard – avant l'exécution de la commande – recevra une valeur réelle. Comme la commande est conçue à l'avance, la base de données sait quel type d'entrée – dans ce cas deux chaînes et un nombre – est attendu. Les différentes positions se distinguent par leur numéro (à partir de 1).

Ensuite, les valeurs sont transférées avec des instructions appropriées et la commande SQL est exécutée. Ici, les valeurs sont tirées des contrôles de formulaire, mais elles peuvent également provenir d'autres macros ou être données sous forme de texte brut :

```
oInstruction_SQL.setString(1, oZoneTexte1. Text) ' Texte pour le nom
oInstruction_SQL.setString(2, oZoneTexte2. Text) ' Texte pour le prénom
oInstruction_SQL.setLong(3, oChampNum1. Value) ' valeur pour l'ID approprié
```

La liste complète des affectations est en "Paramètres des commandes SQL préparées" (page 433).

Pour plus d'informations sur les avantages de cette méthode, voir ci-dessous (liens externes) :

- Injection SQL (Wikipedia)
- Why use PreparedStatement (Java JDBC)
- Commandes SQL (Wikilivres)
- Utilité des requêtes préparées (Openclassrooms)

Lecture et utilisation des enregistrements

Il existe plusieurs façons, selon les besoins, de transférer des informations d'une base de données vers une macro afin qu'elles puissent être traitées ultérieurement.

Remarque : si l'on parle de « formulaire », il peut également s'agir d'un sous-formulaire. Dans ce cas, il s'agit toujours du (sous-)formulaire qui est associé à un certain nombre de données.

Utiliser des formulaires

L'enregistrement courant et ses données sont toujours disponibles via le formulaire qui affiche les données pertinentes (table, requête, SELECT). Il existe plusieurs méthodes **getdata_type**, telles que :

```
Dim ID As Long
Dim sNom As String
Dim dValeur As Currency
Dim dEntree As New com.sun.star.util.Date ' objet UNO de type Date
ID = oFormulaire.getLong(1)
sNom = oFormulaire.getString(2)
dValeur = oFormulaire.getDouble(4)
dEntree = oFormulaire.getDate(7)
```

Note

Toutes les méthodes qui fonctionnent avec des bases de données comptent à partir de 1. Cela s'applique aussi bien aux colonnes qu'aux lignes.

Si vous préférez utiliser des noms de colonne au lieu de numéros de colonne pour travailler avec la source de données sous-jacente (table, requête, vue), le nom de colonne peut être déterminé à l'aide de **findColumn**. Voici un exemple de recherche de la colonne appelée *Nom*.

```
Dim sNom As String
numNom = oFormulaire.findColumn("Nom") ‘ numéro du champ
stNom = oFormulaire.getString(numNom) ‘ contenu du champ
```

Le type de valeur renvoyé correspond toujours au type de méthode, mais les cas spéciaux suivants doivent être notés :

- Il n'y a pas de méthodes pour les données des types **Decimal**, **Currency (devise)**, etc. qui sont utilisées pour des calculs commerciaux exacts. Comme Basic effectue automatiquement la conversion appropriée, vous pouvez utiliser **getDouble**
- Lorsque vous utilisez **getBoolean**, vous devez tenir compte de la façon dont TRUE et FALSE sont définis dans la base de données. Les définitions usuelles (valeurs logiques, 1 pour TRUE et 0 pour FALSE) sont traitées correctement.
- Les valeurs de date peuvent être définies non seulement avec le type de données **Date**, mais également comme **util.Date** (comme ci-desus, objet UNO). Cela facilite la lecture et la modification de l'année, du mois et du jour.

- Avec des nombres entiers, méfiez-vous des différents types de données. L'exemple ci-dessus utilise **getLong** ; la variable Basic ID doit également avoir le type de données **Long**, car il correspond au type **Integer** dans la base de données.

La liste complète de ces méthodes se trouve dans « Modification des lignes (enregistrements) de données » (page 430).

Conseil

Si les valeurs d'un formulaire doivent être utilisées directement pour un traitement ultérieur en SQL (par exemple pour une entrée dans une autre table), il est beaucoup plus simple de ne pas avoir à interroger le type de champ.

La macro suivante, qui est liée à **Propriétés : Bouton> Evénements> Exécuter l'action** lit le premier champ du formulaire indépendamment du type nécessaire pour le traitement futur dans Basic.

```
SUB LireValeurs(oEvent As Object)
        Dim oFormulaire As Object
        Dim stChamp1 As String
        oFormulaire = oEvent.Source.Model.Parent
        stChamp1 = oFormulaire.getString(1)
End Sub
```

Si les champs sont lus à l'aide de **getString()**, tout le formatage nécessaire pour un traitement SQL ultérieur est conservé. Une date qui s'affiche sous la forme 08.03.19 est lue au format 2019-03-08 et peut être utilisée directement dans SQL.

La lecture dans un format correspondant au type n'est obligatoire que si la valeur doit être traitée ultérieurement dans la macro, par exemple dans un calcul.

Résultat d'une requête

L'ensemble des résultats d'une requête peut être utilisé de la même manière. Dans la section des commandes SQL et ci-dessus, vous trouverez la variable **oResultat** pour cet ensemble de résultats, qui est généralement lue de cette manière :

```
While oResultat.next ' un enregistrement après l'autre
       REM transfère le résultat dans des variables
       stVar = oResultat.getString(1)
       inVar = oResultat.getLong(2)
       boVar = oResultat.getBoolean(3)
REM faire quelque chose avec ces valeurs
Wend
```

Selon le type de commande SQL, le résultat attendu et son objectif, la boucle **WHILE** peut être raccourcie ou supprimée complètement. Mais fondamentalement, un jeu de résultats peut toujours être évalué de cette manière.

Si seul le premier enregistrement doit être évalué

```
oResultat.next
```

accède à la ligne de cet enregistrement et avec

```
stVar = oResultat.getString(1)
```

lit le contenu du premier champ. La boucle se termine ici.

La requête de l'exemple ci-dessus contient du texte dans la première colonne, un nombre entier dans la seconde (un **Integer** dans la base de données correspond à **Long** en Basic) et un champ

Oui/Non dans la troisième. Les champs sont accessibles via un index de champ qui, contrairement à un index de tableau, commence à 1 (cf. Note page précédente).

La navigation dans un tel résultat n'est pas possible. Seules les étapes simples vers l'enregistrement suivant sont autorisées. Pour naviguer dans l'enregistrement, le **ResultSetType** doit être connu lorsque la requête est créée. Ceci est accessible en utilisant

```
oSQL_Resultat.ResultSetType = 1004
```

ou

```
oSQL_Resultat.ResultSetType = 1005
```

Le type **1004 – SCROLL_INTENSIVE** vous permet de naviguer librement mais ne prend pas en compte les changements dans les données d'origine. Le type **1005 – SCROLL_SENSITIVE** reconnaît les modifications des données d'origine susceptibles d'affecter le résultat de la requête.

Le nombre total de lignes dans l'ensemble de résultats ne peut être déterminé qu'après la spécification d'un type numérique pour le résultat. Il se déroule comme suit :

```
Dim iResultat As Long
If oResultat.last ' va au dernier enregistrement si possible
          iResultat = oResultat.getRow  ' iResultat reçoit le nombre de lignes
                                                ‘ qui est le numéro de la
dernière.
Else
          iResultat = 0
End If
```

Utilisation d'un contrôle

Si un contrôle est lié à une source de données, la valeur peut être lue directement, comme décrit dans la section suivante. Cependant, cela peut entraîner des problèmes. Il est plus sûr d'utiliser la procédure décrite dans "Utiliser des formulaires" (page 415) ou bien la méthode suivante, qui est présentée pour plusieurs types de contrôle différents :

```
sValue = oZoneTexte1.BoundField.Text ' exemple pour une Zone de Texte
nValue = oChampNumerique.BoundField.Value ' exemple pour un champ numérique
dValue = oChampDate.BoundField.Date ' exemple pour un champ Date
```

BoundField représente le lien entre le contrôle visible et le contenu réel de l'ensemble de données.

Naviguer dans un ensemble de données

Dans l'exemple page précédente (While… Wend), la méthode **Next** a été utilisée pour passer d'une ligne du jeu de résultats à la suivante. Il existe d'autres méthodes et tests similaires qui peuvent être utilisés à la fois pour les données dans un formulaire – représenté par la variable **oFormulaire** – et pour un ensemble de résultats. Par exemple, en utilisant la méthode décrite dans « Mise à jour automatique des formulaires » (page 434), l'enregistrement précédent peut être sélectionné à nouveau :

```
Dim laLigne As Long
laLigne = oFormulaire.getRow() ' enregistre le numéro de ligne courant
oFormulaire.reload() ' recharge l’ensemble d’enregistrements
oFormulaire.absolute(laLigne) ' revient à la même ligne
```

La section « Mise à jour automatique des formulaires » montre toutes les méthodes qui conviennent pour cela.

Note

Malheureusement, depuis le début de LibreOffice, il y a eu un bogue (reporté d'OpenOffice) qui affecte les formulaires. Il définit le numéro de ligne actuel lorsque les données sont modifiées dans un formulaire sur « 0 ». Voir https://bugs.documentfoundation.org/show_bug.cgi?id=82591. Pour obtenir le numéro de ligne actuel correct, liez la macro suivante à l'événement **Formulaire> Propriétés> Événements> Après le changement d'enregistrement**.

```
Global laLigne As Long
Sub CompteurLignes(oEvent As Object)
      laLigne = oEvent.Source.Row
End Sub
```

Le nouveau numéro de ligne est lu et affecté à la variable globale laLigne. Cette variable doit être placée au début de tous les modules et conservera son contenu jusqu'à ce que vous quittiez Base ou que vous changiez la valeur en appelant à nouveau *CompteurLignes*.

Il semblerait que ce bogue est corrigé depuis la version 5.0

Édition d'enregistrements (ajout, modification, suppression)

Pour éditer des enregistrements, plusieurs choses doivent fonctionner ensemble :

- Les informations doivent être saisies par l'utilisateur dans un champ, à l'aide du clavier ;
- L'ensemble de données sous-jacent au formulaire doit être informé du changement. Cela se produit lorsque vous quittez le champ pour un autre ;
- La base de données elle-même doit être modifiée. Cela se produit lorsque vous passez d'un enregistrement à un autre.

Lorsque vous effectuez cette opération via une macro, ces étapes partielles doivent toutes être prises en compte. Si l'une d'entre elles fait défaut ou est mal exécutée, les modifications seront perdues et ne finiront pas dans la base de données. Tout d'abord, la modification ne doit pas se trouver uniquement dans la valeur affichée du contrôle mais dans l'ensemble de données lui-même. Cela rend inutile la modification de la propriété **Text** d'un contrôle.

Veuillez noter que les tableaux sont les seuls ensembles de données qui peuvent être modifiés sans causer de problèmes. Pour les autres ensembles de données, l'édition n'est possible que dans des circonstances spéciales.

Modifier le contenu d'un contrôle

Si vous souhaitez modifier une seule valeur, la propriété **BoundField** du contrôle peut être utilisée avec une méthode appropriée. Ensuite, la modification doit être transmise à la base de données. Voici un exemple de champ de date dans lequel la date réelle doit être saisie :

```
Dim unoDate As New com.sun.star.util.Date
unoDate.Year = Year(Date)
unoDate.Month = Month(Date)
unoDate.Day = Day(Date)
oChampDate.BoundField.updateDate(unoDate)
oFormulaire.updateRow() ' la modification est transmise à la base
```

Pour **BoundField**, vous utilisez la méthode **updateXxx** qui correspond au type de données du champ. Dans cet exemple, le champ est un champ **Date**. La nouvelle valeur est transmise comme argument – dans ce cas la date actuelle, convertie au format requis par la macro.

Modifier les lignes (enregistrements) d'un ensemble de données

La méthode précédente ne convient pas lorsque plusieurs valeurs dans une ligne doivent être modifiées, d'une part, un contrôle devrait exister sur le formulaire pour chaque champ, ce qui n'est souvent pas souhaité et inutile. En outre, un objet doit être récupéré pour chaque champ. La manière simple et directe utilise le formulaire comme ceci :

```
Dim unoDate As New com.sun.star.util. Date
unoDate.Annee = Year(Date)
unoDate.Mois = Month(Date)
unoDate.Jour = Day(Date)
oFormulaire.updateDate(3, unoDate)
oFormulaire.updateString(4, "un Texte")
oFormulaire.updateDouble(6, 3.14)
oFormulaire.updateInt(7, 16)
oFormulaire.updateRow()
```

Pour chaque colonne de l'ensemble de données, la méthode **updateXxx** appropriée à son type est appelée. Les arguments sont le numéro de colonne (à partir de 1) et la valeur souhaitée. Ensuite, les modifications sont transmises à la base de données.

Créer, modifier et supprimer des lignes

Les modifications nommées font référence à la ligne actuelle de l'ensemble de données sous-jacent au formulaire. Dans certaines circonstances, il est nécessaire d'appeler une méthode depuis « Naviguer dans un ensemble de données » (page 429). Les étapes suivantes sont nécessaires :

1. Choisissez l'enregistrement actuel.
2. Modifiez les valeurs comme décrit dans la section précédente.
3. Confirmez la modification avec la commande suivante :
   `oFormulaire.updateRow()`
4. Dans certains cas particuliers, il est possible d'annuler et de revenir à l'état précédent :
   `oFormulaire.cancelRowUpdates()`

Pour un nouvel enregistrement, il existe une méthode spéciale, comparable au passage à une nouvelle ligne dans un contrôle de table. Cela se fait comme suit :

1. Préparez-vous pour un nouvel enregistrement :
   `oFormulaire.moveToInsertRow()`
2. Saisissez toutes les valeurs souhaitées/requises. Cela se fait en utilisant les méthodes **updateXxx** comme indiqué dans la section précédente.
3. Confirmez les nouvelles données avec la commande suivante :
   `oFormulaire.insertRow()`
4. La nouvelle entrée ne peut pas être facilement annulée. Au lieu de cela, vous devrez supprimer le nouvel enregistrement.

Il existe une commande simple pour supprimer un enregistrement, procédez comme suit :

1. Choisissez l'enregistrement souhaité et rendez le courant, comme pour une modification.
2. Utilisez la commande suivante pour le supprimer :
   `oFormulaire.deleteRow()`

Conseil

Pour garantir que les modifications sont transférées dans la base de données, elles doivent être confirmées explicitement avec **updateRow** ou **insertRow**, selon le cas. Appuyer sur le bouton Enregistrer utilisera automatiquement la fonction appropriée, avec une macro, vous devez déterminer avant d'enregistrer si l'enregistrement est nouveau (**Insert**) ou une modification d'un existant (**Update**).

```
If oFormulaire.isNew Then
        oFormulaire.insertRow()
Else
        oFormulaire.updateRow()
End If
```

Tester et modifier les contrôles

Outre le contenu de l'ensemble de données, beaucoup plus d'informations peuvent être lues à partir d'un contrôle, éditées et modifiées. Cela est particulièrement vrai pour les propriétés, comme décrit dans le chapitre 4, Formulaires.

Divers exemples dans « Améliorer la convivialité » (page 433) utilisent la propriété *Complément d'information* du champ :

```
Dim stTag As String
stTag = oEvent.Source.Model.Tag
```

Comme mentionné dans la section précédente, la propriété **Text** ne peut être modifiée utilement que si le contrôle n'est pas lié à un ensemble de données. Cependant, il existe d'autres propriétés qui sont déterminées dans le cadre de la définition du formulaire mais qui peuvent être adaptées au moment de l'exécution. Par exemple, une étiquette peut avoir une couleur de texte différente si elle représente un avertissement plutôt qu'une information :

```
Sub MontreAvertissement(oChamp As Object, iType As Integer)
      Select Case iType
          Case 1
              oChamp.TextColor = RGB(0,0,255) ' 1 = bleu
          Case 2
              oChamp.TextColor = RGB(255,0,0) ' 2 = rouge
          Case Else
              oChamp.TextColor = RGB(0,255,0) ' 0 = vert (ni 1 ni 2)
      End Select
End Sub
```

Noms anglais dans les macros

Alors que le concepteur d'un formulaire peut utiliser des désignations en langue native pour les contrôles et l'accès aux données, seuls les noms anglais peuvent être utilisés dans Basic. Ceux-ci sont présentés dans le synopsis suivant.

Les propriétés qui ne sont normalement déterminées que dans la définition de formulaire ne sont pas incluses ici. il en va de même pour les méthodes (fonctions et/ou procédures) qui ne sont que rarement utilisées ou requises uniquement pour des déclarations plus complexes.

Le synopsis comprend les éléments suivants :

- **Nom** Nom à utiliser pour la propriété dans le code de macro
- **Type de donnée** Un type de données Basic. Pour les fonctions, le type de retour. Non inclus pour les procédures.

- **L/E** Indique comment vous pouvez utiliser la valeur:

L	Lecture seule (read only)
E	Écriture seule (modification) (write)
(L)	Lecture possible, pas adapté pour l'édition
(E)	Écriture possible mais pas utile
L+E	Adapté à la lecture et à l'écriture

Vous trouverez de plus amples informations dans la référence API en recherchant le nom anglais du contrôle. Il existe un outil plus qu'utile, indispensable, appelé Xray pour découvrir quelles propriétés et méthodes sont disponibles pour un élément.

L'utilitaire Xray

Xray a été initialement conçue pour Apache Open Office, mais continue de fonctionner sur Libre Office après, quelques aménagements. Cet outil peut-être téléchargé sur la page personnelle de Bernard Marcelly, son concepteur (Lien vers Xray). Pour l'installation et l'utilisation de Xray se reporter au document « XrayTool60_fr.odt **»** téléchargé.

Pour la configuration de Xray, se reporter au chapitre 3.5, page 22, de ce document.
On y constate qu'il est souhaitable d'installer le SDK (Software development kit – Kit de développement logiciel) de LibreOffice en local (sur l'ordinateur utilisé).

Cette installation n'est pas obligatoire mais à l'avantage de donner accès, en local, à la documentation de l'API de LibreOffice.

Depuis la version 4.2 de LibreOffice, le SDK de LibreOffice ne fournit plus les fichiers html de l'IDL, ni les références croisées des éléments de l'API, utilisés par Xray. Pour pallier à cela, mais malheureusement uniquement sous Windows, il existe un utilitaire (createLibO_IDL.vbs) contenu dans « LibO_IDL_creator pour XRay.zip », à télécharger sur la même page que précédemment.

Sous Windows, il n'est pas utile d'installer le SDK dans un répertoire où l'utilisateur à tous les droits. Il suffit, après l'installation dans le répertoire par défaut (LibreOffice), de modifier les autorisations accordées à l'utilisateur (Clic droit sur le répertoire « sdk » > Propriétés > Onglet sécurité > bouton Modifier > Sélectionner l'utilisateur > Cocher la case Contrôle total).

Il est alors possible d'appliquer les consignes d'utilisation de ce script « .vbs » contenues dans le fichier ReadMe.txt.

On veillera à appliquer la modification du code Basic de Xray indiqué à la dernière ligne de ce fichier ReadMe.

Pour les utilisateurs de Linux, de Mac OS X, et autres, se reporter aux documentations respectives, relatives à la gestion des droits sur les dossiers ou répertoires.
Pour ceux-ci, qui ne peuvent exécuter le script « .vbs », et pour les utilisateurs Windows qui ne souhaitent pas l'exécuter, un fichier « LibO6.4_SDK_Addon4Xray.zip » a été préparé. Il suffit de le dézipper, puis de copier-coller le répertoire « Common » et son contenu dans le sous-répertoire « docs » du répertoire dans lequel le SDK a été installé, par exemple :

- sous Windows 10 : « C:\Program Files\LibreOffice\sdk\docs »
- sous Linux Debian, Ubuntu : « /opt/libreoffice6.4/sdk/docs/ »

Le fichier LibO6.4_SDK_Addon4Xray.zip pourra être trouvé dans les « **Fichiers additionnels** » à la page : https://wiki.documentfoundation.org/FR/Doc#Base_6.4.

Propriétés des formulaires et des contrôles

Le modèle d'un contrôle décrit ses propriétés. Selon la situation, la valeur d'une propriété est accessible en lecture seule ou en écriture seule. L'ordre suit celui des listes de « Propriétés des champs de contrôle » du chapitre 4, Formulaires.

Police de caractère

Dans chaque contrôle qui affiche du texte, les propriétés de police peuvent être personnalisées.

Nom	*Type*	*L/E*	*Propriété*
FontName	string	L+E	Nom de la police
FontHeight	single	L+E	Taille de la police
FontWeight	single	L+E	Gras ou normal
FontSlant	integer	L+E	Italique ou roman
FontUnderline	integer	L+E	Souligné
FontStrikeout	integer	L+E	Barré

Formulaire

Terme anglais : Form

Nom	*Type*	*L/E*	*Propriété*
ApplyFilter	boolean	L+E	Filtre appliqué
Filter	string	L+E	Filtre actuel pour l'enregistrement
FetchSize	long	L+E	Nombre d'enregistrements chargés à la fois
Row	long	L	Numéro de la ligne actuelle
RowCount	long	L	Nombre d'enregistrements

Propriétés applicables à tous les contrôles

Control – voir aussi FormComponent

Nom	*Type*	*L/E*	*Propriété*
Name	string	L+(E)	Nom du champ/contrôle
Enabled	boolean	L+E	Actif : le champ peut être sélectionné
EnableVisible	boolean	L+E	Le champ est affiché
ReadOnly	boolean	L+E	Le contenu du champ ne peut être modifié
TabStop	boolean	L+E	Le champ est accessible via la touche Tab
Align	integer	L+E	Alignement horizontal : 0 = gauche, 1 = centré, 2 = droite
BackgroundColor	long	L+E	Couleur de l'arrière plan

Nom	Type	L/E	Propriété
Tag	string	L+E	Information additionnelle
HelpText	string	L+E	Texte d'aide sous forme d'info-bulle

S'appliquant à de nombreux types de contrôles

Nom	Type	L/E	Propriété
Text	string	(L+E)	Contenu affiché du champ. Dans les champs de texte, cela peut être lu et traité ultérieurement, mais cela ne fonctionne généralement pas pour les autres types.
Spin	boolean	L+E	Compteur incorporé dans un champ formaté.
TextColor	long	L+E	Couleur du texte (premier plan).
DataField	string	L	Nom du champ dans l'ensemble de données.
BoundField	object	L	Objet représentant la connexion à l'ensemble de données et donnant accès au contenu du champ.

Champ de texte – autres propriétés (TextField)

Nom	Type	L/E	Propriété
String	string	L+E	Contenu du champ affiché.
MaxTextLen	integer	L+E	Longueur maximale du texte.
DefaultText	string	L+E	Texte par défaut.
MultiLine	boolean	L+E	Indique s'il peut contenir plus d'une ligne
EchoChar	(integer)	L+E	Caractère affiché pour la saisie du mot de passe.

Champ numérique (NumericField)

Nom	Type	L/E	Propriété
ValueMin	double	L+E	Valeur d'entrée minimale acceptable
ValueMax	double	L+E	Valeur d'entrée maximale acceptable
Value	double	L+(E)	Valeur actuelle (Ne pas utiliser pour les valeurs de l'ensemble de données).
ValueStep	double	L+E	Intervalle correspondant à un clic pour la molette de la souris ou le compteur.
DefaultValue	double	L+E	Valeur par défaut..
DecimalAccuracy	integer	L+E	Nombre de décimales.
ShowThousandsSeparator	boolean	L+E	Afficher le séparateur de paramètres régionaux pour des milliers.

Champ Date (DateField)

Les valeurs de date sont définies par le type de données **long** et sont stockées au format ISO : AAAAMMJJ, par exemple 20190304 pour le 4 mars 2019. Pour utiliser ce type avec **getDate** et **updateDate**, et avec le type **com.sun.star.util. Date**, voir les exemples.

Nom	***Type de données***	***Type de données depuis LO 4.1.1***	***L/E***	***Propriété***
DateMin	long	com.sun.star. util.Date	L+E	Date d'entrée minimale acceptable.
DateMax	long	com.sun.star. util.Date	L+E	Date d'entrée maximale acceptable.
Date	long	com.sun.star. util.Date	L+(E)	Valeur actuelle (Ne pas utiliser pour les valeurs de l'ensemble de données).
DateFormat	integer		L+E	Format de date spécifique au système d'exploitation: 0 = date courte (simple) 1 = date courte jj.mm.aa (année à 2 chiffres) 2 = date courte jj.mm.aaaa (année à 4 chiffres) 3 = date longue (comprend le jour de la semaine et le nom du mois) D'autres possibilités peuvent être trouvées dans la définition du formulaire ou dans la référence API.
DefaultDate	long	com.sun.star. util.Date	L+E	Valeur par défaut
DropDown	boolean		L+E	Afficher un calendrier mensuel déroulant.

Champ durée (TimeField)

Les valeurs de temps sont également du type **long**.

Nom	***Type de données***	***Type de données depuis LO 4.1.1***	***L/E***	***Propriété***
TimeMin	long	com.sun.star. util.Time	L+E	Valeur d'entrée minimale acceptable
TimeMax	long	com.sun.star. util.Time	L+E	Valeur d'entrée maximale acceptable
Time	long	com.sun.star. util.Time	L+(E)	Valeur actuelle (Ne pas utiliser pour les valeurs de l'ensemble de données).

Nom	Type de données	Type de données depuis LO 4.1.1	L/E	Propriété
TimeFormat	integer		L+E	Format de l'heure : 0 = court comme hh : mm (heures, minutes, 24 heures) 1 = long hh : mm : ss (même chose avec Secondes, horloge sur 24 heures) 2 = court comme hh : mm (horloge 12 heures avec AM / PM) 3 = long comme hh : mm : ss (horloge 12 heures avec AM / PM) 4 = entrée courte pour une durée 5 = entrée longue pour une durée
DefaultTime	long	com.sun.star.util.Time	L+E	Valeur par défaut.

Champ de devise (CurrencyField)

Un champ monétaire est un champ numérique avec les possibilités supplémentaires suivantes.

Nom	Type	L/E	Propriété
CurrencySymbol	string	L+E	Symbole de la monnaie pour affichage seulement.
PrependCurrencySymbol	boolean	L+E	Le symbole est affiché avant le chiffre.

Champ formaté (FormattedControl)

Un contrôle formaté peut être utilisé à volonté pour les nombres, la devise ou la date/heure. De très nombreuses propriétés déjà décrites s'appliquent ici mais avec des noms différents.

Nom	Type	L/E	Propriété
CurrentValue	variant	L	Valeur actuelle du contenu. Le type de données réel dépend du contenu et du format.
EffectiveValue	variant	L+(E)	
EffectiveMin	double	L+E	Valeur d'entrée minimale acceptable
EffectiveMax	double	L+E	Valeur d'entrée maximale acceptable
EffectiveDefault	variant	L+E	Valeur par défaut
FormatKey	long	L+(E)	Format d'affichage et de saisie. Il n'y a pas de moyen facile de modifier cela à l'aide d'une macro.
EnforceFormat	boolean	L+E	Le format est testé lors de la saisie. Seuls certains caractères et combinaisons sont autorisés.

Zone de liste (ListBox)

L'accès en lecture et en écriture à la valeur située derrière la ligne sélectionnée est quelque peu compliqué mais possible.

Nom	*Type*	*L/E*	*Propriété*
ListSource	array of string	L+E	Source de données : source du contenu de la liste ou nom de l'ensemble de données qui fournit l'entrée visible.
ListSourceType	integer	L+E	Type de source de données : 0 = Liste de valeurs 1 = Tableau 2 = Requête 3 = Ensemble de résultats d'une commande SQL 4 = Résultat d'une commande de base de données 5 = Noms de champs d'une table de base de données
StringItemList	array of string	L	Liste des entrées disponibles pour la sélection.
ItemCount	integer	L	Nombre d'entrées de liste disponibles
ValueItemList	array of string	L	Liste des valeurs à passer du formulaire à la table.
DropDown	boolean	L+E	La liste déroulante.
LineCount	integer	L+E	Nombre total des lignes affichées une fois complètement déroulées.
MultiSelection	boolean	L+E	Sélection multiple prévue.
SelectedItems	array of integer	L+E	Liste des entrées sélectionnées sous forme de liste de positions dans la liste globale des entrées.

Le premier élément sélectionné dans le champ de liste est obtenu comme ceci :

```
oControle = oFormulaire.getByName("Nom de la Zone de Liste")
sEntree = oControl.ValueItemList(oControl.SelectedItems(0))
```

Note

Depuis LibreOffice 4.1, la valeur transmise à la base de données peut être déterminée directement.

```
oControle = oFormulaire.getByName("Nom de la Zone de Liste")
iD = oControle.getCurrentValue()
```

`getCurrentValue()` renvoie la valeur qui sera stockée dans la table de base de données. Dans les Zones de liste, cela dépend du champ auquel elles sont liées (`BoundField`).

Jusqu'à LibreOffice 4.0 inclus, cette fonction renvoyait le contenu affiché, pas la valeur sous-jacente dans la table.

Veuillez noter que l'entrée est un « tableau de chaînes » (array of string), si la requête d'un champ de liste est exécutée pour restreindre une option de sélection :

```
Sub FiltreZoneListe
	Dim stSql(0) As String
	Dim oDoc As Object
	Dim oDrawpage As Object
	Dim oFormulaire As Object
	Dim oChamp As Object
	oDoc = thisComponent
	oDrawpage = oDoc.drawpage
	oFormulaire = oDrawpage.forms.getByName("Formulaire")
	oChamp = oFormulaire.getByname("Zone de liste")
	stSql(0) = "SELECT ""Nom"", ""ID"" FROM ""Nom_Requete_Filtre"" ORDER BY
""Nom"""
	oChamp.ListSource = stSql
	oChamp.refresh
End Sub
```

Zone combinée (ComboBox)

Malgré le fait d'avoir des fonctionnalités similaires à celles des Zones de liste, les propriétés des Combobox sont quelque peu différentes. Voir l'exemple « Boîtes combinées sous forme de Zone de liste avec une option d'entrée » en page 461.

Nom	*Type*	*L/E*	*Propriété*
Autocomplete	boolean	L+E	Remplissage automatique.
StringItemList	array of string	L+E	Liste des entrées disponibles pour utilisation.
ItemCount	integer	L	Nombre d'entrées de liste disponibles.
DropDown	boolean	L+E	Liste déroulante.
LineCount	integer	L+E	Nombre de lignes affichées lorsqu'elle est déroulée.
Text	string	L+E	Texte actuellement affiché.
DefaultText	string	L+E	Entrée par défaut
ListSource	string	L+E	Nom de la source de données qui fournit les entrées de liste.
ListSourceType	integer	L+E	Type de source de données. Mêmes possibilités que pour les Zones de liste (seul le choix de ValueItemList est ignoré).

Cases à cocher (CheckBox) et Boutons Radio (RadioButton)

Les boutons d'options peuvent également être utilisés.

Nom	*Type*	*L/E*	*Propriété*
Label	string	L+E	Titre (label)
State	short	L+E	Statut 0 = non sélectionné 1 = sélectionné 2 = indéfini
MultiLine	boolean	L+E	Sauts de ligne pour le texte long.

Champ de motif (PatternField)

Outre les propriétés du texte simple, les éléments suivants sont intéressants :

Nom	*Type*	*L/E*	*Propriété*
EditMask	string	L+E	Masque de saisie.
LiteralMask	string	L+E	Masque de caractères.
StrictFormat	boolean	L+E	Test de format lors de la saisie.

Contrôle de Table (GridControl)

Nom	*Type*	*L/E*	*Propriété*
Count	long	L	Le nombre de colonnes.
ElementNames	array of string	L	Liste des noms de colonnes.
HasNavigationBar	boolean	L+E	Barre de navigation disponible.
RowHeight	long	L+E	Hauteur de ligne

Étiquette ou Texte fixe (Label)

Nom	*Type*	*L/E*	*Propriété*
Label	string	L+E	Text displayed.
MultiLine	boolean	L+E	Sauts de ligne pour le texte long.

Zones de groupe (GroupBox)

Il n'y a pas de propriétés pour les zones de groupe qui sont traitées normalement à l'aide de macros. C'est le statut des champs d'options individuels qui compte.

Boutons (CommandButton ou ImageButton)

Nom	*Type*	*L/E*	*Propriété*
Label	string	L+E	Titre – Texte de l'étiquette.
State	short	L+E	État par défaut sélectionné pour le basculement.
MultiLine	boolean	L+E	Sauts de ligne pour le texte long.
DefaultButton	boolean	L+E	S'il s'agit d'un bouton par défaut.

Barre de Navigation (NavigationBar)

D'autres propriétés et méthodes associées à la navigation – par exemple les filtres et la modification du pointeur d'enregistrement – sont contrôlées à l'aide du formulaire.

Nom	*Type*	*L/E*	*Propriété*
IconSize	short	L+E	Taille des icônes (Petites, Grandes)

Nom	Type	L/E	Propriété
ShowPosition	boolean	L+E	La position peut être saisie et s'affiche.
ShowNavigation	boolean	L+E	Permet la navigation.
ShowRecordActions	boolean	L+E	Permet des actions sur les enregistrements.
ShowFilterSort	boolean	L+E	Permet le tri par filtre.

Méthodes pour les formulaires et les contrôles

Le type de données du paramètre est indiqué par une abréviation :

- c numéro de colonne pour le champ souhaité dans l'ensemble de données, à partir de 1
- n valeur numérique – peut être un entier ou un nombre décimal
- s Chaîne, la longueur maximale dépend de la définition de la table.
- b booléen (logique) – vrai ou faux
- d Valeur de date

Naviguer dans un ensemble de données

Ces méthodes fonctionnent à la fois dans les formulaires et dans l'ensemble de résultats d'une requête.

« Curseur » dans la description signifie le pointeur d'enregistrement.

Nom	Type	Description
Test de la position du curseur		
isBeforeFirst	boolean	Le curseur se trouve avant le premier enregistrement. C'est le cas s'il n'a pas encore été réinitialisé après l'entrée.
isFirst	boolean	Indique si le curseur est sur la première entrée.
isLast	boolean	Indique si le curseur se trouve sur la dernière entrée.
isAfterLast	boolean	Le curseur se trouve après la dernière ligne lorsqu'il est déplacé avec suivant.
getRow	long	Numéro de ligne actuel.
Positionnement du curseur Pour les types de données booléens, True signifie que la navigation a réussi.		
beforeFirst	–	Se déplace avant la première ligne.
first	boolean	Passe à la première ligne.
previous	boolean	Remonte d'une ligne.
next	boolean	Avance d'une ligne.
last	boolean	Va au dernier enregistrement.
afterLast	–	Va après le dernier enregistrement.
absolute(n)	boolean	Accède à la ligne avec le numéro de ligne donné.

Nom	Type	Description
relative(n)	boolean	Va en arrière ou en avant du nombre donné : en avant pour les arguments positifs et en arrière pour les arguments négatifs.
Méthodes affectant l'état actuel de l'enregistrement		
refreshRow	–	Relit les valeurs d'origine de la ligne.
rowInserted	boolean	Indique s'il s'agit d'une nouvelle ligne.
rowUpdated	boolean	Indique si la ligne actuelle a été modifiée.
rowDeleted	boolean	Indique si la ligne actuelle a été supprimée.

Modification des lignes (enregistrements) de données

Les méthodes utilisées pour la lecture sont disponibles pour n'importe quel formulaire ou ensemble de données. Les méthodes de modification et de stockage ne peuvent être utilisées que pour les ensembles de données modifiables (généralement des tables, pas des requêtes).

Nom	Type	Description
Méthodes pour ligne entière.		
insertRow	–	Enregistre une nouvelle ligne.
updateRow	–	Confirme la modification de la ligne actuelle.
deleteRow	–	Supprime la ligne actuelle.
cancelRowUpdates	–	Annule les modifications de la ligne actuelle.
moveToInsertRow	–	Déplace le curseur sur une ligne correspondant à un nouvel enregistrement.
moveToCurrentRow	–	Après la saisie d'un nouvel enregistrement, ramène le curseur à sa position précédente.
Lecture des valeurs		
getString(c)	string	Donne le contenu de la colonne sous forme de chaîne de caractères.
getBoolean(c)	boolean	Donne le contenu de la colonne sous forme de valeur booléenne.
getByte(c)	byte	Donne le contenu de la colonne sous la forme d'un octet unique.
getShort(c)	short	Donne le contenu de la colonne sous forme d'entier.
getInt(c)	integer	
getLong(c)	long	
getFloat(c)	float	Donne le contenu de la colonne sous forme de nombre décimal de précision unique.

Nom	*Type*	*Description*
getDouble(c)	double	Donne le contenu de la colonne sous forme de nombre décimal à double précision. Les conversions automatiques effectuées par Basic en font un type approprié pour les champs décimaux et monétaires.
getBytes(c)	array of bytes	Donne le contenu de la colonne sous forme de tableau d'octets uniques.
getDate(c)	Date	Donne le contenu de la colonne sous forme de date.
getTime(c)	Time	Donne le contenu de la colonne sous forme de valeur de temps.
getTimestamp(c)	DateTime	Donne le contenu de la colonne sous forme d'horodatage (date et heure).
Dans Basic lui-même, les valeurs de date et d'heure ont toutes deux le type DATE. Pour accéder aux dates dans les ensembles de données, différents types sont disponibles : **com.sun.star.util.Date** pour une date, **com.sun.star.util.Time** pour une heure et **com.sun.star.util.DateTime** pour un horodatage.		
wasNull	boolean	Indique si la valeur de la colonne la plus récemment lue était NULL.
Enregistrer les valeurs		
updateNull(c)	–	Définit le contenu de la colonne sur NULL.
updateBoolean(c,b)	–	Modifie le contenu de la colonne c en valeur logique b.
updateByte(c,x)	–	Stocke l'octet x dans la colonne c.
updateShort(c,n)	–	Stocke n (Integer, Long ou Float) dans la colonne c.
updateInt(c,n)	–	
updateLong(c,n)	–	
updateFloat(c,n)	–	Stocke le nombre décimal n dans la colonne c.
updateDouble(c,n)	–	
updateString(c,s)	–	Stocke la chaîne s dans la colonne c.
updateBytes(c,x)	–	Stocke le tableau d'octets x dans la colonne c.
updateDate(c,d)	–	Stocke la date d dans la colonne c.
updateTime(c,d)	–	Stocke l'heure d dans la colonne c.
updateTimestamp(c,d)	–	Stocke l'horodatage d dans la colonne c.

Modifier des valeurs individuelles

Cette méthode utilise la propriété **BoundField** d'un contrôle pour lire ou modifier le contenu de la colonne correspondante. Cela correspond presque exactement à la méthode décrite dans la section précédente, sauf que le numéro de colonne n'est pas indiqué.

Nom	*Type*	*Description*
Lecture des valeurs		
getString	string	Donne le contenu du champ sous forme de chaîne de caractères.
getBoolean	boolean	Donne le contenu du champ sous forme de valeur logique.
getByte	byte	Donne le contenu du champ sous la forme d'un octet unique.
getShort	short	Donne le contenu du champ sous forme d'entier.
getInt	integer	
getLong	long	
getFloat	float	Donne le contenu du champ sous la forme d'une valeur décimale simple précision.
getDouble	double	Donne le contenu du champ sous forme de nombre décimal à double précision. Les conversions automatiques effectuées par Basic en font un type approprié pour les champs décimaux et monétaires.
getBytes	array of bytes	Donne le contenu du champ sous forme de tableau d'octets.
getDate	Date	Donne le contenu du champ sous forme de date.
getTime	Time	Donne le contenu du champ sous forme d'heure.
getTimestamp	DateTime	Donne le contenu du champ sous forme d'horodatage.
Dans Basic lui-même, les valeurs de date et d'heure ont toutes deux le type **DATE**. Pour accéder aux dates dans les ensembles de données, différents types sont disponibles : **com.sun.star.util.Date** pour une date, **com.sun.star.util.Time** pour une heure et **com.sun.star.util.DateTime** pour un horodatage.		
wasNull	boolean	Indique si la valeur de la colonne la plus récemment lue était NULL.
Stockage des valeurs		
updateNull	–	Définit le contenu de la colonne sur NULL.
updateBoolean(b)	–	Définit le contenu de la colonne sur la valeur logique b.
updateByte(x)	–	Stocke l'octet x dans la colonne.
updateShort(n)	–	Stocke l'entier n dans la colonne.
updateInt(n)	–	
updateLong(n)	–	
updateFloat(n)	–	Stocke le nombre décimal n dans la colonne.
updateDouble(n)	–	

Nom	*Type*	*Description*
updateString(s)	–	Stocke la chaîne de caractères s dans la colonne.
updateBytes(x)	–	Stocke le tableau d'octets x dans la colonne.
updateDate(d)	–	Stocke la date d dans la colonne.
updateTime(d)	–	Stocke l'heure d dans la colonne.
updateTimestamp(d)	–	Stocke l'horodatage d dans la colonne.

Paramètres des commandes SQL préparées

Les méthodes qui transfèrent la valeur d'une commande SQL pré-préparée (voir « Commandes SQL pré-préparées avec paramètres » en page 414) sont similaires à celles de la section précédente. Le premier paramètre (indiqué par i) est une position numérotée dans la commande SQL.

Nom	*Type*	*Description*
setNull(i,n)	–	Définit le contenu de la colonne sur NULL. N est le type de données SQL comme indiqué dans la référence API.
setBoolean(i,b)	–	Place la valeur logique b donnée dans la commande SQL.
setByte(i,x)	–	Place l'octet x donné dans la commande SQL.
setShort(i,n)	–	Place l'entier n donné dans la commande SQL.
setInt(i,n)		
setLong(i,n)		
setFloat(i,n)	–	Place le nombre décimal donné dans la commande SQL.
setDouble(i,n)		
setString(i,s)	–	Place la chaîne de caractères s dans la commande SQL.
setBytes(i,x)	–	Place le tableau d'octets x donné dans la commande SQL.
setDate(i,d)	–	Place la date donnée d dans la commande SQL.
setTime(i,d)	–	Place le temps donné d dans la commande SQL.
setTimestamp(i,d)	–	Place l'horodatage donné d dans la commande SQL.
clearParameters	–	Supprime les valeurs précédentes de tous les paramètres d'une commande SQL.

Améliorer la convivialité

Pour cette première catégorie d'utilisation de macro, nous montrons différentes possibilités pour améliorer la convivialité des formulaires de base.

Mise à jour automatique des formulaires

Souvent, quelque chose est modifié dans un formulaire et cette modification doit apparaître dans un deuxième formulaire sur la même page. L'extrait de code suivant appelle la méthode de rechargement sur le deuxième formulaire, provoquant son actualisation.

```
Sub MiseAJour
```

Tout d'abord, la macro est nommée. La désignation par défaut d'une macro est **Sub**. Cela peut être écrit en majuscules ou en minuscules. Sub permet à un sous-programme de s'exécuter sans renvoyer de valeur. Plus bas, en revanche, une fonction est décrite, qui renvoie une valeur.

La macro porte le nom *MiseAJour*. Vous n'avez pas besoin de déclarer des variables car LibreOffice Basic crée automatiquement des variables lorsqu'elles sont utilisées. Malheureusement, si vous avez mal orthographié une variable, LibreOffice Basic crée silencieusement une nouvelle variable sans se plaindre. Utilisez **Option Explicit** Pour empêcher LibreoOffice Basic de créer automatiquement des variables, ceci est recommandé par la plupart des programmeurs.

Par conséquent, nous commençons généralement par déclarer des variables. Toutes les variables déclarées ici sont des objets (pas des nombres ou du texte), nous ajoutons donc **As Object** à la fin de la déclaration. Pour nous rappeler plus tard le type des variables, nous faisons précéder leurs noms d'un "o" (pour Object). En principe, cependant, vous pouvez choisir presque tous les noms de variables que vous désirez.

```
        Dim oDoc As Object
        Dim oDrawpage As Object
        Dim oFormulaire As Object
```

Le formulaire se trouve dans le document actuellement actif. Le conteneur, dans lequel tous les formulaires sont stockés, est nommé **drawpage**. Dans le navigateur de formulaires, il s'agit du concept de niveau supérieur duquel tous les formulaires sont subsidiaires.

Dans cet exemple, le formulaire auquel accéder est nommé Affiche, nom visible dans le navigateur de formulaire. Par défaut, le premier formulaire par défaut est appelé Formulaire1.

```
        oDoc = thisComponent
        oDrawpage = oDoc.drawpage
        oFormulaire = oDrawpage.forms.getByName("Affiche")
```

Le formulaire étant désormais accessible et le point par lequel il est accessible est enregistré dans la variable **oFormulaire**, il est maintenant rechargé (rafraîchi) avec la commande **reload()**.

```
        oFormulaire.reload()
End Sub
```

Le sous-programme commence par **Sub** et doit donc se terminer par **End Sub**.

Cette macro peut maintenant être sélectionnée pour s'exécuter lorsqu'un autre formulaire est enregistré. Par exemple, sur une caisse enregistreuse (caisse), si le nombre total d'articles vendus et leurs numéros de stock (lus par un lecteur de codes-barres) sont entrés dans un formulaire, un autre formulaire dans la même fenêtre ouverte peut afficher les noms de tous les articles, et le coût total, dès que le formulaire est enregistré.

Filtrage des enregistrements

Le filtre lui-même peut parfaitement fonctionner sous la forme décrite au Chapitre 8, Astuces. La variante ci-dessous remplace le bouton Enregistrer et lit à nouveau les zones de liste, de sorte qu'un filtre choisi dans une zone puisse restreindre les choix disponibles dans l'autre zone[9].

```
Sub Filtre
```

En premier lieu, les variables sont déclarées.

```
    Dim oDoc As Object
    Dim oDrawpage As Object
    Dim oFormulaire1 As Object
    Dim oFormulaire2 As Object
    Dim oZoneListe1 As Object
    Dim oZoneListe2 As Object
```

Puis, elles sont paramétrées pour accéder à l'ensemble des formulaires. Cet ensemble comprend les deux formulaires « Filtre » et « Affichage ». Les zones de liste sont dans le formulaire « Filtre » et portent les noms « ZoneListe_1 » et « ZoneListe_2.

```
    oDoc = thisComponent
    oDrawpage = oDoc.drawpage
    oFormulaire1 = oDrawpage.forms.getByName("Filtre")
    oFormulaire2 = oDrawpage.forms.getByName("Affiche")
    oZoneListe1 = oFormulaire1.getByName("ZoneListe_1")
    oZoneListe2 = oFormulaire1.getByName("ZoneListe_2")
```

Ensuite, le contenu des zones de liste est transféré vers le formulaire sous-jacent en utilisant **commit()**. Le transfert est nécessaire, car sinon le changement dans une Zone de liste ne sera pas reconnu lors de l'enregistrement. L'instruction **commit()** n'a besoin d'être appliquée qu'à la Zone de liste qui vient d'être accédée. Après cela, l'enregistrement est réalisé à l'aide de **updateRow()**.

```
    oZoneListe1.commit()
    oZoneListe2.commit()
    oFormulaire1.updateRow()
```

En principe, notre table de filtres ne contient qu'un seul enregistrement, qui est écrit une fois au début. Cet enregistrement est donc écrasé en permanence à l'aide d'une commande de mise à jour.

```
    oZoneListe1.refresh()
    oZoneListe2.refresh()
    oFormulaire2.reload()
End Sub
```

Les listes de sélection sont destinées à s'influencer mutuellement. Par exemple, si une zone de liste est utilisée pour limiter les médias affichés aux CD, l'autre zone de liste ne doit pas inclure tous les auteurs de livres dans sa liste d'auteurs. Une sélection dans la deuxième zone de liste aboutirait alors trop souvent à un filtre vide. C'est pourquoi les zones de liste doivent être relues. À proprement parler, la commande **refresh()** ne doit être exécutée que sur la zone de liste qui n'a pas été accédée.

Après cela, Formulaire2, qui devrait afficher le contenu filtré, est à nouveau lu (reload).

9 Voir la base de données Exemple_Cherche_et_ Filtre.odb associée à ce manuel.

Les zones de liste qui doivent être influencées à l'aide de cette méthode peuvent recevoir du contenu à l'aide de diverses requêtes.

La variante la plus simple consiste à faire en sorte que la zone de liste tire son contenu des résultats du filtre. Ensuite, un seul filtre détermine quel contenu de données sera filtré davantage.

```
SELECT "Champ_1" || ' – ' || "Nombre" AS "Affiche", "Champ_1"
FROM
        (SELECT COUNT("ID") AS "Nombre", "Champ_1"
        FROM "TableRecherche" GROUP BY "Champ_1")
ORDER BY "Champ_1"
```

Le contenu du champ et le nombre de résultats s'affichent. Pour obtenir le nombre de résultats, une sous-requête est utilisée. Ceci est nécessaire car sinon, seul le nombre de résultats, sans plus d'informations sur le champ, sera affiché dans la zone de liste.

La macro crée des zones de liste assez rapidement par cette action, elles ne contiennent qu'une seule valeur. Si une zone de liste n'est pas NULL, elle est prise en compte par le filtrage. Après l'activation de la deuxième zone de liste, seuls les champs vides et une valeur affichée sont disponibles pour les deux zones de liste. Cela peut sembler pratique pour une recherche limitée. Mais que se passe-t-il si un catalogue de bibliothèque montre clairement la classification d'un élément, mais ne montre pas uniquement s'il s'agit d'un livre, d'un CD ou d'un DVD ? Si la classification est choisie en premier et que la seconde zone de liste est alors mise à « CD », elle doit être remise à NULL afin d'effectuer une recherche ultérieure qui inclut des livres. Il serait plus pratique que la deuxième zone de liste affiche directement les différents types de médias disponibles, avec le nombre de succès correspondant.

Pour atteindre cet objectif, la requête suivante est construite, qui n'est plus alimentée directement à partir des résultats du filtre. Le nombre de succès doit être obtenu d'une manière différente.

```
SELECT
IFNULL("Champ_1" || ' ' || "Compte", 'vide – ' || "Compte")
AS "Affiche", "Champ_1"
FROM
      (SELECT COUNT("ID") AS "Compte", "Champ_1" FROM "Table"
         WHERE "ID" IN
         (SELECT "Table"."ID" FROM "Filtre", "Table"
            WHERE "Table"."Champ_2" = IFNULL("Filtre"."Filtre_2",
            "Table"."Champ_2"))
      GROUP BY "Champ_1")
ORDER BY "Champ_1"
```

Cette requête très complexe peut être décomposée. En pratique, il est courant d'utiliser une **VUE** pour la sous-requête. La zone de liste reçoit son contenu à partir d'une requête relative à cette **VUE**.

La requête en détail : La requête présente deux colonnes. La première colonne contient la vue, vue par une personne qui a ouvert le formulaire. Cette vue montre le contenu du champ et, séparés par un tiret, les résultats pour ce contenu de champ. La deuxième colonne transfère son contenu vers la table sous-jacente du formulaire. Ici, nous n'avons que le contenu du champ. Les zones de liste tirent ainsi leur contenu de la requête, qui est présentée comme résultat du filtre dans le formulaire. Seuls ces champs sont disponibles pour un filtrage supplémentaire.

La table à partir de laquelle ces informations sont tirées est en fait une requête. Dans cette requête, les champs de clé primaire sont comptés (`SELECT COUNT("ID") AS "Nombre"`). Ceci est ensuite regroupé par le terme de recherche dans le champ (`GROUP BY "Champ_1"`). Cette requête présente le terme dans le champ lui-même en tant que deuxième colonne. Cette requête à son tour est basée sur une autre sous-requête :

```
SELECT "Table"."ID" FROM "Filtre", "Table"
      WHERE
           "Table"."Champ_2" = IFNULL("Filtre"."Filtre_2", Table"."Champ_2")
```

Cette sous-requête traite de l'autre champ à filtrer. En principe, cet autre champ doit également correspondre à la clé primaire. S'il existe d'autres filtres, cette requête peut être étendue :

```
SELECT "Table"."ID" FROM "Filtre", "Table"
      WHERE
          "Table"."Champ_2" = IFNULL("Filtre"."Filtre_2", Table"."Champ_2")
AND
            "Table"."Champ_3" = IFNULL("Filtre"."Filtre_3",
"Table"."Champ_3")
```

Cela permet à tous les champs supplémentaires qui doivent être filtrés de contrôler ce qui apparaît finalement dans la zone de liste du premier champ, "Champ_1".

Enfin, toute la requête est triée par le champ sous-jacent.

À quoi ressemble réellement la requête finale sous-jacente au formulaire affiché, peut être vu dans le Chapitre 8, « Trucs et Astuces".

La macro suivante peut contrôler, via une zone de liste, quelles zones de liste doivent être enregistrées et lesquelles doivent être lues à nouveau. Le sous-programme suivant suppose que la propriété Complément d'information pour chaque zone de liste contient une liste séparée par des virgules de tous les noms de zone de liste sans espaces. Le premier nom de la liste doit être le nom de cette zone de liste elle-même.

```
Sub Filtre_info_Additionnelle(oEvent As Object)
       Dim oDoc As Object
       Dim oDrawpage As Object
       Dim oFormulaire1 As Object
       Dim oFormulaire2 As Object
       Dim sTag As String
       Dim arListe()
          sTag = oEvent.Source.Model.Tag
```

Un tableau (une collection de données accessible via un numéro d'index) est établi et rempli avec les noms de champs des zones de liste. Le premier nom de la liste est le nom de la zone de liste liée à l'événement.

```
       arListe() = Split(sTag, ",")
       oDoc = thisComponent
       oDrawpage = oDoc.drawpage
       oFormulaire1 = oDrawpage.forms.getByName("Filtre")
          oFormulaire2 = oDrawpage.forms.getByName("Affiche")
```

Le tableau est parcouru de sa borne inférieure ('**Lbound()**') à sa borne supérieure ('**Ubound()**') en une seule boucle. Toutes les valeurs séparées par des virgules dans les informations supplémentaires sont maintenant transférées successivement.

```
        For i = LBound(aList()) To UBound(arListe())
            If i = 0 Then
```

La zone de liste qui a déclenché la macro doit être enregistrée. Elle se trouve dans la variable **arListe(0)**. Tout d'abord, les informations de la zone de liste sont transmises à la table sous-jacente, puis l'enregistrement est enregistré.

```
                oFormulaire1.getByName(arListe(i)).commit()
                oFormulaire1.updateRow()
                Else
```

Les autres zones de liste doivent être actualisées, car elles contiennent désormais des valeurs différentes selon le choix effectué dans la première zone de liste.

```
                oFormulaire1.getByName(arListe(i)).refresh()
            End If
        Next
        oFormulaire2.reload()
End Sub
```

Les requêtes pour cette macro plus utilisable sont naturellement les mêmes que celles déjà présentées dans la section précédente.

