Guide Base

Chapitre 3
Tables

Droits d’auteur

Contributeurs

De cette édition

Pulkit Krishna
Dan Lewis

Des éditions précédentes

Pulkit Krishna
Robert Großkopf
Hazel Russman
Dan Lewis
Jost Lange
Steve Schwettman
Jean Hollis Weber
Jochen Schiffers

Traduction

Jean-Michel Coste

Relecteurs

Philippe Clement
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 avril 2021. Basé sur LibreOffice 6.4.

Table des matières

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 1 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 %.

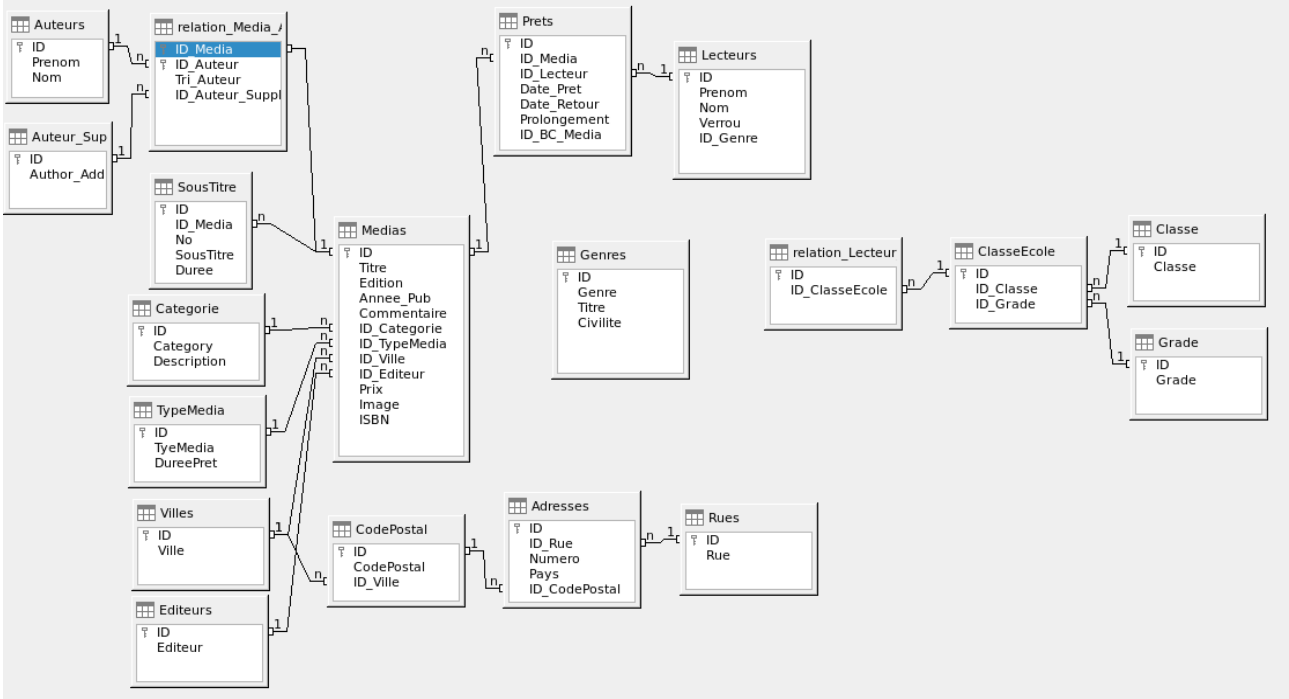


Figure 1: Diagramme de relations pour l'exemple de base de données Media_sans_Macros.

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 2 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.

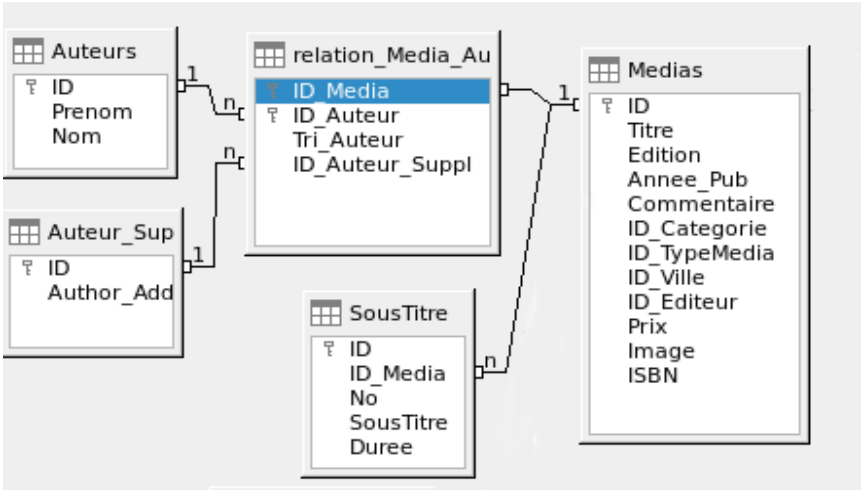


Figure 2: Exemple relation 1: n ; relation n : m

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)

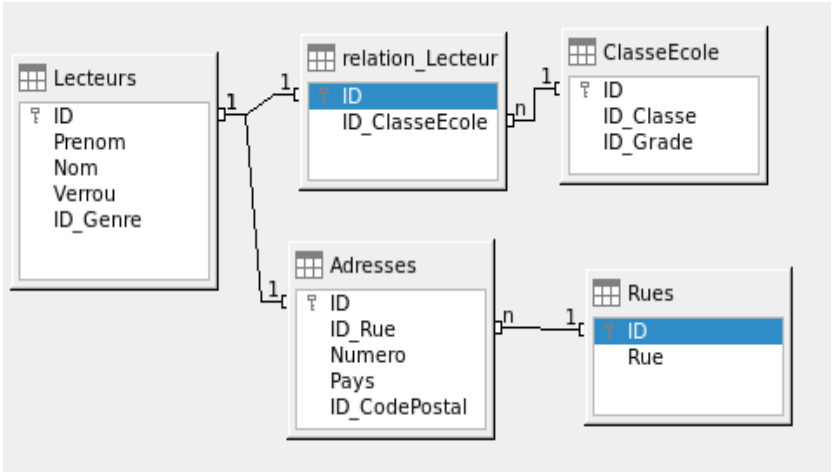


Figure 3: Exemple relation 1:1

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 3) , 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 4).