Filtrage des données via le filtre de formulaires

Comme alternative à cette procédure, il est également possible de modifier directement la fonction de filtrage du formulaire.

```
SUB DefinirFiltrage
        DIM oDoc AS OBJECT
        DIM oForm AS OBJECT
        DIM oChamp  AS OBJECT
        DIM stFiltre As String
        oForm = thisComponent.Drawpage.Forms.getByName("Formulaire")
        oChamp = oForm.getByName("Filtre")
        stFiltre = oChamp.Text
        oForm.filter = " UPPER(""Nom"") LIKE '%'||'" + UCase(stFiltre) + "'||'%'"
        oForm.ApplyFilter = TRUE
        oForm.reload()
End Sub
```

Le champ **Filtre** est recherché dans le formulaire Formulaire et le contenu est lu dans la variable **stFiltre**. Le filtre est réglé en conséquence (**oForm.filter**). Le filtre est activé et le formulaire est rechargé.

```
SUB SupprimeFiltre
        DIM oForm AS OBJECT
        oForm = thisComponent.Drawpage.Forms.getByName("Formulaire")
        oForm.ApplyFilter = False
        oForm.reload()
END SUB
```

Bien entendu, le filtre peut également être supprimé via la barre de navigation.

Avec cette fonction de filtrage, un formulaire peut également être lancé directement avec un filtre, par exemple « Dupont » pour un seul enregistrement.

Faire défiler les enregistrements avec une barre de défilement

Un contrôle de barre de défilement ne peut être utilisé que par le biais de macros. L'exemple_ suivant montre comment faire défiler des enregistrements de données avec une telle barre de défilement. Elle peut alors être utilisée à la place de la barre de navigation[10].

```
GLOBAL loPosistion AS LONG
```

La position de l'enregistrement en cours est stockée sous la forme d'une variable globale, de sorte qu'elle peut être lue dans toutes les procédures et modifiée dans toutes les procédures.

```
SUB LigneMax(oEvent AS OBJECT)
       ' Déclenché les événements du formulaire
       ' "Lors du chargement" et "Après l'action d'enregistrement".
       DIM oFormulaire AS OBJECT
       DIM oDefileChamp AS OBJECT
       DIM loMax AS LONG
       oFormulaire = oEvent.Source
       oDefileChamp = oFormulaire.getByName("Barre_defilement")
       loPosition = oFormulaire.getRow
       oFormulaire.last
       loMax = oFormulaire.getRow
       oFormulaire.absolute(loPos)
       oDefileChamp.ScrollValueMax = loMax
       oDefileChamp.ScrollValue = loPos
END SUB
```

Le nombre total d'enregistrements de données ne peut pas être déterminé à l'aide de la fonction RowCount du formulaire, car cette fonction ne compte que les enregistrements de données qui ont déjà été chargés dans la mémoire cache. Par conséquent, le numéro de la ligne actuelle du formulaire est lu en premier, puis le saut à la fin des données à lire est effectué et le numéro de la ligne est déterminé comme valeur maximale. Ensuite, il est nécessaire de revenir à la ligne d'origine avec **oFormulaire.absolute()**.

La barre de défilement est informée de la valeur maximale déterminée en tant que ScrollValueMax et de la position actuelle en tant que loPosition. Si cette valeur c'est pas affectée, la position dans la barre de défilement ne correspond pas nécessairement à l'ensemble des données actuelles.

```
SUB Navigation(oEvent AS OBJECT)
       ' Déclenché par l'événement du formulaire "Après le changement
d'enregistrement".
       ' Synchronise la position du défilement avec la position de l'ensemble des
données
       DIM oFormulaire AS OBJECT
       DIM oDefileChamp AS OBJECT
       oFormulaire = oEvent.Source
       oDefileChamp = oFormulaire.getByName("Barre_defilement")
       loPos = oFormulaire.getRow
       IF loPos = 0 THEN
           loPos = oFormulaire.RowCount + 1
       END IF
       oDefileChamp.ScrollValue = loPos
END SUB
```

Lors de la navigation dans les enregistrements de données, l'affichage de la barre de défilement et l'affichage de la barre de navigation doivent toujours correspondre. C'est pourquoi le numéro de la

10 Voir la base de données Exemple_Defiler_Enregistrements.odb associée à ce manuel.

ligne en cours est toujours déterminé après le changement d'enregistrement de données. Pour le numéro de la ligne en cours, un "0" est affiché lorsque le curseur se déplace au-delà de la dernière ligne pour ajouter un nouvel enregistrement. Dans ce cas, cependant, la barre de défilement ne doit pas revenir à la position de départ, mais doit être positionnée à 1 au-dessus de la valeur maximale précédente, comme la barre de navigation.

```
SUB Defilement(oEvent AS OBJECT)
          ' Déclenché par l'événement de la "Barre_defilement" - "A l'ajustement"
          DIM oFormulaire AS OBJECT
          DIM oDefiler AS OBJECT
          oDefiler = oEvent.Source
          oFormulaire = oDefiler.Model.Parent
          loPos = oDefiler.getValue()
          oFormulaire.absolute(loPos)
END SUB
```

Dans cette procédure, une nouvelle valeur pour la ligne du formulaire est déterminée à partir de la barre de défilement. La valeur est lue dans la barre de défilement avec getValue() et affectée au formulaire avec oFormulaire.absolute(loPos).

Adapter les données des champs de textes aux conventions du langage SQL

Lorsque les données sont stockées dans une commande SQL, les guillemets simples (') dans des noms tels que "O'Connor" peuvent poser des problèmes. Cela est dû au fait que les guillemets simples (' et ') sont utilisés pour encadrer le texte à saisir dans les enregistrements. Dans de tels cas, nous avons besoin d'une fonction intermédiaire pour préparer les données de manière appropriée[11].

```
Function String_to_SQL(st As String)
       If InStr(st,"'") Then
          st = Join(Split(st,"'"),"''")
       End If
       String_to_SQL = st
End Function
```

Notez que c'est une fonction, pas une procédure. Une fonction prend une valeur comme argument, puis renvoie une valeur.

Le texte à transférer est d'abord scanné pour voir s'il contient une apostrophe. Si tel est le cas, le texte est divisé à ce stade – l'apostrophe elle-même est le délimiteur de la division – et joint à nouveau avec deux apostrophes. Cela masque le code SQL. La fonction donne son résultat via l'appel suivant :

```
stNouveauTexte = String_to_SQL(stAncienTexte)
```

Cela signifie simplement que la variable stAncienTexte est retravaillée et le résultat stocké dans stNouveauTexte. Les deux variables n'ont pas besoin d'avoir des noms différents. L'appel peut être fait avec :

```
stTexte = String_to_SQL(stTexte)
```

Cette fonction est utilisée à plusieurs reprises dans les macros suivantes afin que les apostrophes puissent également être stockées à l'aide de commandes SQL.

11 Voir la base de données Exemple_ZoneCombinée_ListeChamps.odb associée à ce manuel.

Enregistrer les commandes SQL et les exécuter à la demande

Seules les requêtes peuvent être lancées via le module d'interrogation. Le code SQL qui a également un effet modificateur sur les données ne peut pas y être exécuté, sauf si le pilote de la base de données l'autorise, comme c'est le cas pour le pilote direct de MySQL. L'éditeur disponible via **Outils** → **SQL** permet l'exécution de code, mais n'offre pas de possibilité de stockage. Cela signifie que les commandes récurrentes doivent être stockées séparément quelque part et peuvent ensuite être copiées dans l'éditeur via le presse-papiers. La macro suivante fournit un remède[12].

Le code destiné à une opération telle que UPDATE, DELETE ou INSERT est stocké dans une table dont le premier champ contient la clé primaire **ID** et dont le second champ contient le code SQL. Un formulaire présente, dans un contrôle de table, l'ensemble des requêtes enregistrées. Après sélection de l'enregistrement souhaité la requête SQL qu'il contient est exécutée via un bouton.

```
Sub ChangeDonnees(oEvent As Object)
       Dim oConnexion As Object
       Dim oForm As Object
       Dim stSql As String
       Dim oSql_Instruction As Object
       Dim inRetourBouton As Integer
       oForm = oEvent.Source.Model.Parent
       oConnexion = oForm.activeConnection()
       stSQL = oForm.getString(2)'c'est le 2ème champ qui contient le code SQL.
       inRetourBouton = MsgBox("Exécuter le code SQL suivant ?" & CHR(13) & _
                        stSQL & CHR(13), 20, "Exécuter une requête SQL")
       If inRetourBouton = 6 Then
          oSQL_Instruction = oConnexion.createStatement()
             oSQL_Instruction.executeUpdate(stSql)
       End If
End Sub
```

Pour des raisons de sécurité, ce code SQL est d'abord affiché à nouveau dans une boîte de dialogue. Si vous ne confirmez pas par « Oui », le code n'est pas exécuté. Cette confirmation est déterminée par la valeur de retour de la boîte de dialogue (6 correspond à Oui ou la constante IDYES). Ensuite, la connexion pour la déclaration est d'abord établie, puis la déclaration est exécutée.

Pré-calculs et calculs dans un formulaire

Les remarques suivantes s'appliquent à tous les formulaires utilisés dans Base, pas seulement à celui figurant dans la base d'exemple sur laquelle s'appuie ce chapitre[13]

Les valeurs calculées, dans un formulaire, ne sont pas stockées dans la base de données. Elles ne sont utiles que pour leur visualisation à l'intérieur du formulaire, ou lors de l'impression de ce dernier.

Le ou les calculs peuvent être réalisés lors :

- de l'ouverture du formulaire lorsqu'il utilise des données provenant d'une requête SQL réalisant ce ou ces calculs. Il serait possible de produire le même effet à l'aide d'une

12 Voir la base de données Exemple_InsertUpdateDelete_SQL.odb associée à ce manuel.
13 Voir la base de données Exemple_Calcul_Direct_Formulaire.odb associée à ce manuel.

macro, déclenchée à l'ouverture du formulaire, mais cela est beaucoup plus complexe à mettre en œuvre ;
- de la saisie ou de la modification d'un champ numérique lorsqu'une procédure basic (macro) liée à ce champ est exécutée.

Les valeurs calculées peuvent être lues :

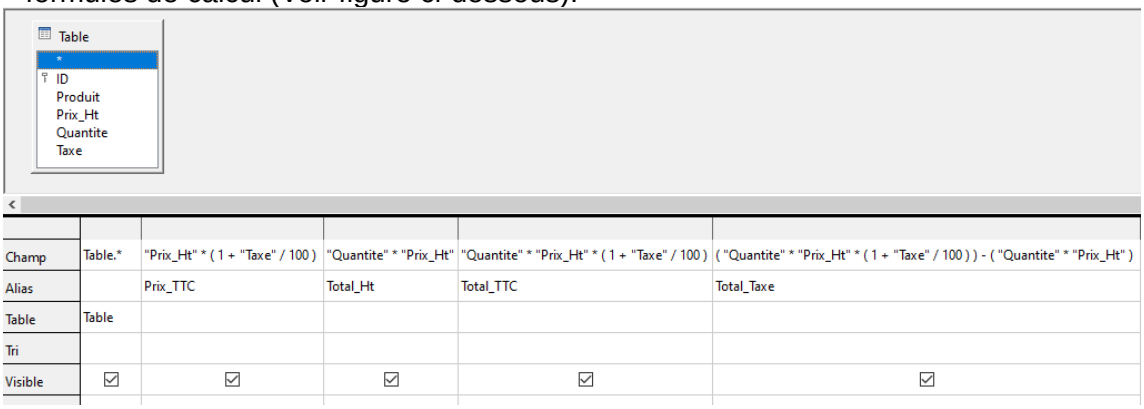
- immédiatement après l'ouverture du formulaire lorsqu'il se réfère à une requête SQL (cf. ci-dessus) ;
- dès qu'une valeur est modifiée dans un champ "numérique" grâce à une macro Basic.

Pré-calculs grâce à une requête SQL

Dans la base exemple il existe une requête SQL dénommée "Requête".

Elle peut être créée de deux manières :

- en mode ébauche, où après ajout de la table "Table", on définit les champs calculés et les formules de calcul (Voir figure ci-dessous).



- ou directement, de manière plus simple, en mode « Créer une requête en mode SQL ». La requête listée ci-après montre comment cela est possible.

```
SELECT "Table".*,
       "Prix_Ht" * ( 1 + "Taxe" / 100 ) AS "Prix_TTC",
       "Quantite" * "Prix_Ht" AS "Total_Ht",
       "Quantite" * "Prix_Ht" * ( 1 + "Taxe" / 100 ) AS "Total_TTC",
       ( "Quantite" * "Prix_Ht" * ( 1 + "Taxe" / 100 ) ) - ( "Quantite" *
"Prix_Ht" ) AS "Total_Taxe"
FROM "Table"
```

`SELECT "Table".*,`
`.....`
`FROM "Table"`
	On sélectionne tous les champs de la table "Table"

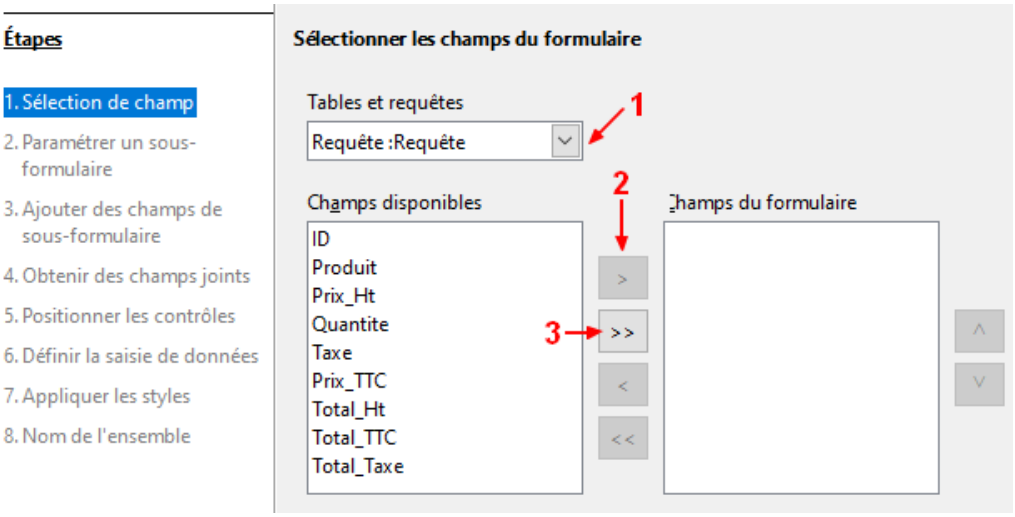
`"Prix_Ht" * ( 1 + "Taxe" / 100 ) AS "Prix_TTC",`	On calcule la valeur du prix TTC de l'article. Valeur qui est affecté à "Prix_TTC"

`"Quantite" * "Prix_Ht" AS "Total_Ht",`
`"Quantite" * "Prix_Ht" * ( 1 + "Taxe" / 100 ) AS "Total_TTC",`
`( "Quantite" * "Prix_Ht" * ( 1 + "Taxe" / 100 ) ) - ( "Quantite" * "Prix_Ht" ) AS "Total_Taxe"`
	On procède de même pour les calculs des :
	"Total_Ht", total hors taxe de l'article,
	"Total_TTC", total toute taxe comprise,
	"Total_Taxe", cout total de la taxe appliquée.

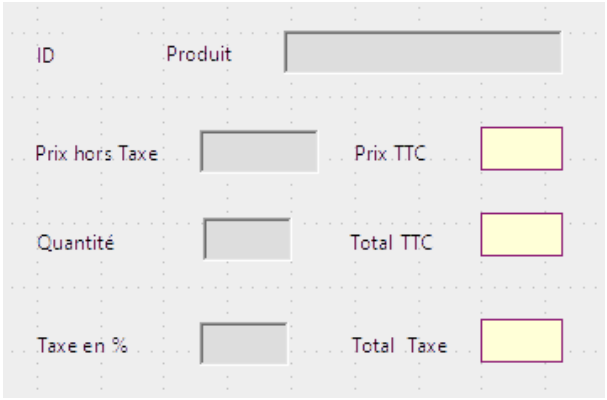
Lors de l'exécution de cette requête, on obtient le tableau suivant.

ID	Produit	Prix_Ht	Quantite	Taxe	Prix_TTC	Total_Ht	Total_TTC	Total_Taxe
1	Papier	3,24	2	20	3,89	6,48	7,78	1,3
2	Equerre	0,79	3	19	0,94	2,37	2,82	0,45
3	Dossier	0,7	24	20	0,84	16,8	20,16	3,36
5	Bloc	2,34	6	5,5	2,47	14,04	14,81	0,77

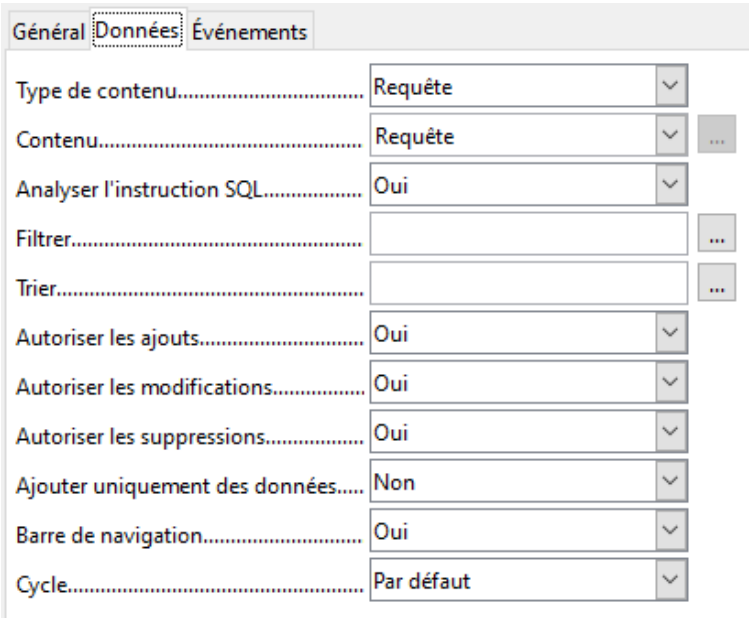
Ensuite un formulaire est créé. Le plus simple est d'utiliser "l'assistant pour créer un formulaire", qui permet de sélectionner la source de données (1), puis de transférer les champs disponibles, vers champs du formulaire à l'aide de divers boutons > , >>, <, << (2). Ici, on transfère tous les champs (3). Pour les étapes suivantes, se reporter au chapitre "Créer des formulaires" du guide Base "GB6404-Formulaires.odt").



On obtient, après mise en page, etc. un formulaire tel que celui-ci :



Afin qu'il affiche les valeurs résultantes de la requête il est nécessaire de modifier ses propriétés par le menu **Formulaire > Propriétés du Formulaire**, ou l'icône dans la barre d'outils ébauche de formulaire. Dans l'onglet « **Données** » on sélectionne : pour « **Type de contenu** » Requête, pour « **Contenu** » Requête, etc. comme sur la figure ci-dessous.



Ainsi, lors de l'ouverture du formulaire, positionné sur le premier enregistrement de la table, on pourra lire les champs calculés par la requête SQL.

Calculs directs dans un formulaire

Dans le même formulaire, il est possible de réaliser des calculs à l'aide de procédures Basic (Macros). Les deux macros suivantes montrent comment réaliser cela. Elles sont liées à la sortie de certains champs(évènement perte de focus). La compréhension des lignes de code est facilitée par les commentaires, mais sera développée par la suite.

```
Option Explicit
' ------------------Sub Calcul_Prix_TTC---------------------------
' Lancé par : évènement perte de focus des champs : Prix_Ht & Taxe
' Calcule le prix Toutes Taxes Comprises
' Champs pour le calcul : Prix_Ht (hors taxe) & Taxe
' Champ modifié : Prix_TTC (Toutes Taxes Comprises)
' Lance ensuite le calcul des totaux
' ------------------------------------------------------------------
Sub Calcul_Prix_TTC(oEvent As Object)
	' Déclaration des variables
	Dim oChamp As Object
	Dim oFormulaire As Object
	Dim oChamp1 As Object
	Dim oChamp2 As Object
	Dim oChamp3 As Object
	Dim dPrix_Ht As Double ‘ d : décimal
	Dim dTaxe As Double
	Dim dPrix_TTC As Double
	' Procédure principale
	oChamp = oEvent.Source.Model
	oFormulaire = oChamp.Parent
	' Récupère dans dPrix_Ht le montant du prix hors taxe du produit
	oChamp1 = oFormulaire.getByName("Prix_Ht")
	dPrix_Ht = oChamp1.getCurrentValue
	' Récupère dans dTaxe le montant de la Taxe (en france TVA)
	oChamp2 = oFormulaire.getByName("Taxe")
	dTaxe = oChamp2.getCurrentValue
	' Calcul du prix_TTC et affichage dans le champ Adhoc du formulaire
```

```
        dPrix_TTC = dPrix_Ht * (1 + dTaxe / 100)
        oChamp3 = oFormulaire.getByName("Prix_TTC")
        oChamp3.BoundField.UpdateDouble(dPrix_TTC)
        If Not IsEmpty(oFormulaire.getByName("Quantite").getCurrentValue()) Then
           Calcul_Totaux(oEvent)
        End If
End Sub


'---------------Sub Calcul_Totaux-------------------------------
' Lancé par : - Calcul_Prix_TTC ;
'             - événement perte de focus du champ : Quantite
' Calcule le Total_TTC & le Total_Taxe
' Champs pour le calcul : Prix_TTC, Quantite, Total_TTC, Prix_Ht
' Champs modifiés : Total_TTC & Total_Taxe
'---------------------------------------------------------------
Sub Calcul_Totaux(oEvent As Object)
        Dim oChamp As Object
        Dim oFormulaire As Object
        Dim oChamp1 As Object
        Dim oChamp2 As Object
        Dim oChamp3 As Object
        Dim oChamp4 As Object
        Dim oChamp5 As Object
        Dim dPrix_TTC As Double
        Dim iQuantite As Integer
        Dim dPrix_Ht As Double
        Dim dTotal_TTC As Double
        Dim dTotal_Taxe As Double
        oChamp = oEvent.Source.Model
        oFormulaire = oChamp.Parent
        oChamp1 = oFormulaire.getByName("Prix_TTC")
        dPrix_TTC = oChamp1.getCurrentValue
        oChamp2 = oFormulaire.getByName("Quantite")
        iQuantite = oChamp2.getCurrentValue
        oChamp3 = oFormulaire.getByName("Prix_Ht")
        dPrix_Ht = oChamp3.getCurrentValue
        oChamp4 = oFormulaire.getByName("Total_TTC")
        oChamp4.BoundField.UpdateDouble(dPrix_TTC * iQuantite)
        oChamp5 = oFormulaire.getByName("Total_Taxe")
        oChamp5.BoundField.UpdateDouble((dPrix_TTC * iQuantite) - (dPrix_Ht *
iQuantite))
End Sub
```

Pour que ces macros fonctionnent automatiquement, il faut paramétrer les propriétés des champs du formulaire "Prix_Ht", "Quantite", et "Taxe". Pour cela, en mode édition, on affectera une des macros à l'évènement "Perte de Focus" de ces champs (Voir "Évènement dans un formulaire" du présent guide).

La macro Calcul_Prix_TTC sera affectée aux pertes de focus des champs : Prix_Ht et Taxe. La seconde, Calcul_Totaux le sera, pour le même type d'évènement, pour le champ "Quantite".

Ainsi, si une valeur est saisie, modifiée, supprimée dans les champs précités, la macro liée à la sortie (Perte de Focus) de ces champs est exécutée et modifie les valeurs des champs calculés (Prix_TTC, Total_TTC, et Total_Taxe).

Par exemple, le champ Prix_TTC est mis à jour suite à l'exécution de l'instruction :

```
        oChamp3.BoundField.UpdateDouble(dPrix_TTC)
```

UpdateDouble est une méthode de **BoundField** (littéralement champ lié), elle-même propriété de l'objet oChamp3.

On peut traduire cette instruction, en langage courant, par :

Mets à jour au format double (**UptdateDouble**), le champ lié (**BoundField**) à l'objet oChamp3, c'est-à-dire le champ "Prix_TTC", avec la valeur calculée dPrix_TTC.

La liaison entre oChamp3 et "Prix_TTC" étant réalisée par l'instruction :

```
oChamp3 = oFormulaire.getByName("Prix_TTC")
```

Si le champ "Quantite" contient une valeur (`Not IsEmpty` - n'est pas vide)

```
If Not IsEmpty(oFormulaire.getByName("Quantite").getCurrentValue()) Then
	Calcul_Totaux(oEvent)
End If
```

des calculs supplémentaires sont effectués pour les champs qui lui sont liés, à savoir les totaux. Cela est fait en lançant la seconde macro "Calcul_Totaux".

Cette seconde procédure n'appelle aucun commentaire, car les instructions qui la composent sont similaires à celles de la première.

Enfin, il est possible de réaliser la même chose avec un formulaire ne comportant qu'un champ unique de Contrôle de Table. Voir pour cela dans la base exemple, le formulaire Form_Controle_Table.

Concernant l'initialisation des deux variables object, au début des macros,

```
oChamp = oEvent.Source.Model
oFormulaire = oChamp.Parent
```

se référer aux deux derniers paragraphes du sous-chapitre "Accéder aux formulaires" de ce guide.

Fournir la version actuelle de LibreOffice

LibreOffice version 4.1 a apporté des modifications aux champs de liste et aux valeurs de date qui rendent nécessaire la détermination de la version actuelle lors de l'exécution de macros dans ces zones. Le code suivant sert cet objectif :

```
Function OfficeVersion()
	Dim aSettings, aConfigProvider
	Dim aParams2(0) As New com.sun.star.beans.PropertyValue
	Dim sProvider$, sAccess$
	sProvider = "com.sun.star.configuration.ConfigurationProvider"
	sAccess = "com.sun.star.configuration.ConfigurationAccess"
	aConfigProvider = createUnoService(sProvider)
	aParams2(0).Name = "nodepath"
	aParams2(0).Value = "/org.openoffice.Setup/Product"
	aSettings = aConfigProvider.createInstanceWithArguments(sAccess, aParams2())
	OfficeVersion() = Array(aSettings.ooName, aSettings.ooSetupVersionAboutBox)
End Function
```

Cette fonction renvoie un tableau dans lequel le premier élément est "LibreOffice" et le second est le numéro de version complet, par exemple 6.4.6.2.

Attention

On ne peut pas récupérer directement la valeur de la fonction. Il faut d'abord l'affecter à une variable tableau.

```
Dim aVersion()
aVersion = OfficeVersion() ' on peut ensuite utiliser les deux valeurs
```

Renvoyer la valeur des zones de liste

Depuis LibreOffice 4.1, la valeur renvoyée par une zone de liste à la base de données est stockée dans *CurrentValue*. Ce n'était pas le cas dans les versions précédentes, ni dans OpenOffice ou Apache OpenOffice. La fonction suivante fera le calcul. La version de LibreOffice doit être vérifiée pour voir si elle est postérieure à LibreOffice 4.0.

```
Function ID_Determination(oChamp As Object) As Integer
    a() = OfficeVersion()
    If a(0) = "LibreOffice" And (LEFT(a(1),1) = 4 And RIGHT(LEFT(a(1),3),1) > 0) Or
        LEFT(a(1),1) > 4 Then
        stContenu = oChamp.currentValue
    Else
```

Avant LibreOffice 4.1, la valeur transmise était lue dans la liste de valeurs de la zone de liste. L'enregistrement visiblement choisi est SelectedItems (0). "0" car plusieurs valeurs supplémentaires peuvent être sélectionnées dans une zone de liste.

```
        stContenu = oChamp.ValueItemList(oChamp.SelectedItems(0))
    End If
    If IsEmpty(stContenu) Then
```

- 1 est une valeur qui n'est pas utilisée comme valeur automatique et qui n'existera donc pas dans la plupart des tables en tant que clé étrangère.

```
        ID_Determination = -1
    Else
        ID_Determination = Cint(stContenu)
```

Converti en entier

```
    End If
End Function
```

La fonction transmet la valeur sous forme d'entier. La plupart des clés primaires incrémentent automatiquement des entiers. Lorsqu'une clé étrangère ne satisfait pas ce critère, la valeur de retour doit être ajustée au type approprié.

La valeur affichée d'un champ de liste peut être autrement déterminée à l'aide de la propriété d'affichage du champ.

```
Sub ListeChampAffiche
    Dim oDoc As Object
    Dim oFormulaire As Object
    Dim oZoneListe As Object
    Dim oController As Object
    Dim oVue As Object
    oDoc = thisComponent
    oFormulaire = oDoc.Drawpage.Forms(0)
```

```
        oZoneListe = oFormulaire.getByName("ZoneListe")
        oController = oDoc.getCurrentController()
        oVue = oController.getControl(oZoneListe)
        print "Contenu affiché : " & oVue.SelectedItem
End Sub
```

Le contrôleur est utilisé pour accéder à la vue du formulaire. Cela détermine ce qui apparaît dans l'interface visuelle. La valeur sélectionnée est **SelectedItem**.

Limiter les listes de sélection en saisissant les premières lettres

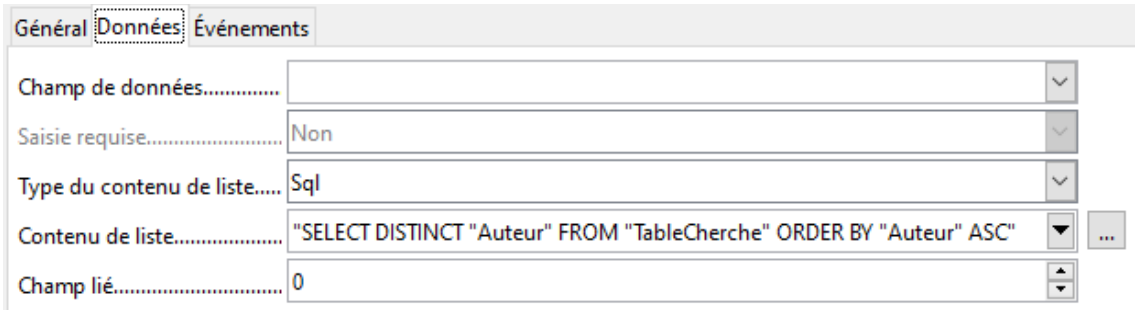
Il peut parfois arriver que le contenu des listes de sélection devienne interminable et trop volumineux à gérer. Dans de tels cas, il est nécessaire de positionner la liste déroulante sur le premier champ correspondant à des critères saisis par l'utilisateur, ou de limiter le contenu d'une zone de liste avec ces mêmes critères[14]. Ceci est réalisé grâce aux procédures décrites par la suite.

Une première solution est utilisée dans le formulaire « Filtre_Macro_Tableau_Direct », qui comporte deux contrôles, une zone de liste (liste déroulante) et un contrôle de table.

Pour la première, dans ses **Propriétés**, onglet **Données**, une requête SQL est nécessaire.

```
SELECT DISTINCT "Auteur" FROM "TableCherche" ORDER BY "Auteur" ASC
```

L'ensemble du paramétrage de la zone de liste figure sur l'illustration suivante :



Champ Lié est défini sur « 0 », car il n'y a qu'une seule colonne dans la requête (Auteur).

La macro (Standard.Module1.Filtre_direct (document, Basic)) ci-après est liée à **Événements – Modifié(es).** La chaîne suivante **Auteur = ''** est lié aux **Propriétés** du formulaire, onglet **Données,** rubrique **Filtre.**

```
Sub Filtre_direct(oEvent AS Object)
       ' Le filtre du formulaire est défini directement, il peut être
       ' activé et désactivé dans la barre de navigation
       Dim oFormulaire AS Object
       Dim oChamp AS Object
       oChamp = oEvent.Source.Model
       oFormulaire = oChamp.Parent
       stListeValeurs = oChamp.getCurrentValue()
       If stListeValeurs = "" Then
          oFormulaire.Filter = ""
       Else
          oFormulaire.Filter = "Auteur = '" + stListeValeurs + "'"
       End If
       oFormulaire.reload()
```

14 Pour l'utilisation de cette macro, voir l'exemple Exemple_Cherche_et_Filtre, associée à ce manuel.

```
End Sub
```

Après l'initialisation des variables oChamp et oFormulaire, stListeValeurs reçoit la valeur courante de oChamp, c'est-à-dire le ou les caractères saisis par l'utilisateur. Ensuite, le champ « Auteur » est filtré à l'aide de cette chaîne de caractère, puis le formulaire est rechargé (reload()).

Après cela le pointeur dans la liste de sélection se place automatiquement sur le premier « Auteur » commençant par les caractères saisis. Exemple : **Ber** amène l'affichage sur **Beran, Armgard**, et le contrôle de table n'affiche que les enregistrements concernant cet auteur. Le choix d'un auteur différent, dans la liste de sélection, modifie l'affichage dans le contrôle de table.

La seconde solution (formulaire : Limite_Liste) proposée ci-dessous passe par la saisie d'un critère de recherche d'**Auteur**, toujours par la saisie de quelques caractères dans une « zone de texte » indépendante. Ces caractères pouvant être situés au début, dans, ou à la fin de la chaîne de caractères. Ces choix étant possibles à l'aide de trois boutons radio liés entre eux en un groupe dénommé « Choix » (**Propriétés – Général – Nom du groupe**).

Les propriétés du formulaire et des divers contrôles pourront aisément être vues en ouvrant le formulaire en mode édition.

La première procédure est lancée par l'appui sur la touche TAB ou sur la touche Entrée, après la saisie de quelques caractères. Le commentaire se suffit à lui-même.

```
Sub Lanceur (oEvent As Object)
'-------------------------------------------------------------
' Verifie si le KeyCode de la touche enfoncée est
' 1280 - Entrée ou 1282 - Tabulation
'-------------------------------------------------------------
      If oEvent.KeyCode = 1280 or oEvent.KeyCode = 1282 Then
         Limiter (oEvent)
      Else
         Exit Sub
      End If
End Sub
```

Si l'une des touches attendue est actionnée la macro « Limiter » est déclenchée avec oEvent en paramètre, sinon on sort de la macro (Exit Sub). La liste complète des « Key_Code » peut être obtenue dans l'API : **com::sun::star::awt::Key Constant Group Reference**

En premier lieu de nombreuses variables sont déclarées avec leurs types respectifs.

```
Sub Limiter (oEvent As Object)
      Dim oDrawpage As Object
      Dim oFormulaire As Object
      Dim oZone_Texte As Object
      Dim oZone_Liste As Object
      Dim oChamp As Object
      Dim oRadio As Object
      Dim oRadio1 As Object
      Dim oRadio2 As Object
      Dim oRadio3 As Object
      Dim oSourceDonnees As Object
      Dim oConnexion As Object
      Dim oCommande_Sql As Object
      Dim oResultat As Object
      Dim stCritere As String
      Dim stRequete As String
```

```
Dim arContenu()
Dim i As Integer
' - Initialise les zones de texte et de liste à "" (vide)
oDrawpage = thisComponent.drawpage
oFormulaire = oDrawpage.forms.getByName("Formulaire")
oZone_Texte = oFormulaire.getByName("Critere")
oZone_Texte.DefaultText = ""
```

Ici le texte par défaut de « oZone_Texte » est fixé à vide(""), il s'agit du texte par défaut contenu dans la zone de texte, nommée « Critere », dédié à la saisie de caractères par l'utilisateur.

```
oZone_Liste = oFormulaire.getByName("Liste")
oZone_Liste.StringItemList = arContenu()
```

Là, pour le contrôle de liste déroulante, nommée « Liste » c'est également un contenu vide qui est attribué. Vide car « arContenu » a été déclaré sans qu'aucune valeur ne lui soit affecté.

```
' - Récupère les caractères entrés comme critère
oChamp = oEvent.Source.Model
stCritere = oChamp.getCurrentValue()
' - Récupère l'état des boutons radio pour connaître celui qui est activé
oRadio = oChamp.Parent
oRadio1 =oRadio.getByName("Bouton radio 1")
oRadio2 =oRadio.getByName("Bouton radio 2")
oRadio3 =oRadio.getByName("Bouton radio 3")
```

Les commentaires pour les lignes au-dessus sont suffisants à la compréhension.

```
If stCritere <>"" Then
   If oRadio1.State = 1 Then           ' - Critère au début - bouton 1
      stCritere = stCritere & "%"
   ElseIf oRadio2.State = 1 Then       ' - Critère au milieu - bouton 2
      stCritere = "%" & stCritere & "%"
   ElseIf oRadio3.State = 1 Then       ' - Critère à la fin - bouton 3
      stCritere = "%" & stCritere
   End If
Else
   MsgBox ("Vous n'avez saisi aucun critère")
End If
```

Si « stCritere » n'est pas vide alors, suivant l'état (State) des boutons radio (activé = 1 – non activé = 0), on modifie cette chaîne de caractère pour qu'elle soit ensuite utilisée dans la requête SQL (interne à la procédure) suivante.

Pour rappel le symbole « **%** » remplace un ou plusieurs caractères dans une chaîne utilisée avec un LIKE.

Ainsi :

- xyz% = toute chaîne commençant par "xyz" ;
- %xyz% = toute chaîne dans laquelle se trouve"xyz" ;
- %xyz = toute chaîne se terminant par "xyz".

```
' - Construction de la Requête
stRequete = "SELECT DISTINCT ""Auteur"" FROM ""TableCherche"" WHERE _
             ""Auteur"" LIKE '" & stCritere & "' ORDER BY ""Auteur"" ASC"
' - Connexion à la source de donnée si ce n'est pas le cas
oSourceDonnees = ThisComponent.Parent.CurrentController
```

```
        If Not (oSourceDonnees.isConnected()) Then
            oSourceDonnees.connect()
        End IF
        oConnexion = ThisComponent.Parent.CurrentController.ActiveConnection()
        ' - Exécution de la requête SQL
        oCommande_Sql = oConnexion.createStatement()
        oResultat = oCommande_Sql.executeQuery(stRequete)
```

Sans commentaires.

```
        ' Crée un tableau 'arContenu' avec l'ensemble des données résultant de la
requête
        i = 0
        While oResultat.next
            ReDim Preserve arContenu(i)
            arContenu(i)=oResultat.getString(1)
            i=i+1
        Wend
        oZone_Liste.StringItemList = arContenu()
End Sub
```

Après la boucle de lecture des résultats de la requête et leur affectation à la variable tableau arContenu() on remplit la liste déroulante avec ces résultats. Ainsi cette zone de liste sera réduite aux seuls champs « Auteur » correspondants aux critères saisis.

Lors de la sélection d'un « Auteur » dans la liste déroulante, la même macro que précédemment (`Filtre_direct(oEvent AS Object)`) est utilisée.

Il est possible de procéder à plusieurs recherches les unes après les autres.

```
Sub Ferme_Formulaire_Sans_Sauvegarder(oEvent As Object)
        oEvent.Source.Model.Parent.isModified = False
        ThisDatabaseDocument.FormDocuments.getByName("Limite_Liste").close
End Sub
```

Enfin, pour quitter le formulaire la macro précédente est liée au bouton « Quitter la recherche » (**Propriétés – Événement – Exécuter l'action**) ainsi qu'au formulaire lui-même (**Propriétés – Événement – Lors du déchargement**). En effet, il n'est pas souhaitable ici d'enregistrer les éléments contenus dans les divers contrôles.

Conversion de dates d'un formulaire en une variable de date

```
Function ValeurDate(oChamp As Object) As Date
        a() = OfficeVersion()
        If a(0) = "LibreOffice" And (LEFT(a(1),1) =4 And RIGHT(LEFT(a(1),3),1) > 0) _
            Or LEFT(a(1),1) > 4 Then
```

Ici, toutes les versions de LibreOffice à partir de la 4.1 sont interceptées. À cette fin, le numéro de version est divisé en ses éléments individuels, et les numéros de version majeure et mineure sont vérifiés. Cela fonctionnera jusqu'à LibreOffice 9.

```
            Dim stAn As String
            Dim stMois As String
            Dim stJour As String
            stAn = Right(Ltrim(Str(oChamp.CurrentValue.Year)), 4)
            stMois = Right("0" & Lrim(Str(oChamp.CurrentValue.Month)),2)
            stJour = Right("0" & Lrim(Str(oChamp.CurrentValue.Day)),2)
```

```
            ValeurDate = CDateFromIso(stAn & stMois & stJour)
        Else
            ValeurDate = CDateFromIso(oChamp.CurrentValue)
        End If
End Function
```

Depuis LibreOffice 4.1.2, les dates sont stockées sous forme de tableaux dans les contrôles de formulaire. Cela signifie que la valeur actuelle du contrôle ne peut pas être utilisée pour accéder à la date elle-même. La date doit être recréée à partir du jour, du mois et de l'année si elle doit être utilisée ultérieurement dans les macros[15].

Recherche d'enregistrements de données

Dans une base de données vous pouvez rechercher tous les enregistrements qui contiennent, dans un de leurs champs, une donnée (numérique, chaîne de caractère) sans utiliser de macro. Cependant, la conception d'une requête SQL sera très complexe. Une macro peut résoudre ce problème grâce aux boucles **While**… **Wend** et **For… Next**[16].

Lancée par une procédure principale, qui lui transmet le nom de la table à parcourir en tant que variable, la sous-procédure suivante réalise la recherche.

Pour ce faire elle :

- lit le critère de recherche dans le formulaire ;
- vérifie que la connexion à la base de donnée est bien établie ;
- exécute en interne une requête SQL ;
- parcourt tous les champs de la table ;
- écrit finalement une liste de numéros, correspondant aux clés primaires des enregistrements de la table qui sont extraits par le terme de recherche, dans une table temporaire.

Cette table Cherchetmp, se compose d'une clé primaire à incrémentation automatique (ID) et d'un champ appelé (Nr). Il contient toutes les clés primaires extraites de la table dans laquelle la recherche est effectuée.

Pour obtenir un résultat correct, la table doit contenir le contenu que vous recherchez sous forme de texte et non de clé étrangère.

S'il n'y a pas qu'une seule table, dans laquelle effectuer la recherche, le contenu des tables doit être joint dans une VUE que la macro utilisera.

```
Sub Rechercher(stTable As String)
        Dim oSourceDonnees As Object
        Dim oConnexion As Object
        Dim oSQL_Commande As Object
        Dim stSql As String
        Dim oResultat As Object
        Dim oDoc As Object
        Dim oDrawpage As Object
        Dim oFormulaire As Object
        Dim oFormulaire2 As Object
        Dim oChamp As Object
        Dim stContenu As String
        Dim arContenu() As String
```

15 Voir la base de données Exemple_DateFormulaire_en_Variable.odb
16 Voir la base de données Exemple_Cherche_et_Filtre.odb, Formulaire RequeteCherche, associée à ce manuel.

```
Dim inI As Integer
Dim inK As Integer
oDoc = thisComponent
oDrawpage = oDoc.drawpage
oFormulaire = oDrawpage.forms.getByName("FormulaireRecherche")
oChamp = oFormulaire.getByName("TexteCherche")
stContenu = oChamp.getCurrentValue()
stContenu = LCase(stContenu)
```

Le contenu (`stContenu`) du champ de texte de recherche (`"TexteCherche"`) est converti en minuscules (`Lcase`) de sorte que la fonction de recherche suivante n'a besoin que de comparer les écritures en minuscules. La connexion à la source de données est vérifiée et si besoin établie.

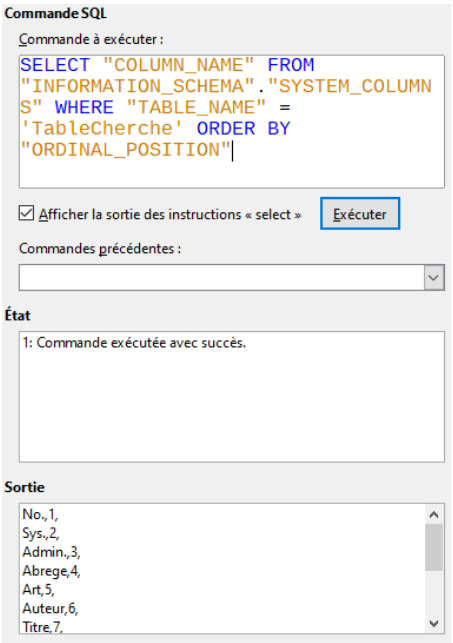
```
oSourceDonnees = ThisComponent.Parent.DataSource
oConnexion = oSourceDonnees.GetConnection("","")
oSQL_Commande = oConnexion.createStatement()
```

Il faut d'abord déterminer si un terme de recherche a été saisi. Si le champ est vide, on supposera qu'aucune recherche n'est requise. Tous les enregistrements seront affichés.

Si un terme de recherche a été saisi, les noms de colonne sont lus à partir de la table recherchée, afin que la requête puisse accéder aux champs.

```
If stContenu <> "" Then
	stSql = "SELECT ""COLUMN_NAME"" FROM ""INFORMATION_SCHEMA"".""SYSTEM_COLUMNS""
WHERE ""TABLE_NAME"" = '" + stTable + "' ORDER BY ""ORDINAL_POSITION"""
	oResultat = oSQL_Commande.executeQuery(stSql)
```

Dans les bases de données relationnelles, le schéma d'information (`INFORMATION_SCHEMA`) est un ensemble de vues, en lecture seule, également appelées tables système. Cet ensemble fournit des informations sur toutes les tables, vues, colonnes et procédures d'une base de données. Ici la vue utilisée est `SYSTEM_COLUMNS`, qui permet d'obtenir tous les noms des colonnes de la table `stTable`. Il est possible de visualiser le résultat de cette requête par **OUTILS > SQL…** comme montré sur l'illustration suivante :



Note

Les formules SQL dans les macros doivent d'abord être placées entre guillemets comme les chaînes de caractères normales. Les noms de champ et les noms de table sont déjà entre guillemets dans la formule SQL. Pour créer un code final qui transmet correctement les guillemets doubles, les noms de champ et les noms de table doivent recevoir deux ensembles de ces guillemets.

`stSql = "SELECT ""Nom"" FROM ""Table"";"`, devient lorsqu'il est affiché avec la commande `MsgBox stSql` : SELECT "Nom" FROM "Table";

L'index du tableau dans lequel les noms de champs sont écrits est initialement défini sur 0. Ensuite, la requête commence à être lue. La taille du tableau étant inconnue, elle doit être ajustée en continu. C'est pourquoi la boucle commence par '**ReDim Preserve arContenu (inI)**' pour définir la taille du tableau et en même temps conserver son contenu existant. Ensuite, les champs sont lus et l'index du tableau est incrémenté de 1. Puis, le tableau est de nouveau dimensionné et une autre valeur peut être stockée.

```
        InI = 0
        While oResultat.next
            ReDim Preserve arContenu(inI)
            arContenu(inI) = oResultat.getString(1)
            inI = inI + 1
        Wend
        stSql = "DROP TABLE ""Cherchetmp"" IF EXISTS"
            oSQL_Commande.executeUpdate (stSql)
```

La requête est maintenant regroupée dans une boucle et ensuite appliquée à la table définie au début. Toutes les combinaisons de casse sont autorisées, car le contenu du champ de la requête est converti en minuscules.

La requête est construite de telle sorte que les résultats aboutissent dans la table « Cherchetmp ». On suppose que la clé primaire est le premier champ de la table (arContenu (0)).

```
        stSql = "SELECT """+arContenu(0)+""" INTO ""Cherchetmp"" FROM """ + stTable _
         + """ WHERE "
        For inK = 0 To (inI - 1)
            stSql = stSql+"LCase("""+arContenu(inK)+""") LIKE '%"+stContenu+"%'"
            If inK < (inI - 1) Then
                stSql = stSql+" OR "
            End If
        Next
        oSQL_Commande.executeQuery(stSql)
    Else
        stSql = "DELETE FROM ""Cherchetmp"""
        oSQL_Commande.executeUpdate (stSql)
    End If
```

Le formulaire d'affichage doit être rechargé. Sa source de données est une requête (dans cet exemple RequeteCherche).

```
        oFormulaire2 = oDrawpage.forms.getByName("Affiche")
        oFormulaire2.reload()
    End Sub
```

Dans la mesure du possible, cette requête doit être construite de manière à pouvoir être modifiée ultérieurement.

```
SELECT * FROM "TableCherche" WHERE "No." IN (SELECT "No." FROM "Cherchetmp") OR
"No." = CASE WHEN (SELECT COUNT("No.") FROM "Cherchetmp") > 0 THEN '0' ELSE "No."
END
```

Tous les éléments de la **TableCherche** sont inclus, y compris la clé primaire. Aucune autre table n'apparaît dans la requête directe. Par conséquent, aucune clé primaire d'une autre table n'est nécessaire et le résultat de la requête reste modifiable.

La clé primaire est enregistrée dans cet exemple sous le nom « **No**. ». La macro lit précisément ce champ. Il y a une vérification initiale pour voir si le contenu du champ « **No**. » apparaît dans la table Cherchetmp. L'opérateur **IN** est compatible avec plusieurs valeurs. La sous-requête peut également générer plusieurs enregistrements.

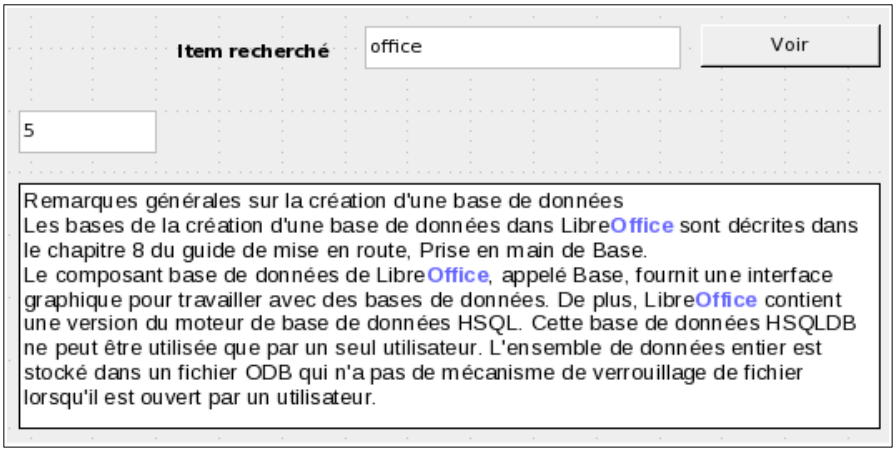
Pour de plus grandes quantités de données, la correspondance des valeurs à l'aide de l'opérateur **IN** ralentit rapidement. Par conséquent, il n'est pas judicieux d'utiliser un champ de recherche vide simplement pour transférer tous les champs de clé primaire de la table de recherche dans la table **Cherchetmp**, puis d'afficher les données de la même manière. Au lieu de cela, un champ de recherche vide crée une table **Cherchetmp** vide, de sorte qu'aucun enregistrement n'est disponible. C'est le but de la seconde moitié de la condition :

```
OR "No." = CASE WHEN (SELECT COUNT("No.") FROM "Cherchetmp") > 0 THEN '-1'
ELSE "No." END
```

Si un enregistrement est trouvé dans la table Cherchetmp, cela signifie que le résultat de la première requête est supérieur à 0. Dans ce cas : **"No." = '-1'** (ici nous avons besoin d'un nombre qui ne peut pas apparaître comme clé primaire, donc **'-1'** est une bonne valeur). Si la requête renvoie précisément 0 (ce qui sera le cas si aucun enregistrement n'est présent), alors **"No." = "No."**. Cela listera tous les enregistrements qui ont un **"No."**. Comme **No.** est la clé primaire, cela signifie tous les enregistrements.

Recherche dans les formulaires et mise en évidence des résultats

Avec un grand champ de texte, il est souvent difficile de savoir où les correspondances à un terme de recherche se produisent. Ce serait bien si le formulaire pouvait mettre en évidence les correspondances. Cela devrait ressembler à ce que montre l'illustration suivante :



Pour qu'un formulaire fonctionne de cette manière, nous avons besoin de quelques éléments supplémentaires dans notre boîte d'astuces.[17]

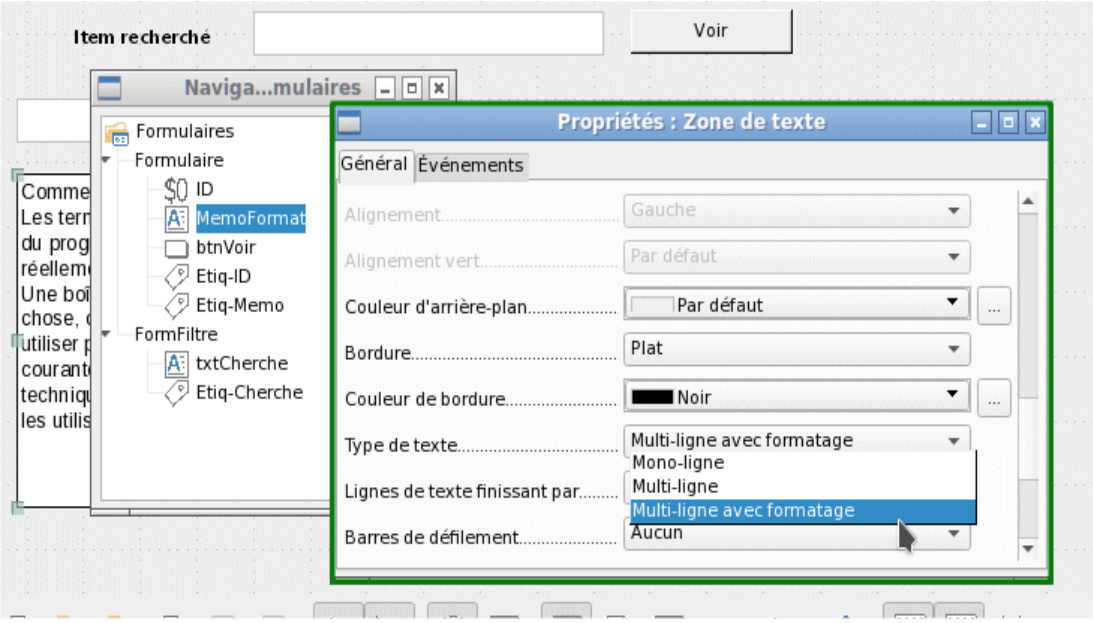
17 Voir la base de données Exemple_Autotexte_Recherche_Orthographe.odb associée à ce manuel.

Le fonctionnement d'un champ de recherche comme celui-ci a déjà été expliqué. Une table de filtres est créée et un formulaire est utilisé pour écrire les valeurs actuelles d'un seul enregistrement dans cette table. Le formulaire principal est fourni avec son contenu à l'aide d'une requête qui ressemble à ceci :

```
SELECT "ID", "Memo" FROM "Table" WHERE LOWER ("Memo") LIKE '%' || LOWER (
(SELECT "TexteCherche" FROM "Filtre" WHERE "ID" = TRUE)) || '%'
```

Lorsque le texte de recherche est entré, tous les enregistrements de la table « Table » qui ont le texte de recherche dans le champ « Memo » sont affichés. La recherche n'est pas sensible à la casse.