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 4.

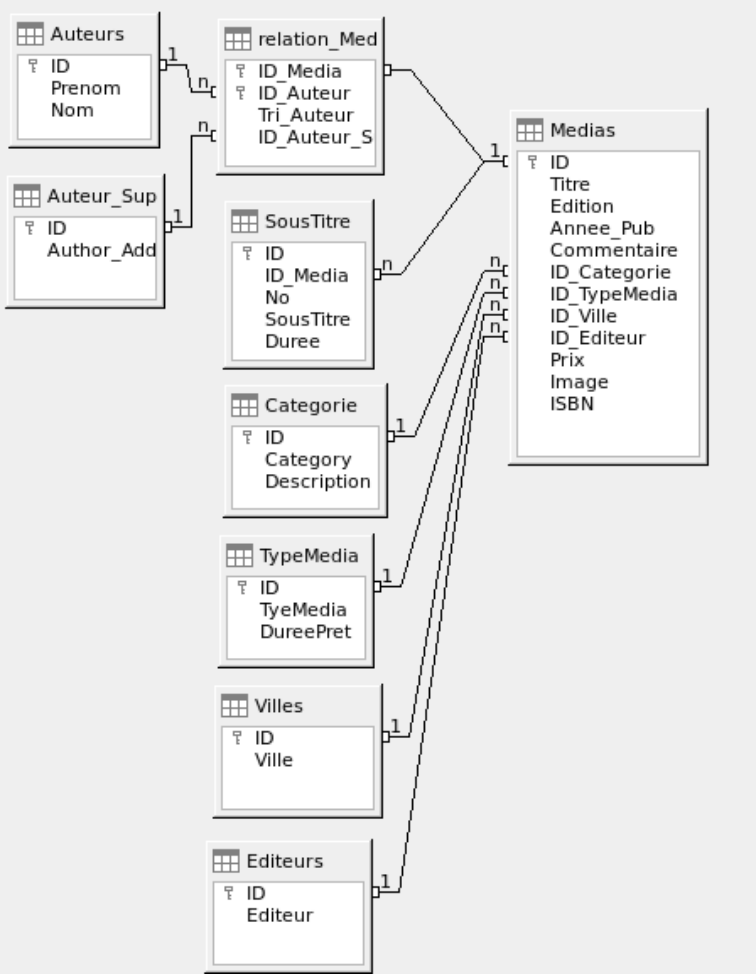


Figure 4: Ajout de médias

Table Prêt

La table centrale est *Pret* (voir Figure 5). 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.

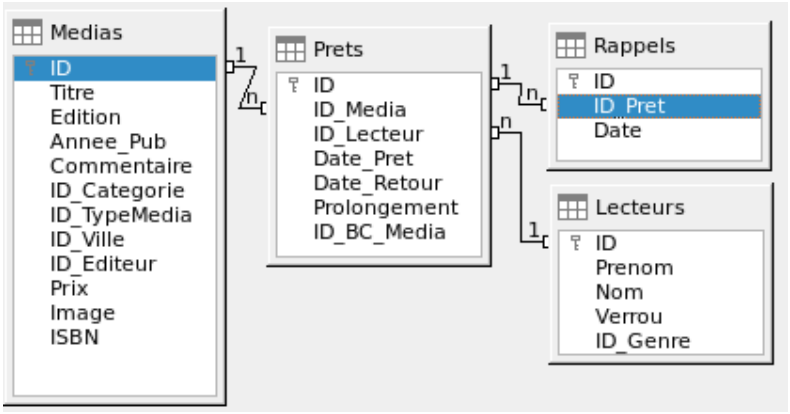


Figure 5: Prêts

Table d'administration des utilisateurs

Pour cette conception de table, deux scénarios sont envisagés. La chaîne de tables illustrée à la figure 6 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 6.

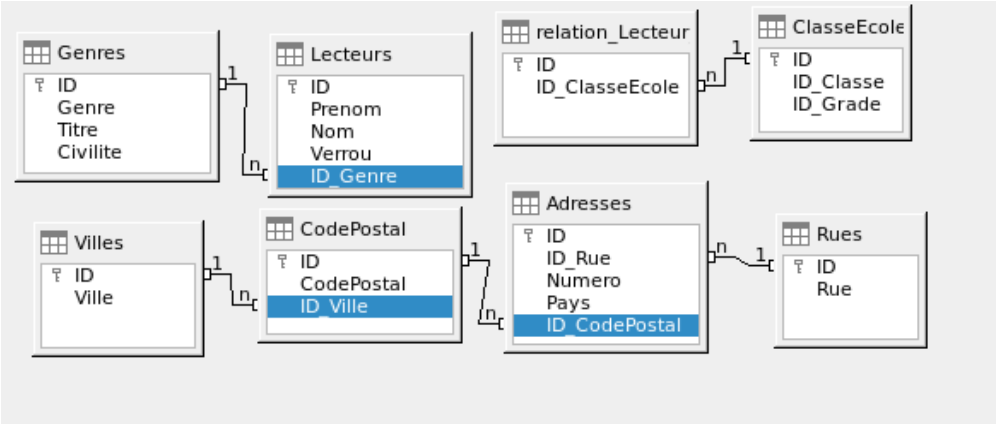


Figure 6: Lecteurs – une chaîne de classe scolaire et une chaîne d'adresses

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.