Si aucun texte de recherche n'est entré, tous les enregistrements du tableau sont affichés. La clé primaire de cette table étant incluse dans la requête, cette dernière peut être éditée.



Dans le formulaire, en plus du champ ID de la clé primaire, il y a un champ appelé MemoFormat qui a été configuré (en utilisant **Propriétés> Général> Type de texte> Multi-lignes avec formatage)** pour afficher le texte en couleur et en noir. Un examen attentif des propriétés du champ de texte révèle que l'onglet Données a maintenant disparu. En effet, les données ne peuvent pas être saisies dans un champ qui a une mise en forme supplémentaire que la base de données elle-même ne peut pas stocker. Néanmoins, il est toujours possible de mettre du texte dans ce champ, de le baliser, et de le transférer après une mise à jour en utilisant une macro.

La procédure *LireContenu* sert à transférer le contenu du champ de base de données « Memo » dans le champ de texte formaté MemoFormat, et à le formater de sorte que tout texte correspondant à celui du champ de recherche soit mis en évidence.

La procédure est liée à **Formulaire> Evénements> Après le changement d'enregistrement.**

```
Sub LireContenu(oEvent As Object)
	Dim inMemo As Integer
	Dim oChamp As Object
	Dim stChercheTexte As String
	Dim oCurseur As Object
	Dim inCherche As Integer
	Dim inChercheVieux As Integer
```

```
Dim inLongueur As Integer
oFormulaire = oEvent.Source
inMemo = oFormulaire.findColumn("Memo")
oChamp = oFormulaire.getByName("MemoFormat")
   oChamp.Text = oFormulaire.getString(inMemo)
```

Les variables sont d'abord définies. Ensuite, le champ de table « Memo » est recherché dans le formulaire et la fonction **getString()** est utilisée pour lire le texte de la colonne numérotée. Celui-ci est transféré dans le champ qui peut être formaté mais qui n'a pas de lien vers la base de données : MemoFormat.

Les tests initiaux ont montré que le formulaire s'ouvrait mais que la barre d'outils du formulaire en bas n'était plus créée. Par conséquent, une très courte attente de 5/1000 secondes a été intégrée. Après cela, le contenu affiché est lu à partir du FormFiltre (qui est parallèle au formulaire dans la hiérarchie des formulaires).

```
Wait 5
   stChercheTexte = oFormulaire.Parent.getByName("FormFiltre").
         getByName("txCherche").Text
```

Pour pouvoir mettre en forme du texte, un **TextCursor** (invisible) doit être créé dans le champ qui contient le texte. L'affichage par défaut du texte utilise une police serif de 12 points qui peut ne pas apparaître dans d'autres parties du formulaire et ne peut pas être directement personnalisée à l'aide des propriétés de contrôle de formulaire. Dans cette procédure, le texte est défini sur l'apparence souhaitée dès le début. Si cela n'est pas fait, les différences de formatage peuvent entraîner la coupure de la limite supérieure du texte dans le champ. Lors des premiers tests, seuls 2/3 de la première ligne étaient lisibles.

Pour que le curseur invisible marque le texte, il est défini initialement au début du champ, puis à la fin. L'argument dans les deux cas est **True**. Viennent ensuite les spécifications concernant la taille de la police, le type de la police, la couleur et la taille. Ensuite, le curseur revient au début.

```
oCurseur = oChamp.createTextCursor()
oCurseur.gotoStart(true)
oCurseur.gotoEnd(true)
oCurseur. CharHeight = 10
oCurseur. CharFontName = "Arial, Helvetica, Tahoma"
oCurseur. CharColor = RGB(0,0,0)
oCurseur. CharWeight = 100.000000      'com:: sun:: star:: awt:: FontWeight
oCurseur.gotoStart(false)
```

S'il y a du texte dans le champ et qu'une entrée a été faite demandant une recherche, ce texte est maintenant recherché pour trouver la chaîne de recherche. La boucle externe demande d'abord si ces conditions sont remplies ; celle de l'intérieur établit si la chaîne de recherche est vraiment dans le texte du champ MemoFormat. Ces paramètres peuvent en fait être omis, car la requête sur laquelle le formulaire est basé n'affiche que le texte qui remplit ces conditions.

```
If oChamp. Text <> "" And stChercheTexte <> "" Then
   If inStr(oChamp.Text, stChercheTexte) Then
      inCherche = 1
      inChercheVieux = 0
         inLen = Len(stChercheTexte)
```

Le texte est parcouru pour la chaîne de recherche. Cela se déroule dans une boucle qui se termine lorsqu'aucune autre correspondance n'est affichée. **InStr()** renvoie l'emplacement du premier caractère de la chaîne de recherche dans le format d'affichage spécifié, indépendamment

de la casse. La boucle est contrôlée par l'exigence qu'à la fin de chaque cycle, le début de **inCherche** soit incrémenté de 1 (-1 dans la première ligne de la boucle et +2 dans la dernière ligne). Pour chaque cycle, le curseur est déplacé vers la position initiale sans marquage à l'aide de **oCurseur.goRight (Position, false)**, puis vers la droite avec un marquage par la longueur de la chaîne de recherche. Ensuite, le formatage souhaité (bleu et un peu plus gros) est appliqué et le curseur est ramené à son point de départ suivant pour la prochaine exécution.

```
                Do While inStr(inCherche, oChamp.Text, stChercheTexte) > 0
                    inCherche = inStr(inCherche, oChamp.Text, stChercheTexte) - 1
                    oCurseur.goRight(inCherche-inChercheVieux, false)
                    oCurseur.goRight(inLen, true)
                    oCurseur.CharColor = RGB(102,102,255)
                    oCurseur.CharWeight = 110.000000
                    oCurseur.goLeft(inLen, false)
                    inChercheVieux = inCherche
                    inCherche = inCherche + 2
                Loop
            End If
        End If
End Sub
```

La procédure **EcrireContenu** sert à transférer le contenu du champ de texte formatable MemoFormat dans la base de données. Cela se déroule indépendamment du fait qu'une modification ait lieu ou non.

La procédure est liée à **Formulaire> Événements> Avant le changement d'enregistrement**.

```
Sub EcrireContenu(oEvent As Object)
    Dim oFormulaire As Object
    Dim inMemo As Integer
    Dim loID As Long
    Dim oChamp As Object
    Dim stMemo As String
    oFormulaire = oEvent.Source
        If InStr(oFormulaire.ImplementationName, "ODatabaseForm") Then
```

L'événement déclencheur est implémenté deux fois. Seul le nom d'implémentation qui se termine par OdatabaseForm donne l'accès correct à l'enregistrement (les implémentations sont expliquées à la page 480).

```
                If Not oFormulaire.isBeforeFirst() And Not oFormulaire.isAfterLast()
Then
```

Lorsque le formulaire est lu ou rechargé, le curseur se place avant l'enregistrement en cours. Ensuite, si une tentative est effectuée, vous obtenez le message "État du curseur non valide".

```
                inMemo = oFormulaire.findColumn("Memo")
                loID = oFormulaire.findColumn("ID")
                oChamp = oFormulaire.getByName("MemoFormat")
                stMemo = oChamp.Text
                If stMemo <> "" Then
                    oFormulaire.updateString(inMemo, stMemo)
                End If
                If stMemo <> "" And oFormulaire.getString(loID) <> "" Then
                    oFormulaire.UpdateRow()
                End If
            End If
        End If
```

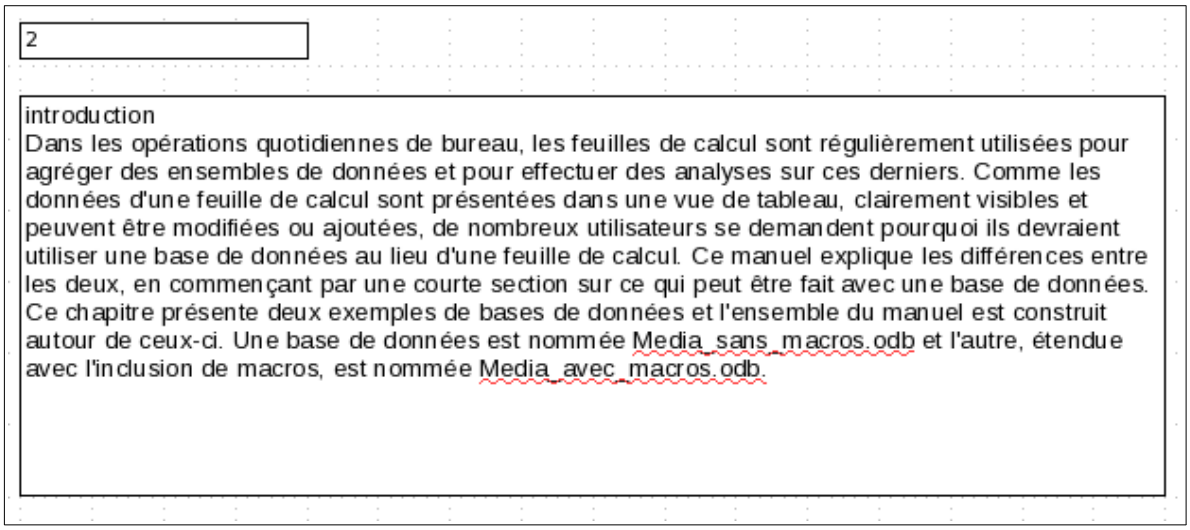
```
End Sub
```

Le champ de la table « Memo » est localisé à partir de la source de données du formulaire, avec celui de « ID ». Si le champ MemoFormat contient du texte, il est transféré dans le champ Memo de la source de données à l'aide de **oFormulaire.updateString ()**. Une mise à jour est effectuée uniquement s'il y a une entrée dans le champ ID (en d'autres termes, une clé primaire a été définie). Sinon, un nouvel enregistrement est inséré dans le cadre du fonctionnement normal du formulaire, le formulaire reconnaît le changement et le stocke indépendamment.

Un bouton situé dans le formulaire principal permet de rafraîchir le formulaire en cas de modification du champ texte recherché, et donc de mettre à jour. L'affichage du champ mémo du compteur d'enregistrements. **Propriétés> Général> Action> Rafraîchir le formulaire**.

Vérification orthographique lors de la saisie des données

Cette macro peut être utilisée pour les **champs de texte multilignes avec formatage**. Comme dans le chapitre précédent, le contenu de chaque enregistrement doit d'abord être écrit, puis le nouvel enregistrement peut être chargé dans le contrôle de formulaire. Les procédures LireContenu et EcireContenu ne diffèrent que par le point auquel la fonction de recherche peut être mise entre guillemets.



Le vérificateur d'orthographe de LibreOffice est lancé dans le formulaire ci-dessus chaque fois qu'un espace ou un retour est frappé dans le contrôle de formulaire. En d'autres termes, il s'exécute à la fin de chaque mot. Il pourrait également être lié au contrôle perdant le focus pour s'assurer que le dernier mot est vérifié.

La procédure est liée à **Formulaire> Événements> Touche relâchée**.

```
Sub MarqueMauvaisMotDirect(oEvent As Object)
	GlobalScope.BasicLibraries.LoadLibrary("Tools")
```

La fonction **RTrimStr** est utilisée pour supprimer tout signe de ponctuation à la fin de la chaîne. Sinon, tous les mots qui se terminent par une virgule, un point ou un autre signe de ponctuation apparaîtront comme des fautes d'orthographe. De plus, **LTrimChar** est utilisé pour supprimer les guillemets au début des mots[18].

18 Pour l'utilisation de cette macro, voir l'exemple Exemple_Autotexte_Recherche_Orthographe.odb, associée à ce manuel.

```
Dim aProp() As New com.sun.star.beans.PropertyValue
Dim oLinuSvcMgr As Object
Dim oSpellChk As Object
Dim oChamp As Object
Dim arTexte()
Dim stMot As String
Dim inLongueurMot As Integer
Dim ink As Integer
Dim i As Integer
Dim oCurseur As Object
Dim stTexte As String
oLinguSvcMgr = createUnoService("com.sun.star.linguistic2.LinguServiceManager")
If Not IsNull(oLinguSvcMgr) Then
    oSpellChk = oLinguSvcMgr.getSpellChecker()
    End If
```

Tout d'abord, toutes les variables sont déclarées. Ensuite, le module de vérification orthographique de base **SpellChecker** est accédé. Ce sera ce module qui vérifiera l'exactitude des mots individuels.

```
oChamp = oEvent.Source.Model
ink = 0
    If oEvent.KeyCode = 1280 Or oEvent.KeyCode = 1284 Then
```

L'événement qui lance la macro est une frappe. Cet événement comprend un code, le KeyCode, pour chaque touche individuelle. Le **KeyCode** de la touche « Entrée » est 1280, celui de l'espace est 1284. Comme beaucoup d'autres informations, ces éléments ont été récupérés via l'outil Xray. Si vous appuyez sur espace ou entrée, l'orthographe est vérifiée. Il est lancé, en d'autres termes, à la fin de chaque mot. Seul le test du dernier mot ne se produit pas automatiquement.

Chaque fois que la macro s'exécute, tous les mots du texte sont vérifiés. La vérification de mots individuels pourrait également être possible, mais demanderait beaucoup plus de travail.

Le texte est divisé en mots simples. Le délimiteur est le caractère d'espace. Avant cela, les mots séparés par des sauts de ligne doivent être réunis à nouveau, sinon les morceaux peuvent être confondus avec des mots complets.

```
    stTexte = Join(Split(oChamp.Text, CHR(10)),"  ")
    stTexte = Join(Split(stTexte, CHR(13)),"  ")
    arTexte = Split(RTrim(stTexte),"  ")
    For i = LBound(arText) To Ubound(arText)
        stMot = arTexte(i)
        inLongueurMot = len(stMot)
        stMot = Trim(RtrimStr(RtrimStr(RtrimStr(RtrimStr(RtrimStr( _
            RtrimStr(stMot,","), "."),"?"),"!"),"."),")"))
            stMot = LTrimChar(stMot,"(")
```

Les mots individuels sont lus. Leur longueur non tronquée est nécessaire pour l'étape d'édition suivante. Ce n'est qu'ainsi que la position du mot dans tout le texte (qui est nécessaire pour la mise en évidence spécifique des fautes d'orthographe) peut être déterminée.

Trim est utilisé pour supprimer les espaces, tandis que **RTrimStr** supprime les virgules et les points à la fin du texte et **LTrimChar** tous les signes de ponctuation au début.

```
    If stMot <> "" Then
        oCurseur = oChamp.createTextCursor()
        oCurseur.gotoStart(false)
        oCurseur.goRight(ink, false)
```

```
                oCurseur.goRight(inLongueurMot, true)
                If Not oSpellChk.isValid(stMot, "fr", aProp()) Then
                    oCurseur.CharUnderline = 9
                    oCurseur.CharUnderlineHasColor = True
                    oCurseur.CharUnderlineColor = RGB(255,51,51)
                Else
                    oCurseur.CharUnderline = 0
                End If
            End If
            ink = ink + inLongueurMot + 1
            Next
        End If
End Sub
```

Si le mot n'est pas nul, un curseur de texte est créé. Ce curseur est déplacé sans surligner au début du texte dans le champ de saisie. Ensuite, il saute vers la droite, toujours sans mise en évidence, au terme stocké dans la variable **ink**. Cette variable débute par 0, mais après l'exécution de la première boucle, elle est égale à la longueur du mot (+1 pour l'espace suivant). Ensuite, le curseur est déplacé vers la droite de la longueur du mot actuel. Les propriétés de police sont modifiées pour créer la région en surbrillance.

Le correcteur orthographique est lancé. Il requiert le mot et le code du pays comme arguments, sans code de pays tout compte comme correct. L'argument tableau est généralement vide.

Si le mot ne figure pas dans le dictionnaire, il est marqué d'une ligne ondulée rouge. Ce type de soulignement est ici représenté par « **9** ». Si le mot est trouvé, il n'y a pas de soulignement (**0**). Cette étape est nécessaire car sinon, un mot reconnu comme faux puis corrigé continuerait à être affiché avec la ligne ondulée rouge. Il ne serait jamais supprimé car aucun format conflictuel n'a été indiqué.

Boîtes combinées sous forme de Zone de liste avec une option d'entrée

Une table avec un seul enregistrement peut être directement créée en utilisant des Zones combinées et des champs numériques invisibles et la clé primaire correspondante entrée dans une autre table.[19]

Le contrôle Zone combinée traite les champs de formulaire en combinant l'entrée et le choix des valeurs (comboboxes) comme des Zones de liste avec une option de saisie. À cet effet, en plus des combobox du formulaire, les valeurs de champ clé qui doivent être transférées à la table sous-jacente sont stockées dans des champs numériques séparés. Les champs peuvent être déclarés comme invisibles. Les clés de ces champs sont lues lorsque le formulaire est chargé et la zone de liste déroulante est définie pour afficher le contenu correspondant. Si le contenu de la liste déroulante est modifié, il est enregistré et la nouvelle clé primaire est transférée dans le champ numérique correspondant pour être stockée dans la table principale.

Si des requêtes modifiables sont utilisées à la place des tables, le texte à afficher dans les combinaisons de champs peut être directement déterminé à partir de la requête. Une macro n'est alors pas requise pour cette étape.

Une hypothèse pour le fonctionnement de la macro est que la clé primaire de la table qui est la source de données pour le champ de combinaison est un entier à incrémentation automatique. On suppose également que le nom de champ de la clé primaire est ID.

19 Pour l'utilisation de zones combinées au lieu de zones de liste, consultez la base de données Exemple_ZoneCombinee_ListeChamps.odb associée à ce manuel.

Affichage du texte dans les zones combinées

Ce sous-programme consiste à afficher le texte dans la zone de liste déroulante en fonction de la valeur des champs de clé étrangère invisibles du formulaire principal. Il peut également être utilisé pour les listes de sélection qui renvoient à deux tables différentes. Cela peut se produire si, par exemple, le code postal d'une adresse postale est stocké séparément de la ville. Dans ce cas, le code postal peut être lu à partir d'une table contenant uniquement une clé étrangère pour la ville. La zone de liste doit afficher le code postal et la ville ensemble[20].

```
Sub AfficheTexte(oEvent As Object)
```

Cette macro doit être liée à l'événement de formulaire suivant : "Après le changement d'enregistrement".

La macro est appelée directement depuis le formulaire. L'événement déclencheur est la source de toutes les variables dont la macro a besoin. Certaines variables ont déjà été déclarées globalement dans un module séparé et ne sont plus déclarées ici.

```
Dim oFormulaire As Object
Dim oChampListe As Object
Dim stValeurChamp As String
Dim inZones As Integer
Dim stRequete As String
oFormulaire = oEvent. Source
```

Dans le formulaire, il y a un contrôle caché à partir duquel les noms de toutes les différentes listes déroulantes peuvent être obtenus. Une par une, ces zones de liste sont traitées par la macro.

```
aZonesCombinees() = Split(oFormulaire.getByName("ZonesCombinees").Tag,",")
For inZones = LBound(aZonesCombinees) TO Ubound(aZonesCombinees)
   …
   Next inZone
```

Le complément d'information (Tag) attaché au contrôle masqué contient cette liste de noms de zones de liste, séparés par des virgules. Les noms sont écrits dans un tableau puis traités dans une boucle. La boucle se termine par l'instruction **NEXT**.

La zone combinée, qui a remplacé une zone de liste, s'appelle oListeChamps. Pour obtenir la clé étrangère, nous avons besoin de la colonne correcte dans la table qui sous-tend le formulaire. Ceci est accessible en utilisant le nom du champ de table, qui est stocké dans les informations supplémentaires de la liste déroulante.

```
oChampListe = oFormulaire.getByName(Trim(aZonesCombinees(inZone)))
oChamp = oFormulaire.getString(oFormulaire.findColumn(oChampListe.Tag))
oChampListe.Refresh()
```

La zone combinée est lue à nouveau en utilisant **Refresh()** au cas où le contenu du champ a été modifié par l'entrée de nouvelles données.

La requête nécessaire pour fournir le contenu visible de la zone combinée est basée sur le champ sous-jacent du contrôle et la valeur déterminée pour la clé étrangère. Pour rendre le code SQL utilisable, toute opération de tri qui pourrait être présente est supprimée. Ensuite, une vérification est effectuée pour toutes les définitions de relation (qui commenceront par le mot **WHERE**). Par défaut, la fonction **InStr()** ne fait pas la distinction entre les majuscules et les minuscules, donc toutes les combinaisons de casse sont couvertes. S'il existe une relation, cela signifie que la

20 Des applications de cette procédure se trouvent dans l'exemple Media_avec_Macros

requête contient des champs de deux tables différentes. Nous devons trouver la table qui fournit la clé étrangère pour le lien. La macro dépend ici du fait que la clé primaire de chaque table s'appelle ID.

Si aucune relation n'est définie, la requête accède à une seule table. Les informations de table peuvent être ignorées et la condition formulée directement à l'aide de la valeur de clé étrangère.

```
If NOT IsEmpty(oChamp.getCurrentValue()) Then
    stRequete = oChampListe.ListSource
    If InStr(stRequete,"order by") > 0 Then
        stSql = Left(stRequete, InStr(stRequete,"order by")-1)
    Else
        stSql = stRequete
    End If
    If InStr(stSql,"where") Then
        st = Right(stSql, Len(stSql)-InStr(stSql,"where")-4)
        If InStr(Left(st, InStr(st,"=")),"."""ID""") Then
            a() = Split(Right(st, Len(st)-InStr(st,"=")-1),".")
        Else
            a() = Split(Left(st, InStr(st,"=")-1),".")
        End If
        stSql = stSql + "AND "+a(0)+"."""ID"" = " + oChamp.getCurrentValue
    Else
        stSql = stSql + "WHERE ""ID"" = " + oChamp.getCurrentValue
    End If
```

Chaque nom de champ et de table doit être entré dans la commande SQL avec deux ensembles de guillemets. Les guillemets sont normalement interprétés par Basic comme des délimiteurs de chaîne de texte, ils n'apparaissent donc plus lorsque le code est transmis à SQL. Le fait de doubler les guillemets garantit qu'un jeu est transmis. ""ID"" signifie que le champ "ID" sera accessible dans la requête, avec l'ensemble unique de guillemets requis par SQL.

La requête stockée dans la variable **stSql** est maintenant exécutée et son résultat enregistré dans **oResultat**.

```
    oSourceDonnees = ThisComponent. Parent. CurrentController
    If Not (oSourceDonnees.isConnected()) Then
        oSourceDonnees.connect()
    End If
    oConnexion = oSourceDonnees. ActiveConnection()
    oSQL_Commande = oConnexion.createStatement()
    oResultat = oSQL_Commande.executeQuery(stSql)
```

Le résultat de la requête est lu dans une boucle. Comme pour une requête dans l'interface graphique, plusieurs champs et enregistrements peuvent être affichés. Mais la construction de cette requête ne nécessite qu'un seul résultat, qui se trouvera dans la première colonne (**1**) du jeu de résultats de la requête. C'est l'enregistrement qui fournit le contenu affiché de la zone combinée. Le contenu est du texte (**getString()**), d'où la commande **oResultat.getString(1)**.

```
    While oResultat.next
        stValeurChamp = oResultat.getString(1)
    Wend
```

La zone combinée doit maintenant être définie sur la valeur de texte récupérée par la requête.

```
oChampListe.Text = stValeurChamp
Else
```

S'il n'y a pas de valeur dans le champ de la clé étrangère **oChamp**, la requête a échoué et la zone de liste déroulante est définie sur une chaîne vide.

```
            oChampListe.Text = ""
        End If
        Next inZone
End Sub
```

Cette procédure gère le contact entre la zone combinée et la clé étrangère disponible dans un champ de la source de données du formulaire. Cela devrait être suffisant pour afficher les valeurs correctes dans les zones combinées. Le stockage de nouvelles valeurs nécessiterait une procédure supplémentaire.

Transférer une valeur de clé étrangère d'une zone combinée vers un champ numérique

Si une nouvelle valeur est entrée dans la zone de liste déroulante (et c'est après tout le but pour lequel cette macro a été construite), la clé primaire correspondante doit être entrée dans la table sous-jacente du formulaire en tant que clé étrangère.

```
Sub EnregistrerValeurTexteSelection(oEvent As Object)
```

Cette macro doit être liée à l'événement de formulaire suivant : "Avant l'action d'enregistrement".

Une fois que les variables ont été déclarées (non affichées ici), nous devons d'abord déterminer exactement quel événement doit lancer la macro. Avant l'action d'enregistrement, deux implémentations sont appelées successivement. Il est important que la macro elle-même récupère l'objet de formulaire. Cela peut être fait dans les deux implémentations mais de différentes manières. Ici, l'implémentation appelée *ODatabaseForm* est filtrée.

```
        If InStr(oEvent. Source. ImplementationName,"ODatabaseForm") Then
            …
        End If
End Sub
```

Cette boucle se construit avec le même début que la procédure **AfficheTexte :**

```
            oFormulaire = oEvent. Source
            aZonesListe() = Split(oFormulaire.getByName("ZonesCombinees"). Tag,",")
            For inZone = LBound(aZonesListe) To Ubound(aZonesListe)
                …
            Next inZone
```

Le champ **oChampListe** affiche le texte. Il peut se trouver à l'intérieur d'un champ table, auquel cas il n'est pas possible d'y accéder directement depuis le formulaire. Dans de tels cas, les informations supplémentaires pour les zones combinées de contrôle masqué doivent contenir le chemin d'accès au champ à l'aide de la zone de liste déroulante du "ControleTable". Le fractionnement de cette entrée révélera comment accéder à la zone de liste déroulante.

```
            a() = Split(Trim(aZonesListe(inZone)),">")
                If Ubound(a) > 0 Then
                    oChampListe = oFormulaire.getByName(a(0)).getByName(a(1))
                Else
                    oChampListe = oFormulaire.getByName(a(0))
                End If
```

Ensuite, la requête est lue à partir de la zone de liste déroulante et divisée en ses parties individuelles. Pour les combobox simples, les informations nécessaires sont le nom du champ et le nom de la table :

```
SELECT "Champ" FROM "Table"
```

Cela pourrait dans certains cas être complété par une instruction de tri. Chaque fois que deux champs doivent être réunis dans la zone de liste déroulante, plus de travail sera nécessaire pour les séparer.

```
SELECT "Champ1"||' '||"Champ2" FROM "Table"
```

Cette requête associe deux champs avec un espace entre eux. Comme le séparateur est un espace, la macro le recherchera et divisera le texte en deux parties en conséquence. Naturellement, cela ne fonctionnera de manière fiable que si Champ1 ne contient pas déjà du texte dans lequel les espaces sont autorisés. Sinon, si le prénom est "Anne Marie" et le nom de famille "Müller", la macro traitera "Anne" comme prénom et "Marie Müller" comme nom de famille. Dans de tels cas, un séparateur plus approprié doit être utilisé, qui peut ensuite être trouvé par la macro. Dans le cas des noms, cela peut être "Nom, Prénom".

Les choses se compliquent encore si les deux champs proviennent de tables différentes :

```
SELECT "Table1"."Champ1"||' > '||"Table2"."Champ2"
FROM "Table1", "Table2"
WHERE "Table1"."ID" = "Table2"."ID_Etrangère"
ORDER BY "Table1"."Champ1"||' > '||"Table2"."Champ2" ASC
```

Ici, les champs doivent être séparés les uns des autres, la table à laquelle appartient chaque champ doit être établie et les clés étrangères correspondantes déterminées.

```
stRequete = oChampListe.ListSource
aChamps() = Split(stRequete, """")
stContenu = ""
For i=LBound(aChamps)+1 To UBound(aChamps)
```

Le contenu de la requête est dépourvu de scories inutile. Les pièces sont réassemblées dans un tableau avec une combinaison de caractères inhabituelle comme séparateur. FROM sépare l'affichage du champ visible des noms de table. WHERE sépare la condition des noms de table. Les jointures ne sont pas prises en charge.

```
   If Trim(UCASE(aChamps(i))) = "ORDER BY" Then
      Exit For
   ElseIf Trim(UCASE(aChamps(i))) = "FROM" Then
      stContenu = stContenu+" §§ "
   ElseIf Trim(UCASE(aChamps(i))) = "WHERE" Then
      stContenu = stContenu+" §§ "
   Else
   stContenu = stContenu+Trim(aChamps(i))
   End If
Next i
acontenu() = Split(stContenu, " §§ ")
```

Dans certains cas, le contenu de l'affichage du champ visible provient de différents champs :

```
aPremier() = Split(aContenu(0),"||")
If UBound(aPremier) > 0 Then
   If UBound(aContenu) > 1 Then
```

La première partie contient au moins deux champs. Les champs commencent par un nom de table. La deuxième partie contient deux noms de table, qui peuvent être déterminés à partir de la première partie. La troisième partie contient une relation avec une clé étrangère, séparée par = :

```
                aTest() = Split(aFirst(0),".")
                NomTable1 = aTest(0)
                NomChampTable1 = aTest(1)
                Erase aTest
                stFieldSeperator = Join(Split(aFirst(1),"'"),"")
                aTest() = Split(aFirst(2),".")
                NomTable2 = aTest(0)
                NomChampTable2 = aTest(1)
                Erase aTest
                aTest() = Split(acontenu(2),"=")
                aTest1() = Split(aTest(0),".")
                If aTest1(1) <> "ID" Then
                   NomIDTab2 = aTest1(1)
                   IF aTest1(0) = NomTable1 Then
                      Position = 2
                   Else
                      Position = 1
                   End If
                Else
                   Erase aTest1
                   aTest1() = Split(aTest(1),".")
                   NameTab12ID = aTest1(1)
                   If aTest1(0) = NomTable1 Then
                      Position = 2
                   Else
                      Position = 1
                   End If
                End If
                Else
```

La première partie contient deux noms de champs sans noms de table, éventuellement avec des séparateurs. La deuxième partie contient les noms de table. Il n'y a pas de troisième partie :

```
                If UBound(aFirst) > 1 Then
                   NomChampTable1 = aFirst(0)
                   stFieldSeperator = Join(Split(aFirst(1),"'"),"")
                   NomChampTable2 = aFirst(2)
                Else
                   NomChampTable1 = aFirst(0)
                   NomChampTable2 = aFirst(1)
                End If
                NomTable1 = acontenu(1)
             End If
          Else
```

Il n'y a qu'un seul champ dans une table :

```
             NomChampTable1 = aFirst(0)
             NomTable1 = acontenu(1)
          End If
```

La longueur de caractère maximale qu'une entrée peut avoir est donnée par la fonction ColumnSize. La zone combinée ne peut pas être utilisée pour limiter la taille, car elle peut avoir besoin de contenir deux champs en même temps.

```
        LongueurChamp1 = TailleColonne(NomTable1, NomChampTable1)
        If NomChampTable2 <> "" Then
            If NomTable2 <> "" Then
                LongueurChamp2 = TailleColonne(NomTable2, NomChampTable2)
            Else
                LongueurChamp2 = TailleColonne(NomTable1, NomChampTable2)
            End If
        Else
            LongueurChamp2 = 0
        End If
```

Le contenu de la combobox est lu :

```
        stContenu = oChampListe.getCurrentValue()
```

Les espaces de début et de fin et les caractères non imprimables sont supprimés si nécessaire.

```
stContenu = Trim(stContenu)
If stContenu <> "" Then
    If NomChampTable2 <> "" Then
```

Si un deuxième champ de table existe, le contenu de la zone de liste déroulante doit être fractionné. Pour déterminer où le fractionnement doit se produire, nous utilisons le séparateur de champ fourni à la fonction comme argument.

```
        a_stParts = Split(stContenu, FieldSeparator, 2)
```

Le dernier paramètre signifie que le nombre maximum de parties est de 2.

En fonction de l'entrée correspondant au champ 1 et de celle du champ 2, le contenu de la zone de liste déroulante est maintenant alloué aux variables individuelles. "Position = 2" sert ici d'indicateur que la deuxième partie du contenu correspond au champ 2.

```
        If Position = 2 Then
            stContenu = Trim(a_stParts(0))
            If UBound(a_stParts()) > 0 Then
                stContenuChamp2 = Trim(a_stParts(1))
            Else
                stContenuChamp2 = ""
            End If
            stContenuChamp2 = Trim(a_stParts(1))
        Else
            stContenuChamp2 = Trim(a_stParts(0))
            If UBound(a_stParts()) > 0 Then
                stContenu = Trim(a_stParts(1))
            Else
                stContenu = ""
            End If
            stContenu = Trim(a_stParts(1))
        End If
    End If
```

Il peut arriver qu'avec deux contenus séparables, la taille installée de la zone de liste déroulante (longueur du texte) ne corresponde pas aux champs du tableau à enregistrer. Pour les zones combinées qui représentent un seul champ, cela est normalement géré en configurant correctement le contrôle de formulaire. Ici, en revanche, nous avons besoin d'un moyen de détecter ces erreurs. La longueur maximale autorisée du champ concerné est vérifiée.

```
If (LongueurChamp1 > 0 And Len(stContenu) > LongueurChamp1) Or _
(LongueurChamp2 > 0 And Len(stContenuChamp2) > LongueurChamp2) Then
```

Si la longueur du champ de la première ou de la deuxième partie est trop grande, une chaîne par défaut est stockée dans l'une des variables. Le caractère **Chr(13)** est utilisé pour insérer un saut de ligne.

```
            stmsgbox1 = "Le champ " + NomChampTable1 + " ne peut dépasser " + _
LongueurChamp + "caractères en longueur." + Chr(13)
            stmsgbox2 = "Le champ " + NomChampTable2 + " ne peut dépasser " + _
LongueurChamp2 + "caractères en longueur." + Chr(13)
```

Si le contenu des deux champs est trop long, les deux textes sont affichés.

```
            If ( LongueurChamp1 > 0 And Len(stContenu) > LongueurChamp1) And _
               ( LongueurChamp2 > 0 And Len(stContenuChamp2) > LongueurChamp2) Then
               MsgBox("Le texte saisi est trop long." + Chr(13) + stmsgbox1 + _
stmsgbox2 + "Raccourcir SVP.",64,"Entrée invalide")
```

L'affichage utilise la fonction **MsgBox()**. Cela attend comme premier argument une chaîne de texte, puis éventuellement un nombre (qui détermine le type de boîte de message affiché), et enfin une chaîne de texte optionnelle comme titre de la fenêtre. La fenêtre aura donc le titre "Entrée invalide" et le numéro "64" fournit une boîte contenant le symbole d'information.

Le code suivant couvre tout autre cas de texte excessivement long qui pourrait survenir.

```
         ElseIf (LongueurChamp1 > 0 And Len(stContenu) > LongueurChamp) Then
            MsgBox("Le texte saisi est trop long." + Chr(13) + stmsgbox1 + "Raccourcir
SVP.",64,"Entrée invalide")
         Else
            MsgBox("Le texte saisi est trop long." + Chr(13) + stmsgbox2 + "Raccourcir
SVP.",64,"Entrée invalide")
         End If
         Else
```

S'il n'y a pas de texte trop long, la fonction peut continuer. Sinon, elle sort ici.

Désormais, les entrées sont masquées afin que les guillemets éventuellement présents ne génèrent pas d'erreur.

```
         stContenu = String_to_SQL(stContenu)
         If stContenuChamp2 <> "" Then
            stContenuChamp2 = String_to_SQL(stContenuChamp2)
         End If
```

Les premières variables sont préallouées et peuvent ensuite être modifiées par la requête. Les variables inID1 et inID2 stockent le contenu des champs de clé primaire des deux tables. Si une requête ne donne aucun résultat, Basic attribue à cette variable entière une valeur de 0. Cependant, cette valeur peut également indiquer une requête réussie renvoyant une valeur de clé primaire de 0 ; donc la variable est préréglée à -1. HSQLDB ne peut pas définir cette valeur pour un champ de valeur automatique.

Ensuite, la connexion à la base de données est établie, si elle n'existe pas déjà.

```
            inID1 = -1
            inID2 = -1
            oSourceDonnees = ThisComponent. Parent. CurrentController
            If Not (oSourceDonnees.isConnected()) Then
               oSourceDonnees.connect()
            End If
            oConnexion = oSourceDonnees. ActiveConnection()
```

```
            oSQL_Commande = oConnexion.createStatement()
            If NomChampTable2 <> "" And Not IsEmpty(stContenuChamp2) And _
    NomTable2 <> "" Then
```

Si un deuxième champ de table existe, une deuxième dépendance doit d'abord être déclarée.

```
            stSql = "SELECT ""ID"" FROM """ + NomTable2 + """ WHERE """ + _
    NomChampTable2 + """='" + stContenuChamp2 + "'"
            oResultat = oSQL_Commande.executeQuery(stSql)
                While oResultat.next
                    inID2 = oResultat.getInt(1)
                Wend
            If inID2 = -1 Then
                stSql = "INSERT INTO """ + NomTable2 + """ (""" + NomChampTable2 + """)
    VALUES ('" + stContenuChamp2 + "') "
                oSQL_Commande.executeUpdate(stSql)
                    stSql = "CALL IDENTITY()"
```

Si le contenu de la zone de liste n'est pas présent dans le tableau correspondant, il y est inséré. La valeur de la clé primaire qui en résulte est ensuite lue. S'il est présent, la clé primaire existante est lue de la même manière. La fonction utilise les champs de clé primaire générés automatiquement (**IDENTITY**).

```
                oResultat = oSQL_Commande.executeQuery(stSql)
                    While oResultat.next
                        inID2 = oResultat.getInt(1)
                    Wend
                End If
```

La clé primaire de la deuxième valeur est temporairement stockée dans la variable **inID2**, puis écrite en tant que clé étrangère dans la table correspondant à la première valeur. Selon que l'enregistrement de la première table était déjà disponible, le contenu est nouvellement enregistré (**INSERT**) ou modifié (**UPDATE**) :

```
                If inID1 = -1 Then
                    stSql = "INSERT INTO """ + NomTable1 + """ (""" + NomChampTable1 +
    """,""" + NomIDTab2 + """) VALUES ('" + stContent + "','" + inID2 + "') "
                    oSQL_Commande.executeUpdate(stSql)
```

Et l'ID correspondant directement lu :

```
                    stSql = "CALL IDENTITY()"
                    oResultat = oSQL_Commande.executeQuery(stSql)
                    While oResultat.next
                        inID1 = oResultat.getInt(1)
                    Wend
```

La clé primaire de la première table doit enfin être lue à nouveau pour pouvoir être transférée vers la table sous-jacente du formulaire.

```
                Else
                    stSql = "UPDATE """ + NomTable1 + """ SET """ + NameTab12ID + _
    """='" + inID2 + "' WHERE """ + NomChampTable1 + """ = '" + stContent + "'"
                    oSQL_Commande.executeUpdate(stSql)
                End If
            End If
```

Dans le cas où les deux champs sous-jacents à la zone combinée sont dans la même table (par exemple Nom et Prénom dans la table Noms), une requête différente est nécessaire :

```
        If NomChampTable2 <> "" And NomTable2 = "" Then
            stSql = "SELECT ""ID"" FROM """ + NomTable1 + """ WHERE """ + _
NomChampTable1 + """='" + stContent + "' AND """ + NomChampTable2 + _
"""='" + stContentField2 + "'"
            oResultat = oSQL_Commande.executeQuery(stSql)
                While oResultat.next
                    inID1 = oResultat.getInt(1)
                Wend
                If inID1 = -1 Then
```

… et une deuxième table n'existe pas :

```
                stSql = "INSERT INTO """ + NomTable1 + """ (""" + NomChampTable1 +
""","""" + NomChampTable2 + """) VALUES ('" + stContent + "','" + stContentField2 +
"') "
                    oSQL_Commande.executeUpdate(stSql)
```

Ensuite, la clé primaire est à nouveau lue.

```
                stSql = "CALL IDENTITY()"
                oResultat = oSQL_Commande.executeQuery(stSql)
                    While oResultat.next
                        inID1 = oResultat.getInt(1)
                    Wend
            End If
        End If
        IF NomChampTable2 = "" Then
```

Considérons maintenant le cas le plus simple : le deuxième champ de table n'existe pas et l'entrée n'est pas encore présente dans la table. En d'autres termes, une seule nouvelle valeur a été entrée dans la zone de liste déroulante.

```
        stSql = "SELECT ""ID"" FROM """ + NomTable1 + """ WHERE """ + NomChampTable1 +
"""='" + stContent + "'"
        oResultat = oSQL_Commande.executeQuery(stSql)
            While oResultat.next
                inID1 = oResultat.getInt(1)
            Wend
                If inID1 = -1 Then
```

S'il n'y a pas de deuxième champ, le contenu de la zone est inséré en tant que nouvel enregistrement.

```
                stSql = "INSERT INTO """ + NomTable1 + """ (""" + NomChampTable1 + """)
VALUES ('" + stContenu + "') "
                    oSQL_Commande.executeUpdate(stSql)
```

… Et l'ID qui en résulte est lu directement.

```
                stSql = "CALL IDENTITY()"
                oResultat = oSQL_Commande.executeQuery(stSql)
                    While oResultat.next
                        inID1 = oResultat.getInt(1)
                    Wend
            End If
        End If
```

La valeur du champ de clé primaire doit être déterminée afin de pouvoir être transférée vers la partie principale du formulaire.

Ensuite, la valeur de clé primaire résultant de toutes ces boucles est transférée dans le champ invisible de la table principale et de la base de données sous-jacente. Le champ de table lié au champ de formulaire est atteint en utilisant "BoundField"**."updateInt"** place un entier (voir sous les définitions de type numérique) dans ce champ.

```
                oFormulaire.updateLong(oFormulaire.findColumn(oChampListe.Tag), inID1)
            End If
        ELSE
```

Si aucune clé primaire ne doit être saisie, parce qu'il n'y avait aucune entrée dans la zone de liste déroulante ou que cette entrée a été supprimée, le contenu du champ invisible doit également être supprimé. updateNull() est utilisé pour remplir le champ avec l'expression spécifique à la base de données pour un champ vide, NULL.

```
                oFormulaire.updateNULL(oFormulaire.findColumn(oChampListe.Tag), NULL)
                End If
            NEXT inZone
        End If
End Sub
```

Fonction pour mesurer la longueur de l'entrée zone combinée

La fonction suivante donne le nombre de caractères dans la colonne de table respective, de sorte que les entrées trop longues ne soient pas simplement tronquées. Une **Function** est choisie ici pour fournir des valeurs de retour. Un **SUB** n'a pas de valeur de retour qui peut être transmise et traitée ailleurs.

```
Function TailleColonne(Tablename As String, NomChamp As String) As Integer
       oSourceDonnees = ThisComponent. Parent. CurrentController
       If Not (oSourceDonnees.isConnected()) Then
          oSourceDonnees.connect()
       End If
       oConnexion = oSourceDonnees. ActiveConnection()
       oSQL_Commande = oConnexion.createStatement()
       stSql = "SELECT ""COLUMN_SIZE"" FROM ""INFORMATION_SCHEMA"".""SYSTEM_COLUMNS""
WHERE ""TABLE_NAME"" = '" + Tablename + "' AND ""COLUMN_NAME"" = '" + NomChamp + "'"
       oResultat = oSQL_Commande.executeQuery(stSql)
          While oResultat.next
             i = oResultat.getInt(1)
          Wend
       TailleColonne = i
End Function
```

Générer des actions de base de données

```
Sub GenereActionEnregistrement(oEvent As Object)
```

Cette macro doit être liée à l'événement *Texte modifié* de la zone de liste. Il est nécessaire que dans tous les cas où la zone de liste est modifiée, la modification soit enregistrée. Sans cette macro, il n'y aurait aucun changement dans la table réelle que Base pourrait reconnaître, car la zone de liste déroulante n'est pas liée au formulaire.

Cette macro modifie directement les propriétés du formulaire :

```
       Dim oFormulaire As Object
       oFormulaire = oEvent.Source.Model.Parent
       oFormulaire. IsModified = TRUE
```

```
End Sub
```

Cette macro n'est pas nécessaire pour les formulaires qui utilisent des requêtes pour le contenu des zones combinées. Les changements dans les zones combinées sont enregistrés directement.

Navigation d'un formulaire à un autre

Dans un formulaire lorsqu'un événement particulier se produit, un second formulaire peut être ouvert[21]. Dans les propriétés du contrôle de formulaire, sur la ligne « Complément d'information » (balise), entrez le nom du second formulaire suivit du nom du premier, séparés par une virgule (ex : Form_2, Form_1). Dans l'exemple utilisé ceci est réalisé dans les propriétés de deux boutons.

De plus amples informations peuvent également être saisies ici, puis séparées à l'aide de la fonction **Split()**.

```
Sub De_formulaire_a_formulaire(oEvent As Object)
	Dim stTag As String
	stTag = oEvent.Source.Model.Tag
	aFormulaire() = Split(stTag, ",")
```

Le tableau est déclaré et rempli avec les noms des formulaires, d'abord celui du formulaire à ouvrir et deuxièmement celui du formulaire actuel, qui sera fermé après l'ouverture de l'autre.

```
	ThisDatabaseDocument.FormDocuments.getByName(Trim(aFormulaire(0))).open
	ThisDatabaseDocument.FormDocuments.getByName(Trim(aFormulaire(1))).close
End Sub
```

Si au contraire, le formulaire ne doit être ouvert que lorsque le formulaire actuel est fermé, par exemple lorsqu'un formulaire principal existe et que tous les autres formulaires sont contrôlés à partir de celui-ci à l'aide de boutons, la macro suivante doit être liée au formulaire avec **Outils> Personnaliser> Evénements> Document fermé** :

```
Sub Ouvre_Formulaire_Principal
	ThisDatabaseDocument.FormDocuments.getByName("Formulaire").open
End Sub
```

Si les documents de formulaire sont triés dans le fichier ODB dans des dossiers, la macro de changement de formulaire doit être plus étendue :

```
Sub De_Formulaire_a_Formulaire_avec_dossiers(oEvent As Object)
	REM Le formulaire à ouvrir est donné en premier.
	REM Si un formulaire se trouve dans un dossier, utilisez "/" pour définir
	REM la relation afin que le sous-dossier puisse être trouvé.
	Dim stTag As String
	stTag = oEvent.Source.Model.Tag 'La balise est entrée dans les informations
complémentaires
	aFormulaires() = Split(stTag, ",") 'Ici, le nom du nouveau formulaire vient en
premier, puis celui de l'ancien formulaire
	aFormulaires1() = Split(aFormulaires(0),"/")
	aFormulaires2() = Split(aFormulaires(1),"/")
	If UBound(aFormulaires1()) = 0 Then
	   ThisDatabaseDocument.FormDocuments.getByName(Trim(aFormulaires1(0))).open
	Else
	   ThisDatabaseDocument.FormDocuments.getByName(Trim(aFormulaires1(0))).
	      getByName(Trim(aFormulaires1(1))).open
```

21 Voir la base Exemple_Formulaire_a_formulaire.odb, associée à ce manuel. Des usages sont visibles également dans la base Media_avec_Macros.

```
        End If
        If UBound(aFormulaires2()) = 0 Then
            ThisDatabaseDocument. FormDocuments.getByName _
                (Trim(aFormulaires2(0))).close
        Else
            ThisDatabaseDocument. FormDocuments.getByName _
                (Trim(aFormulaires2(0))).getByName(Trim(aFormulaires2(1))).close
        End If
End Sub
```

Les documents de formulaire qui se trouvent dans un dossier sont saisis dans le champ Informations supplémentaires en tant que dossier/formulaire. Cela doit être converti en :

```
…getByName("Dossier").getByName("Formulaire").
```

Ouvrir des rapports, tableaux, requêtes depuis un formulaire

Comme dans le chapitre précédent, les rapports, eux aussi, peuvent être ouverts depuis un formulaire. Tout comme les formulaires, les rapports sont des documents distincts inclus dans le dossier de base. Au lieu de **FormDocuments**, **ReportDocuments** doit être utilisé ici. En outre, il faut noter que les formulaires et les rapports peuvent être localisés dans des sous-répertoires. Cependant, il est plus difficile d'accéder aux tableaux, requêtes et vues, car ils ne sont pas disponibles sous forme de documents séparés[22].

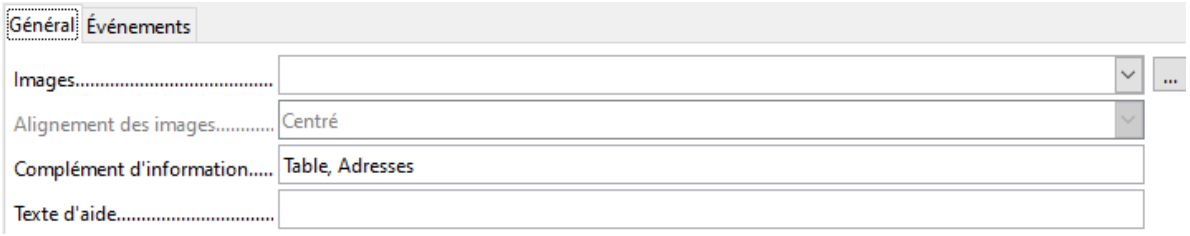
```
Sub Ouvre_Element(oEvent As Object)
        Dim stTag As String
        Dim inType As Integer
        Dim aOuvrir()
        Dim oDoc As Object
        stTag = oEvent.Source.Model.Tag
        aOuvrir() = Split(stTag, ",")
        Select Case Trim(aOuvrir(0))
            Case "Formulaire", "Rapport"
                Rem Formulaires et rapports peuvent être
                Rem localisés dans des sous-répertoires.
                aForms1() = Split(Trim(aOuvrir(1)),"/")
                If Trim(aOuvrir(0)) = "Formulaire" Then
                    oDoc = ThisDatabaseDocument.FormDocuments
                Else
                    oDoc = ThisDatabaseDocument.ReportDocuments
                End If
                If Ubound(aForms1()) > 0 Then
                    oDoc.getByName(Trim(aForms1(0))).getByName(Trim(aForms1(1))).open
                Else
                    oDoc.getByName(Trim(aForms1(0))).open
                End If
                If Trim(aOuvrir(0)) = "Formulaire" And Ubound(aOuvrir()) > 1 Then
                Rem Le formulaire qui lance la macro pourrait également être fermé
                    aForms2() = Split(Trim(aOuvrir(2)),"/")
                    If Ubound(aForms2()) > 0 Then
                        ThisDatabaseDocument.FormDocuments.getByName(Trim(aForms2(0))). _
                            getByName(Trim(aForms2(1))).close
                    Else

        ThisDatabaseDocument.FormDocuments.getByName(Trim(aForms2(0))).close
                    End If
                End If
```

22 Voir la base de données Exemple_Formulaire_a_formulaire.odb associée à ce manuel.

```
            Exit Sub
        Case "Requête"
            inType = 1
            Open_Table_Query_View(Trim(aOuvrir(1)),inType)
        Case "Table"
            inType = 0
            Open_Table_Query_View(Trim(aOuvrir(1)),inType)
    End Select
End Sub
```

Dans le formulaire de base (dans l'exemple Formulaire_Menu) les informations nécessaires sont lues dans les **propriétés** du bouton utilisé, onglet **Général**.



Deux termes sont utilisés, le premier pour indiquer que l'on désire ouvrir une Table (ou un Formulaire, une Requête, un Rapport), le second pour donner le nom de l'objet à ouvrir (Exemple : Table, Adresses).

Ces deux informations sont récupérées dans **oTag**, puis placées dans un tableau **aOuvrir**. **Trim** est utilisé pour supprimer tous les espaces superflus devant et après les expressions.

Si le premier terme est « **Formulaire** » ou « **Rapport** », le formulaire ou le rapport nommé dans le second terme est ouvert.

Pour les requêtes et les tables (ou vues), il faut adopter une approche différente.

Une variable (**inType**) détermine s'il s'agit d'une **Requête** (**1**), d'une **Table** ou d'une **Vue** (**0**), puis la sous-procédure suivante est lancée, avec pour paramètres : **aOuvrir(1)** – le nom de l'élément à ouvrir, et **inType**.

```
Sub Ouvre_Form_Rapp_Tabl_Vue_Req(stNom AS STRING, inType AS INTEGER)
    Rem Form pour Formulaire – Rapp pour Rapport
    Rem Tabl pour Table – Vue – Req pour Requête
    DIM oControleur AS OBJECT
    DIM oConnexion AS OBJECT
    oControleur = ThisDatabasedocument.CurrentController
    IF NOT oControleur.isconnected THEN
        oControleur.connect
    END IF
    oConnexion = oControleur.ActiveConnection
    DIM URL AS NEW com.sun.star.util.URL
    DIM Args(5) AS NEW com.sun.star.beans.PropertyValue
    URL.Complete = ".component:DB/DataSourceBrowser"
    Dispatch = StarDesktop.queryDispatch(URL,"_Blank",8)
    Args(0).Name = "ActiveConnection"
    Args(0).Value = oConnexion
    Args(1).Name = "CommandType"
    Args(1).Value = inType    '0=Table 1=Requête_SQL 2=Commande
    Args(2).Name = "Command"
    Args(2).Value = stNom
    Args(3).Name = "ShowMenu"
    Args(3).Value = True
```

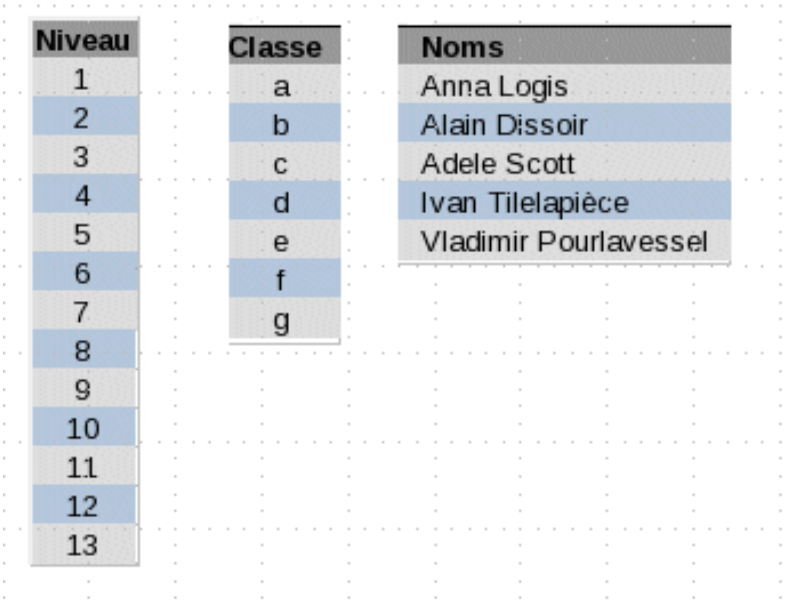
```
        Args(4).Name = "ShowTreeView"
        Args(4).Value = False
        Args(5).Name = "ShowTreeViewButton"
        Args(5).Value = False
        Dispatch.dispatch(URL, Args)
END SUB
```

Tout d'abord, la connexion à la base de données est établie, si elle n'existe pas encore. Cette connexion est transmise dans un tableau avec quelques informations supplémentaires, notamment le type d'élément à ouvrir (table, requête, ...) et le nom de l'élément.

L'élément est finalement ouvert via le **StarDesktop.queryDispatch** avec la commande **dispatch**.

Zones de liste hiérarchiques

Les paramètres d'un champ de liste sont destinés à influencer directement les paramètres d'un autre. Pour les cas simples, cela a déjà été décrit ci-dessus dans la section sur le filtrage des enregistrements. Mais en supposant que la première zone de liste est censée affecter le contenu de la deuxième, qui affecte alors le contenu d'une troisième zone de liste, et ainsi de suite[23].



Exemples de zones de liste pour une sélection hiérarchique

Dans cet exemple, la première zone de liste (Niveau) contient toutes les années scolaires. Les Classes de chaque niveau sont représentées par des lettres. Les noms sont ceux des membres de la classe.

Dans des circonstances normales, la liste des niveaux affiche les 13 années, la liste des classes toutes les lettres des classes et la liste des noms tous les élèves de l'école.

S'il s'agit de listes hiérarchiques, le choix des classes est restreint une fois que le niveau est choisi. Seules les lettres de classe qui sont effectivement présentes à ce niveau-là sont affichées. Cela peut varier car, si le nombre d'élèves augmente, le nombre de classes par niveau peut également augmenter. La dernière zone de liste, Noms, est très restreinte. Au lieu de plus de 1000 élèves, il n'en afficherait que 30.

23 Un exemple de contrôles hiérarchiques se trouve dans la base Exemple_Cherche_et_Filtre.odb

Au début, seul le niveau peut être sélectionné. Une fois cela fait, la liste (restreinte) des classes est mise à disposition. Ce n'est qu'à la fin que la liste des noms est donnée.

Si la zone de liste Niveaux est modifiée, la séquence doit recommencer. Si seule la zone de liste Classes est modifiée, le numéro de niveau de la dernière zone de liste reste valide

Pour créer une telle fonction, le formulaire doit pouvoir stocker une variable intermédiaire. Cela a lieu dans un contrôle caché.

La macro est liée à une modification du contenu d'une zone de liste : **Propriétés Zone de liste> Événements> Modifiée**. Les variables nécessaires sont stockées dans le complément d'information de la zone de liste.

Voici un exemple d'informations complémentaires fournies:
Formulaire, **`Niveau`**, Contrôle_masqué, **ZoneListe_2**

Le formulaire s'appelle Formulaire. La Zone de liste actuelle est appelée ZoneListe_1. Cette zone de liste affiche le contenu du champ de table Niveau et les zones de liste suivantes doivent être filtrées en fonction de cette entrée. Le contrôle masqué est désigné par Controle_Masque et l'existence d'une deuxième zone de liste (ZoneListe_2) est transmise à la procédure de filtrage.

```
Sub Controles_Hierarchiques(oEvent As Object)
	Dim oDoc As Object
	Dim oDrawpage As Object
	Dim oFormulaire As Object
	Dim oChampCache As Object
	Dim oChamp As Object
	Dim oChamp1 As Object
	Dim stSql As String
	Dim acontenu()
	Dim stTag As String
	oChamp = oEvent.Source.Model ' la zone de liste origine de l'événement
	stTag = oChamp. Tag
	oFormulaire = oChamp.Parent
	REM La balise entre dans le champ Complément d'informations
	REM Elle contient :
	REM 0. Nom du champ du champ à filtrer dans la table
	REM 1. Nom de champ du contrôle masqué qui stockera la valeur filtrée
	REM 2. Zone de liste supplémentaire éventuelle
	REM La balise est lue à partir de l'élément qui lance la macro. La variable est
	REM passée à la procédure, et si nécessaire à toutes les autres zones de liste.
	aFiltre() = Split(stTag, ",")
	stFiltre = ""
```

Une fois que les variables ont été déclarées, le contenu de la balise est passé (découpée) à un tableau, afin que les éléments individuels soient accessibles. Ensuite, l'accès aux différents champs du formulaire est déclaré.

La zone de liste qui a appelé la macro est déterminée et sa valeur lue. Ce n'est que si cette valeur n'est pas NULL qu'elle sera combinée avec le nom du champ à filtrer, dans notre exemple Niveau, pour créer une commande SQL. Sinon, le filtre restera vide. Si les zones de liste sont destinées à filtrer un formulaire, un contrôle masqué n'est pas utile. Dans ce cas, la valeur du filtre est stockée directement dans le formulaire.

```
	If Trim(aFiltre(1)) = "" Then
		If oChamp.getCurrentValue <> "" Then
```

```
                stFiltre = """"+Trim(aFiltre(0))+"""='"+oChamp.getCurrentValue()+"'"
```

Si un filtre existe déjà (par exemple celui traitant de ZoneListe_2, qui est en cours d'accès), le nouveau contenu est attaché au contenu précédent stocké dans le champ masqué.

```
            If oFormulaire.Filter <> ""
```

Cela ne doit se produire que lorsque le même champ n'a pas encore été filtré. Par exemple, si nous filtrons par Niveau, une répétition du filtre ne trouvera aucun enregistrement supplémentaire pour la zone de liste Nom. Une personne ne peut être trouvée que dans un niveau. Il faut donc exclure la possibilité que le nom du filtre ait déjà été utilisé.

```
                And InStr(oFormulaire.Filter, """"+Trim(aFiltre(0))+"""='") = 0 Then
                stFiltre = oFormulaire.Filter + " AND " + stFiltre
```

Si un filtre existe et que le champ qui sera utilisé pour le filtrage est déjà présent dans le filtre, le filtrage précédent sur ce nom de champ doit être supprimé et un nouveau filtre créé.

```
            ElseIf oForm.Filter <> "" Then
                stFiltre = Left(oFormulaire.Filter,
                InStr(oFormulaire.Filter, """"+Trim(aFiltre(0))+"""='")-1) + _
                    stFiltre
            End If
        End If
```

Ensuite, le filtre est entré dans le formulaire. Ce filtre peut également être vide si la première zone de liste a été sélectionnée et n'a pas de contenu.

```
        oFormulaire.Filter = stFiltre
        oFormulaire.reload()
```

La même procédure s'exécutera si le formulaire n'a pas besoin d'être filtré immédiatement. Dans ce cas, la valeur du filtre est stockée dans l'intervalle de temps dans un champ masqué.

```
    Else
        oChampCache = oFormulaire.getByName(Trim(aFiltre(1)))
        If oChamp.getCurrentValue <>"" Then
            stFiltre = """"+Trim(aFiltre(0))+"""='"+oChamp.getCurrentValue()+"'"
            If oChampCache.HiddenValue <> ""
            And InStr(oChampCache.HiddenValue, """"+Trim(aFiltre(0))+"""='") = 0
Then
                stFiltre = oChampCache.HiddenValue + " AND " + stFiltre
            ElseIf oChampCache.HiddenValue <> "" Then
                stFiltre = Left(oChampCache.HiddenValue,
                    InStr(oChampCache.HiddenValue, """"+Trim(aFiltre(0))+"""='")-1) +
_
                        stFiltre
            End If
        End If
        oChampCache.HiddenValue = stFiltre
    End If
```

Si les informations supplémentaires ont une entrée numérotée 4 (la numérotation commence à 0), la zone de liste suivante doit être réglée sur l'entrée correspondante de la zone de liste origine de l'appel.

```
    If UBound(aFiltre()) > 1 Then
        oChamp1 = oFormulaire.getByName(Trim(aFiltre(2)))
        aFiltre1() = Split(oChamp1.Tag,",")
```

Les données nécessaires au filtrage sont lues à partir des informations supplémentaires (étiquette) dans la zone de liste correspondante. Malheureusement, il n'est pas possible d'écrire uniquement le nouveau code SQL dans la zone de liste, puis de lire les valeurs de la zone de liste. Au lieu de cela, les valeurs correspondant à la requête doivent être écrites directement dans la zone de liste.

La création du code part du fait que la table à laquelle le formulaire se réfère est la même que celle à laquelle se réfèrent les zones de liste. Une telle zone de liste n'est pas conçue pour transférer des clés étrangères vers la table.

```
            If oChamp.getCurrentValue <> "" Then
                stSql = "SELECT DISTINCT """+Trim(aFiltre1(0))+""" FROM """+ _
                    oFormulaire.Command+ """ WHERE "+stFiltre+" ORDER BY """+ _
                    Trim(aFiltre1(0))+"""""
                oSourceDonnees = ThisComponent. Parent. CurrentController
                If Not (oSourceDonnees.isConnected()) Then
                    oSourceDonnees.connect()
                End If
                oConnexion = oSourceDonnees.ActiveConnection()
                oSQL_Instruction = oConnexion.createStatement()
                oResultat_Requete = oSQL_Instruction.executeQuery(stSql)
```

Les valeurs sont lues dans un tableau. Le tableau est transféré directement dans la zone de liste. Les indices correspondants pour le tableau sont incrémentés dans une boucle.

```
                inIndex = 0
                While oResultat_Requete.next
                    ReDim Preserve acontenu(inIndex)
                    acontenu(inIndex) = oResultat_Requete.getString(1)
                    inIndex = inIndex+1
                Wend
            Else
                acontenu(0) = ""
            End If
            oChamp1.StringItemList = acontenu()
```

Le contenu de la zone de liste a été recréé. Maintenant, il doit être relu. Ensuite, à l'aide de la propriété "*Complément d'information"* de la zone de liste qui a été actualisée, chacune des zones de liste dépendantes qui suivent est vidée, lançant une boucle pour toutes les zones de liste suivantes jusqu'à ce qu'une zone soit atteinte qui n'a pas de quatrième terme dans son Complément d'information.

```
            oChamp1.refresh()
            While UBound(aFiltre1()) > 1
                Dim aVide()
                oChamp2 = oFormulaire.getByName(Trim(aFiltre1(2)))
                Dim aFiltre1()
                aFiltre1() = Split(oChamp2.Tag,",")
                oChamp2.StringItemList = aVide()
                oChamp2.refresh()
            Wend
        End If
    End Sub
```

Le contenu visible des zones de liste est stocké dans `oChamp1.StringItemList`. Si une valeur supplémentaire doit être stockée pour être transmise à la table sous-jacente en tant que clé étrangère, comme d'habitude pour les zones de liste dans les formulaires, cette valeur doit être transmise à la requête séparément, puis stockée avec `oChamp1.ValueItemList`.

Une telle extension nécessite des variables supplémentaires telles que, en plus de la table dans laquelle les valeurs du formulaire doivent être stockées, la table à partir de laquelle est tiré le contenu de la zone de liste.

Une attention particulière doit être portée à la formulation du filtre.

```
stFiltre = """"+Trim(aFiltre(1))+"""='"+oChamp.getCurrentValue()+"'"
```

Ceci ne fonctionnera que si la version sous-jacente de LibreOffice est 4.1 ou ultérieure, car c'est la valeur à stocker qui est donnée comme `CurrentValue()`, et non la valeur affichée. Pour vous assurer qu'il fonctionne dans différentes versions, définissez Propriétés : Zone de liste> Données> Champ lié> '0'.

Saisie des temps en millisecondes

Afin de stocker les temps avec une précision de l'ordre de la milliseconde, il faut un champ d'horodatage dans la table, adapté séparément par une requête SQL à cet effet (voir « Création de table » – « Clés primaires [trois derniers paragraphes] au chapitre 3). Un tel champ peut être représenté sur un formulaire par un champ formaté au format **MM:SS,00**. Cependant, lors de la première tentative d'écriture, l'entrée d'enregistrement échouera. Cela peut être corrigé avec la macro suivante, qui doit être liée à la propriété « Avant l'action d'enregistrement » du formulaire :

```
SUB Horodatage
	Dim unoStmp As New com.sun.star.util.DateTime
	Dim oDoc As Object
	Dim oDrawpage As Object
	Dim oFormulaire As Object
	Dim oChamp As Object
	Dim stTemps As String
	Dim arMinEtSec()
	Dim arMandS()
	Dim loNano As Long
	Dim inSecond As Integer
	Dim inMinute As Integer
	oDoc = thisComponent
	oDrawpage = oDoc.Drawpage
	oFormulaire = oDrawpage.Forms.getByName("Formulaire")
	oChamp = oFormulaire.getByName("Temps")
	stTemps = oChamp.Text
```

Les variables sont déclarées en premier. Le reste du code est exécuté uniquement lorsque le champ de temps contient quelque chose. Sinon, le mécanisme interne du formulaire agira pour définir le champ sur **NULL**.

```
	If stTemps <> "" Then
	   arMinEtSec() = Split(stTemps,".")
	   loNano = CLng(ar(1)&"0000000")
	   arMinEtSec() = Split(ar(0),":")
	   inSeconde = CInt(arMinEtSec(1))
	   inMinute = Cint(arMinEtSec(0))
```

L'entrée dans le champ temps est décomposée en ses éléments.

Tout d'abord, la partie décimale est séparée et complétée à droite avec des caractères nuls jusqu'à un total de neuf chiffres. Un nombre aussi élevé ne peut être stocké que dans une variable Long.

Ensuite, le reste du temps est divisé en minutes et secondes, en utilisant les deux points comme séparateur, et ceux-ci sont convertis en nombres entiers.

```
            With unoStmp
                .NanoSeconds = loNano
                .Seconds = inSecond
                .Minutes = inMinute
                .Hours = 0
                .Day = 30    ' jour du mois
                .Month = 12
                .Year = 1899
            End With
```

Les valeurs d'horodatage sont attribuées à la date standard de LibreOffice du 30/12/1899. Bien sûr, la date actuelle réelle peut être stockée à côté.

```
            oChamp.BoundField.updateTimestamp(unoStmp)
    End If
    End Sub
```

Maintenant, l'horodatage que nous avons créé est transféré dans le champ à l'aide de **updateTimestamp** et stocké dans le formulaire.

Note

Obtenir et stocker la date actuelle :

```
Dim Maintenant As Date
      Maintenant = Now()
      With unoStmp
            .NanoSeconds = loNano
            .Seconds = inSecond
            .Minutes = inMinute
            .Hours = Hour(now)
            .Day = Day(now)
            .Month = Month(now)
            .Year = Year(now)
      End With
```

Dans les didacticiels précédents (notamment Apache Open Office), les **NanoSeconds** étaient appelées **HundrethSeconds**. Cela ne correspond pas à l'API LibreOffice et provoquera un message d'erreur.

Un événement – plusieurs implémentations

Lors de l'utilisation de formulaires, une macro liée à un seul événement peut être exécutée deux fois. Cela se produit parce que plusieurs processus sont liés simultanément, par exemple, au stockage d'un enregistrement modifié. Les différentes causes d'un tel événement peuvent être déterminées de la manière suivante :

```
Sub Determine_CauseEvenement(oEvent As Object)
      Dim oFormulaire As Object
      oFormulaire = oEvent.Source
      MsgBox oFormulaire.ImplementationName
End Sub
```

Lorsqu'un enregistrement modifié est stocké, deux implémentations sont impliquées, nommées **org.openoffice.comp.svx. FormController** et **com.sun.star.comp.forms. ODatabaseForm**. En

utilisant ces noms, nous pouvons nous assurer qu'une macro n'exécute son code qu'une seule fois. Une exécution en double provoque généralement juste une (petite) pause dans l'exécution du programme, mais cela peut conduire à ce qu'un curseur soit reculé de deux enregistrements au lieu d'un. Chaque implémentation n'autorise que des commandes spécifiques, il peut donc être important de connaître le nom de l'implémentation.

Enregistrer avec confirmation

Pour les modifications d'enregistrements complexes, il est logique de demander à l'utilisateur avant l'exécution si la modification doit réellement être effectuée. Si la réponse dans la boîte de dialogue est « Non », la sauvegarde est abandonnée, la modification est annulée et le curseur reste sur l'enregistrement en cours.

```
Sub Confirmation_Enregistre(oEvent As Object)
       Dim oFonctionFormulaire As Object
       Dim oOperationsFormulaire As Object
       Dim inReponse As Integer
       oFonctionFormulaire = com.sun.star.form.runtime.FormFeature
       Select Case oEvent.Source.ImplementationName
          Case "org.openoffice.comp.svx.FormController"
          inReponse = MsgBox("Faut-il modifier l'enregistrement ?",4, "Modification _
enregistrement")
             Select Case inReponse
                Case 6 ' Oui, aucune autre action
                Case 7 ' Non, interrompre l'enregistrement
                   oOperationsFormulaire = oEvent.Source.FormOperations

       oOperationsFormulaire.execute(oFonctionFormulaire.UndoRecordChanges)
                Case Else
             End Select
          Case "com.sun.star.comp.forms.ODatabaseForm"
       End Select
End Sub
```

Il existe deux moments de déclenchement avec des noms d'implémentation différents. Ces deux implémentations se distinguent dans **SELECT CASE**. Le code sera exécuté uniquement pour l'implémentation **FormController**. Cela est dû au fait que seul **FormController** a la variable **FormOperations**.

Outre Oui et Non, l'utilisateur peut également cliquer sur le bouton Fermer. Cela donne cependant la même valeur que Non, à savoir 7.

Si le formulaire est parcouru avec la touche de tabulation, l'utilisateur ne voit que la boîte de dialogue avec l'invite de confirmation. Cependant, les utilisateurs qui utilisent la barre de navigation verront également un message indiquant que l'enregistrement ne sera pas modifié.

Clé primaire à partir du numéro courant et de l'année

Lorsque les factures sont préparées, les soldes annuels sont affectés. Cela conduit souvent à une volonté de séparer les tableaux de facturation d'une base de données par année et de commencer une nouvelle table chaque année.

La solution de macro suivante utilise une méthode différente. Il écrit automatiquement la valeur du champ ID dans la table mais prend également en compte le champ « Annee » qui existe dans la

table comme clé primaire secondaire. Ainsi, les clés primaires suivantes peuvent apparaître dans le tableau :

Annee	*ID*
2014	1
2014	2
2014	3
2015	1
2015	2

De cette manière, un aperçu de l'année est plus facilement obtenu pour les documents.

```
Sub ID_et_Date_Courante
          Dim oSourceDonnees As Object
          Dim oConnexion As Object
          Dim oSQL_Commande As Object
          Dim stSql As String
          Dim oResultat As Object
          Dim oDoc As Object
          Dim oDrawpage As Object
          Dim oFormulaire As Object
          Dim oChamp1 As Object
          Dim oChamp2 As Object
          Dim oChamp3 As Object
          Dim inIDnouveau As Integer
          Dim inAnnee As Integer
          Dim unoDate
          oDoc = thisComponent
          oDrawpage = oDoc.drawpage
          oFormulaire = oDrawpage.forms.getByName("Formulaire")
          oChamp1 = oFormulaire.getByName("fmt_Annee")
          oChamp2 = oFormulaire.getByName("fmtID")
          oChamp3 = oFormulaire.getByName("dat_Date")
          If IsEmpty(oChamp2.getCurrentValue()) Then
              If IsEmpty(oChamp3.getCurrentValue()) Then
                  unoDate = createUnoStruct("com.sun.star.util.Date")
                  unoDate.Year = Year(Date)
                  unoDate.Month = Month(Date)
                  unoDate.Day = Day(Date)
                  inAnnee = Year(Date)
              Else
                  inAnnee = oChamp3.CurrentValue.Year
              End If
              oSourceDonnees = ThisComponent.Parent.CurrentController
              If Not (oSourceDonnees.isConnected()) Then
                  oSourceDonnees.connect()
              End If
              oConnexion = oSourceDonnees. ActiveConnection()
              oSQL_Commande = oConnexion.createStatement()
              stSql = "SELECT MAX(""ID"")+1 FROM ""orders"" WHERE ""year"" = '"+
                     inAnnee + "'"
              oResultat = oSQL_Commande.executeQuery(stSql)
              While oResultat.next
```

```
                inIDnouveau = oResultat.getInt(1)
            Wend
            If inIDnouveau = 0 Then
                inIDnouveau = 1
            End If
            oChamp1.BoundField.updateInt(inAnnee)
            oChamp2.BoundField.updateInt(inIDnouveau)
            If IsEmpty(oChamp3.getCurrentValue()) Then
                oChamp3.BoundField.updateDate(unoDate)
            End If
        End If
End Sub
```

Toutes les variables sont déclarées. Les contrôles de formulaire dans le formulaire principal sont accessibles. Le reste du code s'exécute uniquement si l'entrée du champ fmtID est toujours vide. Ensuite, si aucune date n'a été saisie, une structure de date est créée afin que la date et l'année en cours puissent être reportées dans les champs appropriés. Ensuite, une connexion est établie à la base de données, si elle n'existe pas déjà. La valeur la plus élevée du champ ID pour l'année en cours est incrémentée de 1. Si l'ensemble de résultats est vide, cela signifie qu'il n'y a aucune entrée dans le champ ID. À ce stade, 0 peut être entré dans le contrôle fmtID, mais la numérotation des commandes doit commencer à 1 afin que la variable inIDnouveau reçoive la valeur 1.

La valeur retournée pour l'année, l'ID et la date du jour (si aucune date n'a été saisie) sont transférés vers le formulaire.

Dans le formulaire, les champs de l'ID et de l'année des clés primaires sont protégés en écriture. Par conséquent, ils ne peuvent recevoir des valeurs qu'à l'aide de cette macro.

Cette macro est utilisée dans la base exemple Exemple_Numero_Serie_Annee.odb

Tâches de base de données développées à l'aide de macros

Établir une connexion à une base de données

```
oSourceDonnees = ThisComponent.Parent.DataSource
If Not oSourceDonnees.IsPasswordRequired Then
        oConnexion = oSourceDonnees.GetConnection("","")
```

Ici, il serait possible de fournir un nom d'utilisateur et un mot de passe, si nécessaire. Dans ce cas, les guillemets contiendraient ("Utilisateur", "MotdePasse"). Au lieu d'inclure le nom d'utilisateur et un mot de passe en texte clair, la boîte de dialogue de protection par mot de passe est appelée :

```
Else
        oAuthentication = createUnoService("com.sun.star.sdb.InteractionHandler")
        oConnexion = oSourceDonnees.ConnectWithCompletion(oAuthentication)
End If
```

Si toutefois un formulaire dans le même fichier de base accède à la base de données, il vous suffit de :

```
oSourceDonnees = ThisComponent. Parent.CurrentController
If Not (oSourceDonnees.isConnected()) Then
        oSourceDonnees.connect()
End If
```

```
oConnexion = oSourceDonnees.ActiveConnection()
```

Ici, la base de données est connue, donc un nom d'utilisateur et un mot de passe ne sont pas nécessaires, car ils sont déjà désactivés dans la configuration de base HSQLDB pour la version interne.

Pour les formulaires en dehors de Base, la connexion se fait via le premier formulaire :

```
oSourceDonnees = Thiscomponent.Drawpage.Forms(0)
oConnexion = oSourceDonnees.activeConnection
```

Copie de données d'une base de données à une autre

La base de données interne est une base de données mono-utilisateur. Les enregistrements sont stockés dans un fichier *.odb. L'échange de données entre différents fichiers de base de données n'était pas autorisé mais est néanmoins possible à l'aide de l'exportation et de l'importation.

Mais souvent, les fichiers *.odb sont configurés pour permettre l'échange automatique de données entre les bases de données. La procédure suivante peut être utile ici.

Une fois les variables déclarées, le chemin d'accès à la base de données actuelle est lu à partir d'un bouton du formulaire. Le nom de la base de données est séparé du reste du chemin. Le fichier cible des enregistrements est également présent dans ce dossier. Le nom de ce fichier est attaché au chemin pour permettre une connexion à la base de données cible.

La connexion à la base de données source est déterminée par rapport au formulaire qui contient le bouton : **ThisComponent.Parent.CurrentController**. La connexion à la base de données externe est établie à l'aide de **DatabaseContext** et du chemin.

```
Sub CopieDonnees
          Dim oContexteBase As Object
          Dim oSourceDonnees As Object
          Dim oSourceDonneesCible As Object
          Dim oConnexion As Object
          Dim oConnexionCible As Object
          Dim oDB As Object
          Dim oSQL_Commande As Object
          Dim oSQL_CommandeCible As Object
          Dim oResultat As Object
          Dim oResultatCible As Object
          Dim stSql As String
          Dim stSqlCible As String
          Dim inID As Integer
          Dim inIDCible As Integer
          Dim stNom As String
          Dim stVille As String
          Dim Drapeau As Boolean
          Drapeau = False
          oDB = ThisComponent.Parent
          stDir = Left(oDB.Location, Len(oDB.Location)-Len(oDB.Title))
          stDir = ConvertToUrl(stDir & "BDCible.odb")
          oSourceDonnees = ThisComponent.Parent.CurrentController
          If Not (oSourceDonnees.isConnected()) Then
             oSourceDonnees.connect()
          End If
          oConnexion = oSourceDonnees.ActiveConnection()
          oContexteBase = createUnoService("com.sun.star.sdb.DatabaseContext")
          oSourceDonneesTarget = oContexteBase.getByName(stDir)
```

```
            oConnexionTarget = oSourceDonneesTarget.GetConnection("","")
            oSQL_Commande = oConnexion.createStatement()
            stSql = "SELECT * FROM ""Table"""
            oResultat = oSQL_Commande.executeQuery(stSql)
            While oResultat.next
                inID = oResultat.getInt(1)
                stNom = oResultat.getString(2)
                stVille = oResultat.getString(3)
                oSQL_CommandeCible = oConnexionTarget.createStatement()
                stSqlCible = "SELECT ""ID"" FROM ""Table"" WHERE ""ID"" = '"+inID+"'"
                oResultatCible = oSQL_CommandeCible.executeQuery(stSqlCible)
                inIDCible = - 1
                While oResultatCible.next
                    inIDCible = oResultatCible.getInt(1)
                Wend
                If inIDCible = - 1 Then
                    stSqlCible = "INSERT INTO ""Table"" (""ID"",""Nom"",""Ville"") _
                     VALUES ('"+inID+"','"+stNom+"','"+stVille+"')"
                    oSQL_CommandeCible.executeUpdate(stSqlCible)
                    Drapeau = True
                End If
            Wend
            If Drapeau Then Msgbox("Données copiées") Else Msgbox ("Données non
copiées")
End Sub
```

Les tables complètes de la base de données source sont lues et insérées, ligne par ligne, dans la table de la base de données cible à l'aide de la connexion qui a été établie. Avant l'insertion, une vérification est effectuée pour voir si une valeur a déjà été définie pour la clé primaire dans la table cible. Si tel est le cas, l'enregistrement n'est pas copié.

Il est également possible de faire en sorte qu'au lieu de copier un nouvel enregistrement, un enregistrement existant soit mis à jour. Dans tous les cas, cela garantit que la base de données cible contient des enregistrements avec la clé primaire correcte de la base de données source.

Cette macro est utilisée dans la base Exemple_CopieDonnees_Source_Cible.odb.

Importation directe des données de Calc

Il arrive souvent que Calc soit utilisé à la place de Base pour entrer des données dans un tableau. Ces données peuvent ensuite être copiées dans une table de base de données via le presse-papiers ou par glisser-déposer. Il est également possible d'exporter dans un fichier texte *.csv déjà inclus dans Base.

Mais, l'utilisateur peut aussi vouloir transférer simplement les données saisies sous Calc dans une base de données, afin de profiter des fonctionnalités d'un SGDB. C'est là qu'entre en jeu la macro suivante, qui est lancée à partir d'un formulaire via un bouton. [24]

La macro suppose les conditions suivantes :

- Les données doivent se trouver sur la « **feuille1** » du document Calc ;
- Sur cette feuille de travail les données sont disposées en colonnes ;
- La première ligne contient les noms de champs, qui doivent correspondre exactement aux noms des champs de la table dans la base de données. Ils peuvent être classés dans un ordre différent ;

24 Voir la base de données "Exemple_Import_Données_Calc" associée à ce manuel.

- Les colonnes dont les en-têtes ne correspondent pas aux champs du tableau sont ignorées.

Les noms des variables ont volontairement été laissés en langue anglaise. Les commentaires et noms des éléments utilisés ont été traduits en français.

```
Sub ImportDonneesCalc(oEvent As Object)
          Dim oSourceDonnees As Object
          Dim oConnexion As Object
          Dim oSQL_Commande As Object
          Dim oResultat As Object
          Dim oDB As Object
          Dim oDoc As Object
          Dim oDocView As Object
          Dim oEtendue As Object
          Dim Arg()
          Dim aColonne()
          Dim aColonnes()
          Dim aType()
          Dim stSql As String
          Dim stLigne As String
          Dim stDir As String
          Dim stColonnes As String
          Dim stColonneDebut As String
          Dim stColonneFin As String
          Dim stLigneDebut As String
          Dim stLigneFin As String
          Dim stEtendue As String
          Dim inCompteur As Integer
          Dim loColonnes As Long
          Dim loLignes As Long
          Dim loID As Long
          oSourceDonnees = ThisComponent.Parent.CurrentController
          If Not (oSourceDonnees.isConnected()) Then
             oSourceDonnees.connect()
          End If
          oConnexion = oSourceDonnees.ActiveConnection()
          oSQL_Commande = oConnexion.createStatement()
```

Après avoir déclaré les variables et vérifié la connexion à la source de données, lit les noms et les types de colonnes de la table dans la base de données. Sauf "ID" qui est une clé primaire qui n'existe que dans cette table en tant que clef primaire.

```
          stSql = "SELECT COLUMN_NAME, TYPE_NAME FROM INFORMATION_SCHEMA. _
             SYSTEM_COLUMNS WHERE TABLE_NAME = 'Table' AND NOT COLUMN_NAME = 'ID'"
          oResultat = oSQL_Commande.executeQuery(stSql)
          inCompteur = 0
          stColonnes = ""
```

Note

Pour Firebird, le code SQL doit être adapté :

```
stSql = "SELECT A.RDB$FIELD_NAME, C.RDB$TYPE_NAME
       FROM RDB$RELATION_FIELDS AS A, RDB$FIELDS AS B, RDB$TYPES AS C
       WHERE A.RDB$FIELD_SOURCE = B.RDB$FIELD_NAME
              AND B.RDB$FIELD_TYPE = C.RDB$TYPE
              AND C.RDB$FIELD_NAME = 'RDB$FIELD_TYPE'
              AND A.RDB$RELATION_NAME = 'Table'
              AND NOT A.RDB$FIELD_NAME = 'ID'"
```

Les noms et les types de colonnes sont lus et stockés dans des tableaux séparés. Cela fonctionnerait également dans un tableau bidimensionnel.

```
WHILE oResultat.next
   ReDim Preserve aColonne(inCompteur)
   ReDim Preserve aType(inCompteur)
   aColonne(inCompteur) = oResultat.getString(1)
   aType(inCompteur) = oResultat.getString(2)
   inCompteur = inCompteur+1
WEND
```

La valeur actuelle, la plus élevée, de la clé primaire « **ID** » est déterminée et incrémentée de 1. Dans l'exemple, la valeur de la clé n'est pas automatiquement incrémentée par HSQLDB, mais gérée par la macro.

```
stSql = "SELECT MAX(""ID"") FROM ""Tabe"""
oResultat = oSQL_Commande.executeQuery(stSql)
WHILE oResultat.next
   loID = oResultat.getInt(1) + 1
WEND
```

Le chemin d'accès au fichier Calc est déterminé, ici dans le même répertoire que le fichier de base de données exemple. Ce fichier est ouvert, mais rendu invisible afin qu'il ne passe pas au premier plan.

```
oDB = ThisComponent.Parent
globalscope.basicLibraries.loadLibrary("Tools") ' bibliothèque de fonctions
       stDir = DirectoryNameoutofPath(ConvertFromURL(odb.location), _
GetPathSeparator())
       stDir = Left(oDB.Location,Len(oDB.Location)-Len(oDB.Title))
       stDir = ConvertToUrl(stDir & "Donnees_Calc.ods")
       oDoc = StarDesktop.loadComponentFromURL(stDir,"_blank", 0, Arg() )
       oDocView = oDoc.CurrentController.Frame.ContainerWindow
       oDocView.Visible = False
```

La zone utile de cellules de la feuille de calcul est déterminée. Cela se fait via **ColumnDescriptions** et **RowDescriptions**.

```
       loColonnes = uBound(oDoc.Sheets(0).ColumnDescriptions)     ' Nombre de colones
       stColonneDebut = split(oDoc.Sheets(0).ColumnDescriptions(0))(1)
       stColonneFin = split(oDoc.Sheets(0).ColumnDescriptions(loColonnes))(1)
       loLignes = uBound(oDoc.Sheets(0).RowDescriptions)    ' Nombre de lignes
       stLigneDebut = split(oDoc.Sheets(0).RowDescriptions(0))(1)
       stLigneFin = split(oDoc.Sheets(0).RowDescriptions(loLignes))(1)
       stEtendue = stColonneDebut & stLigneDebut & ":" & stColonneFin & stLigneFin
```

La zone (**stEtendue**) est composée comme une chaîne. Cela se fait de la même manière que dans Calc, par exemple "A1:C7". Les données d'une telle plage peuvent être lues avec la fonction GetDataArray().

```
       oEtendue = oDoc.Sheets(0).getCellRangeByName(stEtendue)
```

```
aDonnees = oEtendue.getDataArray()
```

Les noms des colonnes se trouvent sur la première ligne. Ils peuvent avoir un ordre différent de celui prévu dans le tableau de la base. Ces noms sont donc stockés dans un tableau séparé pour une comparaison ultérieure. Elles ne sont pas interrogées dans la boucle suivante en tant que valeurs à lire dans le tableau. Par conséquent, la boucle commence avec i=1.

```
aColonnes = aDonnees(0)
FOR i = 1 TO uBound(aDonnees)
    aLigne = aDonnees(i)
    stColonnes = """ID"""
    stLigne = loID
    FOR k = 0 TO loColonnes
        FOR n = 0 TO uBound(aColonne)
```

Dans ce qui suit, les noms des colonnes de Calc sont comparés avec ceux du tableau de base. En plus des colonnes de texte, le tableau de base contient ici aussi une colonne pour un montant en devise, qui est défini comme DECIMAL, et une date.

Pour le montant en devises, il peut être nécessaire de convertir le point décimal. C'est pourquoi ici la fonction Cdbl est couplée à la fonction Str.

Dans Calc une cellule au format date est enregistrée sous la forme d'un de numéro ISO. Ce code doit d'abord être vérifié pour voir si une date peut être formée à partir de celui-ci. Si cela est possible, une date conforme à SQL est compilée dans le format AAAA-MM-JJ, même avec des valeurs de jour et de mois à un chiffre.

```
            IF aColonnes(k) = aColonne(n) THEN
                IF aType(n) = "DECIMAL" THEN
                    IF aLigne(k) <> "" THEN
                        aLigne(k) = Str(CDbl(aLigne(k)))
                    END IF
                END IF
                IF aType(n) = "DATE" THEN
                    IF isDate(CDate(aLigne(k))) AND aLigne(k) <> "" THEN
                        aLigne(k) = Year(CDate(aLigne(k)))&"-"
                                &Right("0"&Month(CDate(aLigne(k))),2)
                                &"-"&Right("0"&Day(CDate(aLigne(k))),2)
                    ELSE
                        aLigne(k) = ""
                    END IF
                END IF
```

Si le contenu de la cellule est vide ou non valable pour un champ décimal ou un champ de date, selon le cas, NULL est transmis à la table de la base de données. Sinon, les contenus sont placés entre guillemets. Les contenus sont ensuite reliés entre eux à l'aide de virgules. Les colonnes sont également nommées dans l'ordre dans lequel elles ont été lues à partir de la table Calc.

```
                IF aLigne(k) = "" THEN
                    aLigne(k) = "NULL"
                ELSE
                    aLigne(k) = "'" & aLigne(k) & "'"
                END IF
                stLigne = stLigne & "," & aLigne(k)
                stColonnes = stColonnes & ",""" & aColonnes(k) & """"
            END IF
        NEXT
```

```
            NEXT
            stSql = "INSERT INTO ""Tabelle"" ("& stColonnes &") VALUES ("& stLigne &")"
            oSQL_Commande.executeUpdate(stSql)
```

La valeur de la clé primaire est augmentée de 1 dans la macro.

```
            loID = loID + 1
        NEXT
        oDoc.close(True)
        oEvent.Source.Model.parent.reload()
End Sub
```

Une fois que le contenu de la feuille de calcul Calc a été entièrement lu, le document Calc (invisible) est fermé. Le formulaire est actualisé (rechargé – reload) à partir de l'événement déclencheur initial (oEvent) par oEvent.Source.Model.parent.reload().

Accès aux requêtes

Il est plus facile de créer des requêtes dans l'interface utilisateur graphique que de transférer leur texte dans des macros, avec la complication supplémentaire que des doubles guillemets sont obligatoires pour tous les noms de table et de champ.

```
Sub aContenuRequete
       Dim oFichierDonnees As Object
       Dim oRequete As Object
       Dim stRequete As String
       oFichierDonnees = ThisComponent.Parent.CurrentController.DataSource
       oRequete = oFichierDonnees.getQueryDefinitions()
       stRequete = oRequete.getByName("Requete").Command
       MsgBox stRequete
End Sub
```

Ici, le contenu d'un fichier *.odb est accessible à partir d'un formulaire. La requête est atteinte en utilisant **getQueryDefinitions()**. Le code SQL de la requête se trouve dans son champ **Command**. Cela peut ensuite être utilisé pour utiliser la commande plus loin dans une macro.

Lorsque vous utilisez le code SQL de la requête, vous devez veiller à ce que le code ne fasse pas référence à une autre requête. Cela conduit inévitablement au message que la table (apparente) de la base de données est inconnue. Pour cette raison, il est plus simple de créer des vues à partir de requêtes, puis d'accéder aux vues dans la macro.

Sécuriser votre base de données

Il peut parfois arriver, en particulier lors de la création d'une base de données, que le fichier ODB soit tronqué de manière inattendue. Une sauvegarde fréquente après édition est donc utile, en particulier lors de l'utilisation du module Rapports.

Lorsque la base de données est en cours d'utilisation peut être endommagée par une défaillance du système d'exploitation, notamment si cela se produit juste au moment où le fichier de base est fermé. C'est à ce moment que le contenu de la base de données est écrit dans le fichier.

De plus, il existe les soupçons habituels de panne pour les fichiers qui refusent soudainement de s'ouvrir, comme une panne de disque dur. Il ne fait donc aucun mal d'avoir une copie de sauvegarde la plus à jour possible. L'état des données ne change pas tant que le fichier ODB reste ouvert. Pour cette raison, les sous-programmes de sécurité peuvent être directement liés à

l'ouverture du fichier. Copier simplement le fichier en utilisant le chemin de sauvegarde indiqué dans **Outils> Options> LibreOffice> Chemins**. La macro commence par écraser la version la plus ancienne après un nombre spécifique de copies (**inMax**).

```
Sub SauveBase(inMax As Integer)
	Dim oChemin As Object
	Dim oDoc As Object
	Dim sTitre As String
	Dim sUrl_Destination As String
	Dim sUrl_Source As String
	Dim i As Integer
	Dim k As Integer
	oDoc = ThisComponent
	sTitre = oDoc.Title
	sUrl_Source = oDoc.URL
	Do While sUrl_Source = ""
	    oDoc = oDoc.Parent
	    sTitre = oDoc.Title
	    sUrl_Source = oDoc.URL
	Loop
```

Si la macro est exécutée lorsque vous lancez le fichier ODB, sTitre et sUrl_Source seront corrects. Cependant, si la macro est réalisée par un formulaire, il doit d'abord déterminer si une URL est disponible. Si l'URL est vide, un niveau supérieur (oDoc.Parent) est recherché.

```
	oChemin = createUnoService("com.sun.star.util.PathSettings")
	For i = 1 To inMax + 1
	    If Not FileExists(oChemin.Backup & "/" & i & "_" & sTitre) Then
	        If i > inMax Then
	            For k = 1 To inMax - 1 To 1 Step -1
	            If FileDateTime(oChemin.Backup & "/" & k & "_" & sTitre) <= _
	            FileDateTime(oChemin.Backup & "/" & k+1 & "_" & sTitre) Then
	                If k = 1 Then
	                      i = k
	                      Exit For
	                End If
	            Else
	                i = k + 1
	                Exit For
	            End If
	            Next
	        End If
	        Exit For
	    End If
	Next
	sUrl_Destination = oChemin.Backup & "/" & i &"_" & sTitre
	FileCopy(sUrl_Source, sUrl_Destination)
End Sub
```

Vous pouvez également effectuer une sauvegarde pendant l'exécution de Base, à condition que les données puissent être réécrites du cache dans le fichier avant l'exécution du sous-programme SauveBase. Il peut être utile de le faire, peut-être après un certain temps ou lorsqu'un bouton à l'écran est enfoncé. Cet effacement du cache est géré par le sous-programme suivant :

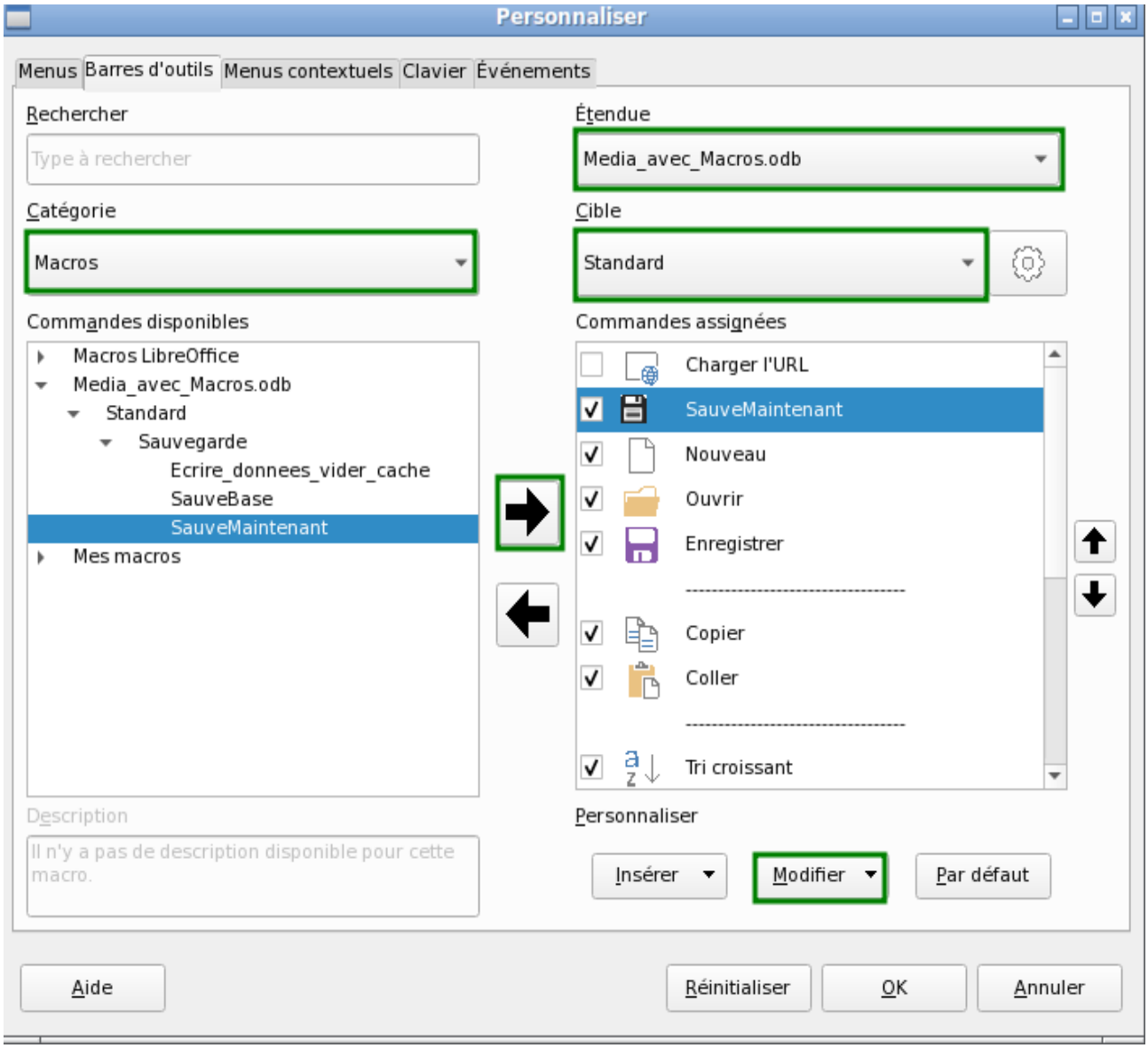
```
Sub Ecrire_donnees_vider_cache
	Dim oDonnees As Object
	Dim oSourceDonnees As Object
	oDonnees = ThisDatabaseDocument.CurrentController
```

```
        If Not (oDonnees.isConnected()) Then oDonnees.connect()
        oSourceDonnees = oDonnees.DataSource
        oSourceDonnees.flush
End Sub
```

Si tout cela doit être lancé à partir d'un seul bouton sur un formulaire, les deux procédures doivent être appelées par une autre procédure :

```
Sub SauveMaintenant
        Ecrire_donnees_vider_cache
        SauveBase(10)
End Sub
```

Surtout pour une macro de sécurité, il peut être judicieux de rendre la macro accessible via la barre d'outils de la base de données. Cela se fait dans la fenêtre principale du fichier de base en utilisant **Outils> Personnaliser> Barres d'outils**.



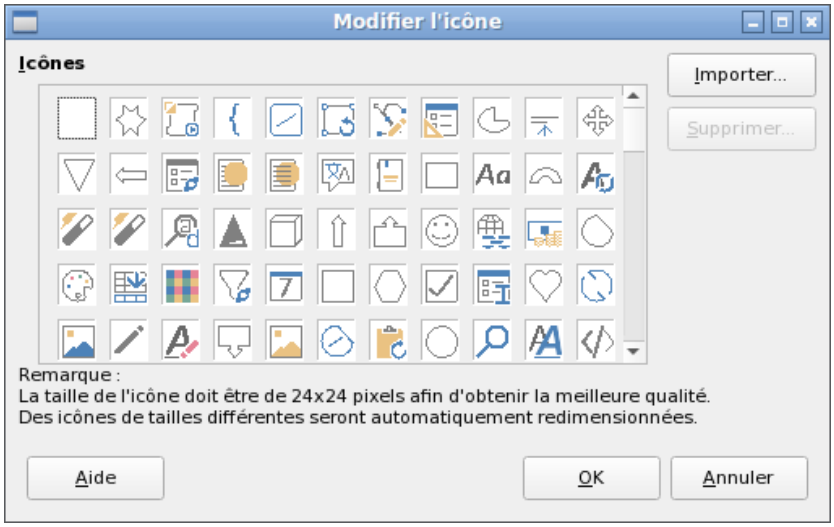
Dans la boîte de dialogue affichée sélectionner :

- dans la liste déroulante « Étendue », le nom du fichier contenant les macros, dans l'exemple « Media_avec_Macros.odb » ;

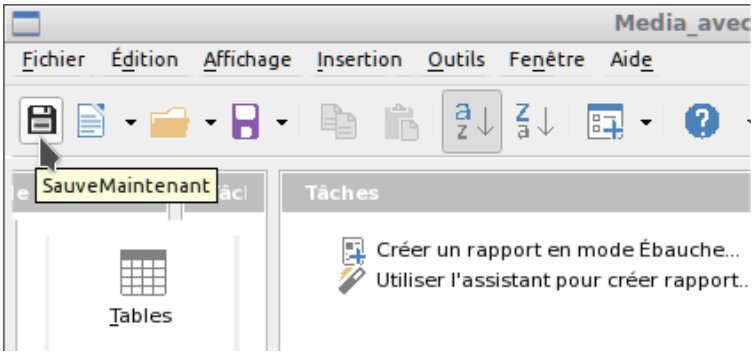
- dans « Catégorie », la macro « SauveMaintenant » ;
- dans « Cible », la barre d'outil « Standard », qui est toujours affichée dans toutes les parties de Base ;
- l'endroit où insérer la nouvelle commande en choisissant par un clic, dans « Commandes assignées », sur celle (existante) qui devra se trouver juste avant « SauveMaintenant ». Sans cela l'insertion se fera à la fin de la liste ;
- enfin, cliquez sur le bouton indiqué sur l'illustration pour transférer cette nouvelle fonction.

La commande est désormais disponible pour une utilisation dans la barre d'outils « Standard ».

Pour affecter une icône à la commande, choisissez **Modifier> Modifier l'icône** pour ouvrir la boîte de dialogue suivante.



Sélectionnez une icône appropriée. Vous pouvez également créer et ajouter votre propre icône.



L'icône apparaît maintenant à la place du nom de la procédure. Le nom devient une info-bulle.

Pour exécuter la procédure, cliquez simplement sur l'icône dans la barre d'outils.

Note

Jusqu'à la version 6.4.7, les icônes personnalisées disparaissent après la fermeture de LibO et sont remplacées par le titre de la macro. Ce bug est corrigé dans LO 7.0.3

Compacter les bases de données

Il s'agit simplement d'une commande SQL (**SHUTDOWN COMPACT**), qui doit être exécutée de temps en temps, surtout après la suppression de nombreuses données. La base de données stocke les nouvelles données, mais réserve toujours l'espace pour les données supprimées. Dans

les cas où les données ont été substantiellement modifiées, vous devez donc compacter la base de données.

Note

Depuis LibreOffice version 3.6, cette commande est automatiquement exécutée pour HSQLDB interne lorsque la base de données est fermée. Par conséquent, cette macro n'est plus nécessaire pour la base de données interne.

Une fois le compactage effectué, les tables ne sont plus accessibles. Le fichier doit être rouvert. Par conséquent, cette macro ferme le formulaire à partir duquel elle est appelée. Malheureusement, vous ne pouvez pas fermer le document lui-même sans provoquer de récupération lors de sa réouverture. Par conséquent, cette fonction fermeture est commentée.

```
Sub Compactage_BaseDeDonnees
	Dim stMessage As String
	oSourceDonnees = ThisComponent.Parent.CurrentController    ' Accessible à
partir du formulaire
	If Not (oSourceDonnees.isConnected()) Then
	   oSourceDonnees.connect()
	End If
	oConnexion = oSourceDonnees.ActiveConnection()
	oSQL_Statement = oConnexion.createStatement()
	stSql = "SHUTDOWN COMPACT"' La base de données sera compactée et fermée
	oSQL_Statement.executeQuery(stSql)
	stMessage = "La base de données est en cours de compactage." + Chr(13) + "Le
formulaire va maintenant se fermer."
	stMessage = stMessage + Chr(13) + "Ensuite, le fichier de base de données doit
être fermé."
	stMessage = stMessage + Chr(13) + "La base de données n'est accessible qu'après
la réouverture du fichier de base de données."
	MsgBox stMessage
	ThisDatabaseDocument.FormDocuments.getByName("Maintenance").close
	REM La fermeture du fichier de base de données entraîne une opération
REM de récupération lorsque vous l'ouvrez à nouveau.
	' ThisDatabaseDocument.close(True)
End Sub
```

Diminution de l'index de table pour les champs à valeur automatique

Si beaucoup de données sont supprimées d'une table, les utilisateurs craignent souvent que la séquence de clés primaires générées automatiquement continue simplement vers le haut au lieu de recommencer à la valeur actuelle la plus élevée de la clé. Le sous-programme suivant lit la valeur actuellement la plus élevée du champ ID dans une table et définit la valeur de clé initiale à 1 de plus que ce maximum[25].

Si le champ de clé primaire n'est pas appelé ID, la macro doit être modifiée en conséquence.

```
Sub Diminue_index_Table(stTable As String)
REM Ce sous-programme définit le champ de clé primaire à incrémentation
REM automatique (AutoChamp) avec le nom prédéfini "ID" sur la valeur
REM la plus basse possible.
	Dim inNombre As Integer
	Dim inValeur_Sequence As Integer
```

25 Cette procédure est utilisée dans le formulaire Maintenance de l'exemple Media_avecMacros.

```
        oSourceDonnees = ThisComponent.Parent.CurrentController 'Accessible via le
formulaire
        If Not (oSourceDonnees.isConnected()) Then
            oSourceDonnees.connect()
        End If
        oConnexion = oSourceDonnees.ActiveConnection()
        oSQL_Statement = oConnexion.createStatement()
' La valeur la plus élevée de "ID" est déterminée
        stSql = "SELECT MAX(""ID"") FROM """+stTable+"""""
        Requete = oSQL_Instruction.executeQuery(stSql)
' La requête est lancée et la valeur de retour stockée dans la variable
' oResultat_Requete
        If Not IsNull(oResultat_Requete) Then
            While oResultat_Requete.next
            ' Le premier champ de données est lu
            inNombre = oResultat_Requete.getInt(1)
            'enregistrement suivant, dans ce cas aucun car un seul
            ' enregistrement existe
            Wend
            ' Si la valeur la plus élevée n'est pas une valeur, ce qui signifie
            ' que la table est vide, la valeur la plus élevée est définie sur -1
            If inNombre = "" Then
                inNombre = -1
            End If
' La valeur la plus élevée est augmentée de 1
            inValeur_Sequence = inNombre+1
            REM Une nouvelle commande est préparée pour la base de données.
            REM L'ID recommencera à partir de inNombre + 1.
            REM Cette instruction n'a pas de valeur de retour,
            REM car aucun enregistrement n'est en cours de lecture
            oSQL_statement = oConnexion.createStatement()
            oSQL_statement.executeQuery("ALTER TABLE """ + stTable + """ _
ALTER COLUMN  ""ID"" RESTART WITH " + inValeur_Sequence + "")
        End If
End Sub
```

Impression depuis Base

La méthode standard pour obtenir un document imprimable depuis Base est d'utiliser un rapport. Alternativement, les tables et les requêtes peuvent être copiées dans Calc et préparées pour être imprimées. Bien entendu, l'impression directe d'un formulaire à partir de l'écran est également possible.

Impression d'un rapport à partir d'un formulaire interne

Normalement, la génération des rapports se fait à partir de l'interface utilisateur de base. Un clic sur le nom du rapport lance la préparation du rapport. Ce serait bien sûr plus facile si le rapport pouvait être lancé directement à partir d'un formulaire.

```
Sub LanceRapport
        ThisDatabaseDocument.ReportDocuments.getByName("Rapport").open
End Sub
```

Tous les rapports sont accessibles par nom depuis leur conteneur **ReportDocuments**. Ils sont ouverts avec l'intruction **open**. Si un rapport est lié à une requête filtrée via le formulaire, cette méthode permet à l'enregistrement en cours d'être imprimé.