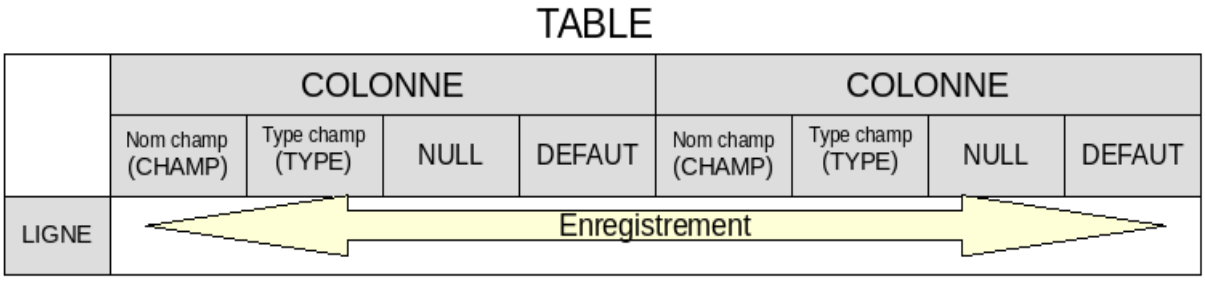
L'adresse est également divisée. La *Rue* est stockée séparément, car les noms de rue dans une zone sont souvent répétés. Le code postal et la ville sont séparés, car il y a souvent plusieurs codes postaux pour une même zone et donc plus de codes postaux que de villes. Ainsi, par rapport à la table Adresses,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

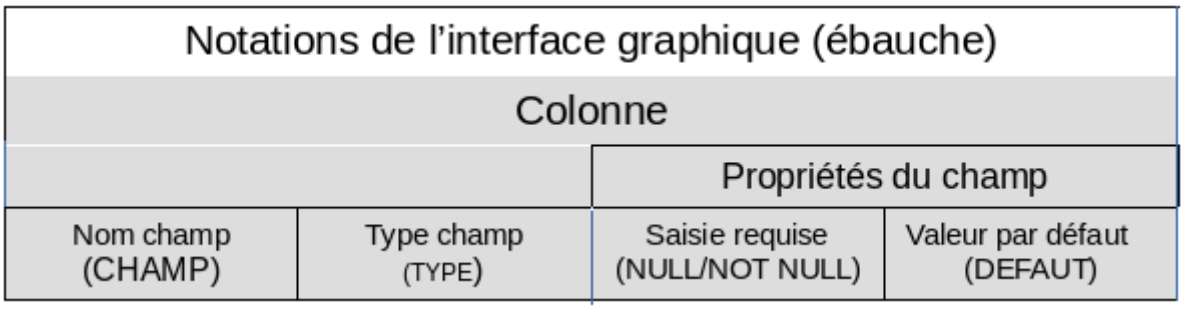
La plupart des utilisateurs de LibreOffice utiliseront généralement l'interface utilisateur graphique (GUI) exclusivement pour créer des tables. La saisie directe de commandes SQL devient nécessaire lorsque, par exemple, un champ doit être inséré ultérieurement à une position particulière, ou une valeur 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 22).

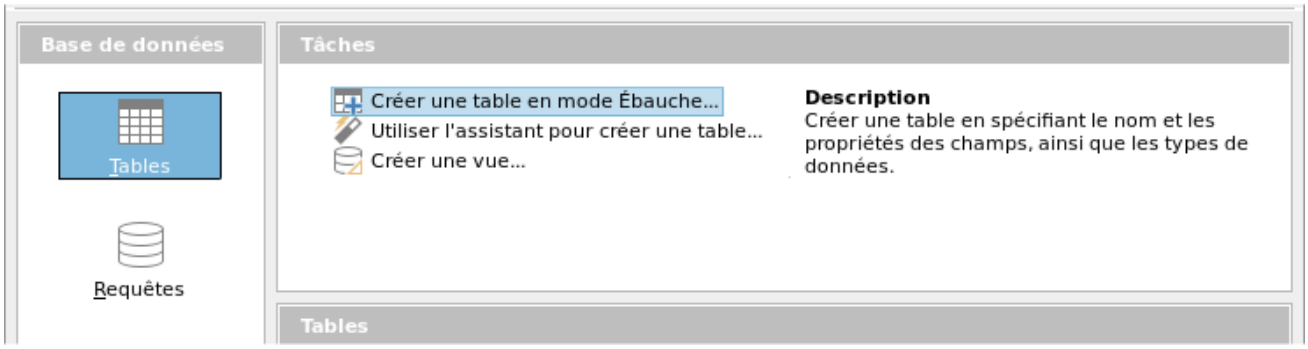
Note

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

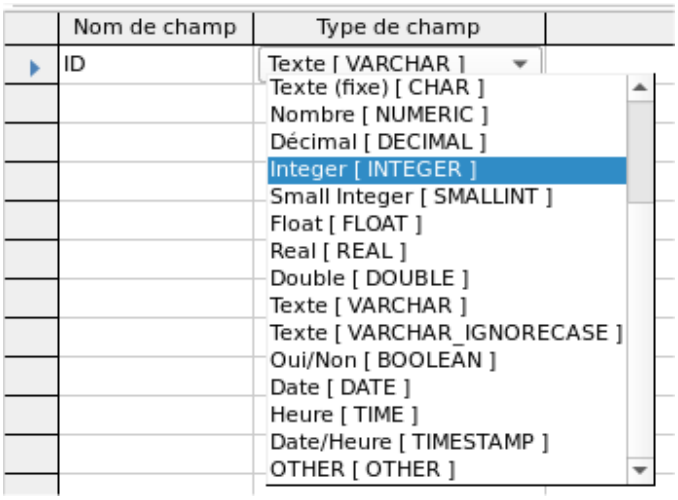
La création de la base de données à l'aide de l'interface utilisateur graphique (GUI) est expliquée étape par étape, en utilisant la table Medias comme exemple.



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

1) Champ ID :

   a) 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*.