Lancement, formatage, impression directe et fermeture d'un rapport

Ce serait encore plus agréable si le rapport pouvait être envoyé directement à l'imprimante. La combinaison de procédures suivante ajoute quelques petites fonctionnalités. Il sélectionne d'abord l'enregistrement actif dans le formulaire, reformate le rapport afin que les champs de texte soient définis automatiquement à la bonne hauteur, puis lance le rapport. Enfin, le rapport est imprimé et éventuellement stocké sous forme de pdf. Et tout cela se produit presque complètement en arrière-plan, car le rapport passe à invisible directement après l'ouverture du formulaire et se referme après l'impression. Des suggestions pour les différentes procédures ont été faites par Andrew Pitonyak, Thomas Krumbein et Lionel Elie Mamane.

```
Sub DemarreRapport(oEvent As Object)
	Dim oFormulaire As Object
	Dim stSql As String
	Dim oSourceDonnees As Object
	Dim oConnexion As Object
	Dim oSQL_commande As Object
	Dim oRapport As Object
	Dim oRapportVue As Object
	oFormulaire = oEvent.Source.model.parent
	stSql = "UPDATE ""Filter"" SET ""Integer"" = '" +
	   oFormulaire.getInt(oFormulaire.findColumn("ID")) + "' WHERE ""ID"" = TRUE"
	oSourceDonnees = ThisComponent. Parent. CurrentController
	If Not (oSourceDonnees.isConnected()) Then
	   oSourceDonnees.connect()
	End If
	oConnexion = oSourceDonnees.ActiveConnection()
	oSQL_commande = oConnexion.createStatement()
	oSQL_commande.executeUpdate(stSql)
	oRapport = ThisDatabaseDocument.ReportDocuments.getByName("Reportname").open
	oRapportVue = oRapport.CurrentController.Frame.ContainerWindow
	oRapportVue.Visible = False
	HauteurLigneRapportAuto(oRapport)
End Sub
```

La procédure DemarreRapport est liée à un bouton du formulaire. À l'aide de ce bouton, la clé primaire de l'enregistrement actuel peut être lue. À partir de l'événement qui lance la macro, nous pouvons accéder au formulaire (oFormulaire). Le nom du champ de clé primaire est donné ici sous la forme « ID ». En utilisant **oFormulaire.getInt (oFormulaire.findColumn ("ID")**), la clé est lue dans le champ sous la forme d'un entier. Cette valeur est stockée dans une table de filtres. La table de filtrage contrôle une requête pour garantir que seul l'enregistrement actuel sera utilisé pour le rapport.

Le rapport peut être ouvert sans référence au formulaire. Il est alors accessible en tant qu'objet (oRapport). La fenêtre de rapport est rendue invisible. Malheureusement, il ne peut pas être invisible lorsqu'il est appelé, il apparaît donc brièvement, puis est rempli avec le contenu approprié en arrière-plan.

Ensuite, la procédure *HauteurLigneRapportAuto* est lancée. Cette procédure reçoit une référence au rapport ouvert en tant qu'argument.

La hauteur de la ligne d'enregistrement peut être réglée automatiquement au moment de l'impression. S'il y a probablement trop de texte dans un champ particulier, le texte est tronqué et le reste est indiqué par un triangle rouge. Lorsque cela ne fonctionne pas, la procédure suivante

garantira que dans toutes les tables portant le nom Détail, le contrôle automatique de la hauteur sera activé.

```
Sub HauteurLigneRapportAuto(oRapport As Object)
	Dim oTables As Object
	Dim oTable As Object
	Dim inT As Integer
	Dim inI As Integer
	Dim oLignes As Object
	Dim oLigne As Object
	oTables = oRapport.getTextTables()
	For inT = 0 TO oTables.count() - 1
	   oTable = oTables.getByIndex(inT)
	   If Left$(oTable.name, 6) = "Detail" Then
	      oLignes = oTable.Rows
	      For inI = 0 To oLignes.count - 1
	         oLigne = oLignes.getByIndex(inI)
	         oLigne.IsAutoHeight = True
	      Next inI
	   End If
	Next inT
	ImprimeFermeRapport(oRapport)
End Sub
```

Lors de la création du rapport, il faut veiller à ce que tous les champs de la même ligne de la section « Détail » aient la même hauteur. Sinon, le contrôle automatique de la hauteur peut soudainement définir une ligne pour doubler la hauteur.

Une fois que toutes les tables portant le nom « Detail » ont été réglées automatiquement en hauteur, le rapport est envoyé à l'imprimante par la procédure ImprimeFermeRapport.

Le tableau « Props » contient les valeurs associées à une imprimante dans un document. Pour la commande d'impression, le nom de l'imprimante par défaut est important. Le rapport doit rester ouvert jusqu'à ce que l'impression soit réellement terminée. Ceci est garanti en donnant le nom de l'imprimante et la commande « Attendre que j'aie fini » (**Wait**) comme arguments.

```
Sub ImprimeFermeRapport(oRapport As Object)
	Dim Props
	Dim stImprimante As String
	Props = oRapport.getPrinter()
	stImprimante = Props(0).value
	Dim arg(1) As New com.sun.star.beans.PropertyValue
	arg(0).name = "Name"
	arg(0).value = "<" & stImprimante & ">"
	arg(1).name = "Wait"
	arg(1).value = True
	oRapport.print(arg())
	oRapport.close(true)
End Sub
```

Ce n'est que lorsque l'impression a été complètement envoyée à l'imprimante que le document est fermé.

Pour les paramètres de l'imprimante, consultez la section Imprimante et paramètres d'impression du wiki.

Si, à la place (ou en plus) d'une impression, vous voulez un pdf du document comme copie de sécurité, la méthode **storeToURL**() peut être utilisée :

```
Sub EnregistreRapportPDF(oRapport As Object)
	Dim stUrl As String
	Dim arg(0) As New com.sun.star.beans.PropertyValue
	arg(0).name = "FilterName"
	arg(0).value = "writer_pdf_Export"
	stUrl = "file:///…"
	oRapport.storeToURL(stUrl, arg())
End Sub
```

L'URL doit bien sûr être une adresse URL complète. Mieux encore, cette adresse doit être liée à un enregistrement permanent du document imprimé tel qu'un numéro de facture. Sinon, il se peut qu'un fichier important soit simplement écrasé par l'impression suivante.

Impression de rapports à partir d'un formulaire externe

Il y a des problèmes lorsque des formulaires externes sont utilisés. Les rapports se trouvent dans le fichier *.odb et ne sont pas disponibles à l'aide du navigateur de sources de données.

```
Sub LanceRapportExterne(oEvent As Object)
	Dim oChamp As Object
	Dim oFormulaire As Object
	Dim oDocument As Object
	Dim oDocVue As Object
	Dim Arg()
	oChamp = oEvent.Source.Model
	oFormulaire = oChamp.Parent
	sURL = oFormulaire.DataSourceName
	oDocument = StarDesktop.loadComponentFromURL(sURL, "_blank", 0, Arg())
	oDocVue = oDocument.CurrentController.Frame.ContainerWindow
	oDocVue.Visible = False
	oDocument.getCurrentController().connect
	Wait(100)
	oDocument.ReportDocuments.getByName("Rapport").open
	oDocument.close(True)
End Sub
```

Le rapport est lancé à partir d'un bouton sur le formulaire externe. Le bouton indique au formulaire le chemin d'accès au fichier *.odb : **oForm. DataSourceName**. Ensuite, le fichier est ouvert à l'aide de **loadComponentFromUrl**. Le fichier doit rester en arrière-plan afin que la vue du document soit accessible et que l'interface soit définie sur **Visible = False**. Idéalement, cela aurait dû être fait directement en utilisant la liste d'arguments **Arg()**, mais les tests montrent que cela ne donne pas le résultat correct.

Le rapport ne peut pas être appelé immédiatement à partir du document ouvert, car la connexion n'est pas encore prête. Le rapport apparaît avec un arrière-plan gris, puis LibreOffice plante. Une courte attente de 100 millisecondes résout ce problème. Des tests pratiques sont nécessaires pour déterminer le temps d'attente minimum. Le rapport est maintenant lancé. Comme le rapport sera dans un fichier texte séparé, le fichier *.odb ouvert peut être refermé. La méthode **oDocument.close (True)** transmet cette instruction au fichier *.odb. Le fichier ne sera fermé que lorsqu'il n'est plus actif, c'est-à-dire qu'aucun autre enregistrement ne doit être transmis au rapport.

Un accès similaire peut être lancé à partir de formulaires dans le fichier *.odb, mais dans ce cas, le document ne doit pas être fermé.

Vous pouvez obtenir des impressions de bonne qualité beaucoup plus rapidement qu'avec le Générateur de rapports en utilisant des macros associées à la fonction de publipostage ou à des champs de texte.

Faire un publipostage depuis Base

Parfois, un rapport est tout simplement insuffisant pour produire des lettres de bonne qualité aux destinataires. Les champs de texte d'un rapport sont d'une utilité très limitée dans la pratique. Au lieu de cela, une lettre de publipostage peut être créée dans Writer. Il n'est cependant pas nécessaire d'ouvrir d'abord Writer, de faire toute la saisie et de la personnalisation là-bas, puis de l'imprimer. Vous pouvez faire tout cela directement depuis Base, en utilisant une macro.

```
Sub ImprimePublipostage
	Dim oPublipostage As Object
	Dim aProps()
	oPublipostage = CreateUnoService("com.sun.star.text.MailMerge")
```

Le nom donné à la source de données est celui sous lequel la base de données est enregistrée dans LibreOffice. Ce nom n'a pas besoin d'être identique au nom de fichier. Le nom enregistré dans cet exemple est Adresses.

```
	oPublipostage.DataSourceName = "Addresses"
```

Le chemin vers le fichier publipostagedoit être formaté selon les conventions de votre système d'exploitation, dans cet exemple un chemin absolu dans un système Linux.

```
	oPublipostage.DocumentURL = ConvertToUrl("home/utilisateur/Documents/ _
	   publipostage.odt")
```

Le type de commande est défini. 0 représente une table, 1 une requête et 2 une commande SQL directe.

```
	oPublipostage.CommandType = 1
```

Ici, une requête a été choisie avec le nom *RequetePublipostage*.

```
	oPublipostage.Command = "RequetePublipostage"
```

Un filtre est utilisé pour déterminer les enregistrements à utiliser pour l'impression de publipostage. Ce filtre peut, par exemple, être spécifié à l'aide d'un contrôle de formulaire et transmis de Base à la macro. L'utilisation de la clé primaire d'un enregistrement peut entraîner l'impression d'un seul document.

Dans cet exemple, le champ Sexe de la RequetePublipostage est sélectionné, puis sont recherchés les enregistrements contenant « m » dans ce champ.

```
	oPublipostage.Filter = """Sexe""="m""
```

Les types de sortie disponibles sont Imprimante (1), Fichier (2) et Courrier (3). Ici, à des fins de test, un fichier de sortie est choisi. Ce fichier est stocké sur le chemin donné. Pour chaque enregistrement de publipostage, il y aura une impression. Pour distinguer cette impression, le champ nom de famille est incorporé au nom de fichier.

```
	oPublipostage.OutputType = 2
	oPublipostage.OutputUrl = ConvertToUrl("home/utilisateur/Documents")
	oPublipostage.FileNameFromColumn = True
	oPublipostage.Filenameprefix = "NomFamille"
	oPublipostage.execute(aProps())
End Sub
```

Si le filtre est fourni avec ses données via le formulaire, cela permet d'effectuer des publipostages sans ouvrir Writer.

Impression via des champs de texte

En utilisant **Insertion> Champ> Autres champs> Fonctions> Substituant**, un modèle peut être créé dans Writer pour un document qui sera imprimé ultérieurement. Les espaces réservés doivent porter les mêmes noms que les champs de la table de base de données ou de la requête sous-jacente au formulaire à partir duquel la macro est appelée.

Pour le cas simple, le type à choisir pour l'espace réservé est « Texte ».

Le chemin d'accès au modèle doit être fourni dans la macro. Un nouveau document Sans nom 1.odt est créé. La macro remplit les espaces réservés avec le contenu de l'enregistrement actuel de la requête. Le document ouvert peut ensuite être modifié selon les besoins.

L'exemple de base de données Exemple_BasedeDonnees_LettreType_directe.odb montre comment une facture complète peut être produite à l'aide de champs de texte et d'accéder à une table préparée dans le document modèle. Contrairement aux factures créées avec le Générateur de rapports, ce type de création de facture n'a pas de limites de hauteur pour les champs de la table. Tout le texte est affiché. Cet exemple permet également de créer des lettres-type depuis un formulaire.

Voici une partie du code, principalement fournie par Dpunch :

```
Sub Remplir_ChampsTexte
	oFormulaire = thisComponent.Drawpage.Forms.MainForm
	If oFormulaire.RowCount = 0 Then
	   MsgBox "Aucun enregistrement disponible pour l'impression"
	   Exit Sub
	End If
```

Le formulaire principal est activé. Le bouton qui lance la macro peut également être utilisé pour trouver le formulaire. Ensuite, la macro établit que le formulaire contient réellement des données imprimables.

L'accès direct à l'URL depuis le formulaire n'est pas possible. Cela doit être fait en utilisant la référence de niveau supérieur à la base de données.

```
	oColonnes = oFormulaire.Columns
	oDB = ThisComponent.Parent
```

Le titre de la base de données est séparé de l'URL.

```
	stDir = Left(oDB.Location, Len(oDB.Location)-Len(oDB.Title))
```

Le modèle est trouvé et ouvert

```
	stDir = stDir & "Exemple_ChampsTexte.ott"
	Dim args(0) As New com.sun.star.beans.PropertyValue
	args(0). Name = "AsTemplate"
	args(0). Value = True
	oNouveauDocument = StarDesktop.loadComponentFromURL(stDir,"_blank",0, args)
```

Les champs de texte sont écrits.

```
	oChampsTexte = oNouveauDocument.Textfields.createEnumeration
	Do While oChampsTexte.hasMoreElements
	   oChampTexte = oChampsTexte.nextElement
	   If oChampTexte.supportsService("com.sun.star.text.TextField.JumpEdit") Then
	      stNomColonne = oChampTexte.PlaceHolder
```

Le substituant représente le champ de texte.

```
                If oColonnes.hasByName(stNomColonne) Then
```

Si le nom du champ de texte est le même que le nom de la colonne dans l'ensemble de données sous-jacent, le contenu de la base de données est transféré dans le champ du document texte.

```
                    inIndex = oFormulaire.findColumn(stNomColonne)
                    oChampTexte. Anchor. String = oFormulaire.getString(inIndex)
                End If
            End If
        Loop
End Sub
```

Appeler des applications externes pour ouvrir des fichiers

Cette procédure permet avec un simple clic dans un champ de texte d'appeler le programme qui est lié au suffixe du nom de fichier dans le système d'exploitation. De cette manière, des liens Internet peuvent être suivis ou un programme de messagerie lancé pour une adresse spécifique stockée dans la base de données.

Pour cette section, voir aussi l'exemple de base de données :
Exemple_Activer_Fichier_Courriel.odb

```
Sub Activer_SiteWeb_ou_Courriel
        Dim oDoc As Object
        Dim oDrawpage As Object
        Dim oFormulaire As Object
        Dim oChamp As Object
        Dim oShell As Object
        Dim stChamp As String
        oDoc = thisComponent
        oDrawpage = oDoc.Drawpage
        oFormulaire = oDrawpage.Forms.getByName("Formulaire")
        oChamp = oFormulaire.getByName("url_ou_courriel")
```

Le contenu du champ nommé est lu. Il peut s'agir d'une adresse Web commençant par « http ://», d'une adresse de courriel contenant «@» ou d'un chemin vers un document (par exemple, une image ou un fichier PDF stocké en externe).

```
        stChamp = oChamp.Text
        If stChamp = "" Then
            Exit Sub
        End If
```

Si le champ est vide, la macro se ferme immédiatement. Lors de la saisie des données, il arrive souvent que les champs soient accessibles à l'aide de la souris, mais le fait de cliquer sur le champ dans le but d'y écrire pour la première fois ne doit pas conduire à l'exécution du code de macro.

Maintenant, le champ est scanné pour un caractère «@». Cela indiquerait une adresse de courriel. Le programme de messagerie doit être lancé pour envoyer du courrier à cette adresse.

```
        If InStr(stChamp,"@") Then
            stChamp = "mailto:"+stChamp
```

S'il n'y a pas de «@», le terme est converti en URL. Si cela commence par 'http://', nous n'avons pas affaire à un fichier dans le système de fichiers local mais à une ressource Internet qui doit être recherchée avec un navigateur Web. Sinon, le chemin commencera par le terme 'file:///'.

```
        Else
            stChamp = convertToUrl(stChamp)
        End If
```

Maintenant, le programme attribué par le système d'exploitation à ces fichiers est recherché. Pour le mot-clé 'mailto:' c'est le programme de messagerie, pour 'http://' le navigateur, sinon le système doit décider d'utiliser le suffixe du nom de fichier.

```
        oShell = createUnoService("com.sun.star.system. SystemShellExecute")
        oShell.execute(stChamp,,0)
End Sub
```

Appeler un programme de messagerie avec un contenu prédéfini

L'exemple précédent peut être étendu pour lancer un programme de messagerie avec un sujet et un contenu prédéfinis.

Pour cette section, voir aussi l'exemple de base de données : `Exemple_Activer_Courriel_Fichier.odb`.

Le programme de messagerie est lancé en utilisant 'mailto:recipient?subject=&body= &cc=&bcc='. Les deux dernières entrées ne sont pas présentes dans le formulaire. Les pièces jointes ne sont pas prévues dans la définition de "mailto" mais parfois 'attachment=' fonctionne.

```
Sub Activer_Courriel
        Dim oDoc As Object
        Dim oDrawpage As Object
        Dim oFormulaire As Object
        Dim oChamp1 As Object
        Dim oChamp2 As Object
        Dim oChamp3 As Object
        Dim oChamp4 As Object
        Dim oShell As Object
        Dim stChamp1 As String
        Dim stChamp2 As String
        Dim stChamp3 As String
        Dim stChamp4 As String
        oDoc = thisComponent
        oDrawpage = oDoc.Drawpage
        oFormulaire = oDrawpage.Forms.getByName("Formulaire")
        oChamp1 = oFormulaire.getByName("Adresse_Courriel")
        oChamp2 = oFormulaire.getByName("Sujet_Courriel")
        oChamp3 = oFormulaire.getByName("Contenu_Courriel")
        stChamp1 = oChamp1.Text
        If stChamp1 = "" Then
            MsgBox "Adrresse courriel manquante." & Chr(13) & _
                "Le programme de messagerie ne sera pas activé", 48, "Envoyer courriel"
            Exit Sub
        End If
```

La conversion en URL est nécessaire pour empêcher les caractères spéciaux et les sauts de ligne d'interférer avec l'appel. Cela préfixe cependant le chemin avec le terme 'file:///'. Ces 8 caractères au début ne sont pas transférés.

```
        stChamp2 = Mid(ConvertToUrl(oChamp2.Text),9)
        stChamp3 = Mid(ConvertToUrl(oChamp3.Text),9)
```

Contrairement à un simple lancement de programme, les détails de l'appel du courrier sont donnés ici dans le cadre de l'appel d'exécution.

```
        oShell = createUnoService("com.sun.star.system. SystemShellExecute")
        oShell.execute("mailto:" + stChamp1 + "?subject=" + stChamp2 + "&body=" +
stChamp3,,0)
End Sub
```

Note

L'envoi de courriels à l'aide d'un programme de messagerie peut également être effectué à l'aide du code suivant, mais le contenu réel du courriel ne peut pas être inséré de cette manière.

```
Dim attaches(0)
oCourrielleur = _ createUnoService("com.sun.star.system.SimpleSystemMail")
oProgrammeCourriel = oCourrielleur.querySimpleMailClient()
oNouveauMessage = oProgrammeCourriel.createSimpleMailMessage()
oNouveauMessage.setRecipient(stField1)
oNouveauMessage.setSubject(stField2)
attachs(0) = "file:///…"
oNouveauMessage.setAttachement(attachs())
oProgrammeCourriel.sendSimpleMailMessage(oNouveauMessage, 0)
```

Pour les paramètres utilisables, voir les détails de l'API : http://api.libreoffice.org/docs/idl/ref/interfacecom_1_1sun_1_1star_1_1system_1_1XSimpleMailMessage.html

Changer le pointeur de la souris lors du parcours d'un contrôle

Ceci est normal sur Internet et Base le recrée : le pointeur de la souris traverse un lien et se transforme en main pointée. Le texte du lien peut également changer ses propriétés, devenant bleu et souligné. La ressemblance avec un lien Internet est parfaite. Tout utilisateur attendra un clic pour ouvrir un programme externe.

Pour cette section, voir l'exemple de base de données Exemple_Activer_Courriel_Fichier.odb

Cette courte procédure doit être liée à l'événement '**Souris à l'intérieur**'

```
Sub Pointeur_Souris(oEvent As Object)
REM Voir aussi Bibliothèques standard : Outils → ModuleControls → SwitchMousePointer
        Dim oPointeur As Object
        oPointeur = createUnoService("com.sun.star.awt. Pointer")
        oPointeur.setType(27) 'pour les types voir com.sun.star.awt.SystemPointer
        oEvent.Source.Peer.SetPointer(oPointeur)
End Sub
```

Affichage des formulaires sans barre d'outils

Les nouveaux utilisateurs de Base sont souvent irrités par le fait qu'une barre d'outils existe mais ne soit pas utilisable dans un formulaire. Ces barres d'outils peuvent être supprimées de différentes manières. Les meilleurs moyens dans toutes les versions de LibreOffice sont les deux décrits ci-dessous.

Les tailles de fenêtre et les barres d'outils sont généralement contrôlées par une macro qui est lancée à partir d'un document de formulaire en utilisant **Outils> Personnaliser> Événements> Ouvrir le document**. Cela fait référence à l'ensemble du document, pas à un formulaire principal ou à un sous-formulaire individuel.

Formulaires sans barre d'outils dans la fenêtre

La taille d'une fenêtre peut être modifiée. En utilisant le bouton approprié, elle peut également être fermée. Ces tâches sont effectuées par le gestionnaire de fenêtres de votre système. La position et la taille d'une fenêtre à l'écran peuvent être fournies par une macro au démarrage du programme[26].

```
Sub Cache_Barres_Outils
        Dim oCadre As Object
        Dim oFenetre As Object
        Dim oLayoutMng As Object
        Dim aElements()
        oCadre = StarDesktop.getCurrentFrame()
' Le titre du formulaire doit être affiché dans la barre de titre de la fenêtre.
        oCadre.setTitle "Mon Formulaire"
        oFenetre = oCadre.getContainerWindow()
```

La fenêtre est agrandie. Ce n'est pas la même chose que le mode plein écran, car la barre des tâches est toujours visible et la fenêtre a une barre de titre, qui peut être utilisée pour changer sa taille ou la fermer.

```
        oFenetre.IsMaximized = true
```

Il est possible de créer une fenêtre avec une taille et une position spécifiques. Ceci est réalisé avec 'o**Fenetre**.setPosSize(0,0,600,400,15)'. Ici, la fenêtre apparaît dans le coin supérieur gauche de l'écran avec une largeur de 600 pixels et une hauteur de 400. Le dernier chiffre indique que toutes les dimensions, en pixels, sont données. On l'appelle Flag. Flag est calculé à partir de la somme des valeurs suivantes : x = 1, y = 2, largeur = 4, hauteur = 8. Comme x, y, la largeur et la hauteur sont toutes données, Flag a la valeur 1+2+4+8=15.

```
        oLayoutMng = oCadre.LayoutManager
        aElements = oLayoutMng.getElements()
        For i = LBound(aElements) To UBound(aElements)
            If aElements(i).ResourceURL = _
                "private:resource/toolbar/formsnavigationbar" Then
            Else
                oLayoutMng.hideElement(aElements(i).ResourceURL)
            End If
        Next
        ThisComponent.CurrentController.Sidebar.Visible = False
End Sub
```

Pour la barre de navigation de formulaire, rien n'est à faire, car il faut la conserver pour que le formulaire reste utilisable. Seules les barres d'outils autres que la barre de navigation doivent être masquées. Pour cette raison, il n'y a aucune action pour ce cas.

Si les barres d'outils ne sont pas restaurées directement après avoir quitté le formulaire, elles seront toujours masquées. Elles peuvent bien sûr être restaurées en utilisant **Affichage** > **Barres**

26 Un exemple de cette macro est visible dans le fichier Exemple_Formulaire_a_formulaire.odb

d'outils. Mais ce serait plutôt ennuyeux si la barre d'outils standard (**Affichage > Barres d'outils > Standard**) ou la barre d'état (**Affichage** > **Barre d'état**) manquait.

Cette procédure restaure ("showElement") les barres d'outils de leur état caché ("hideElement"). Les commentaires contiennent les barres dont l'absence est la plus susceptible d'être remarquée.

```
Sub Affiche_Barre_Outils
	Dim oCadre As Object
	Dim oLayoutMng As Object
	Dim aElements()
	oCadre = StarDesktop.getCurrentFrame()
	oLayoutMng = oCadre.LayoutManager
	aElements = oLayoutMng.getElements()
	For i = LBound(aElements) To UBound(aElements)
	   oLayoutMng.showElement(aElements(i).ResourceURL)
	Next
	' éléments importants qui peuvent être absents :
	'   "private: resource/toolbar/standardbar"
	'   "private: resource/statusbar/statusbar"
End Sub
```

Les macros sont liées à : **Outils > Personnaliser > Evénements > Ouvrir le document> Cache_Barre_Outils** et **Document fermé> Afficher_Barres_Outils**.

Malheureusement, souvent, les barres d'outils ne parviennent pas à revenir. Dans le pire des cas, il peut être utile de ne pas lire les éléments que le gestionnaire de mise en page connaît déjà, mais d'abord de créer des barres d'outils particulières, puis de les afficher :

```
Sub Affiche_Barres_Outils
	Dim oCadre As Object
	Dim oLayoutMng As Object
	Dim i As Integer
	Dim aElements(5) As String
	oCadre = StarDesktop.getCurrentFrame()
	oLayoutMng = oCadre.LayoutManager
	aElements(0) = "private:resource/menubar/menubar"
	aElements(1) = "private:resource/statusbar/statusbar"
	aElements(2) = "private:resource/toolbar/formsnavigationbar"
	aElements(3) = "private:resource/toolbar/standardbar"
	aElements(4) = "private:resource/toolbar/formdesign"
	aElements(5) = "private:resource/toolbar/formcontrols"
	For i = LBound(aElements) To UBound(aElements)
	   If Not(oLayoutMng.requestElement(i)) Then
	      oLayoutMng.createElement(aElements(i))
	   End If
	   oLayoutMng.showElement(aElements(i))
	Next i
ThisComponent.CurrentController.Sidebar.Visible = True
End Sub
```

Les barres d'outils à créer sont nommées explicitement. Si une barre d'outils correspondante n'est pas disponible pour le gestionnaire de disposition, elle est créée à l'aide de createElement puis affichée en utilisant showElement.

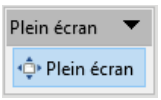
Formulaires en mode plein écran

En mode plein écran, tout l'écran est couvert par le formulaire. Il n'y a pas de barre des tâches ou d'autres éléments qui pourraient indiquer si d'autres programmes sont en cours d'exécution[27].

```
Sub PleinEcran(boSwitch As Boolean)
	Dim oDispatcher As Object
	Dim Props(0) As New com.sun.star.beans.PropertyValue
	oDispatcher = createUnoService("com.sun.star.frame.DispatchHelper")
	Props(0).Name = "FullScreen"
	Props(0).Value = boSwitch
	oDispatcher.executeDispatch(ThisComponent.CurrentController.Frame, _
	   ".uno:FullScreen", "", 0, Props())
End Sub
```

Cette fonction est lancée par les macros suivantes, qui déclenchent également soit la procédure de masquage des barres d'outils (Cache_Barres_Outils), soit celle d'affichage des barres d'outils (Affiche_Barres_Outils). Seule persiste la barre d'outils « Navigation pour formulaire ».

```
Sub PleinEcran_on
	PleinEcran(true)
	Cache_Barres_Outils
End Sub
```

La barre d'outils spécifique au mode plein écran « **CTRL + MAJ + J** » 

n'apparaît pas, ce qui empêche la possibilité d'utiliser la touche « **ESC** » pour désactiver ce mode. Un bouton spécifique doit être utilisé pour la sortie du mode plein écran et le rétablissement des barres d'outils, cette commande figure à la ligne suivante :

```
Sub PleinEcran_off
	PleinEcran(false)
	Affiche_Barres_Outils
End Sub
```

Lancement des formulaires directement depuis l'ouverture de la base de données

Lorsque les barres d'outils ont disparu ou qu'un formulaire doit être affiché en mode plein écran, le fichier de base de données doit lancer le formulaire directement lors de son ouverture. Malheureusement, une simple commande pour ouvrir un formulaire ne fonctionnera pas, car la connexion à la base de données n'existe pas encore.

La macro suivante est lancée depuis **Outils> Personnaliser> Événements> Ouvrir le document**. Utilisez l'option **Enregistrer dans**> « Nom de la base.odb ».

```
Sub Formulaire_Direct
	Dim oSourceDonnees As Object
	oSourceDonnees = ThisDatabaseDocument.CurrentController
	If Not (oSourceDonnees.isConnected()) Then
	   oSourceDonnees.connect()
	End If
	ThisDatabaseDocument.FormDocuments.getByName("MonFormulaire").open
End Sub
```

27 Un exemple de cette macro est visible dans le fichier Exemple_Formulaire_a_formulaire.odb

Tout d'abord, une connexion à la base de données doit être établie. Le contrôleur fait partie de ThisDatabaseDocument, tout comme le formulaire. Ensuite, le formulaire peut être lancé et lire ses données hors de la base de données.

Accéder à une base de données MySQL avec des macros

Toutes les macros présentées jusqu'à présent faisaient partie d'une base de données interne HSQLDB. Lorsque vous travaillez avec des bases de données externes, quelques modifications et extensions sont nécessaires.

Code MySQL dans les macros

Lors de l'accès à la base de données interne, les noms des tables et des champs doivent être placés entre guillemets doubles en double, par rapport au SQL :

```
SELECT "Champ" FROM "Table"
```

Comme ces commandes SQL doivent être préparées à l'intérieur de macros, les guillemets doivent être masqués :

```
stSQL = "SELECT ""Champ"" FROM ""Table"""
```

Les requêtes MySQL utilisent une forme différente de masquage :

```
SELECT `Champ` FROM `Database`.`Table`
```

À l'intérieur du code de macro, cette forme de masquage apparaît comme :

```
stSql = "SELECT `Champ` FROM `Database`.`Table`"
```

Tables temporaires comme stockage intermédiaire individuel

Dans le chapitre précédent, un tableau à une ligne était fréquemment utilisé pour rechercher ou filtrer les tables. Cela ne fonctionnera pas dans un système multi-utilisateur, car les autres utilisateurs seraient alors dépendants de la valeur de filtre de quelqu'un d'autre. Les tables temporaires dans MySQL ne sont accessibles qu'à l'utilisateur de la connexion active, donc ces tables sont accessibles pour la recherche et le filtrage.

Naturellement, de telles tables ne peuvent pas être créées à l'avance. Elles doivent être créées lors de l'ouverture du fichier Base. Par conséquent, la macro suivante doit être liée à l'ouverture du fichier *.odb.

```
Sub Creer_Table_Temporaire
	oSourceDonnees = thisDatabaseDocument.CurrentController
	If Not (oSourceDonnees.isConnected()) Then
	   oSourceDonnees.connect()
	oConnexion = oSourceDonnees.ActiveConnection()
	oSQL_Instruction = oConnexion.createStatement()
	stSql = "CREATE TEMPORARY TABLE IF NOT EXISTS `tmpCherche` _
	      (`ID` INT PRIMARY KEY, `Nom` VARCHAR(50))"
	oSQL_Instruction.executeUpdate(stSql)
End Sub
```

Lorsque le fichier *.odb est ouvert pour la première fois, il n'y a pas de connexion à une base de données MySQL externe. La connexion doit être créée. Ensuite, une table temporaire avec les champs nécessaires peut être mise en place.

Dialogues

Dans Base (comme dans les autres modules), vous pouvez utiliser des boîtes de dialogue aussi bien que des formulaires pour la saisie de données, la modification de données ou la maintenance de la base de données. Les boîtes de dialogue peuvent être directement personnalisées pour l'environnement d'application actuel, mais elles ne sont naturellement pas définies à l'avance aussi facilement que les formulaires. Voici une brève introduction se terminant par un exemple assez complexe à utiliser dans la maintenance des bases de données[28].

Lancement et fin des dialogues

La boîte de dialogue doit d'abord être créée. Pour ce faire, utilisez **Outils> Macros> Gérer les boîtes de dialogue> Nom de fichier de la base de données> Standard> Nouveau**. La boîte de dialogue apparaît avec une surface grise continue et une barre de titre avec une icône de fermeture. Cette boîte de dialogue vide peut maintenant être appelée puis refermée.

Il faut sélectionner la boite de dialogue en cliquant sur le trait fin qui l'entoure. Il est alors possible sous les propriétés générales de définir une taille et une position. Son titre peut également être saisi.

La barre d'outils en bas de la fenêtre contient divers contrôles de formulaire. À partir de là, deux boutons ont été sélectionnés pour notre boîte de dialogue, lui permettant de lancer d'autres boîtes de dialogue. L'édition du contenu et la liaison des macros à des événements s'effectuent de la même manière que pour les boutons dans les formulaires.

Le positionnement des déclarations de variables pour les dialogues nécessite une attention particulière. La boîte de dialogue est déclarée comme une variable globale afin qu'elle soit accessible par différentes procédures. Dans ce cas, la boîte de dialogue s'appelle oDialogue0, car il y aura d'autres boîtes de dialogue avec des numéros de séquence plus élevés

```
Dim oDialogue0 As Object
```

La bibliothèque de la boîte de dialogue est d'abord chargée. Elle se trouve dans le catalogue Standard, si aucun autre endroit n'a été choisi lors de la création de la boîte de dialogue. Le dialogue lui-même peut être atteint dans cette bibliothèque en utilisant l'instruction **Dialogue0**.**Execute()**, qui lance la boîte de dialogue.

```
Sub DebutDialogue0
	DialogLibraries.LoadLibrary("Standard")
	oDialogue0 = createUnoDialog(DialogLibraries.Standard.Dialogue0)
	oDialogue0.Execute()
```

28 Voir la base Exemple_Dialogues.odb, associée à ce manuel ainsi que le formulaire Maintenance de la base Media_avec_Macros.odb

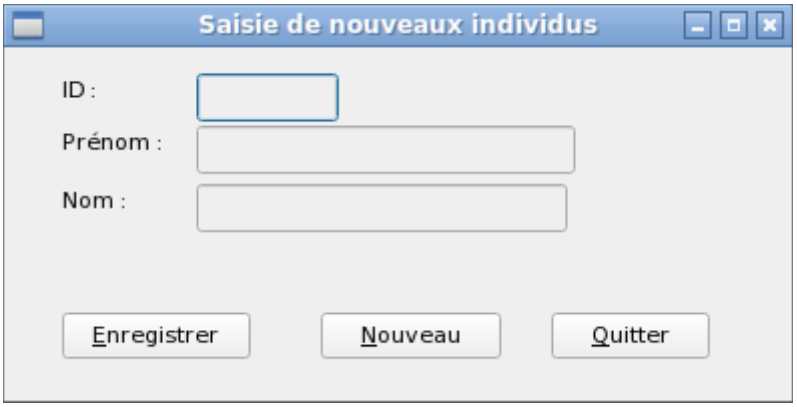
```
End Sub
```

En principe, une boîte de dialogue peut être fermée à l'aide du bouton Fermer sur le cadre. Cependant, si vous souhaitez un autre bouton spécifique pour cela, la commande **EndExecute()** doit être utilisée dans la procédure.

```
Sub FinDialogue0
	oDialogue0.EndExecute()
End Sub
```

Dans ce cadre, n'importe quel nombre de dialogues peuvent être lancés et refermés.

Boîte de dialogue simple pour saisir de nouveaux enregistrements



Cette boîte de dialogue est la première étape de la boîte de dialogue suivante pour la modification des enregistrements. Tout d'abord, l'approche de base de la gestion des tables est clarifiée. Nous traitons ici du stockage des enregistrements avec de nouvelles clés primaires ou de la nouvelle entrée complète des enregistrements. Dans quelle mesure un petit dialogue comme celui-ci peut suffire pour entrer dans une base de données particulière dépend des exigences de l'utilisateur.

```
Dim oDialogue1 As Object
```

Crée directement une variable globale pour la boîte de dialogue au niveau supérieur du module avant toutes les procédures.

La boîte de dialogue s'ouvre et se ferme de la même manière que la boîte de dialogue précédente. Seul le nom passe de Dialogue0 à Dialogue1. La procédure de fermeture de la boîte de dialogue est liée au bouton **Quitter**.

Le bouton **Nouveau** permet d'effacer toutes les entrées précédentes des contrôles de la boîte de dialogue, à l'aide de la procédure **Efface_Champs_Donnees**.

```
Sub Efface_Champs_Donnees
	oDialogue1.getControl("ChampNumerique1").Text = ""
	oDialogue1.getControl("ChampTexte1").Text = ""
	oDialogue1.getControl("ChampTexte2").Text = ""
End Sub
```

Chaque champ inséré dans une boîte de dialogue est accessible par son nom. L'interface utilisateur garantira que les noms ne sont pas dupliqués, ce qui n'est pas le cas des contrôles dans un formulaire.

La méthode **getControl** est utilisée avec le nom du contrôle. Les champs numériques ont également une propriété Text qui peut être utilisée ici. C'est la seule façon de vider un champ

numérique. Le texte vide existe, mais il n'y a pas de nombre vide. Au lieu de cela, un 0 doit être écrit dans le champ de clé primaire.

Le bouton **Enregistrer** lance la procédure **Enregistre_Donnees1** :

```
Sub Enregistre_Donnees1
	Dim oSourceDonnees As Object
	Dim oConnexion As Object
	Dim oSQL_Commande As Object
	Dim loID As Long
	Dim stPrenom As String
	Dim stNom As String
	loID = oDialogue1.getControl("ChampNumerique1").Value
	stPrenom = oDialogue1.getControl("ChampTexte1").Text
	stNom = oDialogue1.getControl("ChampTexte2").Text
	If loID > 0 And stNom <> "" Then
	   oSourceDonnees = thisDatabaseDocument.CurrentController
	   If Not (oSourceDonnees.isConnected()) Then
	      oSourceDonnees.connect()
	   End If
	   oConnexion = oSourceDonnees.ActiveConnection()
	   oSQL_Commande = oConnexion.createStatement()
	   stSql = "SELECT ""ID"" FROM ""Noms"" WHERE ""ID"" = '"+loID+"'"
	   oResultat = oSQL_Commande.executeQuery(stSql)
	   While oResultat.next
	      MsgBox ("la valeur pour le champ "ID" existe déjà",16, _
	         "Valeur dupliquée")
	      Exit Sub
	   Wend
	   stSql = "INSERT INTO ""Noms"" (""ID"", ""Prenom"", ""Nom"") _
	      VALUES ('"+loID+"','"+stPrenom+"','"+stNom+"')"
	   oSQL_Commande.executeUpdate(stSql)
	   Msgbox "Enregistrement effectué"
	   Efface_Champs_Donnees
	End If
End Sub
```

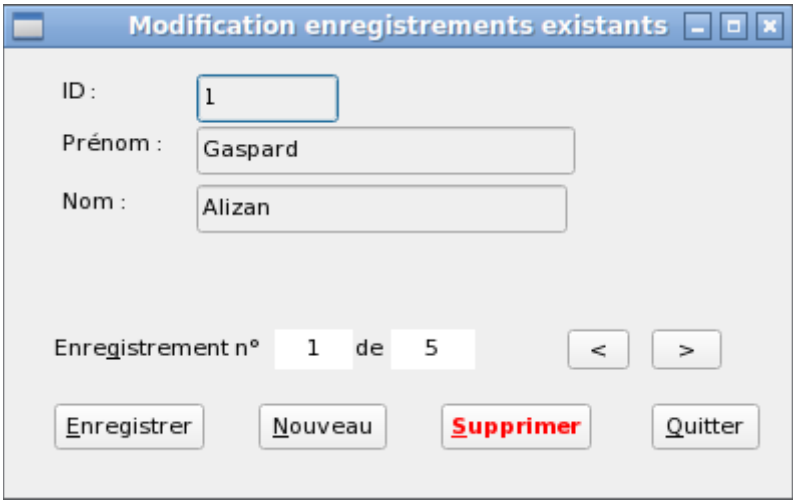
Comme dans la procédure Efface_Champs_Donnees, les champs de saisie sont accessibles. Cette fois, l'accès est en lecture uniquement. L'enregistrement ne sera transmis que si le champ ID a une entrée supérieure à 0 et que le champ Nom contient également du texte. Une valeur nulle pour l'ID peut être exclue car une variable numérique pour les nombres entiers est toujours initialisée à 0. Un champ vide est donc stocké avec une valeur nulle.

Si les deux champs ont été fournis avec du contenu, une connexion est établie avec la base de données. Comme les contrôles ne sont pas dans un formulaire, la connexion à la base de données doit être établie à l'aide de thisDatabaseDocument.CurrentController.

Tout d'abord, la base de données est interrogée pour voir si un enregistrement avec la clé primaire donnée existe déjà. Si cette requête produit un résultat, une boîte de message apparaît contenant un symbole d'arrêt (code : **16**) et le message "Valeur dupliquée". Ensuite, la procédure se termine avec **Exit SUB**.

Si la requête ne trouve aucun enregistrement avec la même clé primaire, le nouvel enregistrement est inséré dans la base de données à l'aide de la commande SQL INSERT. Ensuite, la procédure Efface_Champs_Donnees est appelée pour fournir un nouveau formulaire vide.

Boîte de dialogue pour modifier les enregistrements dans une table



Ce dialogue offre clairement plus de possibilités que le précédent. Ici, tous les enregistrements peuvent être affichés et vous pouvez les parcourir, en créer de nouveaux ou supprimer des enregistrements. Naturellement, le code est beaucoup plus compliqué.

Le bouton **Quitter** est lié à la procédure, modifiée pour Dialogue2, qui a été décrite dans la boîte de dialogue précédente pour la saisie de nouveaux enregistrements. Ici, les boutons restants et leurs fonctions sont décrits.

La saisie des données dans la boîte de dialogue est limitée en ce que le champ ID doit avoir une valeur minimale de 1. Cette limitation concerne la gestion des variables en Basic : les variables numériques sont par définition initialisées à 0. Par conséquent, si les valeurs numériques des champs vides et ceux des champs contenant 0 sont lus, Basic ne peut détecter aucune différence entre eux. Cela signifie que si un 0 devait être utilisé dans le champ ID, il devrait d'abord être lu sous forme de texte et peut-être converti en nombre plus tard.

La boîte de dialogue est chargée dans les mêmes conditions qu'auparavant. Cependant, la procédure de chargement est rendue dépendante d'une valeur nulle pour la variable passée par la procédure Charge_Donnees.

```
Sub Charge_Donnees(loID As Long)
	Dim oSourceDonnees As Object
	Dim oConnexion As Object
	Dim oSQL_Commande As Object
	Dim stPrenom As String
	Dim stNom As String
	Dim loLigne As Long
	Dim loLigneMax As Long
	Dim inDebut As Integer
	oSourceDonnees = thisDatabaseDocument. CurrentController
	If Not (oSourceDonnees.isConnected()) Then
	    oSourceDonnees.connect()
	End If
	oConnexion = oSourceDonnees. ActiveConnection()
	oSQL_Commande = oConnexion.createStatement()
	If loID < 1 Then
	    stSql = "SELECT MIN(""ID"") FROM ""Noms"""
	    oResultat = oSQL_Commande.executeQuery(stSql)
	    While oResultat.next
```

```
            loID = oResultat.getInt(1)
        Wend
        inDebut = 1
    End If
```

Les variables sont déclarées. La connexion à la base de données pour la boîte de dialogue est établie comme décrit ci-dessus. Au début, **loID** vaut **0**. Ce cas fournit la valeur de clé primaire la plus basse autorisée par SQL. L'enregistrement correspondant sera affiché ultérieurement dans la boîte de dialogue. Dans le même temps, la variable **inDebut** est définie sur **1**, de sorte que la boîte de dialogue puisse être lancée plus tard. Si la table ne contient aucun enregistrement, **loID** restera **0**. Dans ce cas, il ne sera pas nécessaire de rechercher le nombre et le contenu des enregistrements correspondants.

Ce n'est que si **loID** est supérieur à 0 que la requête testera les enregistrements disponibles dans la base de données. Ensuite, une deuxième requête comptera tous les enregistrements à afficher. La troisième requête donne la position de l'enregistrement actuel en comptant tous les enregistrements dont la clé primaire est inférieure ou égale à la clé primaire actuelle.

```
    If loID > 0 Then
        stSql = "SELECT * FROM ""Noms"" WHERE ""ID"" = '"+loID+"'"
        oResultat = oSQL_Commande.executeQuery(stSql)
        While oResultat.next
            loID = oResultat.getInt(1)
            stPrenom = oResultat.getString(2)
            stNom = oResultat.getString(3)
        Wend
        stSql = "SELECT COUNT(""ID"") FROM ""Noms"""
        oResultat = oSQL_Commande.executeQuery(stSql)
        While oResultat.next
            loLigneMax = oResultat.getInt(1)
        Wend
        stSql = "SELECT COUNT(""ID"") FROM ""Noms"" WHERE ""ID"" <= '"+loID+"'"
        oResultat = oSQL_Commande.executeQuery(stSql)
        While oResultat.next
            loLigne = oResultat.getInt(1)
        Wend
        oDialogue2.getControl("ChampNumerique1").Value = loID
        oDialogue2.getControl("ChampTexte1").Text = stPrenom
        oDialogue2.getControl("ChampTexte2").Text = stNom
    End If
    oDialogue2.getControl("ChampNumerique2").Value = loLigne
    oDialogue2.getControl("ChampNumerique3").Value = loLigneMax
    If loLigne = 1 Then
        ' ligne précédente
        oDialogue2.getControl("btnSupprimer").Model.enabled = False
    Else
        oDialogue2.getControl("btnSupprimer").Model.enabled = True
    End If
    If loLigne <= loLigneMax Then
        ' ligne suivante | nouvelle ligne | effacer
        oDialogue2.getControl("btnPrecedent").Model.enabled = True
        oDialogue2.getControl("btnEnregistrer").Model.enabled = True
        oDialogue2.getControl("btnSupprimer").Model.enabled = True
    Else
        oDialogue2.getControl("btnSuivant").Model.enabled = False
        oDialogue2.getControl("btnEnregistrer").Model.enabled = False
        oDialogue2.getControl("btnSupprimer").Model.enabled = False
    End If
```

```
        IF inDebut = 1 Then
            oDialogue2.getControl("btnPrecedent").Model.enabled = false
        oDialogue2. Execute()
        End If
End Sub
```

Les valeurs récupérées sont transférées dans les champs de la boîte de dialogue. Les entrées pour le numéro d'enregistrement actuel et le nombre total d'enregistrements récupérés sont toujours écrites, remplaçant la valeur numérique par défaut de 0.

Les boutons de navigation (btnSuivant et btnPrecedent) ne sont utilisables que lorsqu'il est possible d'accéder à l'enregistrement correspondant. Sinon, ils sont temporairement désactivés avec **enabled = False**. Il en va de même pour les boutons btnNouveau et btnSupprimer. Ils ne doivent pas être disponibles lorsque le numéro de la ligne affichée est supérieur au nombre maximal de lignes déterminé. Il s'agit du paramètre par défaut de cette boîte de dialogue lors de la saisie d'enregistrements.

Si possible, la boîte de dialogue ne doit être lancée que lorsqu'elle doit être créée directement à partir d'un fichier de départ à l'aide de **Charge_Donnees (0)**. C'est pourquoi la variable spéciale **inDebut** reçoit la valeur 1 au début de la procédure.

Le bouton "<" est utilisé pour naviguer vers l'enregistrement précédent. Par conséquent, ce bouton n'est actif que lorsque l'enregistrement affiché n'est pas le premier de la liste. La navigation nécessite la lecture de la clé primaire de l'enregistrement actuel à partir du champ ChampNumerique1.

Ici, il y a deux cas possibles :

1) Vous passez à une nouvelle entrée, le champ correspondant n'a donc aucune valeur. Dans ce cas, **loID** a la valeur par défaut qui, selon la définition d'une variable entière, est 0.
2) Sinon, loID contiendra une valeur supérieure à 0. Une requête peut alors déterminer la valeur d'ID immédiatement inférieure à la valeur actuelle.

```
Sub Ligne_Precedente
        Dim loID As Long
        Dim loIDnouveau As Long
        loID = oDialogue2.getControl("ChampNumerique1").Value
        oSourceDonnees = thisDatabaseDocument.CurrentController
        If Not (oSourceDonnees.isConnected()) Then
            oSourceDonnees.connect()
        End If
        oConnexion = oSourceDonnees.ActiveConnection()
        oSQL_Commande = oConnexion.createStatement()
        If loID < 1 Then
            stSql = "SELECT MAX(""ID"") FROM ""Noms"""
        Else
            stSql = "SELECT MAX(""ID"") FROM ""Noms"" WHERE ""ID"" < '"+loID+"'"
        End If
        oResultat = oSQL_Commande.executeQuery(stSql)
        While oResultat.next
            loIDnouveau = oResultat.getInt(1)
        Wend
        If loIDnouveau > 0 Then
            Charge_Donnees(loIDnouveau)
        End If
        If loIDnouveau = 1 Then
            oDialogue2.getControl("btnPrecedent").Model.enabled = false
```

```
        End If
        oDialogue2.getControl("btnSuivant").Model.enabled = True
        oDialogue2.getControl("btnNouveau").Model.enabled = True
End Sub
```

Si le champ ID est vide, l'affichage doit passer à la valeur la plus élevée du numéro de clé primaire. Si, en revanche, le champ ID fait référence à un enregistrement, la valeur précédente d'ID doit être renvoyée.

Le résultat de cette requête est utilisé pour exécuter à nouveau la procédure Charge_Donnees avec la valeur de clé correspondante.

Le bouton ">" est utilisé pour naviguer vers l'enregistrement suivant. Cette possibilité ne doit exister que lorsque la boîte de dialogue n'a pas été vidée pour la saisie d'un nouvel enregistrement. Ce sera naturellement le cas au lancement du dialogue et également avec une table vide.

Une valeur dans ChampNumerique1 est obligatoire. À partir de cette valeur, SQL peut déterminer quelle clé primaire est la suivante la plus élevée de la table. Si l'ensemble de résultats de la requête est vide, car il n'y a pas d'enregistrement correspondant, la valeur de **loIDnouveau = 0**. Sinon, le contenu de l'enregistrement suivant est lu à l'aide de Charge_Donnees.

```
Sub Ligne_Suivante
        Dim loID As Long
        Dim loIDnouveau As Long
        loID = oDialogue2.getControl("ChampNumerique1").Value
        oSourceDonnees = thisDatabaseDocument.CurrentController
        If Not (oSourceDonnees.isConnected()) Then
            oSourceDonnees.connect()
        End If
        oConnexion = oSourceDonnees.ActiveConnection()
        oSQL_Commande = oConnexion.createStatement()
        stSql = "SELECT MIN(""ID"") FROM ""Noms"" WHERE ""ID"" > '"+loID+"'"
        oResultat = oSQL_Commande.executeQuery(stSql)
        While oResultat.next
            loIDnouveau = oResultat.getInt(1)
        Wend
        If loIDnouveau > 0 Then
            Charge_Donnees(loIDnouveau)
        Else
            Efface_Champs_Donnees2
        End If
End Sub
```

Si lors de la navigation vers l'enregistrement suivant, il n'y a pas d'autre enregistrement, la touche de navigation lance la procédure suivante Efface_Champs_Donnees2, qui sert à préparer l'entrée d'un nouvel enregistrement.

La procédure Efface_Champs_Donnees2 ne se contente pas de vider les champs de données eux-mêmes. La position de l'enregistrement en cours est définie sur une valeur supérieure au nombre maximum d'enregistrements, ce qui indique clairement que l'enregistrement en cours de traitement n'est pas encore inclus dans la base de données.

Dès qu'Efface_Champs_Donnees2 a été lancé, la possibilité de sauter à l'enregistrement précédent est activée, les sauts à l'enregistrement suivant et l'utilisation des procédures Nouveau et Suppression sont désactivées.

```
Sub Efface_Champs_Donnees2
	loLigneMax = oDialogue2.getControl("ChampNumerique3").Value
	oDialogue2.getControl("ChampNumerique1").Text = ""
	oDialogue2.getControl("ChampTexte1").Text = ""
	oDialogue2.getControl("ChampTexte2").Text = ""
	oDialogue2.getControl("ChampNumerique2").Value = loLigneMax + 1
	oDialogue2.getControl("btnPrecedent").Model.enabled = True   ' enreg. précédent
	oDialogue2.getControl("btnSuivant").Model.enabled = False    ' enreg. suivant
oDialogue2.getControl("btnNouveau").Model.enabled = False    ' nouvel enreg.
	oDialogue2.getControl("btnSupprimer").Model.enabled = False  ' effacer
End Sub
```

L'enregistrement des données ne devrait être possible que lorsque les champs ID et Nom contiennent des entrées. Si cette condition est remplie, la procédure teste s'il s'agit d'un nouvel enregistrement. Cela utilise le pointeur d'enregistrement qui est défini pour que les nouveaux enregistrements soient un Nombre plus haut que le nombre maximum d'enregistrements

Pour les nouveaux enregistrements, des vérifications sont effectuées pour s'assurer que l'opération de sauvegarde réussira. Si le numéro utilisé pour la clé primaire a déjà été utilisé, un avertissement s'affiche. Si la réponse à la question associée est Oui, l'enregistrement existant avec ce numéro est écrasé. Sinon, la sauvegarde sera abandonnée. S'il n'y a pas d'entrées existantes dans la base de données (**loLigneMax = 0**), ce test n'est pas nécessaire et le nouvel enregistrement peut être sauvegardé directement. Pour un nouvel enregistrement, le nombre d'enregistrements est incrémenté de 1 et les entrées sont effacées pour l'enregistrement suivant.

Les enregistrements existants sont simplement écrasés par une commande de mise à jour.

```
Sub Enregistre_Donnees2(oEvent As Object)
	Dim oSourceDonnees As Object
	Dim oConnexion As Object
	Dim oSQL_Commande As Object
	Dim oDlg As Object
	Dim loID As Long
	Dim stPrenom As String
	Dim stNom As String
	Dim inMsg As Integer
	Dim loLigne As Long
	Dim loLigneMax As Long
	Dim stSql As String
	oDlg = oEvent. Source.getContext()
	loID = oDlg.getControl("ChampNumerique1").Value
	stPrenom = oDlg.getControl("ChampTexte1").Text
	stNom = oDlg.getControl("ChampTexte2").Text
	If loID > 0 And stNom <> "" Then
	   oSourceDonnees = thisDatabaseDocument.CurrentController
	   If Not (oSourceDonnees.isConnected()) Then
	      oSourceDonnees.connect()
	   End If
	   oConnexion = oSourceDonnees.ActiveConnection()
	   oSQL_Commande = oConnexion.createStatement()
	   loLigne = oDlg.getControl("ChampNumerique2").Value
	   loLigneMax = oDlg.getControl("ChampNumerique3").Value
	   If loLigneMax < loLigne Then
	      If loLigneMax > 0 Then
	         stSql = "SELECT ""ID"" FROM ""Noms"" WHERE ""ID"" = '"+loID+"'"
	         oResultat = oSQL_Commande.executeQuery(stSql)
	         While oResultat.next
```

```
                    inMsg = MsgBox ("La valeur du champ "ID" existe déjà." & _
                       CHR(13) & "L'enregistrement doit-il être mis à jour?",20, _
                       "Valeur dupliquée")
                    If inMsg = 6 Then
                       stSql = "UPDATE ""Noms"" SET ""Prenom""='"+stPrenom+"', _
                          ""Nom""='"+stNom+"' WHERE ""ID"" = '"+loID+"'"
                       oSQL_Commande.executeUpdate(stSql)
                       ' Avec la mise à jour, une ligne a été  réécrite
                       ' Le compte de lignes doit être réinitialisé
                       Charge_Donnees(loID)
                    End If
                    Exit Sub
                 Wend
              End If
              stSql = "INSERT INTO ""Noms"" (""ID"", ""Prenom"", ""Nom"") VALUES
                 ('"+loID+"','"+stPrenom+"','"+stNom+"')"
              oSQL_Commande.executeUpdate(stSql)
              oDlg.getControl("ChampNumerique3").Value = loLigneMax + 1
                 ' Après l'insertion, une ligne est ajoutée
              Efface_Champs_Donnees2
                 ' Après l'insertion serait déplacé à l'insertion suivante
           Else
              stSql = "UPDATE ""Noms"" SET ""Prenom""='"+stPrenom+"', _
                 ""Nom""='"+stNom+"' WHERE ""ID"" = '"+loID+"'"
              oSQL_Commande.executeUpdate(stSql)
           End If
        End If
End Sub
```

La procédure de suppression est fournie avec une question supplémentaire pour éviter les suppressions accidentelles. Puisque ce bouton est désactivé lorsque les champs de saisie sont vides, un ChampNumerique1 vide ne doit jamais se produire. Par conséquent, la condition de contrôle **IF loID > 0** peut être omise.

La suppression entraîne une décrémentation du nombre d'enregistrements de 1. Cela doit être corrigé à l'aide de **loLigneMax – 1**. Ensuite, l'enregistrement suivant celui en cours est affiché.

```
Sub Supprime_Ligne(oEvent As Object)
       Dim oSourceDonnees As Object
       Dim oConnexion As Object
       Dim oSQL_Commande As Object
       Dim oDlg As Object
       Dim loID As Long
       oDlg = oEvent.Source.getContext()
       loID = oDlg.getControl("ChampNumerique1").Value
       If loID > 0 Then
          inMsg = MsgBox ("Les données actuelles doivent-elles être supprimées ?",20,
_
             "Supprimer la ligne actuelle")
          If inMsg = 6 Then
             oSourceDonnees = thisDatabaseDocument.CurrentController
             If Not (oSourceDonnees.isConnected()) Then
                oSourceDonnees.connect()
             End If
             oConnexion = oSourceDonnees.ActiveConnection()
             oSQL_Commande = oConnexion.createStatement()
             stSql = "DELETE FROM ""Noms"" WHERE ""ID"" = '"+loID+"'"
             oSQL_Commande.executeUpdate(stSql)
             loLigneMax = oDlg.getControl("ChampNumerique3").Value
```

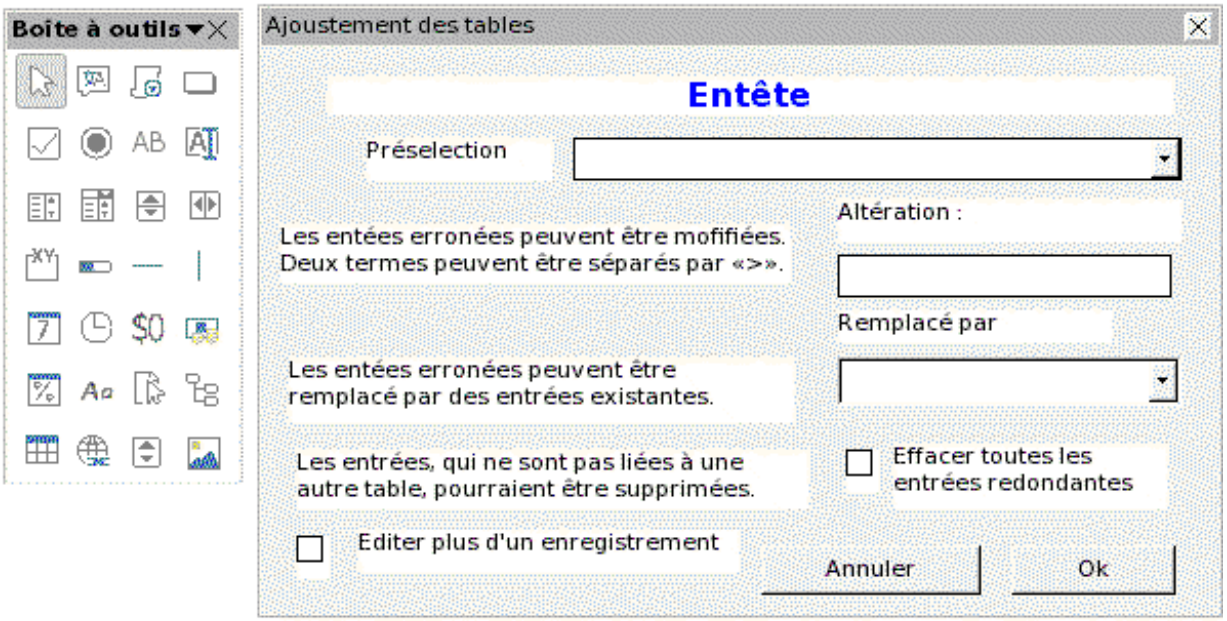
```
                oDlg.getControl("ChampNumerique3").Value = loLigneMax - 1
                Ligne_Suivante
            End If
        ELSE
            MsgBox ("Aucune ligne supprimée." & CHR(13) & _
                "Aucune donnée sélectionnée.",64,"Suppression impossible")
        End If
    End Sub
```

Cette petite boîte de dialogue a montré que l'utilisation du code macro peut fournir une base pour le traitement des enregistrements. L'accès via des formulaires est beaucoup plus facile, mais un dialogue peut être très flexible pour s'adapter aux exigences du programme. Cependant, il ne convient pas pour la création rapide d'une interface de base de données.

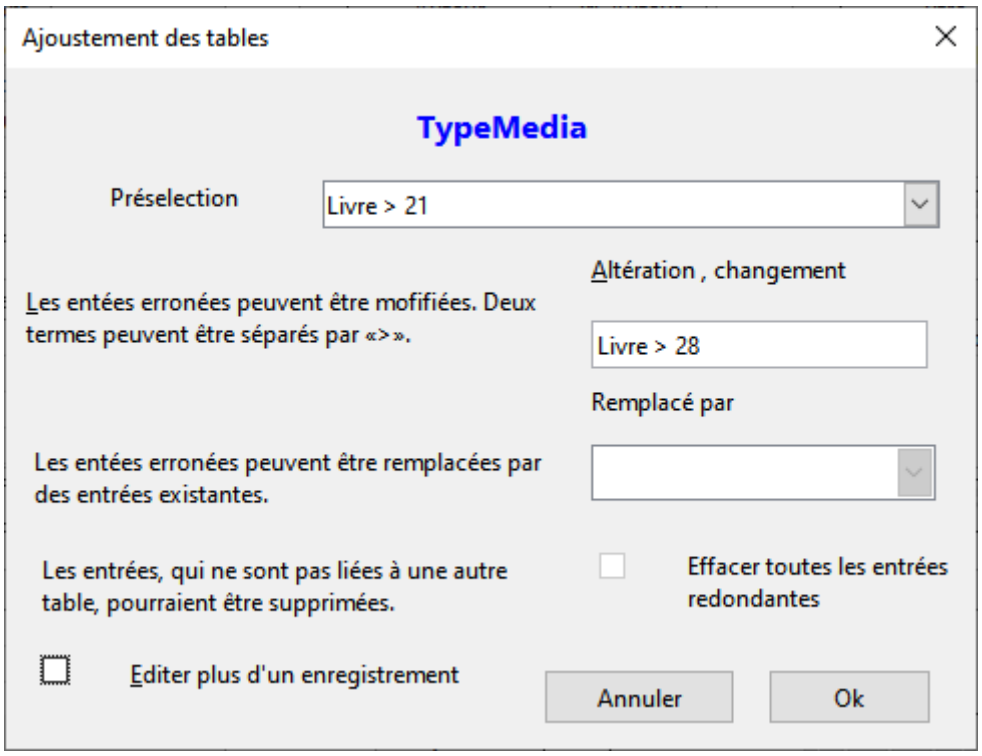
Utilisation d'une boîte de dialogue pour nettoyer les mauvaises données dans les tables

Les erreurs de saisie dans les champs ne sont souvent signalées que plus tard. Il est souvent nécessaire de modifier des entrées identiques dans plusieurs enregistrements en même temps. Il est gênant de devoir faire cela dans une vue de table normale, en particulier lorsque plusieurs enregistrements doivent être modifiés, car chaque enregistrement nécessite une entrée individuelle.

Les formulaires peuvent utiliser des macros pour faire ce genre de chose, mais pour le faire pour plusieurs tables, vous auriez besoin de formulaires construits de la même manière. Les dialogues peuvent faire le travail. Une boîte de dialogue peut être fournie au début avec les données nécessaires pour les tables appropriées et peut être appelée par plusieurs formulaires différents.



Les boîtes de dialogue sont enregistrées avec les modules pour les macros. Leur création s'apparente à celle d'un formulaire. Des champs de contrôle très similaires sont disponibles. Seul le contrôle de table des formulaires est absent en tant que possibilité d'entrée spéciale.



L'apparence des contrôles de boîte de dialogue est déterminée par les paramètres de l'interface utilisateur graphique.

La boîte de dialogue ci-dessus sert dans la base de données d'exemple pour éditer des tables qui ne sont pas utilisées directement comme base d'un formulaire. Ainsi, par exemple, le type de média n'est accessible que via une zone de liste (dans la version macro, il devient une zone combinée). Dans la version macro, le contenu du champ peut être étendu par un nouveau contenu mais une modification du contenu existant n'est pas possible. Dans la version sans macros, les modifications sont effectuées à l'aide d'un champ de table séparé.

Bien que les modifications dans ce cas soient faciles à effectuer sans macros, il est assez difficile de changer le type de support de nombreux supports à la fois. Supposons que les types suivants soient disponibles : « Livre, relié », « Livre, couverture rigide », « Livre broché » et « Reliure anneaux ». Maintenant, il s'avère, après que la base de données a été utilisée pendant une longue période, que des contemporains plus actifs prévoyaient des types de supports supplémentaires similaires pour les œuvres imprimées. La tâche de les différencier est devenue excessive. Nous souhaitons donc les réduire, de préférence à un seul terme. Sans macros, les enregistrements de la table des médias devraient être trouvés (à l'aide d'un filtre) et modifiés individuellement. Si vous connaissez SQL, vous pouvez le faire beaucoup mieux en utilisant une commande SQL. Vous pouvez modifier tous les enregistrements de la table Media avec une seule entrée. Une deuxième commande SQL supprime ensuite les types de média désormais excédentaires qui n'ont plus de lien vers la table Media. Cette méthode est précisément appliquée à l'aide de la boîte de dialogue Remplacer par – seule la commande SQL est d'abord adaptée à la table TypeMedia à l'aide d'une macro qui peut également modifier d'autres tables.

Souvent, les entrées se glissent dans une table qui, avec le recul, peut être modifiée dans le formulaire et ne sont donc plus nécessaires. Il ne fait aucun mal de simplement supprimer ces entrées orphelines, mais elles sont assez difficiles à trouver en utilisant l'interface utilisateur graphique. Là encore, une commande SQL appropriée est utile, associée à une instruction de

suppression. Cette commande pour les tables concernées est incluse dans la boîte de dialogue sous *Supprimer toutes les entrées superflues*.

Si la boîte de dialogue doit être utilisée pour effectuer plusieurs modifications, cela est indiqué par la case à cocher Modifier plusieurs enregistrements. Ensuite, la boîte de dialogue ne se terminera pas simplement lorsque vous cliquerez sur le bouton OK.

Le code de macro associé à cette boîte de dialogue peut être consulté en entier dans la base de données d'exemple (. Seuls des extraits sont expliqués ci-dessous.

```
Sub AjustementTable(oEvent As Object)
```

La macro doit être lancée en entrant dans la section Informations complémentaires pour les boutons concernés :

```
0 : Formulaire, 1 : SousFormulaire, 2 : SousSousFormulaire, 3 : ZoneCombinée ou
contrôle de table, 4 : Champ de clé étrangère dans un formulaire, vide pour un champ
de table, 5 : Nom de table de la table auxiliaire, 6: Champ1 de la table auxiliaire,
7 : Champ de table 1 de la table auxiliaire, or 8 : Nom de table de la table
auxiliaire pour le champ2
```

Les entrées de cette zone sont répertoriées au début de la macro sous forme de commentaires. Les nombres qui leur sont liés sont transférés et l'entrée correspondante est lue à partir d'un tableau. La macro peut éditer des zones de liste, qui ont deux entrées, séparées par « > ». Ces deux entrées peuvent également provenir de tables différentes et être rassemblées à l'aide d'une requête, comme, par exemple, dans la table CodePostal, qui n'a que le champ de clé étrangère ID_Ville pour la ville, ce qui oblige la table Villes à afficher les noms des villes.

```
Dim aTablesEtrangeres(0, 0 to 1)
Dim aTablesEtrangeres2(0, 0 to 1)
```

Parmi les variables définies au début se trouvent deux tableaux. Alors que les tableaux normaux peuvent être créés par la commande **Split()** pendant l'exécution du sous-programme, les tableaux bidimensionnels doivent être définis à l'avance. Les tableaux à deux dimensions sont nécessaires pour stocker plusieurs enregistrements d'une requête lorsque la requête elle-même fait référence à plusieurs champs. Les deux tableaux déclarés ci-dessus doivent être capables d'interpréter les requêtes faisant référence à deux champs de table. Par conséquent, ils sont définis pour deux contenus différents en utilisant 0 à 1 pour la deuxième dimension.

```
stTag = oEvent.Source.Model.Tag
aTable() = Split(stTag, ", ")
For i = LBound(aTable()) To UBound(aTable())
   aTable(i) = trim(aTable(i))
Next
```

Les variables fournies sont lues. La séquence est celle mise en place dans le commentaire ci-dessus. Il y a un maximum de neuf entrées et vous devez déclarer s'il existe une huitième entrée pour le champ de table2 et une neuvième entrée pour une deuxième table.

Si des valeurs doivent être supprimées d'une table, il faut d'abord vérifier qu'elles n'existent pas en tant que clés étrangères dans une autre table. Dans les structures de table simples, une table donnée n'aura qu'une seule connexion de clé étrangère à une autre table. Cependant, dans l'exemple de base de données utilisé, il existe une table Villes qui est utilisée à la fois pour le lieu de publication des médias et la ville pour les adresses. Ainsi, la clé primaire de la table Villes est entrée deux fois dans différentes tables. Ces tables et noms de clés étrangères peuvent naturellement également être saisis à l'aide du champ Informations supplémentaires. Ce serait

plus agréable si elles pouvaient être fournies universellement pour tous les cas. Cela peut être fait à l'aide de la requête suivante.

```
        stSql = "SELECT ""FKTABLE_NAME"", ""FKCOLUMN_NAME"" FROM
""INFORMATION_SCHEMA"".""SYSTEM_CROSSREFERENCE"" WHERE ""PKTABLE_NAME"" = '" +
aTable(5) + "'"
```

Dans la base de données, la zone I**NFORMATION_SCHEMA** contient toutes les informations sur les tables de la base de données, y compris des informations sur les clés étrangères. Les tables contenant ces informations sont accessibles à l'aide de **"INFORMATION_SCHEMA". "SYSTEM_CROSSREFERENCE"**. **"PKTABLE_NAME"**donne la table qui fournit sa clé primaire pour la connexion. **FKTABLE_NAME** donne la table qui utilise cette clé primaire comme clé étrangère. Enfin **FKCOLUMN_NAME** donne le nom du champ de clé étrangère.

La table qui fournit sa clé primaire à utiliser comme clé étrangère se trouve dans le tableau précédemment créé à la position 6. Comme le décompte commence par 0, la valeur est lue à partir du tableau à l'aide de **aTable(5)**.

```
        inCompteur = 0
        stIDEtrangerTab1Tab2 = "ID"
        stIDEtrangerTab2Tab1 = "ID"
        stTableSecondaire = aTable(5)
```

Avant que la lecture des tableaux ne commence, certaines valeurs par défaut doivent être définies. Il s'agit de l'index du tableau dans lequel les valeurs de la table auxiliaire seront écrites, de la clé primaire par défaut si nous n'avons pas besoin de la clé étrangère pour une deuxième table, et de la table auxiliaire par défaut, liée à la table principale, pour le code postal et ville, la table des codes postaux.

Lorsque deux champs sont liés pour être affichés dans une zone de liste, ils peuvent, comme décrit ci-dessus, provenir de deux tables différentes. Pour l'affichage du code postal et de la ville, la requête est :

```
SELECT "CodePostal"."CodePostal" || ' > ' || "Villes"."Ville" FROM
"CodePostal", "Villes" WHERE "CodePostal"."ID_Ville" = "Villes"."ID"
```

La table du premier champ (CodePostal), est liée à la deuxième table par une clé étrangère. Seules les informations de ces deux tables et des champs Code postal et Ville sont transmises à la macro. Toutes les clés primaires sont appelées par défaut ID dans l'exemple de base de données. La clé étrangère de Ville dans CodePostal doit donc être déterminée à l'aide de la macro.

De la même manière, la macro doit accéder à chaque table avec laquelle le contenu de la zone de liste est connecté par une clé étrangère.

```
        oResultat_Requete = oSQL_Instruction.executeQuery(stSql)
        If Not IsNull(oResultat_Requete) Then
           While oResultat_Requete.next
              ReDim Preserve aTablesEtrangeres(inCompteur, 0 to 1)
```

Le tableau doit être fraîchement dimensionné à chaque fois. Afin de préserver le contenu existant, il est sauvegardé à l'aide de (Preserve).

```
              aTablesEtrangeres(inCompteur, 0) = oResultat_Requete.getString(1)
```

Lecture du premier champ avec le nom de la table qui contient la clé étrangère. Le résultat de la table CodePostal est la table Adresses.

```
aTablesEtrangeres(inCompteur, 1) = oResultat_Requete.getString(2)
```

Lecture du deuxième champ avec le nom du champ de clé étrangère. Le résultat de la table CodePostal est le champ ID_CodePostal dans la table Adresses.

Dans les cas où un appel au sous-programme inclut le nom d'une deuxième table, la boucle suivante est exécutée. L'entrée par défaut est modifiée uniquement lorsque le nom de la deuxième table apparaît comme table de clé étrangère pour la première table. Dans notre cas, cela ne se produit pas, car la table Villes n'a pas de clé étrangère de la table CodePostal. L'entrée par défaut de la table auxiliaire reste donc CodePostal. Enfin, la combinaison du code postal et de la ville sert de base à la table Adresses, qui contient une clé étrangère de la table CosePostal.

```
If UBound(aTable()) = 8 Then
    If aTable(8) = aTablesEtrangeres(inCount, 0) Then
        stIDEtrangerTab2Tab1 = aTablesEtrangeres(inCount, 1)
        stTableSecondaire = aTable(8)
    End If
End If
inCompteur = inCompteur + 1
```

Comme d'autres valeurs peuvent avoir besoin d'être lues, l'index est incrémenté pour redimensionner les tableaux. Puis la boucle se termine.

```
    Wend
End If
```

Si, lors de l'appel du sous-programme, un deuxième nom de table existe, la même requête est lancée pour cette table :

```
If UBound(aTable()) = 8 Then
```

Il s'exécute de la même manière, sauf que la boucle teste si le premier nom de table apparaît comme un nom de table de clé étrangère. C'est le cas ici : la table CodePostal contient la clé étrangère ID_Ville de la table Villes. Cette clé étrangère est maintenant affectée à la variable stForeignIDTab1Tab2, afin que la relation entre les tables puisse être définie.

```
    If aTable(5) = aTablesEtrangeres2(inCount, 0) Then
        stIDEtrangerTab1Tab2 = aTablesEtrangeres2(inCompteur, 1)
    End If
```

Après quelques réglages supplémentaires pour garantir un retour à la forme correcte après l'exécution de la boîte de dialogue (détermination du numéro d'enregistrement du formulaire, afin que nous puissions revenir à ce numéro de ligne après une nouvelle lecture), la boucle commence, ce qui recrée la boîte de dialogue lorsque la première action est terminée mais que la boîte de dialogue doit rester ouverte pour d'autres actions. Le réglage de la répétition s'effectue à l'aide de la case à cocher correspondante.

```
Do
```

Avant que la boîte de dialogue ne soit lancée, tout d'abord le contenu des zones de liste est déterminé. Des précautions doivent être prises si les zones de liste représentent deux champs de table et peuvent même être liés à deux tables différentes.

```
If UBound(aTable()) = 6 Then
```

La listbox ne concerne qu'une seule table et un seul champ, car le tableau d'arguments se termine à Tablefield1 de la table auxiliaire..

```
            stSql = "SELECT """ + aTable(6) + """ FROM """ + aTable(5) + _
""" ORDER BY """ + aTable(6) + """"
            ElseIf UBound(aTable()) = 7 Then
```

La zone de liste concerne deux champs de table mais une seule table, car le tableau d'arguments se termine à Tablefield2 de la table auxiliaire.

```
            stSql = "SELECT """ + aTable(6) + """||' > '||""" + aTable(7) + """ FROM
""" _
                + aTable(5) + """ ORDER BY """ + aTable(6) + """"
            Else
```

La zone de liste est basée sur deux champs de table provenant de deux tables. Cette requête correspond à l'exemple avec le code postal et la ville.

```
            stSql = "SELECT """ + aTable(5) + """.""" + aTable(6) + _
"""||' > '||""" + aTable(8) + """.""" + aTable(7) + """ FROM """ + _
aTable(5) + """, """ + aTable(8) + """ WHERE """ + aTable(8) + _
""".""" + stIDEtrangerTab2Tab1 + """ = """ + aTable(5) + """.""" + _
stIDEtrangerTab1Tab2 + """ ORDER BY """ + aTable(6) + """"
            End If
```

Ici, nous avons la première évaluation pour déterminer les clés étrangères. Les variables `stIDEtrangerTab2Tab1` et `stIDEtrangerTab1Tab2` commencent par l'ID de valeur. Pour `stIDEtrangerTab1Tab2`, l'évaluation de la requête précédente donne une valeur différente, à savoir la valeur de ID_Ville. De cette manière, la construction de la requête précédente donne exactement le contenu déjà formulé pour le code postal et la ville – uniquement amélioré par le tri.

Il faut maintenant prendre contact avec les zones de liste, pour leur fournir le contenu renvoyé par les requêtes. Ces zones de liste n'existent pas encore, car la boîte de dialogue elle-même n'a pas encore été créée. Cette boîte de dialogue est d'abord créée en mémoire, en utilisant les lignes suivantes, avant d'être réellement dessinée à l'écran.

```
        DialogLibraries.LoadLibrary("Standard")
        oDlg = CreateUnoDialog(DialogLibraries.Standard.Dialogue_Ajustement_Tables)
```

Viennent ensuite les paramètres des champs de la boîte de dialogue. Voici, par exemple, la listbox qui doit être fournie avec les résultats de la requête ci-dessus :

```
        oCtlListe1 = oDlg.GetControl("ZoneListe1")
        oCtlListe1.addItems(acontenu(),0)
```

L'accès aux champs de la boîte de dialogue s'effectue à l'aide de GetControl avec le nom approprié. Dans les boîtes de dialogue, il n'est pas possible que deux champs utilisent le même nom car cela créerait des problèmes lors de l'évaluation de la boîte de dialogue.

La zone de liste est fournie avec le contenu de la requête, qui a été stocké dans le tableau acontenu(). La zone de liste contient uniquement le contenu à afficher sous forme de champ, donc seule la position 0 est remplie.

Une fois que tous les champs avec le contenu souhaité ont été remplis, la boîte de dialogue s'ouvre.

```
            Select Case oDlg.Execute()
                Case 1  ' Le cas 1 signifie que le bouton "OK" a été cliqué
                Case 0 ' S'il s'agissait du bouton "Annuler"
                    inRepete = 0
```

```
            End Select
        Loop While inRepete = 1
```

La boîte de dialogue s'exécute de manière répétée tant que la valeur de "inRepete" est 1. Ceci est défini par la case à cocher correspondante.

Voici, en bref, le contenu après avoir cliqué sur le bouton "OK":

```
    Case 1
        stContenu1 = String_to_SQL(oCtlListe.getSelectedItem()
        'Lire la valeur de ZoneListe1… et déterminer la valeur ID correspondante.
```

La valeur ID de la première listbox est stockée dans la variable "inZC1".

```
            stTexte = oCtlTexte.Text ' Lire la valeur du champ.
```

Si le champ de texte n'est pas vide, l'entrée du champ de texte est gérée. Ni la zone de liste pour une valeur de remplacement ni la case à cocher pour supprimer tous les enregistrements orphelins ne sont prises en compte. Ceci est rendu clair par le fait que la saisie de texte définit ces autres champs comme inactifs.

```
            If stTexte <> "" Then
```

Si le champ de texte n'est pas vide, la nouvelle valeur est écrite à la place de l'ancienne en utilisant le champ ID précédemment lu dans la table. Il y a la possibilité de deux entrées, comme c'est aussi le cas dans la zone de liste. Le séparateur est ">". Pour deux entrées dans des tables différentes, deux commandes UPDATE doivent être lancées, qui sont créées ici simultanément et transmises, séparées par un point-virgule.

```
            ElseIf oCtlListe2.getSelectedItem() <> "" Then
```

Si le champ de texte est vide et que la zone de liste 2 contient une valeur, la valeur de la listbox 1 doit être remplacée par la valeur de la zone de liste 2. Cela signifie que tous les enregistrements des tables pour lesquels les enregistrements des zones de liste sont des clés étrangères doivent être vérifiés et, si nécessaire, écrit avec une clé étrangère modifiée.

```
            stContenu2 = oCtlListe2.getSelectedItem()
            REM Lire la valeur de la liste.
            REM Déterminer l'ID pour la valeur de la liste.
```

La valeur ID de la deuxième zone de liste est stockée dans la variable inZC2. Ici aussi, les choses évoluent différemment selon que un ou deux champs sont contenus dans la zone de liste, et aussi selon que une ou deux tables sont à la base du contenu de la zone de liste.

Le processus de remplacement dépend de la table définie comme table fournissant la clé étrangère pour la table principale. Pour l'exemple ci-dessus, il s'agit de la table CodePostal, car ID_CodePostal est la clé étrangère qui est transmise via Zone de liste 1 et Zone de liste 2.

```
        If stTableSecondaire = aTable(5) Then
            For i = LBound(aTablesEtrangeres()) To UBound(aTablesEtrangeres())
```

Le remplacement de l'ancienne valeur ID par la nouvelle valeur ID devient problématique dans les relations n : m, car dans de tels cas, la même valeur peut être affectée deux fois. C'est peut-être ce que vous souhaitez, mais cela doit être évité lorsque la clé étrangère fait partie de la clé primaire. Ainsi, dans la table relation_Media_Auteur, un support ne peut pas avoir le même auteur deux fois, car la clé primaire est construite à partir de ID_Media et ID_Auteur. Dans la requête,

tous les champs de clé sont recherchés qui ont collectivement la propriété UNIQUE ou ont été définis comme des clés étrangères avec la propriété UNIQUE à l'aide d'un index.

Donc, si la clé étrangère a la propriété UNIQUE et y est déjà représentée avec le futur souhaité inZC2, cette clé ne peut pas être remplacée.

```
stSql = "SELECT ""COLUMN_NAME"" FROM ""INFORMATION_SCHEMA"". _
""SYSTEM_INDEXINFO"" WHERE ""TABLE_NAME"" = '" + aTablesEtrangeres(i, 0) + "' AND _
""NON_UNIQUE"" = False AND ""INDEX_NAME"" = (SELECT ""INDEX_NAME"" FROM _
""INFORMATION_SCHEMA"".""SYSTEM_INDEXINFO"" WHERE ""TABLE_NAME"" = '" _
+ aTablesEtrangeres(i, 0) + "' AND ""COLUMN_NAME"" = '" + aTablesEtrangeres(i, 1) _
+ "')"
```

""NON_UNIQUE"" = False donne les noms des colonnes UNIQUE. Cependant, tous les noms de colonnes ne sont pas nécessaires, mais uniquement ceux qui forment un index avec le champ de clé étrangère. Ceci est géré par la sous-sélection avec les mêmes noms de table (qui contiennent la clé étrangère) et les noms des champs de clé étrangère.

Si maintenant la clé étrangère est présente dans l'ensemble, la valeur de clé ne peut être remplacée que si d'autres champs sont utilisés pour définir l'index correspondant comme UNIQUE. Lorsque vous effectuez des remplacements, vous devez veiller à ce que l'unicité de la combinaison d'indices ne soit pas compromise.

```
        If aTablesEtrangeres(i, 1) = stNomChamp Then
            inUnique = 1
        Else
            ReDim Preserve aColonnes(inCount)
            aColonnes(inCount) = oQuery_result.getString(1)
            inCompteur = inCompteur + 1
        End If
```

Tous les noms de colonne, à l'exception des noms de colonne connus pour les champs de clé étrangère comme Index avec la propriété UNIQUE, sont stockés dans le tableau. Comme le nom de colonne du champ de clé étrangère appartient également au groupe, il peut être utilisé pour déterminer si l'unicité doit être vérifiée lors de la modification des données.

```
If inUnique = 1 Then
       stSql = "UPDATE """ + aTablesEtrangeres(i, 0) + """ AS ""a"" SET """ + _
aTablesEtrangeres(i, 1) + """='" + inZC2 + "' WHERE """ + _
aTablesEtrangeres(i, 1) + """='" + inZC1 + "' AND (SELECT COUNT(*) FROM """ + _
aTablesEtrangeres(i, 0) + """ WHERE """ + aTablesEtrangeres(i, 1) + _
"""='" + inZC2 + "')"
       If inCompteur > 0 Then
       stGroupeChamp = Join(aColonnes(), """|| ||""")
```

S'il existe plusieurs champs, en dehors du champ de clé étrangère, qui forment ensemble un index UNIQUE, ils sont ici combinés pour un regroupement SQL. Sinon, seul aColonnes(0) apparaît comme stFieldgroup.

```
            stNomDuChamp = ""
            For ink = LBound(aColonnes()) To UBound(aColonnes())
               stNomDuChamp = stNomDuChamp + " AND """ + aColonnes(ink) + """ = _
               ""a"".""" + aColonnes(ink) + """ "
```

Les parties SQL sont combinées pour une sous-requête corrélée.

```
            Next ink
        stSql = Left(stSql, Len(stSql) - 1)
```

La requête précédente se termine par un crochet. À présent, du contenu supplémentaire doit être ajouté à la sous-requête, cette fermeture doit donc être supprimée à nouveau. Après cela, la requête est développée avec les conditions supplémentaires.

```
        stSql = stSql + stNomDuChamp + "GROUP BY (""" + stGroupeChamp + """)) < 1"
End If
```

Si la clé étrangère n'a pas de connexion avec la clé primaire ou avec un index UNIQUE, peu importe si le contenu est dupliqué.

```
Else
        stSql = "UPDATE """ + aTablesEtrangeres(i, 0) + """ SET """ + _
aTablesEtrangeres(i, 1) + """='" + inZC2 + "' WHERE """ + _
aTablesEtrangeres(i, 1) + """='" + inZC1 + "'"
End If
oSQL_Commande.executeQuery(stSql)
NEXT
```

La mise à jour est effectuée aussi longtemps que différentes connexions à d'autres tables se produisent, autrement dit, tant que la table actuelle est la source d'une clé étrangère dans une autre table. C'est le cas deux fois pour la table Villes : dans la table Média et dans la table CodePostal.

Ensuite, l'ancienne valeur peut être supprimée de la zone de liste 1, car elle n'a plus de connexion avec d'autres tables.

```
stSql = "DELETE FROM """ + aTable(5) + """ WHERE ""ID""='" + inZC1 + "'"
oSQL_Commande.executeQuery(stSql)
```

Dans certains cas, la même méthode doit maintenant être utilisée pour une deuxième table qui a fourni des données pour les zones de liste. Dans notre exemple, la première table est la table CodePostal et la seconde est la table Villes.

Si le champ de texte est vide et que la zone de liste 2 ne contient également rien, nous vérifions si la case à cocher correspondante indique que toutes les entrées excédentaires doivent être supprimées. Cela signifie les entrées qui ne sont pas liées à d'autres tables par une clé étrangère.

```
ElseIf oCtlCoche1.State = 1 Then
       stCondition = ""
       If stTableSecondaire = aTable(5) Then
          For i = LBound(aForeignTables()) To UBound(aTablesEtrangeres())
             stCondition = stCondition + """ID"" NOT IN (SELECT """ + _
aTablesEtrangeres(i, 1) + """ FROM """ + aTablesEtrangeres(i, 0) + """) AND "
          Next
       Else
          For i = LBound(aForeignTables2()) To UBound(aTablesEtrangeres2())
             stCondition = stCondition + """ID"" NOT IN (SELECT """ + _
aTablesEtrangeres2(i, 1) + """ FROM """ + aForeignTables2(i, 0) + """) AND "
          Next
       End If
```

Le dernier AND doit être supprimé, sinon l'instruction de suppression se terminerait par AND.

```
       stCondition = Left(stCondition, Len(stCondition) - 4)
       stSql = "DELETE FROM """ + stTableSecondaire + """ WHERE " + stCondition + ""
       oSQL_Commande.executeQuery(stSql)
```

Comme la table a déjà été purgée une fois, l'index de la table peut être vérifié et éventuellement corrigé vers le bas. Voir le sous-programme décrit dans l'une des sections précédentes.

```
MinimiseIndexTable(stTableSecondaire)
```

Ensuite, si nécessaire, la zone de liste sous la forme à partir de laquelle la boîte de dialogue Table_purge a été appelée peut être mise à jour. Dans certains cas, le formulaire entier doit être relu. Dans ce but, l'enregistrement en cours est déterminé au début du sous-programme de sorte qu'après l'actualisation du formulaire, l'enregistrement en cours puisse être rétabli.

```
        oDlg.endExecute()   'Fermeture dialogue…
        oDlg.Dispose()   '… et retirer de la mémoire
End Sub
```

Les boîtes de dialogue se terminent avec la commande endExecute() et sont complètement supprimées de la mémoire avec Dispose().

Écrire des macros avec Access2Base

Les versions de LibreOffice à partir de 4.2 ont intégré Access2Base. Cette bibliothèque introduit une couche de base avec son API spécifique (Application Programming Interface) entre le code de l'utilisateur et l'interface UNO habituelle. L'API fournie n'apporte pas en soi de nouvelles fonctionnalités mais, dans de nombreux cas, elle est plus lisible, concise et plus facile à utiliser que UNO.

L'API ressemble beaucoup à celle conçue par Microsoft pour le logiciel Access. Base et Access ont beaucoup en commun, mais certainement pas leurs styles de programmation natifs. Access2Base comble le vide.

Une documentation en anglais avec des exemples est disponible sur http://www.access2base.com/access2base.html

Pour illustrer brièvement comment Access2Base cache la complexité de UNO :

- La propriété (Access2Base simple) **Value** d'un contrôle a dans UNO comme équivalents, selon le type de contrôle ou son emplacement dans un formulaire, un contrôle de grille ou une boîte de dialogue : *CurrentValue, Date, EffectiveValue, HiddenValue, ProgressValue, RefValue, ScrollValue, SpinValue, State, StringItemList, Text, Time, ValueItemList* ou… *Value*.
- Pour obtenir les N premiers enregistrements d'une table ou d'une requête dans un tableau de base, une méthode consiste simplement à utiliser la méthode *GetRows(N)* sur un objet *Recordset*. Comparez avec les méthodes *getString, getNull, getDouble, getLong*… dans UNO que vous devez appliquer aux champs en fonction de leur type et du système de base de données utilisé.

Il existe deux catégories principales d'objets gérés par Access2Base, ciblant soit :

- L'interface utilisateur. Les classes d'objets typiques sont : Form, SubForm, Dialog, Control, CommandBar, CommandBarControl et Event. Leurs méthodes sont généralement appelées à partir d'une application de base.
- La base de données accèdée. Les classes d'objets typiques sont : Database, TableDef, QueryDef, Recordset et Field. Leurs méthodes sont appelées soit à partir d'une application Base, soit à partir de toute autre application LibreOffice.

Traditionnellement, l'API Access2Base est appelée à partir du langage de base. À partir de LibreOffice 6.4, une passerelle permet également d'accéder à l'API à partir de scripts Python, sans

aucune limitation par rapport à Basic. On peut intégrer de manière transparente des scripts Basic et Python dans la même application, même en partageant les mêmes instances d'objet.

Dans les paragraphes suivants, chaque exemple sera donné à la fois en Basic et en Python. Ils sont strictement équivalents.

Pour accéder à la bibliothèque à partir d'une application Base, attachez la procédure suivante à l'événement OpenDocument de votre fichier de base :

(BASIC)

```
Sub OuvreBDD(Optional oEvent As Object)
	If GlobalScope.BasicLibraries.hasByName("Access2Base") then
	   GlobalScope.BasicLibraries.loadLibrary("Access2Base")
	End If
	Call Application.OpenConnection(ThisDatabaseDocument)
End Sub
```

(PYTHON)

```
from access2base import *
def OuvreBDD(event = None):
	  Application.OpenConnection()
g_exportedScripts = (DBOpen,)
```

Sinon, pour accéder à la base de données à partir d'une application autre Base, exécutez :

(BASIC)

```
Function OuvreBDD() As Object
	If GlobalScope.BasicLibraries.hasByName("Access2Base") then
	   GlobalScope.BasicLibraries.loadLibrary("Access2Base")
	End If
	Set maBDD = Application.OpenDatabase("… Nom de la base de données… ")
End Function
```

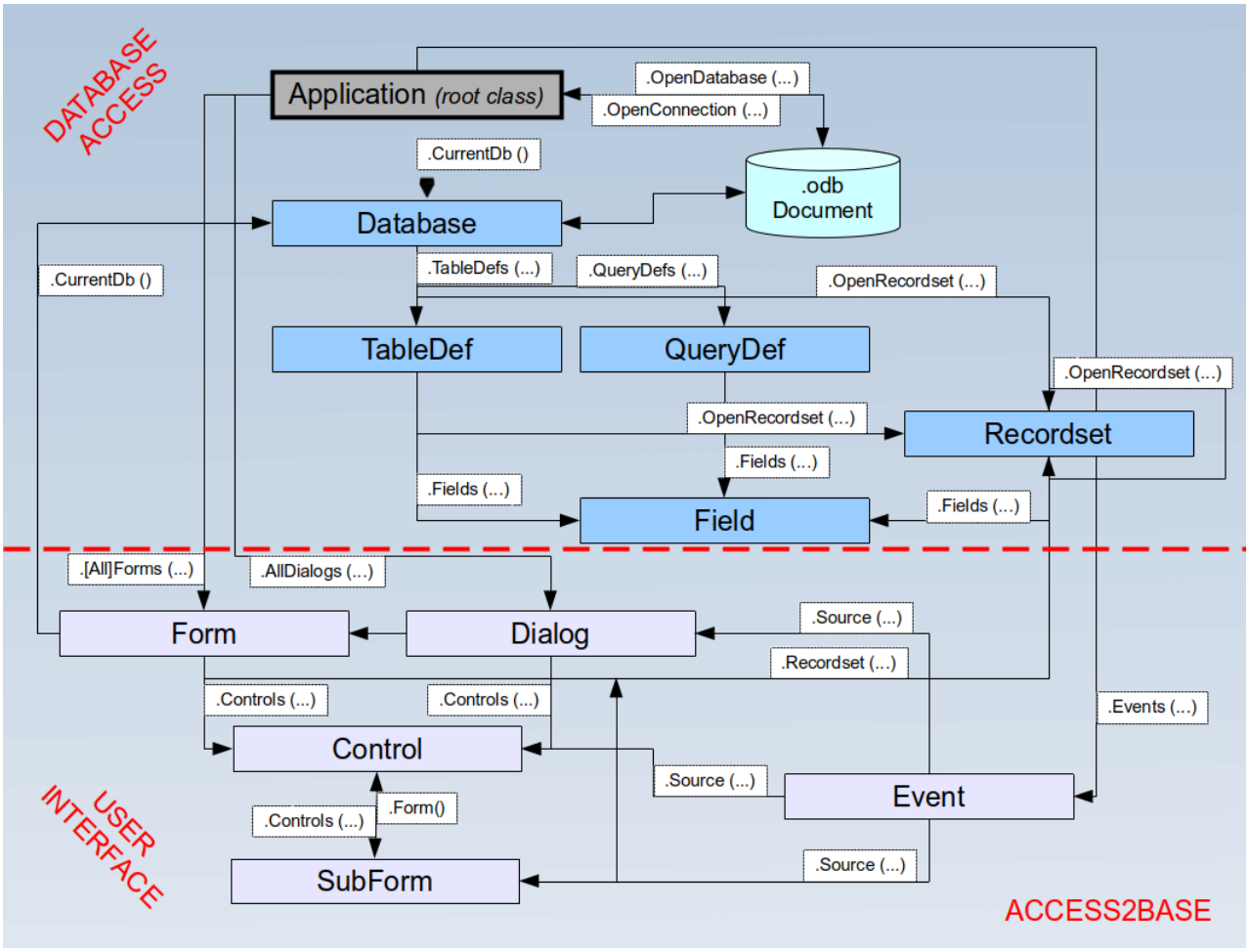
(PYTHON)

```
from access2base import *
def OuvreBDD():
	  return Application.OpenDatabase('… Nom de la base de données… ')
```

Le but de ce manuel n'est pas de reproduire la documentation du site Web susmentionné. Nous limiterons ce document à un résumé des principaux concepts de l'API.

Le modèle Objet

Ci-dessous, à partir de l'objet racine Application (*Application root object*), se trouve un schéma décrivant la navigation à travers les objets les plus utilisés :



À titre d'exemple, pour aider à lire le schéma :

- **Application** est le principal des deux objets racine.
- Les méthodes *CurrentDb()* et *OpenDatabase()* fournissent un objet **Database**.
- La collection *Database.TableDefs()* répertorie toutes les tables stockées dans la base de données. Chaque table est représentée par une instance d'objet **TableDef**.
- La collection *TableDef.Fields()* répertorie tous les champs présents dans la table de base de données. Chaque champ est représenté par une instance d'objet **Field**.
- Les champs peuvent également être les enfants de requêtes stockées (**QueryDefs**) ou d'ensembles de données (**Recordsets**), avec les mêmes attributs.

Quelques exemples

Imprimer une liste de noms de tables et de champs

(BASIC)

```
Sub ScanTables()
Dim oBaseDonnees As Object, oTable As Object, oChamp As Object
Dim i As Integer, j As Integer
       Set oBaseDonnees = Application.CurrentDb()
       With oBaseDonnees
          For i = 0 To .TableDefs.Count - 1
```

```
            Set oTable = .TableDefs(i)       ' Obtenir la définition de chaque table
            DebugPrint oTable.Name
            For j = 0 To oTable.Fields.Count - 1
               Set oChamp = oTable.Fields(j)' Obtenir chaque champ individuellement
               DebugPrint "", oChamp.Name, oChamp.TypeName
            Next j
         Next i
      End With
End Sub
```

(PYTHON)

```
def ScanTables():
      oBaseDonnees = Application.OpenDatabase("/home/somedir/TT NorthWind.odb")
      for oTable in oBaseDonnees. TableDefs():
         DebugPrint(oTable. Name)
         for oField in oTable.Fields():
            DebugPrint("", oField.Name, oField.TypeName)
```

Stocker les données produites par une requête dans un tableau Base ou un tuple Python

(BASIC)

```
Sub ChargeRequete()
Dim oEnregistrements As Object, vDonnees As Variant
      Set oEnregistrements = Application.CurrentDb().OpenRecordset("maRequete")
      vDonnees = oEnregistrements.GetRows(1000)
      oEnregistrements.mClose()
End Sub
```

(PYTHON)

```
def ChargeRequete():
         oEnregistrements= Application.CurrentDb().Openrecordset("maRequete")
         vData = oEnregistrements.GetRow(1000)
         oEnregistrements.Close()
```

Définir les valeurs par défaut dans les entrées de formulaire

Pour spécifier qu'après chaque entrée d'enregistrement, un contrôle est pré-rempli avec la dernière valeur définie, affectez la routine suivante à l'événement *Après le changement d'enregistrement* du formulaire :

(BASIC)

```
Sub SetDefaultNewRec(poEvent As Object)
Dim oFormulaire As Object, oControl As Object
      Set oFormulaire = Application.Events(poEvent).Source '  Obtenir le formulaire
actuel
      Set oControle = oFormulaire. Controls("txtPays")
      oControle.DefaultValue = oControle.Value
End Sub
```

(PYTHON)

```
def SetDefaultNewRec(poEvent):
         oFormulaire = Application.Events(poEvent).Source
         oControl = oForm.Controls("txtCountry")
         oControl.DefaultValue = oControl.Value
```

Fonctions de base de données

Un ensemble de fonctions est fourni pour raccourcir à une seule ligne l'accès aux valeurs de la base de données : *DLookup, DMax, DMin, Dsum*. Ils acceptent tous les mêmes arguments : un nom de champ ou une expression basée sur des noms de champ, un nom de table ou de requête et une clause SQL-where sans le mot clé *WHERE*. Par exemple :

(BASIC)

```
Function Lookup(psField As String, psSearchField As String, _
                        psSearchValue As String) As Variant
      Lookup = Application.DLookup(psField, "maTable", psSearchField & "='" _
                & psSearchValue & "'")
End Function
```

(PYTHON)

```
def Lookup(psField, psSearchField, psSearchValue):
          return Application.Dlookup(psField, "myTable"
          , psSearchField + "='" + psSearchValue + "'")
```

Commandes spéciales

Le DoCmd (2e classe racine) propose un ensemble de fonctions pratiques permettant d'exécuter en une instruction Basic des actions complexes bien que fréquentes et pratiques. Pour n'en nommer que quelques-unes :

CopyObject	Copie une table ou une requête dans la même base de données ou entre deux bases de données.
OpenSQL	Exécute une instruction SQL SELECT donnée et affiche le résultat dans une feuille de données.
OutputTo	Stocke les données d'une table ou d'une requête dans un fichier HTML. Stocke le contenu réel d'un formulaire dans un fichier PDF.
SelectObject	Activer la fenêtre donnée (formulaire, rapport,…)
SendObject	Envoyer par mail avec le formulaire donné en pièce jointe.

L'objet "Basic" en Python

À partir de Python, un objet supplémentaire a été introduit dans la passerelle Access2Base, simplement appelé « Basic », pour permettre d'exécuter avec la bibliothèque BASIC un certain nombre de fonctions intégrées de base bien connues. Elles sont invoquées exactement avec les mêmes arguments et se comportent strictement de la même manière dans les deux environnements. Parmi elles :

- Fonctions système : Basic.ConvertToUrl and Basic.ConvertFromUrl, Basic.GlobalScope, Basic.CreateUnoService.
- Fonctions de l'interface utilisateur : Basic.MsgBox, Basic.InputBox
- Fonctions de date : Basic.DateAdd, Basic.DateDiff
- Introspection d'objets UNO : Basic.Xray

LibreOffice

Guide Base

Chapitre 10 Maintenance de Bases de Données

Remarques générales sur la gestion des bases de données

La modification fréquente des données dans une base de données – en particulier la suppression fréquente des données – a deux effets. Premièrement, la base de données s'agrandit régulièrement, même si elle ne contient en fait pas plus de données. Deuxièmement, la clé primaire créée automatiquement continue de s'accroître, quelle que soit la valeur de la clé suivante nécessaire. Plusieurs importantes méthodes de maintenance sont décrites dans ce chapitre.

Compacter une base de données

Hsqldb a la particularité de continuer à fournir un espace de stockage même pour les données qui ont déjà été supprimées. Les bases de données qui étaient autrefois remplies de données, en particulier d'images, à des fins de tests, ont toujours la même taille, même si toutes ces données ont été supprimées.

Pour libérer à nouveau l'espace de stockage, les fichiers de la base de données doivent être réécrits (tables, descriptions de ces tables, etc.).

Directement sur l'interface de la Base, une commande simple peut être saisie directement via **Outils** → **SQL** qui est réservé à l'administrateur du système dans le cas des bases de données serveur :

```
SHUTDOWN COMPACT
```

La base de données est arrêtée et libérée de tous les anciens chargements. Toutefois, la base doit être redémarrée par la suite si l'on veut y accéder à nouveau.

Depuis la version LO 3.6, la compression est effectuée par défaut au plus tard à la fermeture de la base de données.

Réinitialisation des valeurs automatiques

Lorqu'une base de données est créée, toutes les fonctions possibles testées avec des exemples et des corrections peuvent être apportées jusqu'à ce que tout fonctionne. Alors, avant même qu'une base de données ne soit prête à être utilisée, il est possible que les valeurs de clé primaire prennent une valeur trop importante. Souvent, les clés primaires sont définies sur l'incrémentation automatique. Si les tables sont vidées en vue d'une utilisation normale ou avant de transmettre la base de données à une autre personne, la clé primaire continue à s'incrémenter à partir de sa position actuelle au lieu de se réinitialiser à zéro.

La commande SQL suivante, entrée via **Outils> SQL**, vous permet de réinitialiser la valeur initiale :

```
ALTER TABLE "Nom_Table" ALTER COLUMN "ID" RESTART WITH NouvelleValeur
```

Cela suppose que le champ de clé primaire a pour nom ID et a été défini comme un champ de valeur automatique. La nouvelle valeur doit être celle que vous souhaitez créer automatiquement pour le nouvel enregistrement suivant. Ainsi, par exemple, si les enregistrements actuels vont jusqu'à 4, la nouvelle valeur doit être 5 sans modifier le champ ID. La première valeur d'ID sera la NouvelleValeur dans l'instruction SQL ci-dessus.

Interroger les propriétés de la base de données

Toutes les informations sur les tables de la base de données sont stockées sous forme de table dans une partie distincte de HSQLDB. Cette zone distincte peut être atteinte en utilisant le nom **INFORMATION_SCHEMA**.

La requête suivante peut être utilisée pour trouver les noms de champ, les types de champ, les tailles de colonne et les valeurs par défaut. Voici un exemple de table nommée TableCherche.

```
SELECT "COLUMN_NAME",
       "TYPE_NAME",
       "COLUMN_SIZE",
       "COLUMN_DEF" AS "Valeur defaut"
FROM "INFORMATION_SCHEMA"."SYSTEM_COLUMNS"
WHERE "TABLE_NAME" = "TableCherche"
ORDER BY "ORDINAL_POSITION"
```

Toutes les tables spéciales de HSQLDB sont décrites dans l'annexe A de ce livre. Les informations sur le contenu de ces tables sont facilement obtenues par des requêtes directes :

```
SELECT * FROM "INFORMATION_SCHEMA"."SYSTEM_PRIMARYKEYS"
```

L'astérisque (*) garantit que l'ensemble des colonnes seront retournées La table explorée ci-dessus retourne les informations essentielles sur les clés primaires et les différentes tables.

Ces informations sont surtout utiles pour les macros. Au lieu d'avoir à fournir des informations détaillées sur chaque table ou base de données nouvellement créée, des procédures sont écrites pour extraire ces informations directement de la base de données et sont donc universellement applicables. L'exemple de base de données le montre, entre autres, dans l'un des modules de maintenance, où les clés étrangères sont déterminées.
Firebird contient des informations correspondantes, mais malheureusement un peu plus dispersées dans les bases de données de son système :

```
SELECT "a".RDB$RELATION_NAME AS "Tables",
       "a".RDB$FIELD_NAME AS "Champs",
       "c".RDB$TYPE_NAME AS "Types",
       "a".RDB$Field_POSITION AS "Position_Champs",
       "a".RDB$NULL_FLAG AS "Flag_Null",
       "b".RDB$FIELD_LENGTH AS "Longueur_Champs"
FROM RDB$RELATION_FIELDS AS "a", RDB$FIELDS AS "b", RDB$TYPES AS "c"
WHERE "a".RDB$FIELD_SOURCE = "b".RDB$FIELD_NAME
      AND "b".RDB$FIELD_TYPE = "c".RDB$TYPE
      AND "c".RDB$FIELD_NAME = 'RDB$FIELD_TYPE'
      AND "a".RDB$RELATION_NAME = 'Nom_Relation'
ORDER BY "Tables", "Position_Champs"
```

Exporter des données

Il existe une méthode beaucoup plus simple d'exportation de données que la méthode standard, en ouvrant le fichier *.odb. Directement sur l'interface de base, vous pouvez utiliser **Outils> SQL** pour saisir une commande simple. Dans les bases de données serveur, celle-ci est réservée à l'administrateur système.

```
SCRIPT 'Nom_de_Fichier'
```

'Nom_De_Fichier' doit être de la forme : Chemin\ Nom_De_Fichier

Exemple (sous windows) : C:\Users\...\Exemple_Cherche_et_Filtre.SQL ou
sous Linux *:* /home/.../ *E*xemple_Cherche_et_Filtre.SQL

Cela crée une extraction SQL complète de la base de données avec toutes les définitions de table, les relations entre les tables et les données. Les requêtes et les formulaires ne sont pas extraits, car ils ont été créés dans l'interface utilisateur et ne sont pas stockés dans la base de données interne. Cependant, toutes les vues sont incluses.

Note

Cette procédure peut être utilisée pour mettre à jour une base de données intégrée pour la connexion à la base de données externe avec HSQLDB 2.50. Là encore, les requêtes et les formulaires doivent être remplacés.

Par défaut, le fichier exporté est un fichier texte normal,

Le fichier peut également être fourni sous forme binaire ou compressé (sous forme zippée), ce qui est utile pour les bases de données volumineuses. Cependant, cela rend sa réimportation dans LibreOffice Base un peu plus compliquée.