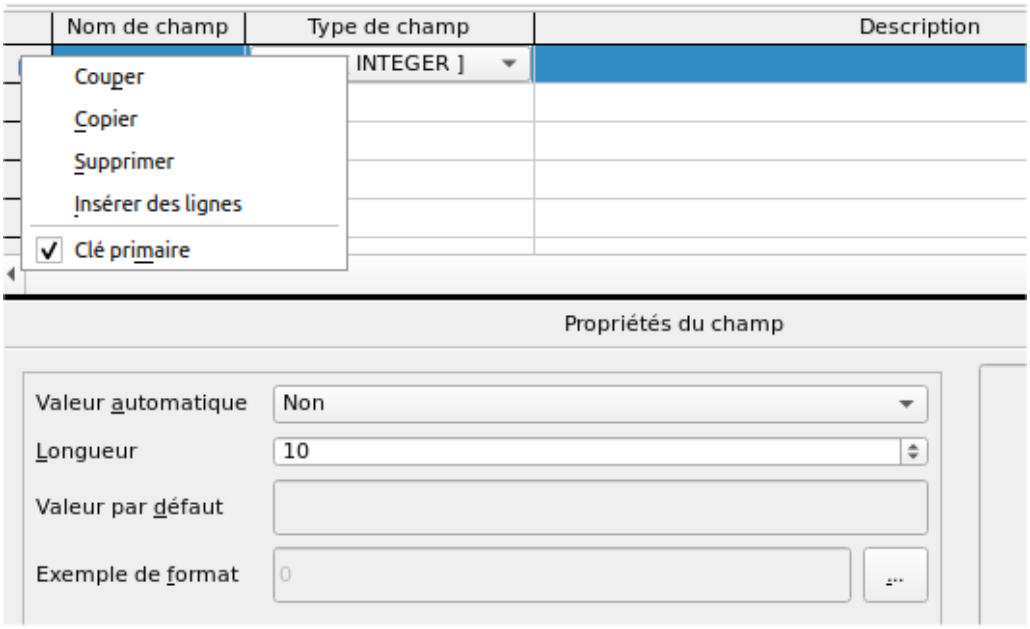


   b) Sélectionnez `Integer [INTEGER]` dans la liste comme type de champ. La valeur par défaut est `Texte [VARCHAR]`. Les nombres entiers peuvent stocker un nombre de 10 chiffres maximum. De plus, Integer est le seul type disponible dans l'interface graphique qui peut recevoir une valeur incrémentée automatiquement.

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.

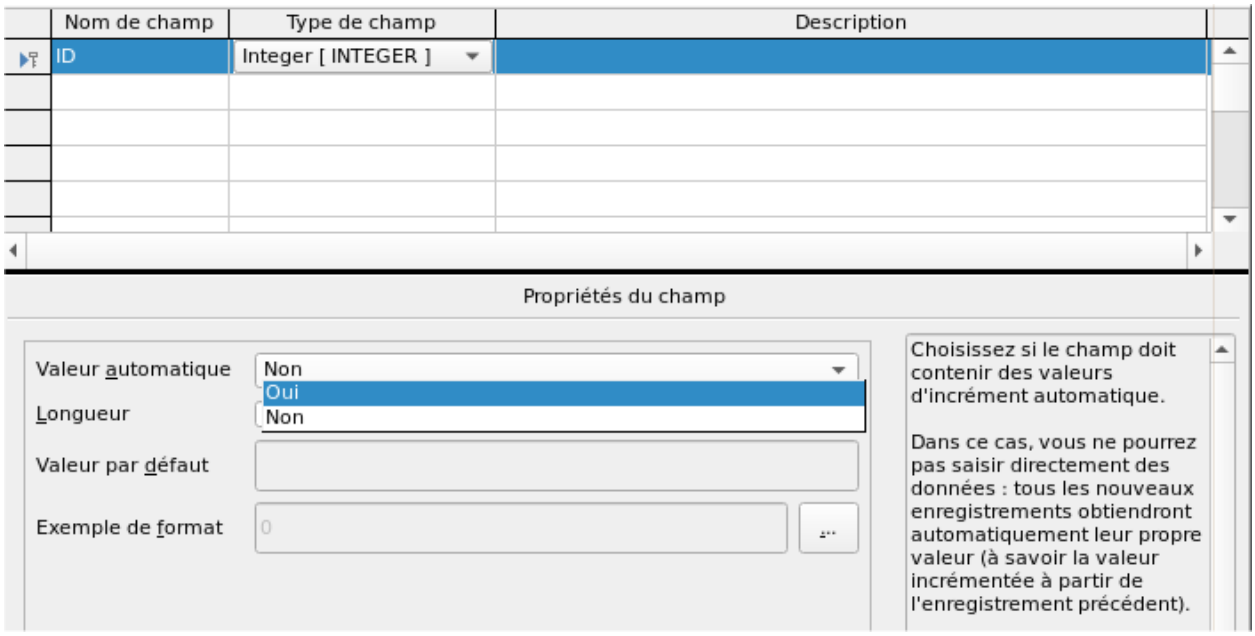
   c) Définissez le champ *ID* comme clé primaire en cliquant sur le rectangle devant son nom de champ avec le bouton droit de la souris et en sélectionnant Clé primaire dans le menu contextuel. Un symbole de clé apparaît avant l'ID.



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.

d) 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.

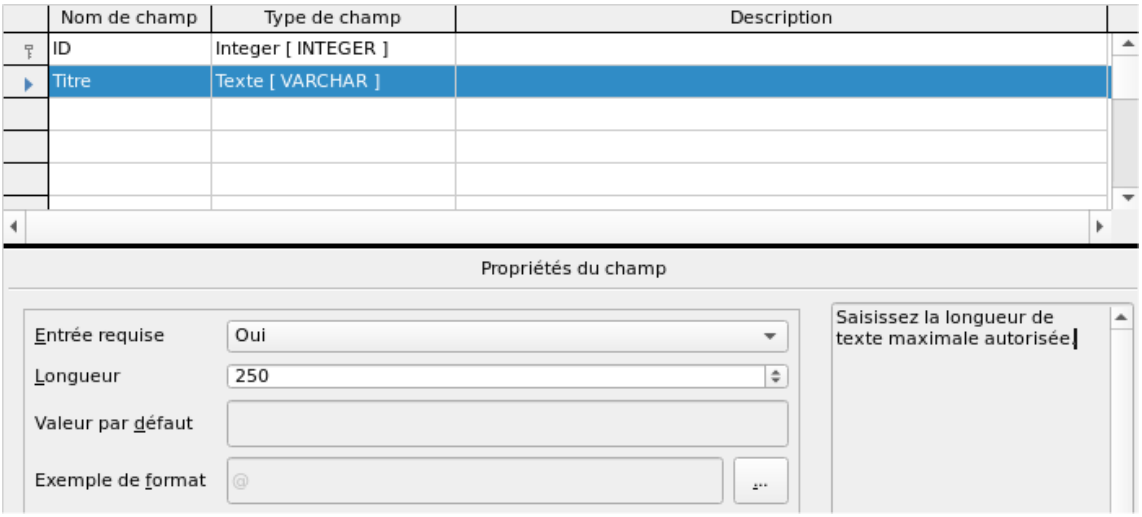


Valeur automatique ne peut être défini que pour un seul champ dans une table. Choisir **Valeur automatique > Oui** définit automatiquement ce champ comme clé primaire si aucune clé primaire n'a déjà été définie.