Le format du fichier exporté peut être modifié en utilisant :

```
SET SCRIPTFORMAT {TEXT | BINARY | COMPRESSED}
```

Pour exporter le fichier, vous devez utiliser ce code SQL une ligne à la fois : :

```
SCRIPT 'Nom_De_Fichier'
SET SCRIPTFORMAT {TEXT | BINARY | COMPRESSED}
CHECKPOINT
SHUTDOWN SCRIPT
```

La documentation concernant ces commandes se trouvent dans le guide (en anglais) pour HSQLDB 1.8 à l'adresse : http://www.hsqldb.org/doc/1.8/guide/guide.html ou guide.pdf.

Le fichier peut être importé en utilisant **Outils> SQL** et en créant une nouvelle base de données avec les mêmes données. Dans le cas d'une base de données interne, les lignes suivantes doivent être supprimées avant l'importation :

```
CREATE SCHEMA PUBLIC AUTHORIZATION DBA
CREATE USER SA PASSWORD ""
GRANT DBA TO SA
SET WRITE_DELAY 60
SET SCHEMA PUBLIC
```

Ces entrées traitent du profil utilisateur et d'autres paramètres par défaut, qui sont déjà définis pour les bases de données internes de LibreOffice. Par conséquent, un message d'erreur apparaît si l'une de ces lignes est présente. Ils se trouvent directement avant le contenu qui sera inséré dans les tables à l'aide de la commande INSERT.

Pour importer ce fichier, son contenu doit être divisé en plusieurs fichiers texte créés par un simple programme d'édition de texte. Le premier fichier doit contenir toutes les tables et vues de création. Copiez toutes les lignes depuis le début avec CREATE TABLE et en arrêtant une ligne au-dessus de la ligne contenant INSERT INTO. Collez-le dans le premier fichier. Copiez et collez les lignes restantes dans le deuxième fichier.

Il y a une limite à la taille du deuxième fichier : il doit être inférieur à 65 Ko. S'il est plus grand que cela, il doit également être divisé en fichiers texte plus petits en coupant et en collant. Assurez-vous simplement que la première ligne de chacun de ces nouveaux fichiers commence par INSERT INTO. Une façon de faire est de couper à partir du bas jusqu'à une telle ligne.

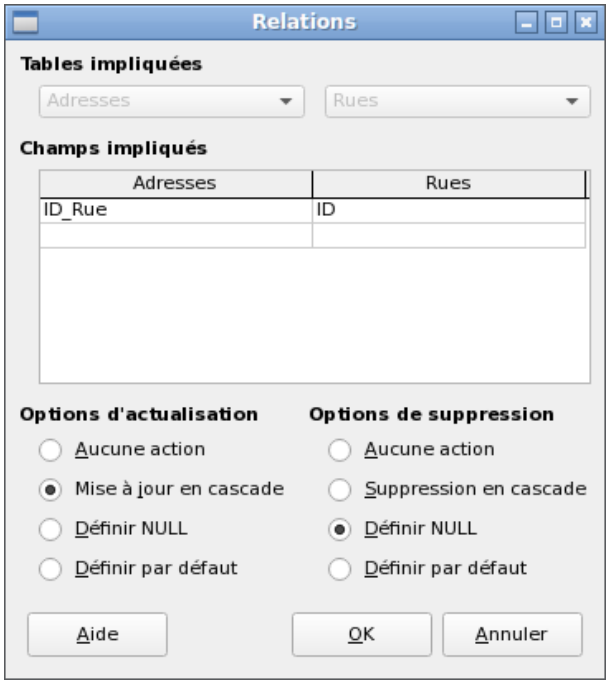
Tester les table pour les entrées inutiles

Une base de données se compose d'une ou plusieurs tables principales, qui contiennent des clés étrangères d'autres tables. Dans l'exemple de base de données, il s'agit des tables Medias et Adresses. Dans la table d'adresses, la clé primaire du code postal apparaît comme une clé étrangère. Si une personne déménage dans une nouvelle maison, l'adresse est modifiée. Le résultat peut être qu'aucune clé étrangère **ID_CodePostal** correspondant à ce code postal n'existe plus. En principe, le code postal lui-même pourrait donc être supprimé. Cependant, lors d'une utilisation normale, il n'est pas évident que l'enregistrement n'est plus nécessaire. Il existe différentes manières d'éviter ce genre de problème.

Test des entrées à l'aide de la définition de relation

L'intégrité des données peut être garantie lors de la définition des relations. En d'autres termes, vous pouvez empêcher la suppression ou la modification de clés de conduire à des erreurs dans la base de données. La boîte de dialogue suivante est accessible via **Outils> Relations**, suivi d'un clic droit sur le connecteur entre deux tables.

Ici, les tables **Adresses** et **Rues** sont considérées. Toutes les actions spécifiées s'appliquent à la table Adresses, qui contient la clé étrangère **ID_Rue**. Les options de mise à jour font référence à une mise à jour du champ **ID** dans la table **Rues**. Si la clé numérique dans le champ **"Rues"."ID"** est modifiée, **Aucune action** signifie que la base de données n'autorise pas ce changement si un **"Rues"."ID"** avec ce numéro de clé apparaît comme clé étrangère dans la table d'adresses.



La **Mise à jour en cascade** signifie que le numéro de clé est simplement reporté. Si la rue 'de la fontaine' dans la table Rues a l'ID '3' et est également représentée dans "Adresses"."ID_Rue", l'ID

peut être modifié en toute sécurité. Par exemple, si elle est remplacée par "67", les valeurs "Adresses"."ID_Rue" correspondantes seront automatiquement remplacées par "67".

Si **Définir Null** est choisi, la modification de l'ID fait de "Adresses"."ID_Rue" un champ vide.

Les options de suppression sont gérées de la même manière.

Pour les deux options, l'interface graphique actuelle n'autorise pas la possibilité **Définir par défaut** (même si les boutons sont présents) car les paramètres par défaut de l'interface graphique sont différents de ceux de la base de données. Voir le chapitre 3, Tables.

La définition de relations permet de garder les relations elles-mêmes propres, mais cela ne supprime pas les enregistrements inutiles qui fournissent leur clé primaire en tant que clé étrangère dans la relation. Il peut y avoir n'importe quel nombre de rues sans adresses correspondantes.

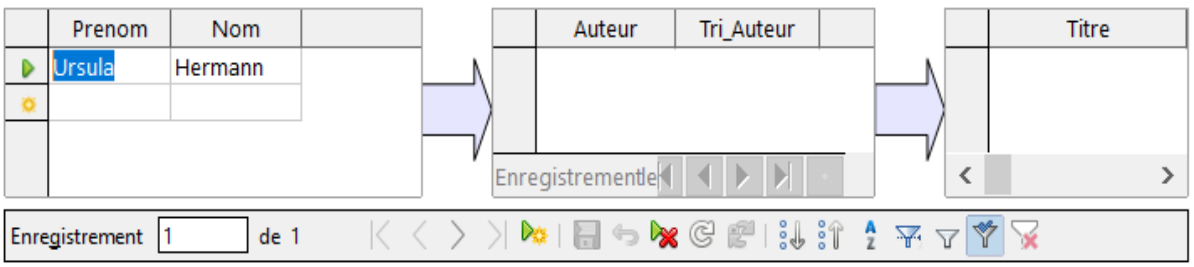
Modification d'entrées à l'aide de formulaires et de sous-formulaires

En principe, toute l'interrelation entre les tables peut être affichée dans des formulaires. Ceci est bien sûr plus simple lorsqu'une table est liée à une seule autre table. Ainsi, dans l'exemple suivant, la clé primaire de l'**auteur** devient la clé étrangère dans la table **relation_Media_Auteur**. rel_Media_Auteur contient également une clé étrangère de **Medias**, de sorte que la disposition suivante montre une relation **n:m** avec trois formes. Chacune est présentée à travers une table.

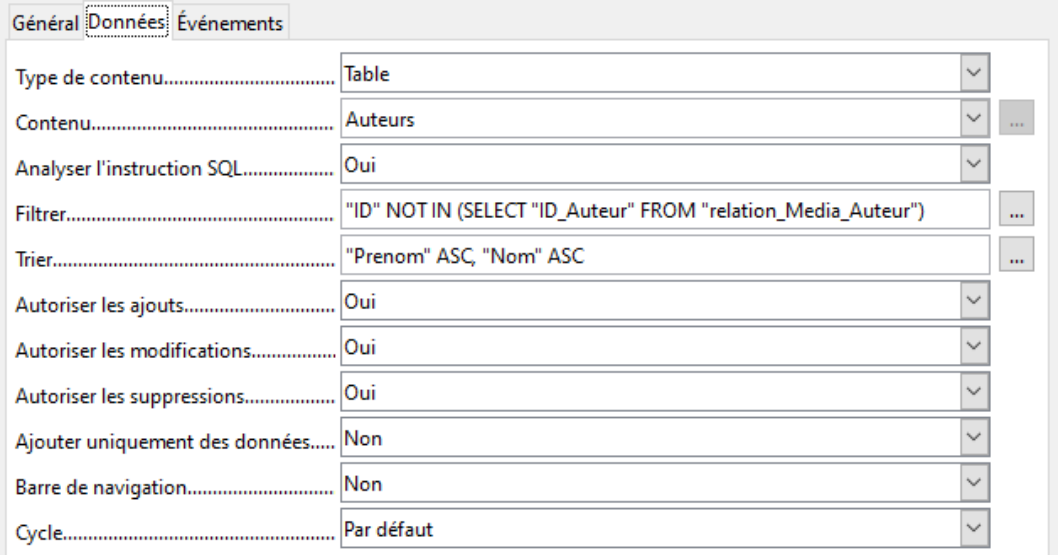
Les illustrations suivantes proviennent de captures d'écran du formulaire : **Maintenance**, dans la base : **Exemple_Media_Sans_Macro.odb**.

La première figure montre que le titre *Les nuits de Reykjavik* appartient à l'auteur *Arnaldur Indridason*. Donc cet auteur ne doit pas être supprimé – sinon les informations requises pour le média *Les nuits de Reykjavik* sera manquant. Cependant, la zone de liste vous permet de choisir un enregistrement différent à la place de *Arnaldur Indridason.*

Dans le formulaire, il y a un filtre intégré dont l'activation peut vous dire quelles catégories ne sont pas nécessaires dans la table **Medias**. Dans le cas qui vient d'être décrit, presque tous les auteurs sont lié à un média. Seul l'enregistrement Ursula Hermann peut être supprimé sans aucune conséquence pour tout autre enregistrement dans Medias. En effet il n'est relié à aucun média.



Le filtre est codé en dur dans ce cas. Il se trouve dans les propriétés du formulaire.



Un tel filtre est activé automatiquement au lancement du formulaire. Il peut être désactivé et activé. S'il est supprimé, il est à nouveau accessible par un rechargement complet du formulaire. Cela signifie plus qu'une simple mise à jour des données l'ensemble du formulaire doit être fermé puis rouvert.

Requêtes pour rechercher des entrées orphelines

Le filtre ci-dessus fait partie d'une requête qui peut être utilisée pour rechercher des entrées orphelines.

```
SELECT "Nom", "Prenom" FROM "Auteurs" WHERE "ID" NOT IN (SELECT "ID_Auteur"
FROM "relation_Media_Auteur")
```

Si une table contient des clés étrangères de plusieurs autres tables, la requête doit être étendue en conséquence. Cela affecte, par exemple, la table **Villes**, qui a des clés étrangères à la fois dans la table **Medias** et dans la table **CodePostal**. Par conséquent, les enregistrements de la table **Villes** qui doivent être supprimés ne doivent être référencés dans aucune de ces tables. Ceci est déterminé par la requête suivante : :

```
SELECT "Ville" FROM "Villes" WHERE "ID" NOT IN (SELECT "ID_Ville" FROM
"Medias") AND "ID" NOT IN (SELECT "ID_Ville" FROM "CodePostal")
```

Les entrées orphelines peuvent ensuite être supprimées en sélectionnant toutes les entrées qui passent le filtre défini et en utilisant l'option Supprimer du menu contextuel du pointeur d'enregistrement, appelée par un clic droit.

Vitesse de recherche dans la base de données

Effet des requêtes

Ce ne sont que ces requêtes, utilisées dans la section précédente pour filtrer les données, qui s'avèrent insatisfaisantes au regard de la vitesse maximale de recherche dans une base de données. Le problème est que dans les grandes bases de données, la sous-requête récupère une

quantité proportionnellement importante de données avec lesquelles chaque enregistrement affichable unique doit être comparé. Seules les comparaisons avec la relation 1-n permettent de comparer une seule valeur avec un ensemble de valeurs. La requête

```
… WHERE "ID" NOT IN (SELECT "ID_Auteur" FROM "relation_Media_Auteur")
```

peut contenir un grand nombre de clés étrangères possibles de la table **relation_Media_Auteur,** qui doivent d'abord être comparées aux clés primaires de la table **Auteurs** pour chaque enregistrement de cette table. Une telle requête n'est donc pas adaptée à une utilisation quotidienne mais peut être nécessaire pour la maintenance de la base de données

Les fonctions de recherche doivent être structurées différemment afin que la recherche de données ne prenne pas un temps interminable et ne gâche pas le travail avec la base de données dans les opérations quotidiennes.

Effet des zones de liste et des zones combinées

Plus un formulaire contient de zones de liste et plus le contenu qu'elles doivent fournir est important, et de ce fait plus la construction du formulaire prend du temps.

Les listes de sélection sont créées par des requêtes qui sont exécutées, pour chaque liste de sélection, lorsque le formulaire est lancé.

Plus Base met à disposition, en priorité, un affichage complet pour l'utilisateur, et ne lit que partiellement les zones de liste, moins la charge correspondante est perceptible.

La même structure de requête, pour davantage de zones de listes, est plus performante en utilisant une vue commune, au lieu de créer à plusieurs reprises des champs avec la même syntaxe en utilisant des commandes SQL stockées dans les zones de listes.

Les vues sont avant tout préférables pour les systèmes de bases de données externes, car dans ce cas le serveur s'exécute beaucoup plus rapidement qu'une requête qui doit être rassemblée par l'interface graphique et envoyée au serveur à chaque fois. Le serveur traite les vues comme des requêtes locales complètes.

Influence du système de base de données utilisé

Le SGBD (Système de Gestion de Base de Données) interne HSQLDB est conçu pour garantir que Base et Java fonctionnent bien ensemble.

Lorsque Base utilise HSQLDB, la taille et la réactivité de votre base sont limitées par rapport à l'utilisation d'un moteur de base de données externe. En particulier lorsque ce serveur de base de données est exécuté sur un ordinateur distinct.

Si les fonctions de votre base de données commencent à ralentir, suivez d'abord les étapes de ce guide pour nettoyer les espaces vides, les données supprimées ou temporaires et vérifiez que vous utilisez des index là où ils ont du sens. Si la réactivité ne revient pas, envisagez de déplacer vos données du fichier de base vers un serveur de base de données externe.

Les bases de données externes s'exécutent beaucoup plus rapidement. En termes de vitesse, les connexions directes à MySQL ou PostgreSQL et les connexions via ODBC (Open DataBase Connectivity) sont presque équivalentes. JDBC (Java DataBase Connectivity) dépend également de l'interaction avec Java, mais fonctionne beaucoup plus rapidement qu'une connexion interne à HSQLDB.

Guide Base

Appendice A
Documentations diverses

Codes à barres

Pour pouvoir utiliser la fonction d'impression de codes-barres, la police ean13.ttf doit être installée. Cette police est disponible gratuitement http://www.codebarre.be/barcodefont/ean13.ttf.

Les codes-barres EAN13 peuvent être créés en utilisant `ean13.ttf` comme suit :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Nombre	Majuscule, A=0 B=1, etc.						*	Minuscule, a=0 b=1, etc.						+

Voir aussi la requête `Barcode_EAN13_ttf_Rapport` et le rapport correspondant dans l'exemple de base de données Media_sans_Macros.

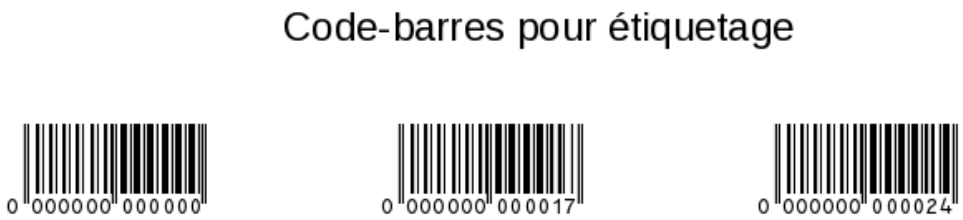


Figure 119: Impression du rapport avec le code ean13

Types de données pour l'éditeur de tables

Entiers

Type	***Option***	***HSQLDB***	***Intervalle***	***Stockage***
Tiny Integer	TINYINT	TINYINT	2^8 = 256 | – 128 to + 127	1 Byte
Small Integer	SMALLINT	SMALLINT	2^{16} = 65536 | – 32768 to + 32767	2 Octets
Integer	INTEGER	INTEGER | INT	2^{32} = 4294967296 | – 2147483648 to + 2147483647	4 Octets
BigInt	BIGINT	BIGINT	2^{64}	8 Octets

Nombres à virgule flottante

Type	***Option***	***HSQLDB***	***Intervalle***	***Stockage***

Decimal	DECIMAL	DECIMAL	Illimité, jusqu'à 50 emplacements dans l'interface graphique, virgule décimale fixe, précision parfaite	variable
Nombre	NUMERIC	NUMERIC	Illimité, jusqu'à 50 emplacements dans l'interface graphique, virgule décimale fixe, précision parfaite	variable
Flottant	FLOAT	(DOUBLE used instead)		
Réel	REAL	REAL		
Double	DOUBLE	DOUBLE [PRECISION] | FLOAT	Réglable, pas exact, 15 décimales maximum	8 Octets

Autres

Type	*Option*	*HSQLDB*	*Intervalle*	*Stockage*
Texte	VARCHAR	VARCHAR	Ajustable	variable
Texte	VARCHAR_ IGNORECASE	VARCHAR_ IGNORECASE	Réglable, la plage affecte le tri	variable
Texte (fixe)	CHAR	CHAR | CHARACTER	Réglable, reste du texte réel remplacé par des espaces	fixe
Memo	LONGVARCHAR	LONGVARCHAR		variable

Date/Temps/Heures

Type	*Option*	*HSQLDB*	**Intervalle**	**Stockage**
Date	DATE	DATE		4 Octets
Time	TIME	TIME		4 Octets
Date/Time	TIMESTAMP	TIMESTAMP | DATETIME	Réglable (0,6 – 6 signifie avec des millisecondes)	8 Octets

Autres

Autres

Type	*Option*	*HSQLDB*	*Range*	*Storage space*
Oui/Non	BOOLEAN	BOOLEAN | BIT		
Champ binaire (fixe)	BINARY	BINARY	Comme Integer	fixe
Champ binaire	VARBINARY	VARBINARY	Comme Integer	variable
Image	LONGVARBINARY	LONGVARBINARY	Comme Integer	variable, destiné aux images plus grandes
OTHER	OTHER	OTHER | OBJECT		

Dans les définitions de table, et lorsque les types de données sont modifiés dans les requêtes utilisant les fonctions « convertir » ou « cast », certains types de données attendent des informations sur le nombre de caractères (a), la précision (g, correspondant au nombre total de caractères) et le nombre de décimales (d). Les types sont CHAR (a), VARCHAR (a), DOUBLE (g), NUMERIC (g, d), DECIMAL (g, d) et TIMESTAMP (g).

TIMESTAMP (g) ne peut avoir que deux valeurs : « 0 » et « 6 ». « 0 » signifie qu'aucune seconde ne sera stockée dans la partie décimale (dixièmes, centièmes…). La précision des horodatages ne peut être donnée que directement à l'aide de commandes SQL. Donc, si vous stockez les horaires d'un type de sport, vous devez définir TIMESTAMP (6) en utilisant **Outils> SQL** à l'avance.

Types de données dans StarBasic

Nombres

Type	*Correspond à HSQLDB*	*Valeur initiale*	*Remarques*	*Exigences de stockage*
Entier	SMALLINT	0	2^{16} = – 32768 bis + 32767	2 Octets
Long	INTEGER	0	2^{32} = – 2147483648 bis + 2147483647	4 Octets
Simple		0.0	Decimal:.	4 Octets
Double	DOUBLE	0.0	Decimal:.	8 Octets
Monétaire	Resembles DECIMAL, NUMERIC	0.0000	4 décimales fixes	8 Octets

Autres

Type	*Corresponds to HSQLDB*	*Initial value*	*Remarks*	*Storage requirements*
Boolean	BOOLEAN	False	1 = oui, tout le reste : non.	1 Octets
Date	TIMESTAMP	00:00:00	Jusqu'à 65536 caractères	8 Octets
String	VARCHAR	Chaine vide		variable
Object	OTHER	Null		variable
Variant		Empty	Peut accepter n'importe quel (autre) type de données	variable

Il existe de grands risques dans la conversion de données, en particulier avec des valeurs numériques. Par exemple, les clés primaires des bases de données sont le plus souvent du type INTEGER. Si ceux-ci sont lus par une macro, la variable dans laquelle ils sont stockés doit être de type Long, car sa taille correspond au type INTEGER dans Base. L'instruction de lecture correspondante est getLong.

Fonctions intégrées et procédures stockées

Les fonctions suivantes sont disponibles dans la HSQLDB intégrée. Malheureusement, une ou deux fonctions ne peuvent être utilisées que lorsque la commande **Exécuter SQL** directement est choisie. Cela empêchera alors ces requêtes d'être mémorisées et modifiées.

Les fonctions qui fonctionnent avec l'interface utilisateur graphique sont marquées [Fonctionne dans l'IGU] (*Interface Graphique Utilisateur*). Les fonctions qui ne fonctionnent que dans les commandes SQL directes sont marquées [SQL direct – ne fonctionne pas dans l'IGU].

Numeriques

Comme nous traitons ici de nombres à virgule flottante, assurez-vous de faire attention aux paramètres des champs dans les requêtes. La plupart du temps, l'affichage des décimales est limité, de sorte que dans certains cas, il peut y avoir des résultats inattendus. Par exemple, la colonne 1 peut afficher 0,00 mais contient en réalité 0,001 et la colonne 2, 1000. Si la colonne 3 est définie pour afficher Colonne 1 * Colonne 2, elle affichera en fait 1.

ABS(d)	Renvoie la valeur absolue d'un nombre. [Fonctionne dans l'IGU]
ACOS(d)	Renvoie l'arc cosinus. [Fonctionne dans l'IGU]
ASIN(d)	Returns the arcsine. [Fonctionne dans l'IGU]
ATAN(d)	Renvoie l'arc tangente. [Fonctionne dans l'IGU]
ATAN2(a, b)	Renvoie l'arc tangente à l'aide de coordonnées, où a est la valeur de l'axe x, b la valeur de l'axe y. [Fonctionne dans l'IGU]
BITAND(a, b)	La forme binaire de a et la forme binaire de b doivent avoir 1 à la même position pour donner 1 dans le résultat. BITAND (3,5) donne 1 ; 0011 ET 0101 = 0001 [Fonctionne dans l'IGU]
BITOR(a, b)	La forme binaire de a ou la forme binaire de b doit avoir 1 à la même position pour donner 1 dans le résultat. BITOR (3,5) donne 7 ; 0011 OU 0101 = 0111 [Fonctionne dans l'IGU]
CEILING(d)	Renvoie le plus petit nombre entier qui n'est pas inférieur à d. [Fonctionne dans l'IGU]
COS(d)	Renvoie le cosinus. [Fonctionne dans l'IGU]
COT(d)	Renvoie la cotangente. [Fonctionne dans l'IGU]
DEGREES(d)	Convertit les radians en degrés. [Fonctionne dans l'IGU]
EXP(d)	Renvoie e^d (e : (2.718…)). [Fonctionne dans l'IGU]
FLOOR(d)	Renvoie le plus grand nombre entier qui n'est pas supérieur à d. [Fonctionne dans l'IGU]

LOG(d)	Renvoie le logarithme naturel en base e. [Fonctionne dans l'IGU]
LOG10(d)	Renvoie le logarithme en base 10. [Fonctionne dans l'IGU]
MOD(a, b)	Renvoie le reste sous forme de nombre entier, dans la division de 2 nombres entiers. MOD (11,3) renvoie 2, car 3 * 3 + 2 = 11 [Fonctionne dans l'IGU]
PI()	Renvoie π (3,1415…). [Fonctionne dans l'IGU]
POWER(a, b)	a^b, POWER(2,3) = 8, car $2^3 = 8$ [Fonctionne dans l'IGU]
RADIANS(d)	Convertit les degrés en radians. [Fonctionne dans l'IGU]
RAND()	Renvoie un nombre aléatoire supérieur ou égal à 0,0 et inférieur à 1,0. [Fonctionne dans l'IGU]
ROUND(a, b)	Arrondit a à b décimales. [Fonctionne dans l'IGU]
ROUNDMAGIC(d)	Résout les problèmes d'arrondi résultant de l'utilisation de nombres à virgule flottante. 3.11-3.1-0.01 n'est pas exactement 0, mais est affiché comme 0 dans l'interface graphique. ROUNDMAGIC en fait une valeur zéro réelle. [Fonctionne dans l'IGU]
SIGN(d)	Renvoie –1 si d est inférieur à 0, 0 si d est égal à 0 et 1 si d est supérieur à 0. [Fonctionne dans l'IGU]
SIN(A)	Renvoie le sinus d'un angle en radians. [Fonctionne dans l'IGU]
SQRT(d)	Renvoie la racine carrée. [Fonctionne dans l'IGU]
TAN(A)	Renvoie la tangente d'un angle en radians. [Fonctionne dans l'IGU]
TRUNCATE(a, b)	Truncates a to b decimal places. TRUNCATE(2.37456,2) = 2.37 [Fonctionne dans l'IGU]

Texte

ASCII(s)	Renvoie le code ASCII de la première lettre de la chaîne. [Fonctionne dans l'IGU]
BIT_LENGTH(str)	Renvoie la longueur de la chaîne de texte str en bits. [Fonctionne dans l'IGU]
CHAR(c)	Returns the letter corresponding to the ASCII code c. [Fonctionne dans l'IGU]
CHAR_LENGTH(str)	Renvoie la longueur de la chaîne str en caractères. [Fonctionne dans l'IGU]
CONCAT(str1, str2)	Concatène str1 et str2. [Fonctionne dans l'IGU]
'str1'||'str2'||'str3' or 'str1'+'str2'+'str3'	Concatenates str1, str2, and str3. A simpler alternative to CONCAT. [Fonctionne dans l'IGU]
DIFFERENCE(s1, s2)	Renvoie la différence de son entre s1 et s2. Seul un nombre entier est émis. 0 signifie qu'ils sonnent de la même manière. Par exemple, « sert » et « serre » donnent 0, « pair » et « faire » donnent 1, « mer » et « mère » donnent 0. [Fonctionne dans l'IGU]
HEXTORAW(s1)	Traduit le code hexadécimal en d'autres caractères. [Fonctionne dans l'IGU]
INSERT(s, start, len, s2)	Renvoie une chaîne de texte, avec une partie du texte remplacée. En commençant à d une longueur l est découpée dans le texte s et remplacée par le texte s2. INSERT (« Consolation », 3, 4, firm) convertit Consolation en Confirmation, où la longueur du texte inséré peut être supérieure à celle du texte supprimé sans causer de problèmes. Donc INSERT (« Bundesbahn », 3, 5, s et B) donne « Bus und Bahn ». [Fonctionne dans l'IGU]
LCASE(s)	Convertit une chaîne en minuscules. [Fonctionne dans l'IGU]
LEFT(s, compte)	Renvoie le nombre de caractères spécifié par compte à partir du début de la chaîne s. [Fonctionne dans l'IGU]
LENGTH(s)	Renvoie la longueur de la chaîne s en caractères. [Fonctionne dans l'IGU]
LOCATE(cherche, s, [début])	Renvoie la première correspondance du terme recherché dans la chaîne s. La correspondance est donnée sous forme de nombre de décalage : (1 = gauche, 0 = introuvable). La définition d'un point de départ dans la chaîne de texte est facultative. [Fonctionne dans l'IGU]

LTRIM(s)	Supprime les espaces de début et les caractères non imprimables au début d'une chaîne de texte. [Fonctionne dans l'IGU]
OCTET_LENGTH(str)	Renvoie la longueur d'une chaîne de texte en octets. Cela correspond à deux fois la longueur de la chaîne. [Fonctionne dans l'IGU]
RAWTOHEX(s1)	Convertit en hexadécimaux, inverse deHEXTORAW(). [Fonctionne dans l'IGU]
REPEAT(s, count)	Répète le nombre de fois de la chaîne de texte. [Fonctionne dans l'IGU]
REPLACE(s, remplace, s2)	Remplace toutes les occurrences de remplace existantes dans la chaîne de texte s par la chaîne s2. [Fonctionne dans l'IGU]
RIGHT(s, count)	Opposé de LEFT ; renvoie le dernier nombre de caractères à la fin d'une chaîne de texte. [Fonctionne dans l'IGU]
RTRIM(s)	Supprime tous les espaces et les caractères non imprimables à la fin d'une chaîne de texte. [Fonctionne dans l'IGU]
SOUNDEX(s)	Renvoie un code à 4 caractères, correspondant au son de s. Correspond à la fonction DIFFÉRENCE(). [Fonctionne dans l'IGU]
SPACE(count)	Renvoie le nombre d'espaces. [Fonctionne dans l'IGU]
SUBSTR(s, start[, len])	Abréviation de SUBSTRING. [Fonctionne dans l'IGU]
SUBSTRING(s, début[, longueur])	Renvoie le texte s à partir de la position de début (1 = gauche). Si longueur est omis, la chaîne entière est renvoyée. [Fonctionne dans l'IGU]
UCASE(s)	Convertit une chaîne en majuscules. [Fonctionne dans l'IGU]
LOWER(s)	Comme LCASE(s). [Fonctionne dans l'IGU]
UPPER(s)	Comme UCASE(s). [Fonctionne dans l'IGU]

Date/Heure

CURDATE()	Renvoie la date actuelle. [Fonctionne dans l'IGU]
CURTIME()	Renvoie l'heure actuelle. [Fonctionne dans l'IGU]
DATEDIFF(string, datetime1, datetime2)	Différence de date entre deux dates – compare les valeurs de date / heure. L'entrée en chaîne détermine les unités dans lesquelles la différence est renvoyée : ms = milliseconde, ss = seconde, mi = minute, hh = heure, jj = jour, mm = mois, aa = année. Les formes longues et courtes peuvent être utilisées pour la chaîne. [Fonctionne dans l'IGU]
DAY(date)	Renvoie le jour du mois (1-31). [Fonctionne dans l'IGU]
DAYNAME(date)	Renvoie le nom anglais du jour. [Fonctionne dans l'IGU]
DAYOFMONTH(date)	Renvoie le jour du mois (1-31). Synonyme de DAY(). [Fonctionne dans l'IGU]
DAYOFWEEK(date)	Renvoie le jour de la semaine sous forme de nombre (1 représente dimanche). [Fonctionne dans l'IGU]
DAYOFYEAR(date)	Renvoie le jour de l'année (1-366). [Fonctionne dans l'IGU]
HOUR(time)	Renvoie l'heure (0-23). [Fonctionne dans l'IGU]
MINUTE(time)	Renvoie les minutes (0-59). [Fonctionne dans l'IGU]
MONTH(date)	Renvoie le mois (1-12). [Fonctionne dans l'IGU]
MONTHNAME(date)	Renvoie le nom anglais du mois. [Fonctionne dans l'IGU]
NOW()	Renvoie la date actuelle et l'heure actuelle ensemble sous forme d'horodatage. Vous pouvez également utiliser CURRENT_TIMESTAMP. [Fonctionne dans l'IGU]
QUARTER(date)	Renvoie le trimestre de l'année (1-4). [Fonctionne dans l'IGU]
SECOND(time)	Renvoie la partie des secondes du temps (0-59). [Fonctionne dans l'IGU]

WEEK(date)	Renvoie la semaine de l'année (1-53). [Fonctionne dans l'IGU]
YEAR(date)	Renvoie la partie année d'une entrée de date. [Fonctionne dans l'IGU]
CURRENT_DATE	Synonyme de CURDATE(), SQL-Standard. [Fonctionne dans l'IGU]
CURRENT_TIME	Synonyme de CURTIME(), SQL-Standard. [Fonctionne dans l'IGU]
CURRENT_TIMESTAMP	Synonyme de NOW(), SQL-Standard. [Fonctionne dans l'IGU]

Connexion à la base de données

À l'exception de IDENTITY(), qui n'a aucune signification dans Base, tout cela peut être effectué à l'aide d'une commande SQL directe.

DATABASE()	Renvoie le nom de la base de données à laquelle appartient cette connexion. [Fonctionne dans l'IGU]
USER()	Renvoie le nom d'utilisateur de cette connexion. [SQL Direct– ne fonctionne pas dans l'IGU]
CURRENT_USER	Fonction standard SQL, synonyme de USER (). [Fonctionne dans l'IGU]
IDENTITY()	Renvoie la dernière valeur d'un champ de valeur automatique, qui a été créé dans la connexion actuelle. Ceci est utilisé dans le codage de macro pour transférer une clé primaire dans une table pour devenir une clé étrangère pour une autre table. [Fonctionne dans l'IGU]

Système

IFNULL(exp, valeur)	Si exp est NULL, la valeur est retournée, sinon exp est retournée. Comme extension, COALESCE() peut être utilisée. Exp et value doivent avoir le même type de données. [Fonctionne dans l'IGU]
CASEWHEN(exp, v1, v2)	Si exp est vrai, v1 est renvoyé, sinon v2. Alternativement, CASE WHEN peut aussi être utilisé. CASE WHEN fonctionne mieux avec l'interface graphique. [Fonctionne dans l'IGU]
CONVERT(terme, type)	Convertit le terme en un autre type de données. [Fonctionne dans l'IGU]
CAST(terme AS type)	Synonyme de CONVERT(). [Fonctionne dans l'IGU]
COALESCE(expr1, expr2, expr3,…)	Si expr1 n'est pas NULL, renvoie expr1, sinon expr2 est vérifié, puis expr3 et ainsi de suite. [Fonctionne dans l'IGU]
NULLIF(v1, v2)	Si v1 est égal à v2, null est renvoyé, sinon la valeur de v1 est renvoyée. Les données doivent être de type comparable. [Fonctionne dans l'IGU]
CASE v1 WHEN v2 THEN v3 [ELSE v4] END	Si v1 est égal à v2, v3 est renvoyé. Sinon, v4 est retourné ou NULL, s'il n'y a pas de condition ELSE. [SQL Direct– ne fonctionne pas dans l'IGU]
CASE WHEN expr1 THEN v1[WHEN expr2 THEN v2] [ELSE v4] END	If expr1 is true, v1 is returned [optionally further conditions can be set]. Otherwise v4 is returned or NULL if there is no ELSE condition. [Fonctionne dans l'IGU]
EXTRACT ({YEAR | MONTH | DAY | HOUR | MINUTE | SECOND} FROM <date ou heure>)	Peut remplacer de nombreuses fonctions de date et d'heure. Renvoie l'année, le mois, le jour, etc. à partir d'une valeur de date ou de date / heure. [Fonctionne dans l'IGU]
POSITION(<string expression> IN <string expression>)	Si la première chaîne est contenue dans la seconde, le décalage de la première chaîne est donné, sinon 0 est renvoyé. [Fonctionne dans l'IGU]
SUBSTRING(<expression chaine> FROM <expression numeriique> [FOR <numeric expression>])	Renvoie une partie d'une chaîne de texte à partir de la position spécifiée dans FROM, éventuellement jusqu'à la longueur indiquée dans FOR. [Fonctionne dans l'IGU]
TRIM([{LEADING | TRAILING | BOTH}] FROM < expression chaine>)	Les caractères spéciaux et les espaces non imprimables sont supprimés. [Fonctionne dans l'IGU]

Caractères de contrôle à utiliser dans les requêtes

Les champs peuvent être liés entre eux dans les requêtes. Deux champs dans

```
SELECT "Prenom", "Nom" FROM "Table"
```

devient un champ unique en utilisant :

```
SELECT "Prenom"||' '||"Nom" FROM "Table"
```

Ici, un espace supplémentaire est inséré. Cela peut être n'importe quel caractère ; tant qu'il est entouré de ", il sera interprété comme du texte. Parfois, cependant, il est nécessaire d'insérer des caractères non imprimables tels que des fin de lignes, par exemple lors de la préparation de rapports. Voici une courte liste de caractères de contrôle. Plus d'informations à ce sujet sont disponibles à l'adresse https://fr.wikipedia.org/wiki/Caract%C3 %A8re_de_contr%C3 %B4le.

CHAR(9)	Tabutation horizontale	Positionne le caractère suivant dans une colonne de type *tab stop* (tabulation).
CHAR(10)	Line feed (saut de ligne)	Dans les lettres de publipostage et le Générateur de Rapports, crée un saut de ligne (Linux, Unix, Mac).
CHAR(13)	Carriage return (retour chariot)	Saut de ligne lorsqu'il est combiné avec le retour chariot dans Windows CHAR (13) || CHAR (10). Peut également être utilisé sous Linux et Mac, d'où la variante universelle.

Note

La combinaison "Ligne suivante – Retour chariot" est un héritage de la machine à écrire manuelle, puis des terminaux télétype et certaines anciennes imprimantes à aiguille, avec lesquels deux commandes étaient nécéssaires pour positionner la tête d'écriture à la ligne suivante puis en début de ligne. A sa création, le système Linux qui ne gérait pas ce type de terminal, n'utilise qu'un seul caractère pour le saut de ligne dans les fichiers texte.

Quelques commandes uno à utiliser avec un bouton

Un bouton peut être lié (par sa propriété Action) à diverses commandes uno. Pour cela, vous devez choisir **Propriétés : Bouton> Action> Ouvrir un document / une page Web**, puis par exemple l'**URL>.uno : RecSearch** pour ouvrir la fonction de recherche. Souvent, vous devrez choisir **Focus sur Clic> Non** si l'action accède directement à un autre contrôle d'une manière qui nécessite qu'il ait le focus, par exemple**.uno: Paste**, qui peut insérer le contenu du presse-papiers dans un contrôle texte.

La liste suivante ne contient que quelques commandes. Toutes les commandes de la barre d'outils de navigation sont déjà utilisables dans le bouton, mais elles peuvent également être créées à l'aide des commandes uno. De nombreuses commandes peuvent être découvertes à l'aide de l'enregistreur de macros, qui utilise souvent un "dispatcher" pour y accéder.

Uno-Command	***Used for…***

.uno:RecSearch	Ouvre la fonction de recherche dans un formulaire.
.uno:Paste	Coller à partir du presse-papiers. Fonctionne uniquement pour **Focus** sur **clic> Non.**
.uno:Copy	Copie le contenu sélectionné dans le presse-papiers. Fonctionne uniquement pour **Focus sur clic> Non.**
.uno:Print	Ouvre la boîte de dialogue d'impression du formulaire.
.uno:PrintDefault	Imprime avec l'imprimante par défaut sans afficher de boîte de dialogue.

Tables d'informations pour HSQLDB

Dans une base de données, les informations sur toutes les propriétés de table et leurs connexions entre elles sont stockées dans la zone *INFORMATION_SCHEMA*. Ces informations permettent de créer des macros BASE qui nécessitent très peu d'arguments pour leurs procédures. Une application est donnée dans l'exemple de base de données du module Maintenance – la procédure Purge_Table pour le contrôle des dialogues.

Dans une requête, des informations individuelles et tous les champs qui en font partie peuvent être fournis de la manière suivante :

```
SELECT * FROM "INFORMATION_SCHEMA"."SYSTEM_ALIASES"
```

Contrairement à une table normale, il est ici nécessaire d'utiliser *INFORMATION_SCHEMA* comme préfixe du nom approprié dans la liste suivante :

```
SYSTEM_ALIASES
SYSTEM_ALLTYPEINFO
SYSTEM_BESTROWIDENTIFIER
SYSTEM_CACHEINFO
SYSTEM_CATALOGS
SYSTEM_CHECK_COLUMN_USAGE
SYSTEM_CHECK_CONSTRAINTS
SYSTEM_CHECK_ROUTINE_USAGE
SYSTEM_CHECK_TABLE_USAGE
SYSTEM_CLASSPRIVILEGES
SYSTEM_COLUMNPRIVILEGES
SYSTEM_COLUMNS
SYSTEM_CROSSREFERENCE
SYSTEM_INDEXINFO
SYSTEM_PRIMARYKEYS
SYSTEM_PROCEDURECOLUMNS
SYSTEM_PROCEDURES
SYSTEM_PROPERTIES
SYSTEM_SCHEMAS
SYSTEM_SEQUENCES
SYSTEM_SESSIONINFO
SYSTEM_SESSIONS
SYSTEM_SUPERTABLES
SYSTEM_SUPERTYPES
SYSTEM_TABLEPRIVILEGES
SYSTEM_TABLES
SYSTEM_TABLETYPES
SYSTEM_TABLE_CONSTRAINTS
SYSTEM_TEXTTABLES
SYSTEM_TRIGGERCOLUMNS
```

```
SYSTEM_TRIGGERS
SYSTEM_TYPEINFO
SYSTEM_UDTATTRIBUTES
SYSTEM_UDTS
SYSTEM_USAGE_PRIVILEGES
SYSTEM_USERS
SYSTEM_VERSIONCOLUMNS
SYSTEM_VIEWS
SYSTEM_VIEW_COLUMN_USAGE
SYSTEM_VIEW_ROUTINE_USAGE
SYSTEM_VIEW_TABLE_USAGE
```

La requête suivante donne un aperçu complet de toutes les tables de la base de données avec les types de champs, les clés primaires et les clés étrangères :

```
SELECT
"A"."TABLE_NAME",
"A"."COLUMN_NAME",
"A"."TYPE_NAME",
"A"."NULLABLE",
"B"."KEY_SEQ" AS "PRIMARYKEY",
"C"."PKTABLE_NAME" || '.' || "C"."PKCOLUMN_NAME" AS "FOREIGNKEY FOR"
FROM "INFORMATION_SCHEMA"."SYSTEM_COLUMNS" AS "A"
      LEFT JOIN "INFORMATION_SCHEMA"."SYSTEM_PRIMARYKEYS" AS "B"
      ON ("B"."TABLE_NAME" = "A"."TABLE_NAME" AND "B"."COLUMN_NAME" =
      "A"."COLUMN_NAME")
      LEFT JOIN "INFORMATION_SCHEMA"."SYSTEM_CROSSREFERENCE" AS "C"
      ON ("C"."FKTABLE_NAME" = "A"."TABLE_NAME" AND "C"."FKCOLUMN_NAME"
      = "A"."COLUMN_NAME")
WHERE "A"."TABLE_SCHEM" = 'PUBLIC'
```

Réparation de base de données pour les fichiers *.odb

Les sauvegardes régulières des données doivent être une pratique courante lors de l'utilisation d'un PC. Les copies de sauvegarde sont le moyen le plus simple de revenir à un état actuel, même à mi-chemin, pour vos données. Cependant, dans la pratique, cela fait souvent défaut.

Conseil

Pour effectuer des sauvegardes régulières de vos documesnt dans des états successifs, vous pouvez utiliser l'extension de Sauvegarde avec incrémentation automatique.

Les formulaires, requêtes et rapports peuvent toujours être copiés à l'aide du presse-papiers dans une nouvelle base de données, à condition qu'une version précédente de la base de données ait été enregistrée. Mais si, pour une raison quelconque, la base de données actuelle ne peut plus être ouverte, le problème principal devient l'accès aux données.

En cas de panne soudaine du PC, il peut arriver que des bases de données ouvertes (bases de données internes HSQLDB ou Firebird) ne puissent plus être ouvertes dans LibreOffice. Au lieu de cela, lorsque vous essayez d'ouvrir la base de données, il vous est demandé un filtre correspondant au format.

Le problème ici est qu'une partie des données d'une base de données ouverte est contenue dans la mémoire de travail et n'est copiée que temporairement dans le stockage intermédiaire. Ce n'est que lorsque le fichier est fermé que toute la base de données est réécrite dans le fichier et reconditionnée.

Récupération du fichier d'archive de la base de données

Pour avoir à nouveau accès à vos données, la procédure suivante peut vous être utile :

1. Créez une copie de votre base de données pour les étapes suivantes.
2. Essayez d'ouvrir la copie avec un programme d'archivage. Dans le cas des fichiers *.odb, nous avons affaire à un format compressé, une archive Zip. Si le fichier ne peut pas être ouvert directement, essayez de le renommer de *.odb en *.zip. Si cela ne l'ouvre pas, votre base de données est en cours d'enregistrement (voir s'il existe un fichier *.lck au même nom dans le dossier de la base).
3. Les dossiers visibles sur la Figure 120 seront toujours visibles après l'ouverture d'un fichier de base de données dans un programme d'archivage.
4. Le fichier de base de données doit être décompressé. Les informations les plus importantes, en ce qui concerne les données, se trouvent dans la base de données des sous-dossiers dans les fichiers **data** et **script**.

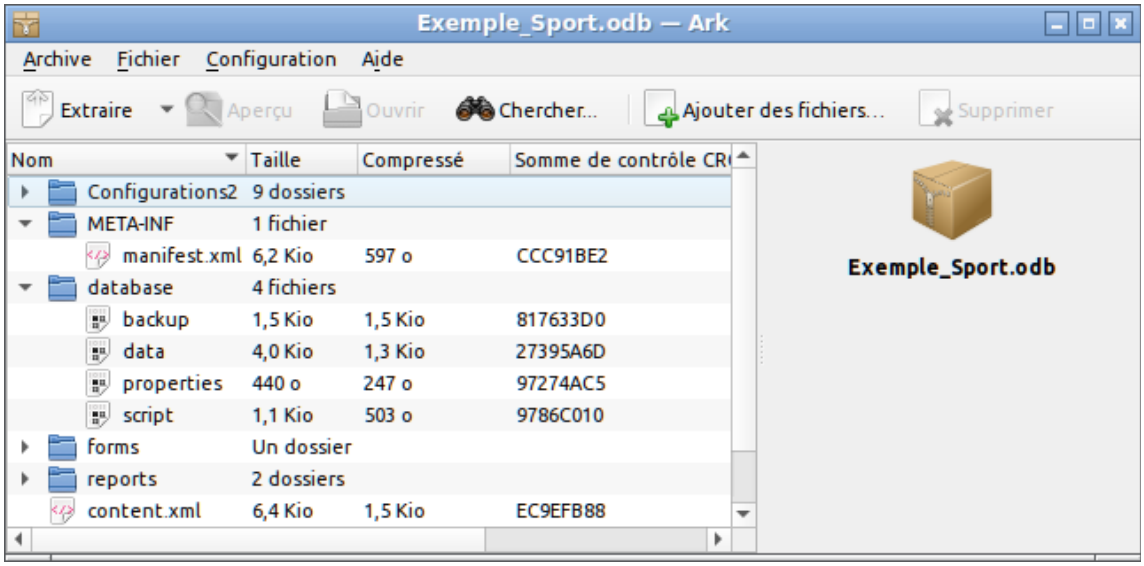


Figure 120: Vue de la structure de l'archive de base de données

5. Il peut être nécessaire de regarder le fichier de script et de le tester pour les contradictions. Cette étape peut cependant être laissée pour la phase de test. Le fichier script contient avant tout la description de la structure des tables.

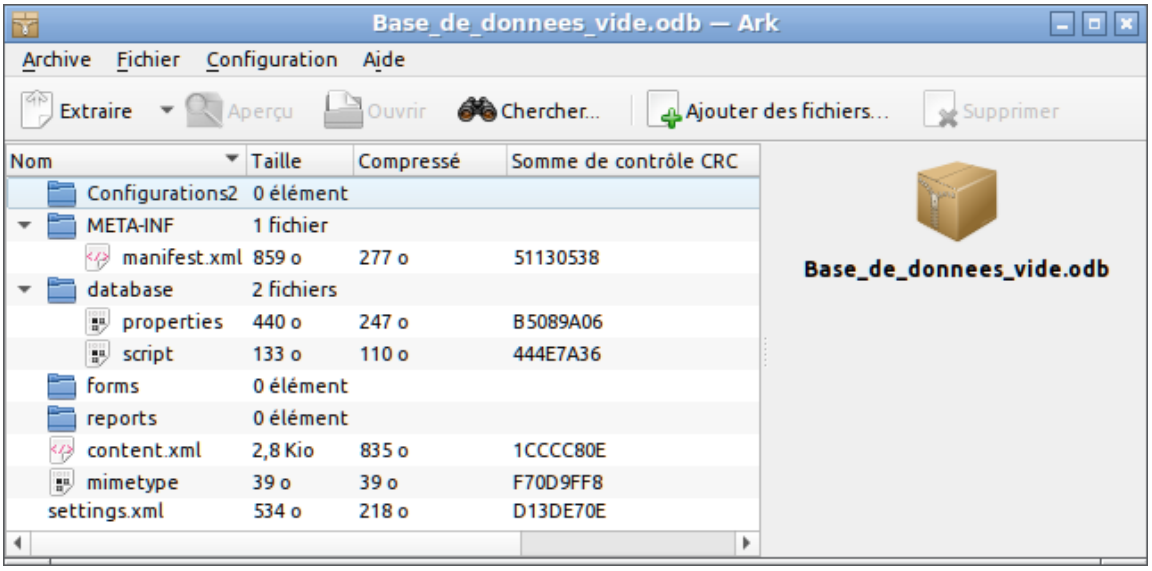


Figure 121: Vue de l'archive de la base vide

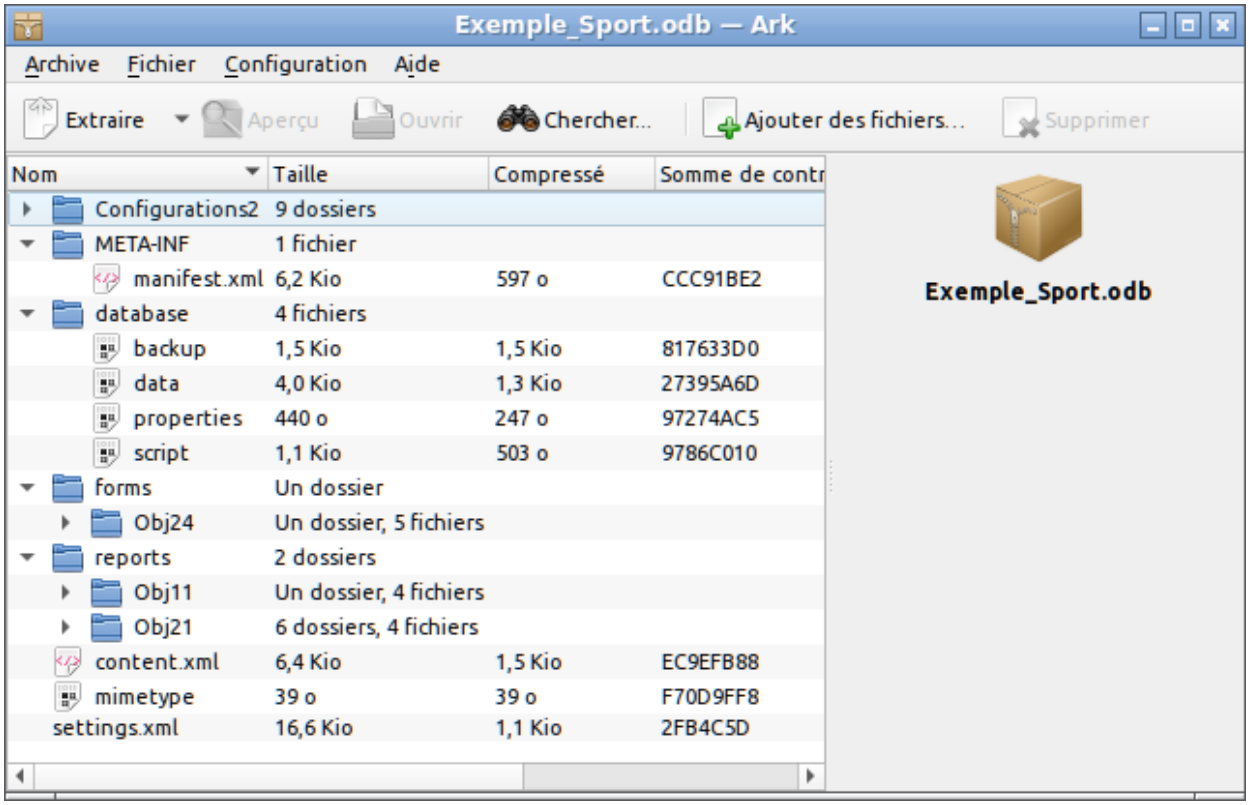
6. Créez un nouveau fichier de base de données vide et ouvrez ce fichier avec le programme d'archivage.
7. Remplacez les fichiers **data** et **script** dans le nouveau fichier de base de données par les fichiers décompressés à l'étape 4.
8. Fermez le programme d'archivage. S'il était nécessaire de renommer le fichier en *.zip avant de l'ouvrir dans le programme d'archivage (cela dépend de votre système d'exploitation), renommez-le maintenant à nouveau en *.odb.
9. Ouvrez le fichier de base de données dans LibreOffice. Vous devriez pouvoir à nouveau accéder à vos tables.
10. La quantité de vos requêtes, formulaires et rapports pouvant être récupérés de la même manière doit faire l'objet de tests supplémentaires.

Voir aussi : http://forum.openoffice.org/en/forum/viewtopic.php?f=83&t=17125

Informations supplémentaires sur les fichiers d'archive de base de données

Dans la pratique, un fichier d'archive de base de données contient non seulement le dossier de base de la base de données et le dossier META-INF qui est spécifié pour le format OpenDocument, mais également des dossiers supplémentaires pour stocker les formulaires et les rapports. Une description de la structure de base du format OpenDocument peut être trouvée sur https://en.wikipedia.org/wiki/OpenDocument_technical_specification.

La vue suivante montre une base de données contenant des tables, des formulaires et des états. Il n'est pas évident que la base de données contienne également une requête. Les requêtes ne sont pas stockées dans des dossiers séparés mais dans le fichier **content.xml**. Les informations nécessaires pour exécuter une requête sont un simple morceau de code SQL.



Fichier de base de données contenant des informations stockées.