2) Le champ suivant est *Titre*.

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

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



c) 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.

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

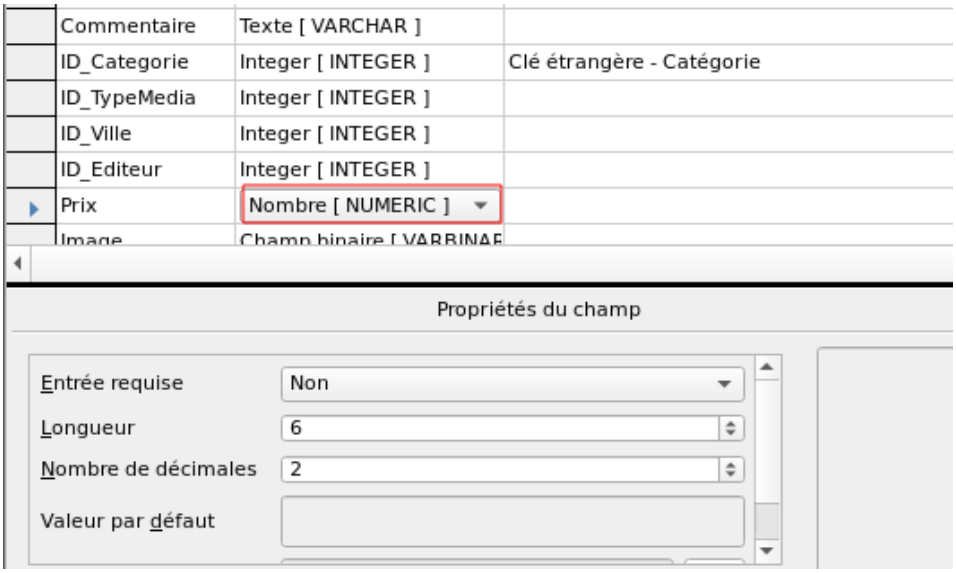
e) *Description* peut être n'importe quoi. Cette colonne peut également être laissée vide. La description sert uniquement à expliquer le contenu du champ aux personnes qui souhaitent afficher la définition de la table ultérieurement.

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

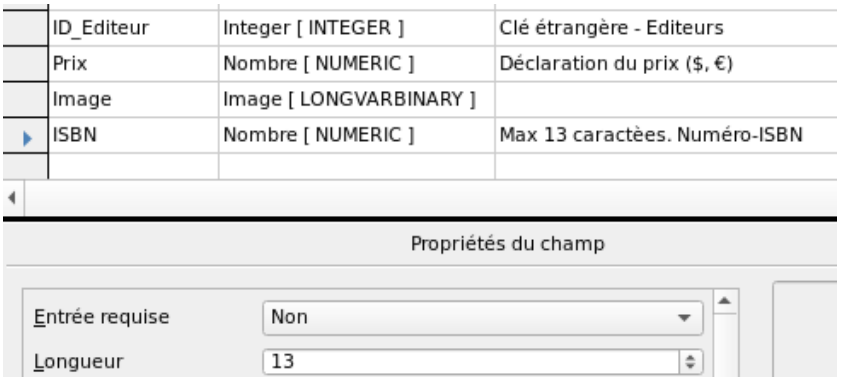
3) Pour le champ *Annee_Pub*, le type `Small Integer [SMALLINT]` a été choisi. Celui-ci peut contenir un entier d'une taille maximale de 5 chiffres. L'année de publication n'est pas utilisée dans les calculs, mais en faire un entier garantit qu'elle ne contiendra aucun caractère `alphabétique`.

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

4) 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.



5) 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.
   a) 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.
   b) Il n'est pas nécessaire de spécifier le caractère € dans le format, car une formule le gérera.



6) 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.
7) 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.

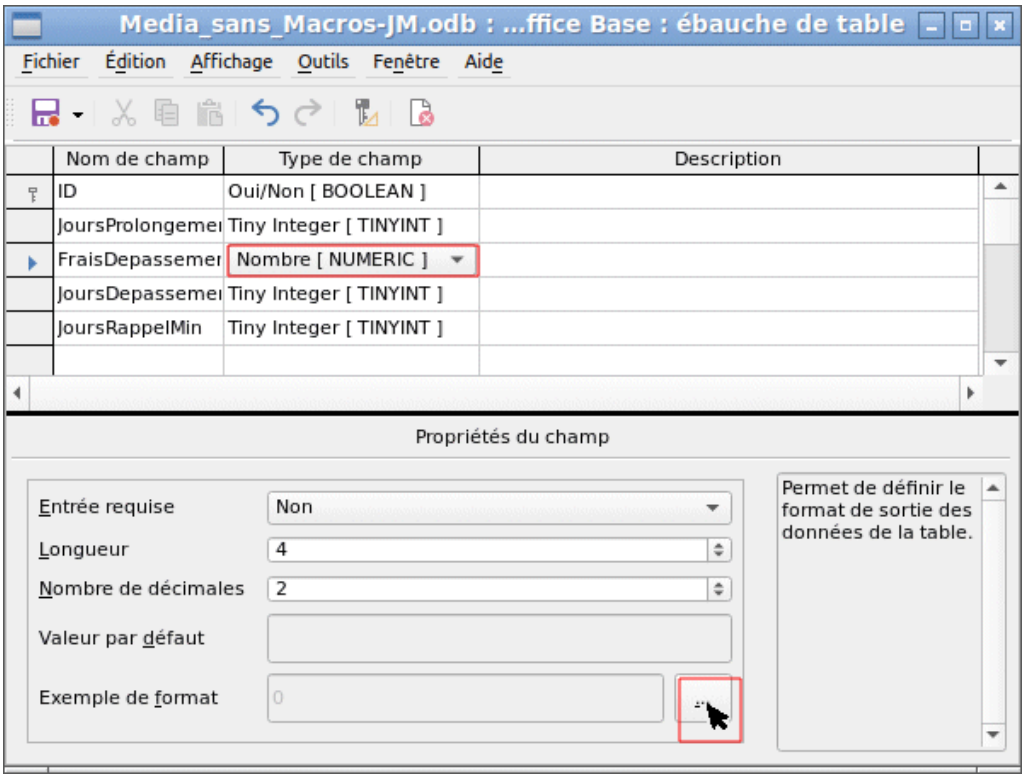
Formater les champs

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.

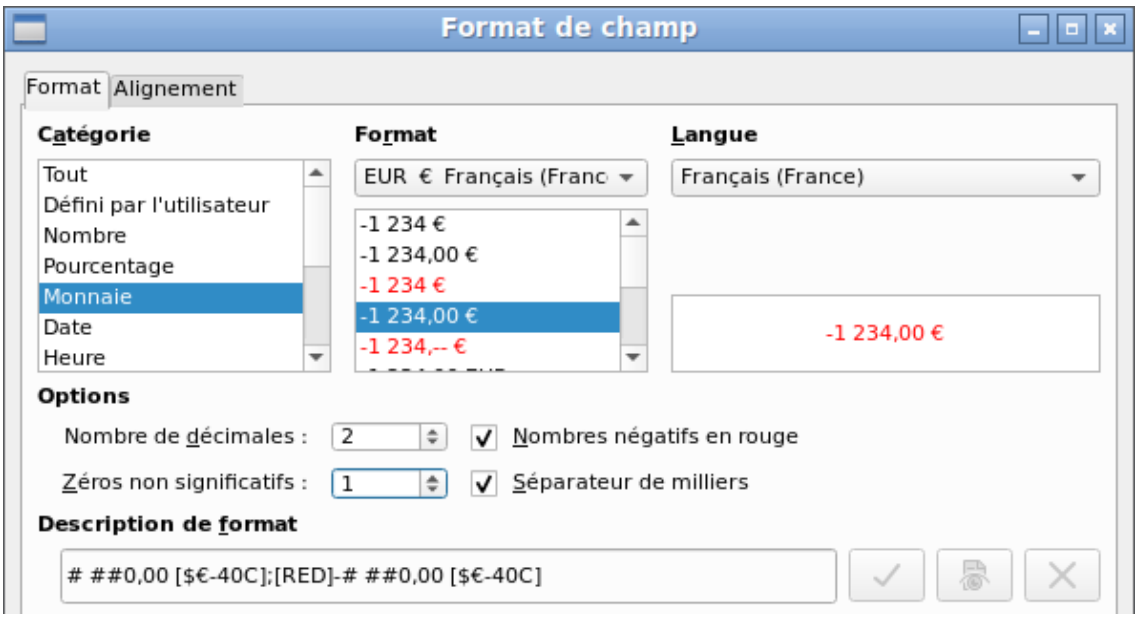
Le formatage ne fournit qu'une représentation du contenu. Une date représentée par un numéro d'année à deux caractères est toujours stockée sous la forme d'une année à quatre caractères. Si un champ est créé pour un nombre avec deux décimales, comme le montant des frais (appelé FraisDepassement) dans 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.

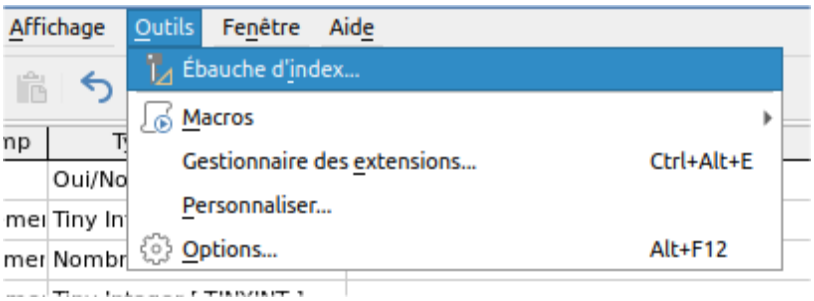


Figure 7: Accès à la conception d'index

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 7).

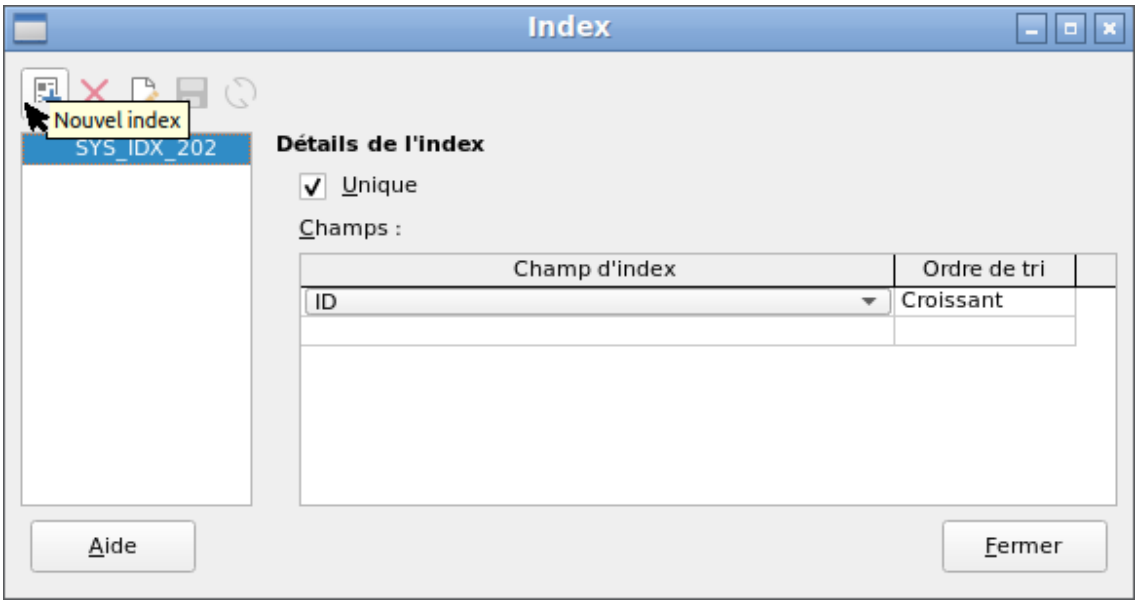


Figure 8: Créer un nouvel index

Dans la boîte de dialogue Index (Figure 8), 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.