Voici un aperçu de l'un des fichiers d'archive de la base de données.

mimetype

```
application/vnd.oasis.opendocument.base
```

Ce petit fichier texte ne contient que l'avertissement que ce fichier d'archive est un fichier de base de données au format OpenDocument.

content.xml pour une base de données sans contenu

```
<?xml version="1.0" encoding="UTF-8"?>
<office: document-content
xmlns: office="urn: oasis: names: tc: opendocument: xmlns: office:1.0" xmlns:
style="urn: oasis: names: tc: opendocument: xmlns: style:1.0" xmlns: text="urn:
oasis: names: tc: opendocument: xmlns: text:1.0" xmlns: table="urn: oasis: names: tc:
opendocument: xmlns: table:1.0" xmlns: draw="urn: oasis: names: tc: opendocument:
xmlns: drawing:1.0" xmlns: fo="urn: oasis: names: tc: opendocument: xmlns: xsl-fo-
compatible:1.0" xmlns: xlink="http://www.w3.org/1999/xlink"
xmlns: dc="http://purl.org/dc/elements/1.1/" xmlns: meta="urn: oasis: names: tc:
opendocument: xmlns: meta:1.0" xmlns: number="urn: oasis: names: tc: opendocument:
xmlns: datastyle:1.0" xmlns: svg="urn: oasis: names: tc: opendocument: xmlns: svg-
compatible:1.0" xmlns: chart="urn: oasis: names: tc: opendocument: xmlns: chart:1.0"
xmlns: dr3d="urn: oasis: names: tc: opendocument: xmlns: dr3d:1.0" xmlns:
math="http://www.w3.org/1998/Math/MathML"
xmlns: form="urn: oasis: names: tc: opendocument: xmlns: form:1.0" xmlns:
script="urn: oasis: names: tc: opendocument: xmlns: script:1.0"
xmlns: ooo="http://openoffice.org/2004/office"
xmlns: ooow="http://openoffice.org/2004/writer"
xmlns: oooc="http://openoffice.org/2004/calc"
xmlns: dom="http://www.w3.org/2001/xml-events"
xmlns: db="urn: oasis: names: tc: opendocument: xmlns: database:1.0" xmlns:
```

```
xforms="http://www.w3.org/2002/xforms"
xmlns: xsd="http://www.w3.org/2001/XMLSchema"
xmlns: xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns: rpt="http://openoffice.org/2005/report"
xmlns: of="urn: oasis: names: tc: opendocument: xmlns: of:1.2" xmlns:
xhtml="http://www.w3.org/1999/xhtml"
xmlns: grddl="http://www.w3.org/2003/g/data-view#"
xmlns: tableooo="http://openoffice.org/2009/table"
xmlns: drawooo="http://openoffice.org/2010/draw"
xmlns: calcext="urn: org: documentfoundation: names: experimental: calc: xmlns:
calcext:1.0" xmlns: field="urn: openoffice: names: experimental: ooo-ms-interop:
xmlns: field:1.0" xmlns: formx="urn: openoffice: names: experimental: ooxml-odf-
interop: xmlns: form:1.0" xmlns: css3t="http://www.w3.org/TR/css3-text/"
office: version="1.2">
      <office: scripts/>
      <office: font-face-decls/>
      <office: automatic-styles/>
      <office: body>
         <office: database>
            <db: data-source>
               <db: connection-data>
                  <db: connection-resource xlink: href="sdbc: embedded:
hsqldb"/>
                  <db: login db: is-password-required="false"/>
               </db: connection-data>
               <db: driver-settings
                  db: system-driver-settings=""
                  db: base-dn=""
                  db: parameter-name-substitution="false"/>
               <db: application-connection-settings
                  db: is-table-name-length-limited="false"
                  db: append-table-alias-name="false"
                  db: max-row-count="100">
                  <db: table-filter>
                     <db: table-include-filter>
                        <db: table-filter-pattern>%</db: table-filter-
pattern>
                     </db: table-include-filter>
                  </db: table-filter>
               </db: application-connection-settings>
            </db: data-source>
         </office: database>
      </office: body>
</office: document-content>
```

Il commence par la version XML et le jeu de caractères utilisé. Tout ce qui suit est en fait une seule ligne continue. La vue préparée ci-dessus devrait rendre les choses plus claires. Les éléments qui vont ensemble sont encadrés par des balises.

Les définitions initiales commençant par xmlns : (espace de noms XML) donnent les espaces de noms accessibles depuis l'intérieur du fichier. Ensuite, des termes un peu plus concrets sont envisagés. Ici, il devient clair que nous avons affaire à une base de données interne HSQLDB, et qu'un mot de passe n'est pas requis pour l'accès.

content.xml pour une base de données avec des contenus
Le contenu suivant n'est qu'un extrait du fichier content.xml, pour clarifier sa structure.

```
<office: scripts/>
<office: font-face-decls>
      <style: font-face style: name="F" svg: font-family=""/>
</office: font-face-decls>
<office: automatic-styles>
      <style: style
         style: name="co1"
         style: family="table-column"
         style: data-style-name="N0"/>
      <style: style
         style: name="co2"
         style: family="table-column"
         style: data-style-name="N107"/>
      <style: style style: name="ce1" style: family="table-cell">
         <style: paragraph-properties fo: text-align="start"/>
      </style: style>
      <number: number-style style: name="N0" number: language="de" number:
country="DE">
         <number: number number: min-integer-digits="1"/>
      </number: number-style>
      <number: currency-style
         style: name="N107P0"
         style: volatile="true"
         number: language="de"
         number: country="DE">
         <number: number
            number: decimal-places="2"
            number: min-integer-digits="1"
            number: grouping="true"/>
         <number: text> </number: text>
         <number: currency-symbol
            number: language="de"
            number: country="DE">€
         </number: currency-symbol>
      </number: currency-style>
```

Ici, un champ est défini comme un champ de devise. Le nombre de décimales est indiqué, la séparation entre les nombres et le symbole monétaire et le symbole monétaire lui-même.

```
      <number: currency-style
         style: name="N107"
         number: language="de"
         number: country="DE">
         <style: text-properties fo: color="#ff0000"/>
         <number: text>-</number: text>
         <number: number
            number: decimal-places="2"
            number: min-integer-digits="1"
            number: grouping="true"/>
         <number: text> </number: text>
         <number: currency-symbol
            number: language="de"
```

```
            number: country="DE">€
         </number: currency-symbol>
         <style: map style: condition="value()&gt;=0" style: apply-style-
name="N107P0"/>
      </number: currency-style>
```

Le deuxième extrait indique que jusqu'à une valeur particulière, la devise doit apparaître en rouge (« ff0000 »).

```
</office: automatic-styles>
<office: body>
      <office: database>
         <db: data-source>
```

Cette entrée du fichier content.xml ci-dessus, avec toutes ses sous-entrées, correspond à un fichier d'archive de base de données vide.

```
         </db: data-source>
         <db: forms>
            <db: component
               db: name="Receipts"
               xlink: href="forms/Obj12"
               db: as-template="false"/>
         </db: forms>
```

Le fichier d'archive de la base de données contient une sous-section dans laquelle les détails d'un formulaire sont stockés. Le formulaire est désigné dans l'interface utilisateur sous le nom de *Receipts*.

```
         <db: reports>
            <db: component
               db : name="Receipts"
               xlink: href="reports/Obj12"
               db: as-template="false"/>
         </db: reports>
```

Le fichier d'archive de la base de données contient également une sous-section dans laquelle les détails d'un rapport sont stockés. Le rapport est également désigné dans l'interface utilisateur sous le nom de *Receipts*.

```
         <db: queries>
            <db: query
               db: name="Sales_calc"
               db: command="SELECT &quot; a&quot;.*, (SELECT &quot; Price&quot; *
                  &quot; a&quot;.&quot; Total&quot; FROM &quot; Stock&quot; WHERE
                  &quot; ID&quot; = &quot; a&quot;.&quot; Stock_ID&quot;) AS
                  &quot; Total*Price&quot; FROM &quot; Sales&quot; AS &quot;
a&quot;"/>
         </db: queries>
```

Toutes les requêtes sont stockées directement dans content.xml. " représente les guillemets doubles. La requête ci-dessus dans cet exemple est en fait assez compliquée avec de nombreuses sous-requêtes corrélées. Il est reproduit ici sous une forme abrégée.

```
         <db: table-representations>
            <db: table-representation db: name="Receipts"/>
            <db: table-representation db: name="Sales"/>
            <db: table-representation db: name="Stock">
```

```
            <db: columns>
               <db: column
                  db: name="ID"
                  db: style-name="co1"
                  db: default-cell-style-name="ce1"/>
               <db: column
                  db: name="MWSt"
                  db: style-name="co1"
                  db: gefault-cell-style-name="ce1"/>
               <db: column
                  db: name="Price"
                  db: style-name="co2"
                  db: default-cell-style-name="ce1"/>
               <db: column
                  db: name="Stock"
                  db: style-name="co1"
                  db: default-cell-style-name="ce1"/>
            </db: columns>
         </db: table-representation>
      </db: table-representations>
```

Cela montre comment les différents tableaux doivent être affichés. Ici, les propriétés d'affichage de colonnes particulières sont stockées : dans cet exemple, les paramètres de la table Stock avec ses champs – ID, MWSt, etc. – sont stockés. Apparemment, quelque chose a été directement entré ici, modifiant un peu les colonnes du tableau.

```
      </office: database>
</office: body>
```

Fondamentalement, content.xml stocke directement le contenu des requêtes et des informations sur l'apparence visuelle des tables. En outre, il existe une définition de la connexion à la base de données. Viennent enfin les informations sur les formulaires et les rapports.

settings.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<office: document-settings
xmlns: office="urn: oasis: names: tc: opendocument: xmlns: office:1.0" xmlns:
table="urn: oasis: names: tc: opendocument: xmlns: table:1.0" xmlns:
xlink="http://www.w3.org/1999/xlink" xmlns: number="urn: oasis: names: tc:
opendocument: xmlns: datastyle:1.0" xmlns: svg="http://www.w3.org/2000/svg" xmlns:
config="urn: oasis: names: tc: opendocument: xmlns: config:1.0"
xmlns: ooo="http://openoffice.org/2004/office" xmlns: db="urn: oasis: names: tc:
opendocument: xmlns: database:1.0"
office: version="1.2"/>
```

Pour une base de données sans autre contenu, seules les définitions de base sont stockées ici. Avec le contenu, divers paramètres sont également enregistrés. Après le début de la définition ci-dessus, les paramètres suivants de la base de données d'exemple sont stockés.

```
<office: settings>
      <config: config-item-set config: name="ooo: view-settings">
         <config: config-item-set config: name="Queries">
            <config: config-item-set config: name="Calculate_sales">
               <config: config-item-set config: name="Tables">
                  <config: config-item-set config: name="Table1">
                     <config: config-item config: name="WindowName"
```

```
                  config: type="string">Verkauf</config: config-item>
               <config: config-item config: name="WindowLeft"
                  config: type="int">153</config: config-item>
               <config: config-item config: name="ShowAll"
                  config: type="boolean">true</config: config-item>
               <config: config-item config: name="WindowTop"
                  config: type="int">17</config: config-item>
               <config: config-item config: name="WindowWidth"
                  config: type="int">120</config: config-item>
               <config: config-item config: name="WindowHeight"
                  config: type="int">120</config: config-item>
               <config: config-item config: name="ComposedName"
                  config: type="string">Verkauf</config: config-item>
               <config: config-item config: name="TableName"
                  config: type="string">Verkauf</config: config-item>
            </config: config-item-set>
         </config: config-item-set>
         <config: config-item config: name="SplitterPosition"
            config: type="int">105</config: config-item>
         <config: config-item config: name="VisibleRows"
            config: type="int">1024</config: config-item>
      </config: config-item-set>
   </config: config-item-set>
</config: config-item-set>
<config: config-item-set config: name="ooo: configuration-settings">
   <config: config-item-set config: name="layout-settings">
      <config: config-item-set config: name="Tables">
         <config: config-item-set config: name="Table1">
            <config: config-item config: name="WindowName"
               config: type="string">Verkauf</config: config-item>
            <config: config-item config: name="WindowLeft"
               config: type="int">186</config: config-item>
            <config: config-item config: name="ShowAll"
               config: type="boolean">false</config: config-item>
            <config: config-item config: name="WindowTop"
               config: type="int">17</config: config-item>
            <config: config-item config: name="WindowWidth"
               config: type="int">120</config: config-item>
            <config: config-item config: name="WindowHeight"
               config: type="int">120</config: config-item>
            <config: config-item config: name="ComposedName"
               config: type="string">Verkauf</config: config-item>
            <config: config-item config: name="TableName"
               config: type="string">Sales</config: config-item>
         </config: config-item-set>
         <config: config-item-set config: name="Table2">
            … (identical config: type-Points as "Table1"
            <config: config-item config: name="TableName"
               config: type="string">Ware</config: config-item>
         </config: config-item-set>
         <config: config-item-set config: name="Table3">
            … (identical config: type-Points as "Table1"
            <config: config-item config: name="TableName"
```

```
                    config: type="string">Receipts</config: config-item>
                </config: config-item-set>
            </config: config-item-set>
        </config: config-item-set>
    </config: config-item-set>
</office: settings>
```

L'ensemble de la vue d'ensemble concerne différentes vues des fenêtres pour la requête Calculate_sales et les tables Sales, Stock et Receipts. Les deux derniers sont présentés ici sous une forme abrégée. Si ces paramètres étaient absents dans un fichier *.odb défectueux, cela n'aurait pas d'importance. Ils seraient recréés lors de la prochaine ouverture de la fenêtre correspondante.

META-INF/manifest.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest: manifest     xmlns: manifest="urn: oasis: names: tc: opendocument:
xmlns: manifest:1.0">
    <manifest: file-entry
        manifest: full-path="/"
        manifest: media-type="application/vnd.oasis.opendocument.base"/>
    <manifest: file-entry
        manifest: full-path="database/script"
        manifest: media-type=""/>
    <manifest: file-entry
        manifest: full-path="database/properties"
        manifest: media-type=""/>
    <manifest: file-entry
        manifest: full-path="settings.xml"
        manifest: media-type="text/xml"/>
    <manifest: file-entry
        manifest: full-path="content.xml"
        manifest: media-type="text/xml"/>
</manifest: manifest>
```

Ce fichier dans le dossier META-INF donne le dossier de contenu pour toute l'archive de la base de données. Comme ce fichier traite d'une base de données vide, il n'y a que cinq entrées de fichier. Une archive de base de données qui contient des formulaires et des rapports aura un fichier META-INF beaucoup plus compliqué.

database/properties

```
#HSQL Database Engine 1.8.0.10
#Sun Jul 14 18:02:08 CEST 2013
hsqldb.script_format=0
runtime.gc_interval=0
sql.enforce_strict_size=true
hsqldb.cache_size_scale=8
readonly=false
hsqldb.nio_data_file=false
hsqldb.cache_scale=13
version=1.8.0
hsqldb.default_table_type=cached
hsqldb.cache_file_scale=1
hsqldb.lock_file=true
hsqldb.log_size=10
modified=no
hsqldb.cache_version=1.7.0
hsqldb.original_version=1.8.0
```

```
hsqldb.compatible_version=1.8.0
```

Le fichier de propriétés contient les paramètres de base de la base de données interne HSQLDB.

database/script

```
SET DATABASE COLLATION "French"
CREATE SCHEMA PUBLIC AUTHORIZATION DBA
CREATE USER SA PASSWORD ""
GRANT DBA TO SA
SET WRITE_DELAY 60
```

Le fichier de script contient les paramètres par défaut pour la connexion à la base de données, le paramètre de langue, etc. L'utilisateur SA, qui sera décrit plus loin, apparaît ici.

Dans une base de données avec contenu, ce fichier contient les définitions de table de base.

```
SET DATABASE COLLATION "French"
CREATE SCHEMA PUBLIC AUTHORIZATION DBA
```

Les tables sont définies avant que l'utilisateur de la base de données ne soit défini. Les tables sont d'abord créées dans le cache avec leurs champs.

```
CREATE CACHED TABLE "Stock"
      ("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL
      PRIMARY KEY,"Stock" VARCHAR(50),"Price" DECIMAL(8,2),"MWSt" TINYINT)
CREATE CACHED TABLE "Sales"
      ("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL
      PRIMARY KEY,"Total" TINYINT,"Stock_ID" INTEGER,"Receipt_ID" INTEGER,
      CONSTRAINT SYS_FK_59 FOREIGN KEY("Stock_ID") REFERENCES "Stock"("ID"))
CREATE CACHED TABLE "Receipts"
      ("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL
      PRIMARY KEY,"Date" DATE)
```

Des modifications sont ensuite apportées à la table pour garantir la cohérence des relations (RÉFÉRENCES).

```
ALTER TABLE "Sales" ADD CONSTRAINT SYS_FK_76 FOREIGN KEY("Receipt_ID")
      REFERENCES "Receipts"("ID")
SET TABLE "Stock" INDEX'608 20'
SET TABLE "Sales" INDEX'1872 1656 1872 12'
SET TABLE "Receipts" INDEX'2232 1'
```

Après avoir défini la position de l'index dans le fichier de données (il n'apparaît ici que dans le fichier de script mais n'est jamais réellement entré directement dans SQL), les champs incrémentés automatiquement dans les tables (AutoValues) sont configurés de manière à fournir le prochain valeur à l'entrée d'un nouvel enregistrement. Supposons que la dernière valeur entrée dans le champ ID de la table Stock soit 19. L'auto-incrémentation commence alors à 20.

```
ALTER TABLE "Stock" ALTER COLUMN "ID" RESTART WITH 20
ALTER TABLE "Sales" ALTER COLUMN "ID" RESTART WITH 12
ALTER TABLE "Receipts" ALTER COLUMN "ID" RESTART WITH 1
CREATE USER SA PASSWORD ""
GRANT DBA TO SA
SET WRITE_DELAY 60
```

Gestion de la base de données interne Firebird

La base de données interne Firebird n'est pour le moment disponible qu'en tant que fonction expérimentale. Pour créer une telle base de données ou pour en modifier une qui a été créée, vous devez sélectionner **Outils> Options> LibreOffice> Avancé> Fonctionnalités optionnelles> Activer les fonctionnalités expérimentales (peut-être instables)**. Ce chemin illustre bien qu'une telle base de données n'est pas adaptée à un usage quotidien.

Le lien suivant permet de signaler et de traiter des bogues importants dans la base de données interne Firebird avec l'équipe LibreOffice : Reporting bugs for Firebird in Base.

Les utilisateurs remarqueront les différences suivantes par rapport à HSQLDB :

1. Si un champ est de type Integer puis déclaré comme clé primaire, il semble possible de lui donner une valeur auto-incrémentée. Cependant, lors de l'enregistrement, ce paramètre disparaît sans préavis.
2. Lorsque de nouveaux enregistrements sont saisis, ils ne sont pas automatiquement enregistrés dans la base de données. Le bouton Enregistrer doit être utilisé pour chaque entrée. Dans HSQLDB intégré, la sauvegarde explicite des enregistrements n'est pas nécessaire.
3. Les alias sont complètement ignorés dans les requêtes. Un alias peut être créé mais il n'apparaîtra pas dans l'en-tête de table de la requête.
4. Il n'est pas possible de créer des conditions, bien que les bases de données externes Firebird les prennent en charge.
5. Les types de données décimal et numérique sont actuellement défectueux. Ce sont les seuls types qui garantissent des valeurs précises, en particulier lorsqu'il y a des décimales. Ce sont donc les champs préférés pour les valeurs monétaires. À l'heure actuelle, seules les valeurs comportant au plus une décimale peuvent être saisies.

Rendre AutoChamp disponible

Le code suivant, entré en utilisant **Outils> SQL**, peut aider à résoudre le problème des valeurs automatiques non fournies.

```
CREATE TABLE "Table1" ("ID" INTEGER NOT NULL PRIMARY KEY, "Nom" VARCHAR(20)
NOT NULL) ;
CREATE GENERATOR GEN_T1_ID ;
SET GENERATOR GEN_T1_ID TO 0 ;
```

Après cela, la fenêtre de saisie SQL doit être fermée et **Affichage> Actualiser les tables** sélectionné. Le déclencheur suivant ne peut être créé que lorsque la table apparaît, et dans certains cas seulement après une tentative (infructueuse) de création d'une entrée.

```
CREATE TRIGGER T1_BI FOR "Table1" ACTIVE BEFORE INSERT POSITION 0 AS
BEGIN
IF (NEW. ID IS NULL) THEN NEW. ID = GEN_ID(GEN_T1_ID, 1) ;
END ;
```

Même après cela, de nombreuses entrées dans Noms peuvent être effectuées dans la table sans créer d'entrée dans ID. Le champ ID indique simplement 0. Les valeurs affectées réelles ne sont affichées que lorsque vous appuyez sur **Actualiser**. Le déclencheur fournit des valeurs commençant par 1.

LibreOffice

Guide Base

Appendice B Comparaison de HSQLDB et Firebird

Types de données et fonctions

Types de données et fonctions dans HSQLDB et Firebird

Les tableaux de cette annexe sont tirés des manuels de HSQLDB et Firebird.

- http://hsqldb.org/doc/guide/
- https://www.firebirdsql.org/en/documentation/

Les informations relatives à HSQLDB interne sont les mêmes que dans l'annexe A de ce manuel.

La base de données interne supplémentaire Firebird est classée expérimentale.

Les tableaux fournissent d'abord une comparaison des fonctions, en particulier les fonctions qui sont populaires dans les forums, telles que :

- Ajouter un certain nombre de jours à une date (DATEADD)
- Valeurs de plusieurs lignes de données regroupées dans une seule ligne de données (LIST)

ne sont actuellement disponibles que dans la base de données externe Firebird, mais pas dans la version interne.

Fonctions intégrées et procédures stockées

Les fonctions suivantes sont disponibles dans les bases de données intégrées. Malheureusement, une ou deux fonctions ne peuvent être utilisées que lorsque la commande **Exécuter directement l'instruction SQL** est choisie. Dans ce mode, les requêtes ne peuvent pas être modifiées.

Les fonctions qui fonctionnent avec l'interface utilisateur graphique sont marquées [Fonctionne dans l'IGU] (Interface Graphique Utilisateur). Les fonctions qui ne fonctionnent que dans les commandes SQL directes sont marquées [Ne fonctionne pas dans l'IGU].

Numériques

Comme nous traitons ici de nombres à virgule flottante, assurez-vous de faire attention aux paramètres des champs dans les requêtes. La plupart du temps, l'affichage des décimales est restreint, ce qui peut entraîner un comportement inattendu dans certains cas. Par exemple, la colonne 1 peut afficher 0,00 mais contient en réalité 0,001 et la colonne 2, 1000. Si la colonne 3 est définie pour afficher Colonne 1 * Colonne 2, elle affichera en fait 1.

HSQLDB		***Firebird***	
`ABS(d)`	Renvoie la valeur absolue d'un nombre. [Fonctionne dans l'IGU]	`ABS(d)`	
`ACOS(d)`	Renvoie l'arc cosinus. [Fonctionne dans l'IGU]	`ACOS(d)`	
`ASIN(d)`	Renvoie l'arc sinus. [Fonctionne dans l'IGU]	`ASIN(d)`	
`ATAN(d)`	Renvoie l'arc tangente. [Fonctionne dans l'IGU]	`ATAN(d)`	
`ATAN2(a, b)`	Renvoie l'arc tangente à l'aide de coordonnées, où a est la valeur de l'axe x, b la valeur de l'axe y. [Fonctionne dans l'IGU]	`ATAN2(x, y)`	
`BITAND(a, b)`	La forme binaire de a et la forme binaire de b doivent avoir 1 à la même position pour donner 1 dans le résultat. BITAND (3,5) donne 1 ; 0011 ET 0101 = 0001 [Fonctionne dans l'IGU]	`BIN_AND(x, y [, z…])`	
`BITOR(a, b)`	La forme binaire de a ou la forme binaire de b doit avoir 1 à la même position pour donner 1 dans le résultat. BITOR (3,5) donne 7 ; 0011 OU 0101 = 0111 [Fonctionne dans l'IGU]	`BIN_OR(x, y [, z…])`	

		`BIN_SHL(n, exp)`	n · 2 exp [Fonctionne dans l'IGU]
		`BIN_SHR(n, exp)`	n / 2 exp Le résultat est affiché sous la forme d'un entier arrondi. [Fonctionne dans l'IGU]
		`BIN_XOR(x, y [, z…])`	Soit la notation binaire de « a » ou la notation binaire de « b ». Doit avoir un « 1 » dans la même position pour donner « 1 » dans le résultat. BIN_XOR(3,5) returns 6 ; 0011 X OR 0101 = 011 0 [Fonctionne dans l'IGU]
`CEILING(d)`	Renvoie le plus petit nombre entier qui n'est pas inférieur à d. [Fonctionne dans l'IGU]	`CEIL(d)` `CEILING(d)`	
`COS(d)`	Renvoie le cosinus. [Fonctionne dans l'IGU]	`COS(radians)`	Les radians peuvent également être représentés à l'aide de l'angle (ici pour le cercle unité) : radians = (2 * PI () * angles / 360)
		`COSH(d)`	
`COT(d)`	Renvoie la cotangente. [Fonctionne dans l'IGU]	`COT(d)`	
`DEGREES(d)`	Convertit les radians en degrés. [Fonctionne dans l'IGU]		
`EXP(d)`	Renvoie e^d (e : (2.718…)). [Fonctionne dans l'IGU]	`EXP(d)`	
`FLOOR(d)`	Renvoie le plus grand nombre entier qui n'est pas supérieur à d. [Fonctionne dans l'IGU]	`FLOOR(d)`	

`LOG(d)`	Renvoie le logarithme naturel en base e. [Fonctionne dans l'IGU]	`LN(d)`	
`LOG10(d)`	Renvoie le logarithme en base 10. [Fonctionne dans l'IGU]	`LOG10(d)`	
		`LOG(base, d)`	Renvoie à nouveau le logarithme dans n'importe quelle base.
`MOD(a, b)`	Renvoie le reste sous forme de nombre entier, dans la division de 2 nombres entiers. MOD (11,3) renvoie 2, car 3 * 3 + 2 = 11 [Fonctionne dans l'IGU]	`MOD(a, b)`	
`PI()`	Renvoie π (3,1415…). [Fonctionne dans l'IGU]	`PI()`	
`POWER(a, b)`	a^b, POWER(2,3) = 8, car $2^3 = 8$ [Fonctionne dans l'IGU]	`POWER(x, y)`	
`RADIANS(d)`	Convertit les degrés en radians. [Fonctionne dans l'IGU]		
`EDGE()`	Renvoie un nombre aléatoire supérieur ou égal à 0,0 et inférieur à 1,0. [Fonctionne dans l'IGU]	`EDGE()`	
`ROUND(a, b)`	Arrondit a à b décimales. [Fonctionne dans l'IGU]	`ROUND(d [, places])`	Arrondit après le nombre de chiffres spécifié à partir du point décimal. ROUND (123.45, 1) renvoie 123.50 ROUND (123.45, –2) renvoie 100.00 [Fonctionne dans l'IGU]

ROUNDMAGIC (d)	Résout les problèmes d'arrondi résultant de l'utilisation de nombres à virgule flottante. 3.11-3.1-0.01 n'est pas exactement 0, mais est affiché comme 0 dans l'interface graphique. ROUNDMAGIC en fait une valeur zéro réelle. [Fonctionne dans l'IGU]		
SIGN(d)	Renvoie –1 si d est inférieur à 0, 0 si d est égal à 0 et 1 si d est supérieur à 0. [Fonctionne dans l'IGU]	SIGN(d)	
SIN(A)	Renvoie le sinus d'un angle en radians. [Fonctionne dans l'IGU]	SIN(radians)	
		Sinh(d)	
SQRT(d)	Renvoie la racine carrée. [Fonctionne dans l'IGU]	SQRT(d)	
TAN(A)	Renvoie la tangente d'un angle en radians. [Fonctionne dans l'IGU]	TAN(radians)	
		TANH(d)	
TRUNCATE (a, b)	Tronque a à b décimales. TRUNCATE(2.37456.2) = 2.37 [Fonctionne dans l'IGU]	TRUNC(d[, jobs])	Définit sur 0 après le nombre de chiffres spécifié à partir du point décimal. TRUNC (123.45, 1) renvoie 123. 4 0 ROUND (123.45, –2) renvoie 100.00 [Fonctionne dans l'IGU]

Texte

HSQLDB		*Firebird*	
`ASCII(s)`	Renvoie le code ASCII de la première lettre de la chaîne. [Fonctionne dans l'IGU]	`ASCII_VAL ("s")`	Renvoie la valeur numérique qui correspond au caractère saisi. [Fonctionne dans l'IGU]
`BIT_LENGTH (str)`	Renvoie la longueur de la chaîne de texte str en bits. [Fonctionne dans l'IGU]	`BIT_LENGTH ("s")`	
`CHAR(c)`	Renvoie la lettre qui appartient au code ASCII. Il ne s'agit pas seulement de lettres, mais aussi de caractères de contrôle. CHAR (13) crée un saut de ligne dans une requête, qui est visible dans les champs multilignes d'un formulaire ou dans les rapports. [Fonctionne dans l'IGU]	`ASCII_CHAR (n)`	Renvoie la lettre qui appartient au code ASCII. Il ne s'agit pas seulement de lettres, mais aussi de caractères de contrôle. [Fonctionne dans l'IGU]
`CHAR_LENGTH (str)`	Renvoie la longueur du texte en nombre de lettres. [Fonctionne dans l'IGU]	`CHAR_LENGTH ("s")` `CHARACTER_LENGTH("s")`	
		`CHAR_TO_UUID`	Converts a 36-character Universally Unique Identifier (UUID) to a 16-octet UUID (outputs unreadable characters). Convertit un identifiant unique universel (UUID) de 36 caractères en un UUID de 16 octets (génère des caractères illisibles).
`CONCAT (STR1, STR2)`	Joint str1 + str2. [Fonctionne dans l'IGU]		

`DIFFERENCE (s1, s2)`	Renvoie la différence de son entre s1 et s2. Seul un nombre entier est émis. 0 signifie qu'ils sonnent de la même manière. Par exemple, « sert » et « serre » donnent 0, « pair » et « faire » donnent 1, « mer » et « mère » donnent 0. [Fonctionne dans l'IGU]		
`HEXTORAW (s1)`	Traduit le code hexadécimal en d'autres caractères. [Fonctionne dans l'IGU]		
`INSERT(s, start, len, s2)`	Renvoie une chaîne de texte, avec une partie du texte remplacée. En commençant à d une longueur l est découpée dans le texte s et remplacée par le texte s2. INSERT ("Consolation", 3, 4, firm) convertit Consolation en Confirmation, où la longueur du texte inséré peut être supérieure à celle du texte supprimé sans causer de problèmes. Donc INSERT (« Bundesbahn », 3, 5, s et B) donne « Bus und Bahn ». [Fonctionne dans l'IGU]	`OVERLAY ("s" PLACING 's2' FROM pos [FOR length])`	Si la position de départ est définie de sorte qu'elle soit supérieure ou égale au texte réel « s », alors « s2 » est directement ajouté à « s ». OVERLAY "Consolation" PLACING "firm" FROM 3 FOR 4) transforme "Consolation" en "Confirmation", où la longueur du texte inséré peut être plus longue que celle du texte coupé. Donc OVERLAY ("Bundesbahn" PLACING 's und B' FROM 3 FOR 5) donne 'Bus und Bahn'. [ne fonctionne pas dans l'interface graphique]
`LCASE(s)`	Convertit une chaîne en minuscules. [Fonctionne dans l'IGU]		
`LEFT(s, count)`	Renvoie le nombre de caractères spécifié par compte à partir du début de la chaîne s. [Fonctionne dans l'IGU]	`LEFT("s", length)`	
`LENGTH(s)`	Renvoie la longueur de la chaîne s en caractères. [Fonctionne dans l'IGU]		

`LOCATE (search, s [, start])`	Renvoie la première correspondance du terme recherché dans la chaîne s. La correspondance est donnée sous forme de nombre de décalage : (1 = gauche, 0 = introuvable). La définition d'un point de départ dans la chaîne de texte est facultative. [Fonctionne dans l'IGU]	`POSITION ('s2' IN "s")` `POSITION ('s2', "s" [, Début])`	
`POSITION (<string expression> IN <string expression>)`	Si la première chaîne est contenue dans la seconde, la position de la première chaîne sera renvoyée, sinon 0 sera affiché. Cela pourrait être utilisé à la place d'une option de recherche avec LIKE. [fonctionne dans l'interface graphique]	`POSITION ('s2' IN "s")` `POSITION ('s2', "s" [, Début])`	
`LTRIM(s)`	Supprime les espaces de début et les caractères non imprimables au début d'une chaîne de texte. [Fonctionne dans l'IGU]		
`OCTET_LENGTH (str)`	Renvoie la longueur d'une chaîne de texte en octets. Cela correspond à deux fois la longueur de la chaîne. [Fonctionne dans l'IGU]	`OCTET_LENGTH (str)`	Renvoie le nombre réel de caractères, en tenant compte des espaces. Le nombre dépend du jeu de caractères. UTF-8 a besoin de deux octets pour les caractères spéciaux.
`RAWTOHEX(s1)`	Convertit en hexadécimaux, inverse deHEXTORAW(). [Fonctionne dans l'IGU]		
`REPEAT(s, count)`	Répète le nombre de fois de la chaîne de texte. [Fonctionne dans l'IGU]		
`REPLACE(s, replace, s2)`	Remplace toutes les occurrences de remplace existantes dans la chaîne de texte s par la chaîne s2. [Fonctionne dans l'IGU]	`REPLACE("s", 's2', remplace)`	REPLACE ("Confirmation", "firm", "test") convertit "confirmation" en "Contestation". Si « s2 » n'apparaît pas dans « s », rien ne sera remplacé. Si N2 apparaît dans « s2 » ou en remplacement, le résultat est NULL.

		LPAD(“s”, Longueur totale [, caractères])	LPAD (“Hello”, 8 , '+') = +++ Hello Place tous les caractères devant une chaîne jusqu’à ce que la longueur totale soit atteinte. Si la longueur totale est inférieure à la longueur de la chaîne, la chaîne est coupée à droite. Si le troisième paramètre reste vide, des espaces sont placés devant lui.
		RPAD(“s”, Longueur totale [, caractères])	RPAD (“Hello”, 8 , '+') = Hello +++ RPAD (“Bonjour”, 8, '+') = Bonjour +++ Place tous les caractères derrière une chaîne jusqu’à ce que la longueur totale soit atteinte. Si la longueur totale est inférieure à la longueur de la chaîne, la chaîne est coupée à droite. Si le troisième paramètre reste vide, des espaces sont placés derrière lui.
		REVERSE(“s”)	Iinverse complètement la chaîne. Cela peut être utile, par exemple, si vous souhaitez trier par terminaisons de caractères (par exemple, les fins de domaine).
RIGHT(s, longueur)	Opposé de LEFT ; renvoie le dernier nombre de caractères à la fin d’une chaîne de texte. [Fonctionne dans l’IGU]	RIGHT(“s”, Longueur)	
RTRIM(s)	Supprime tous les espaces et les caractères non imprimables à la fin d’une chaîne de texte. [Fonctionne dans l’IGU]		
SOUNDEX(s)	Renvoie un code à 4 caractères, correspondant au son de s. Correspond à la fonction DIFFERENCE(). [Fonctionne dans l’IGU]		
SPACE(count)	Renvoie le nombre d’espaces. [Fonctionne dans l’IGU]		

`SUBSTR(s, start [, len])`	Abréviation de SUBSTRING. [Fonctionne dans l'IGU]		
`SUBSTRING (s, début [, longueur])`	Renvoie le texte s à partir de la position de début (1 = gauche). Si longueur est omis, la chaîne entière est renvoyée. [Fonctionne dans l'IGU]	`SUBSTRING ("s" FROM début [FOR longueur])`	Renvoie la partie d'une chaîne à partir de la position de départ spécifiée dans FROM, éventuellement dans la longueur spécifiée dans FOR. Si, par exemple, "Roberta" apparaît dans le champ "Nom", SUBSTRING ("Name" FROM 3 FOR 3) entraîne la sous-chaîne "bert". [Fonctionne dans l'IGU]
`TRIM ([{LEADING | TRAILING | BOTH}] FROM <expression chaine>`	Les caractères spéciaux et les espaces qui ne peuvent pas être imprimés sont supprimés. [Fonctionne dans l'IGU]	`TRIM ([{LEADING | TRAILING | BOTH}] FROM <expression chaine>`	BOTH | LEADING | TRAILING : la norme ici est BOTH pour les deux côtés de « s ». s2 : Le caractère à supprimer. Par défaut, ce sont des espaces (' '), mais peuvent également être d'autres caractères. TRIM (TRAILING '!' FROM 'Hallo !') Renvoie "Hallo"
`UCASE(s)`	Convertit une chaîne en majuscules. [Fonctionne dans l'IGU]		
`LOWER(s)`	Comme LCASE(s). [Fonctionne dans l'IGU]	`LOWER("s")`	
`UPPER(s)`	Comme UCASE(s). [Fonctionne dans l'IGU]	`UPPER("s")`	
		`UUID_TO_CHAR ("s")`	Converts a 16-octet UUID to a 36-character ASCII format.

Date/Heure

HSQLDB	***Firebird***

		DATEADD (n DAY TO date) DATEADD (DAY, n, date)	n est un entier et peut également être négatif pour la soustraction. ANNÉE | MOIS | SEMAINE | JOUR | HOUR | MINUTE | SECOND | MILLISECOND doit être utilisé comme terme pour l'intervalle de temps. Utilisez un champ de date / date, un champ d'heure / heure ou un horodatage comme terme de date.
DATEDIFF (string, datetime1, datetime2)	Différence de date entre deux dates ou heures. L'entrée dans la chaîne décide dans quelle unité la différence est affichée : "ms" = "milliseconde", "ss" = "seconde", "mi" = "minute", "hh" = "heure", "dd" = ' jour ',' mm '=' mois ',' aa '=' année '. La version longue et la version courte de la chaîne peuvent être utilisées. [Fonctionne dans l'IGU]	DATEDIFF (DAY FROM date TO date) DATEDIFF (DAY, date, date)	See DATEADD.
EXTRACT ({YEAR|MONTH|DAY|HOUR| MINUTE| SECOND} FROM <date or time value>)	Peut remplacer de nombreuses fonctions de date et d'heure. Renvoie l'année, le mois, le jour, etc. à partir d'une valeur de date ou d'heure. EXTRACT (DAY FROM "date") indique le jour du mois. [Fonctionne dans l'IGU]	EXTRACT ({YEAR| MONTH | WEEK | DAY | WEEKDAY | YEARDAY | HOUR | MINUTE | SECOND | MILLISECOND } FROM <date temps>)	WEEKDAY Dimanche = 0 YEARDAY 1er Janvier = 0 WEEK 1ère semaine : minimum. 4 jours
DAY(date)	Renvoie le jour du mois (1-31). [Fonctionne dans l'IGU]		
DAYNAME (date)	Renvoie le nom anglais du jour. [Fonctionne dans l'IGU]		

`DAYOFMONTH (date)`	Renvoie le jour du mois (1-31). Synonyme de DAY(). [Fonctionne dans l'IGU]		
`DAYOFWEEK (date)`	Renvoie le jour de la semaine sous forme de nombre (1 représente dimanche). [Fonctionne dans l'IGU]		
`DAYOFYEAR (date)`	Renvoie le jour de l'année (1-366). [Fonctionne dans l'IGU]		
`HOUR(time)`	Renvoie l'heure (0-23). [Fonctionne dans l'IGU]		
`MINUTE(time)`	Renvoie les minutes (0-59). [Fonctionne dans l'IGU]		
`MONTH(date)`	Renvoie le mois (1-12). [Fonctionne dans l'IGU]		
`MONTHNAME (date)`	Renvoie le nom anglais du mois. [Fonctionne dans l'IGU]		
NOW()	Renvoie la date actuelle et l'heure actuelle ensemble sous forme d'horodatage. Vous pouvez également utiliser CURRENT_TIMESTAMP. [Fonctionne dans l'IGU]		
`QUARTER (date)`	Renvoie le trimestre de l'année (1-4). [Fonctionne dans l'IGU]		
`SECOND(time)`	Renvoie la partie des secondes du temps (0-59). [Fonctionne dans l'IGU]		
`WEEK(date)`	Renvoie la semaine de l'année (1-53). [Fonctionne dans l'IGU]		

`YEAR(date)`	Renvoie la partie année d'une entrée de date. [Fonctionne dans l'IGU]		

Connexion à la base de données

HSQLDB		*Firebird*	
`DATABASE()`	Renvoie le nom de la base de données à laquelle appartient cette connexion. [Fonctionne dans l'IGU]		
		`CURRENT_TRANSACTION`	SELECT CURRENT_T RANSACTION FROM RDB $ DATABASE renvoie l'identifiant unique de la transaction sous forme d'entier.
		`CURRENT_CONNECTION`	SELECT CURRENT_CONNECTION FROM RDB $ DATABASE ret renvoie une valeur entière pour la connexion actuelle.
		`CURRENT_ROLE`	SELECT CURRENT_ROLE FROM RDB $ DATABASE reflète le rôle de l'utilisateur actuel. Si aucun rôle n'est défini, le résultat est NONE.
		`RDB $ SET_CONTEXT ('<namespace>', '<variable name>', value | NULL)`	Espace de noms : USER_SESSION | USER_TRANSACTION Le nom de la variable peut avoir un maximum de 80 caractères et la valeur peut avoir un maximum de 255 caractères.
`CURRENT_USER`	Fonction standard SQL, synonyme de USER(). Il convient de noter qu'il n'y a pas de parenthèses ici. [Fonctionne dans l'IGU]	`CURRENT_USER`	
`USER()`	Renvoie le nom d'utilisateur de cette connexion. Le nom d'utilisateur est important si la base de données doit être convertie en base de données externe. [SQL direct – ne fonctionne pas avec l'IGU]	`USER`	

`IDENTITY()`	Renvoie la dernière valeur d'un champ de valeur automatique, qui a été créé dans la connexion actuelle. Ceci est utilisé dans le codage de macro pour transférer une clé primaire dans une table pour devenir une clé étrangère pour une autre table. [Fonctionne dans l'IGU]	`GEN_ID (generator name, <step>)`	Les valeurs automatiques sont créées avec un générateur. La taille du pas doit être donnée ici sous la forme 1. En principe, toute valeur entière est possible. `new.rec_id = gen_id (gen_recnum, 1)`

Système

HSQLDB		***Firebird***	
`IFNULL (exp, valeur)`	Si exp est NULL, la valeur est renvoyée, sinon exp. Au lieu de cela, COALESCE() peut également être utilisé comme une extension. Exp et value doivent avoir le même type de données. IFNULL est une fonction importante lorsque les champs sont liés les uns aux autres par facture ou CONCAT. Le contenu du résultat serait NULL, même si une seule valeur est NULL. « Nom de famille » || ',' || "prénom" entraînerait un champ vide pour les personnes qui, par exemple, ne disposent pas de l'entrée pour "prénom", c'est-à-dire NULL."Nom de famille » || IFNULL (',' || « First Name », '') afficherait simplement « Nom de famille » à la place. [Fonctionne dans l'IGU]		
`CASE WHEN (exp, v1, v2)`	Si exp est vrai, v1 est renvoyé, sinon v2. CASE WHEN peut également être utilisé à la place. CASEWHEN ("a"> 10, 'objectif atteint', 'continuer à pratiquer') renvoie 'objectif atteint' si le contenu du champ "a" est supérieur à 10. CASE WHEN fonctionne mieux avec l'interface graphique [Fonctionne dans l'IGU]	`IIF (<condition>, v1, v2)`	

<table>
<tr><td>CONVERT (term, type)</td><td>Convertit le terme en un autre type de données. CONVERT (“a”, DECIMAL (5,2)) fait du champ “a” un champ à 5 chiffres, dont 2 décimales. Si le nombre est trop grand, une erreur est générée.
[Fonctionne dans l’IGU]</td><td></td><td colspan="2"></td></tr>
<tr><td>CAST (terme AS type)</td><td>Synonyme pour CONVERT ()
[Fonctionne dans l’IGU]</td><td>CAST (terme AS type)</td><td>From</td><td>To</td></tr>
<tr><td></td><td></td><td></td><td>Types numeriques</td><td>Types numeriques
[VAR] CHAR
BLOB</td></tr>
<tr><td></td><td></td><td></td><td>[VAR] CHAR
BLOB</td><td>[VAR] CHAR
BLOB
Types numeriques
DATE
TIME
TIMESTAMP</td></tr>
<tr><td></td><td></td><td></td><td>DATE
TIME</td><td>[VAR] CHAR
BLOB
TIMESTAMP</td></tr>
<tr><td></td><td></td><td></td><td>TIMESTAMP</td><td>[VAR] CHAR
BLOB
DATE
TIME</td></tr>
<tr><td>COALESCE (expr1, expr2, expr3,…)</td><td>Si expr1 n’est pas NULL, expr1 est affiché, sinon expr2 est vérifié, puis expr3, etc. Toutes les expressions doivent avoir au moins un type de données similaire. Il s’agit de la représentation alternative des nombres entiers et des nombres à virgule flottante, mais pas également d’une valeur de date ou d’heure.
[Fonctionne dans l’IGU]</td><td>COALESCE (expr1, expr2 [, expr3…]</td><td colspan="2"></td></tr>
</table>

`NULLIF (v1, v2)`	Si v1 est égal à v2, null est renvoyé, sinon la valeur de v1 est renvoyée. Les données doivent être de type comparable. [Fonctionne dans l'IGU]	`NULLIF (v1, v2)`	
`CASE v1 WHEN v2 THEN v3 [ELSE v4] END`	Si v1 est égal à v2, v3 est renvoyé. Sinon, v4 est retourné ou NULL, s'il n'y a pas de condition ELSE. [SQL Direct – ne fonctionne pas dans l'IGU]	`DECODE (test expression , expression , resultat [, expression2 , Earnings2 …] [, default expression])`	DECODE (UPPER ("sexe"), "M", "homme", "F", "femme", "inconnu")
`CASE WHEN expr1 THEN v1 [WHEN expr2 THEN v2] [ELSE v4] END`	Si expr1 est vrai, v1 est retourné [Facultativement, d'autres cas peuvent être spécifiés]. Sinon, la v4 est reproduite ou NULL si aucun ELSE n'est formulé. CASE WHEN DAYOFWEEK ("date") = 1 THEN "Dimanche" WHEN DAYOFWEEK ("date") = 2 THEN "Lundi"… END pourra afficher le jour de la semaine via SQL, qui n'est autrement disponible qu'en anglais dans la fonction. [Fonctionne dans l'IGU]		DECODE (EXTRACT (WEEK DAY FROM "date"), 0 , "Dimanche", 1 , Lundi', etc.')
		`GEN_UUID ()`	Renvoie un ID unique sous la forme d'un jeu de caractères de 16 octets.
		`HASH (s)`	Renvoie la valeur de hachage d'une chaîne arbitrairement longue. Les valeurs de hachage des mêmes chaînes de caractères doivent être identiques.
		`MAXVALUE (expr [, expr…])`	Renvoie la valeur maximale d'une liste de valeurs. Fonctionne avec des chaînes, des valeurs numériques, des valeurs de date ou d'heure.

		`MINVALUE (expr [, expr…])`	Renvoie la valeur minimale d'une liste de valeurs. Fonctionne avec des chaînes, des valeurs numériques, des valeurs de date ou d'heure.

Fonctions d'agrégat (en particulier avec GROUP BY)

HSQLDB		***Firebird***	
		MAX (expr)	Valeur maximale d'un champ dans une table.
		MIN (expr)	Valeur minimale d'un champ dans une table.
		LIST ([ALL | DISTINCT] 's, [s2'])	Connecte les champs de plusieurs enregistrements de données à un champ avec le terme de connexion correspondant 's2 '. [Fonctionne dans l'IGU]

Variables (selon l'ordinateur)

HSQLDB		***Firebird***	
`CURRENT_TIME`	Synonyme de CURTIME(), standard SQL. [Fonctionne dans l'IGU]	`CURRENT_TIME`	Temps en heures, minutes et secondes. CURRENT_TIME (3) spécifie également les millisecondes.
`CURTIME ()`	Renvoie l'heure actuelle.[Fonctionne dans l'IGU]		
`CURRENT_TIMESTAMP`	Synonyme de NOW (), standard SQL. [Fonctionne dans l'IGU]	`CURRENT_TIMESTAMP [(accuracy)]`	Spécification de l'heure avec date et millisecondes. La précision peut être définie avec [0 | 1 | 2 | 3]. La norme est de 3 décimales. SELECT CURRENT_TIMESTAMP (2) FROM RDB $ DATABASE renvoie l'horodatage avec des dixièmes et des centièmes de seconde (2 décimales).
`NOW ()`	Renvoie la date et l'heure actuelles sous forme d'horodatage. CURRENT_TIMESTAMP peut également être utilisé à la place. [Fonctionne dans l'IGU]	`CAST ("NOW" AS DATE | TIME | TIMESTAMP)` `or DATE "NOW"`	« NOW », écrit seul, est compris comme une chaîne. Avec la conversion appropriée, il devient une date, une heure ou un horodatage (chacun avec 1/1000 s). La forme courte ne fonctionne pas dans l'interface graphique.
`CURRENT_DATE`	Synonyme de CURDATE (), standard SQL. [Fonctionne dans l'IGU]	`CURRENT_DATE`	
`CURDATE ()`	Returns the current date. [Fonctionne dans l'IGU]		

Opérateurs et déclarations

HSQLDB	***Firebird***

`'str1' ||`
`'str2' ||`
`'str3' or`
`'str1' +`
`'str2' +`
`'str3'`
	Concatène str1 + str2 + str3 ; alternative plus simple à CONCAT. [Fonctionne dans l'IGU]	`'s1' || s 2`
		`[||' s3 '...]……`
			Concatène s1, s2, etc. en une nouvelle chaîne [Fonctionne dans l'IGU]

		`ALL`	
		`ANY / SOME`	
		`IN ()`	
		`IS [NOT]` `DISTINCT FROM`	Le résultat est "yes" or "no".
		`NEXT VALUE FOR` `nom sequence`	Voir GEN_ID (), mais n'autorise aucune étape autre que 1.
		`SOME`	

Types de données pour l'éditeur de tables

Entiers (Integer)

Type	*Option*	*HSQLDB*	*Firebird*	*Range*	*Storage Space*
Tiny integer	TINYINT	TINYINT		2^8 = 256 | – 128 à + 127	1 octet
Small integer	SMALLINT	SMALLINT	SMALLINT	2^{16} = 65536 | – 32768 à + 32767	2 octets
integer	INTEGER	INTEGER | INT	INTEGER	2^{32} = 4294967296 | – 2147483648 à + 2147483647	4 octets
BigInt	BIGINT	BIGINT	BIGINT	2^{64} (-2^{63} à $+2^{63}$)	8 octets

Nombres à virgule flottante

Type	*Option*	*HSQLDB*	*Firebird*	*Scope*	*Memory requirements*
Decimal	DECIMAL	DECIMAL	DECIMAL (n, m)	Illimité, via GUI à 50 chiffres, décimales réglables et fixes, précision exacte	2, 4 or 8 octet
Number	NUMERIC	NUMERIC	NUMERIC (n, m)	Illimité, via GUI à 50 chiffres, décimales réglables et fixes, précision exacte	2, 4 or 8 octet
Float	FLOAT	(DOUBLE est utilisé à la place)	FLOAT	$3.4 * 10^{-38}$ to $3.4 * 10^{38}$ réglable, pas exact, 7 décimales maximum	4 octet
Real	REAL	REAL			
Double	DOUBLE	DOUBLE [PRECISION] | FLOAT	DOUBLE PRECISION	$1 , 7 * 10^{-308}$ to $1 , 7 * 10^{308}$ réglable, pas exact, 15 décimales maximum	8 octets

Texte

Type	*Option*	*HSQLDB*	*Firebird*	*Scope*	*Memory requirements*
Text	VARCHAR	VARCHAR	VARCHAR (n)	Ajustable	Variable Firebird : 1 à 32767 octets
Text	VARCHAR_IGNORECASE	VARCHAR_IGNORECASE		Réglable, affecte le tri, ignore les différences entre les majuscules et les minuscules	variable
Text (fixed)	CHAR	CHAR | CHARACTER		Réglable, le reste du texte réel est rempli d'espaces	fixed
Memo	LONGVARCHAR	LONGVARCHAR	BLOB (BLOB SUB_TYPE 1)		variable Firebird : <32 GB

Heure

Type	*Option*	*HSQLDB*	*Firebird*	*Scope*	*Memory requirements*
Date	DATE	DATE	DATE		4 octets
Time	TIME	TIME	TIME	Firebird : 0:00 to 23:59,9999	4 octets
Date/Time	TIME STAMP	TIMESTAMP | DATE TIME	TIME STAMP	Réglable (HSQLDB : 0, 6-6 signifie avec des millisecondes)	8 octets

Autres

Type	*Option*	*HSQLDB*	*Firebird*	*Scope*	*Memory requirements*
Oui Non	BOOLEAN	BOOLEAN | BIT			
Champ binaire (fixed)	BINARY	BINARY		Comme integer	fixe
Champ binaire	VARBINARY	VARBINARY		Comme integer	variable
Image	LONGVARBINARY	LONGVARBINARY	BLOB SUB_TYPE 0 BLOB SUB_TYPE binary	Comme integer	variable, destiné aux images plus grandes Firebird : <32 GB
OTHER	OTHER	OTHER | OBJECT			

LibreOffice

Équipe documentation de LibreOffice

Guide Base

Gérer vos données

6.4

À propos de ce manuel :

Ce manuel s'adresse aux utilisateurs débutants et avancés de LibreOffice Base. Si vous n'avez jamais utilisé Base auparavant, ou si vous voulez une introduction à tous les composants de LibreOffice, vous voudrez peut-être lire Débuter avec LibreOffice.

Ce manuel vous présente les fonctions les plus courantes de LibreOffice Base:
- Création d'une base de données
- Entrée et sortie de données
- Travailler avec des tables, des formulaires, des requêtes, des rapports
- Utiliser des macros
- Maintenir une base de données
- Et beaucoup plus

À propos des auteurs :

Ce manuel a été écrit par des volontaires de la communauté LibreOffice. Les bénéfices des ventes de l'édition imprimée seront utilisés au profit de la communauté.

Une version PDF de ce livre peut être téléchargée gratuitement sur:

À propos de LibreOffice :

LibreOffice est la suite de productivité personnelle gratuite, libre et open source de The Document Foundation. Il fonctionne sous Windows, macOS et GNU/Linux. Le support et la documentation sont gratuits grâce à notre vaste communauté dédiée d'utilisateurs, de contributeurs et de développeurs.
Vous aussi, vous pouvez vous impliquer dans le travail bénévole dans de nombreux domaines: développement, assurance qualité, documentation, traduction, assistance aux utilisateurs, etc.
Vous pouvez télécharger LibreOffice gratuitement sur

Communauté fantastique. Projet passionnant. Suite bureautique gratuite.