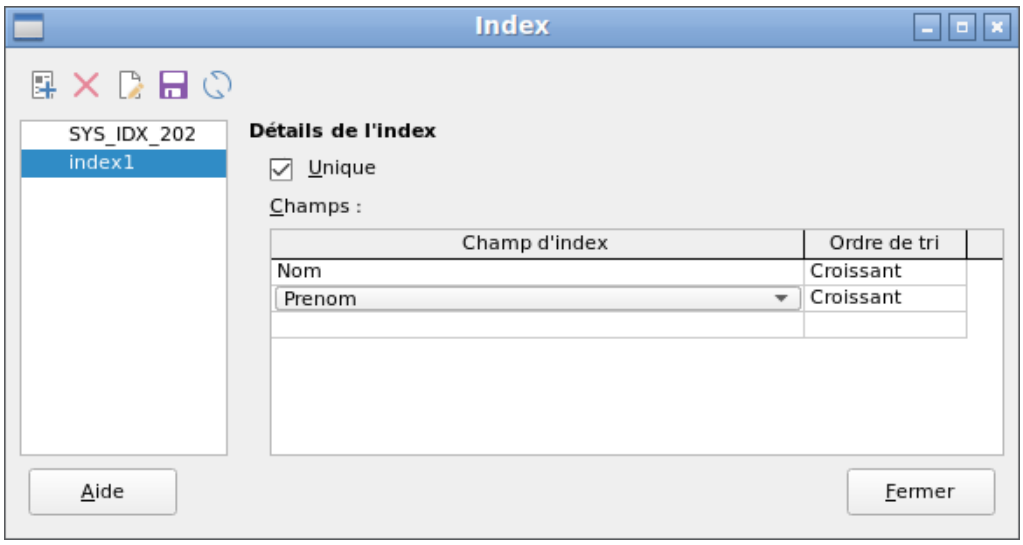


Figure 9: L'index est défini comme Unique

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 9, 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.

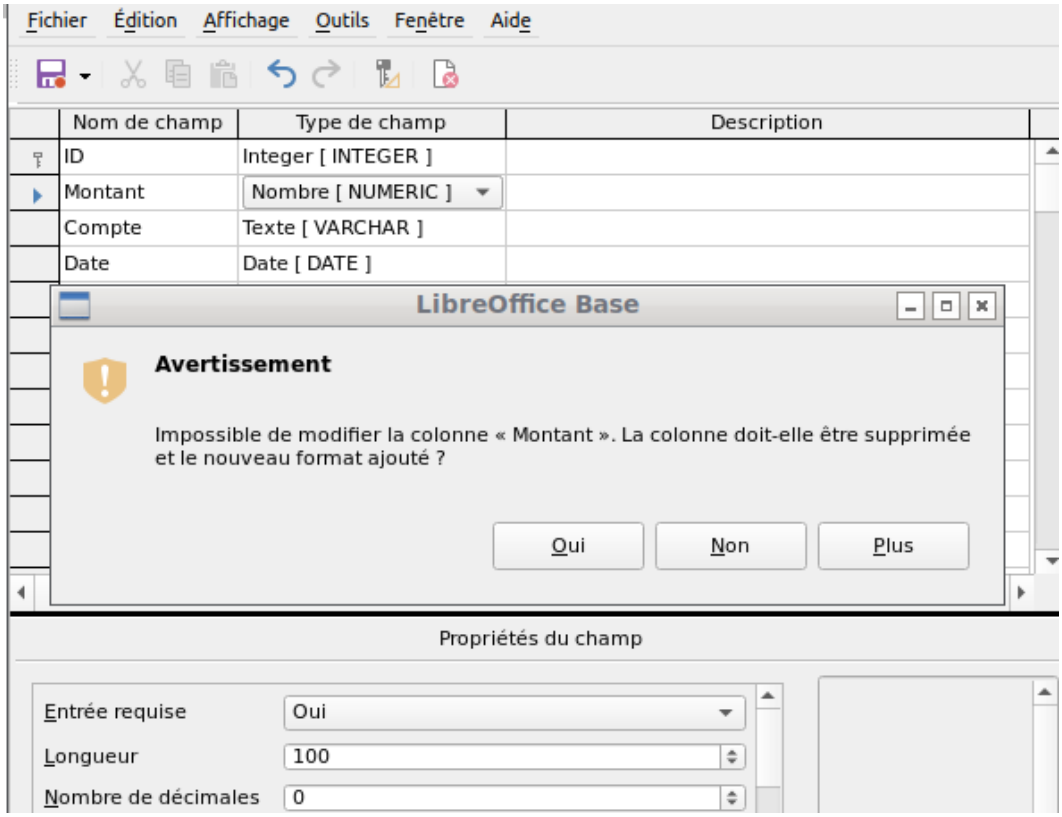
Note

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.

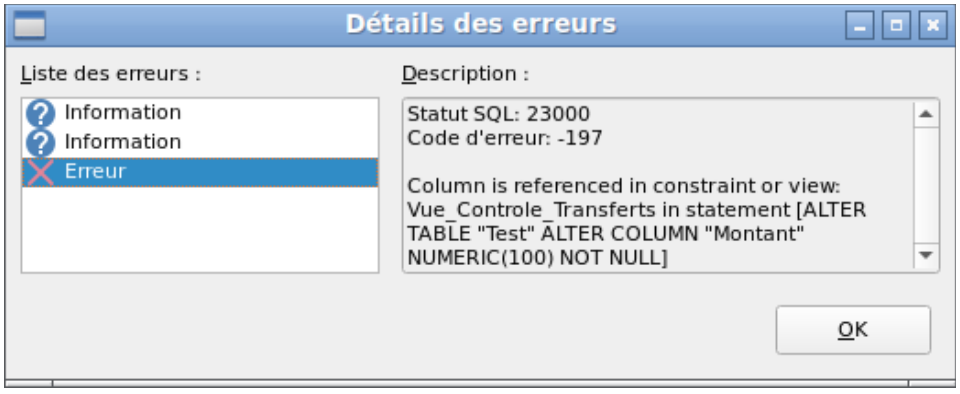
Problèmes lors de la modification des tables

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.

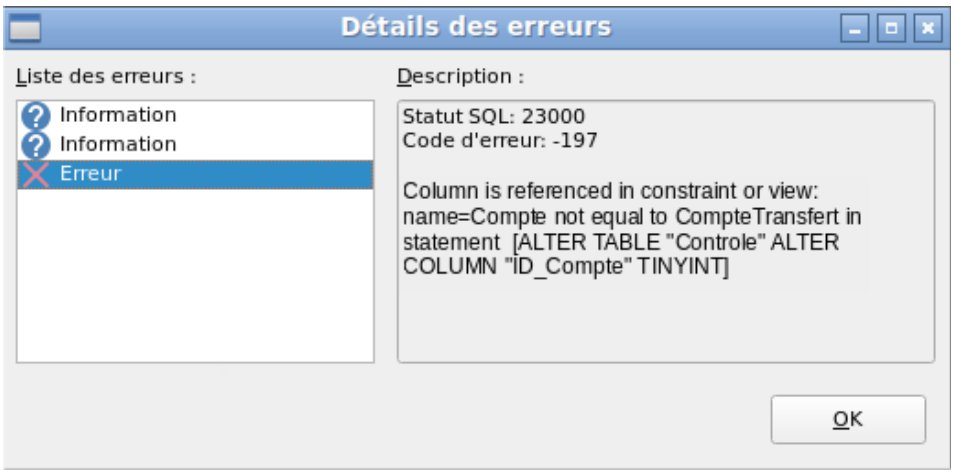


Dans ce cas, le champ Montant doit être réinitialisé sur Entrée requise = oui. Le symbole d'avertissement nous informe que ce changement peut entraîner une perte de données. Une simple modification n'est pas possible, car il se peut qu'il y ait déjà des enregistrements qui n'ont aucune entrée dans ce 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 10). 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 10: 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 table 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.

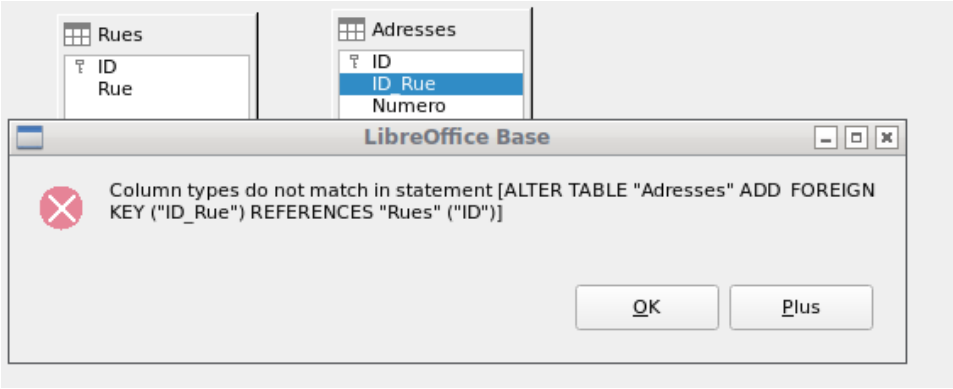
Lier des tables

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.

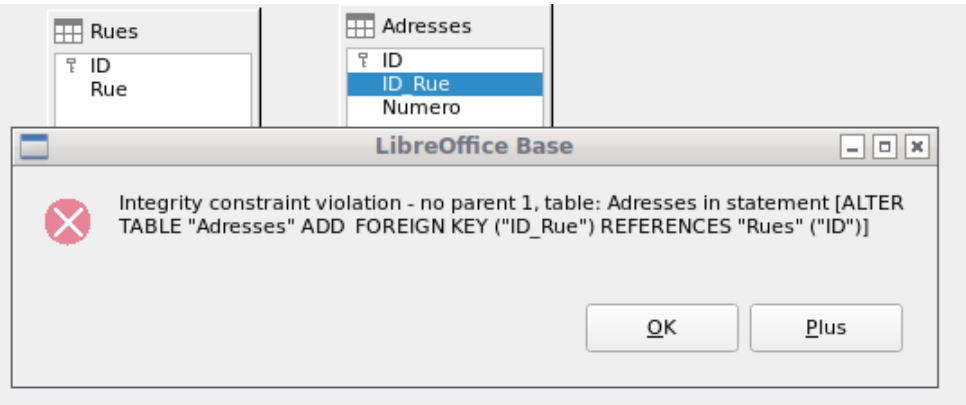
Les liens de base sont créés en utilisant **Outils > Relations**. Cela crée une ligne de connexion de la clé primaire dans une table à la clé étrangère définie dans l'autre.

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



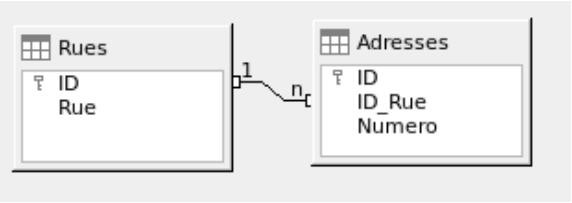
Ce message affiche l'erreur qui s'est produite et la commande SQL interne qui a provoqué l'erreur.

`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.

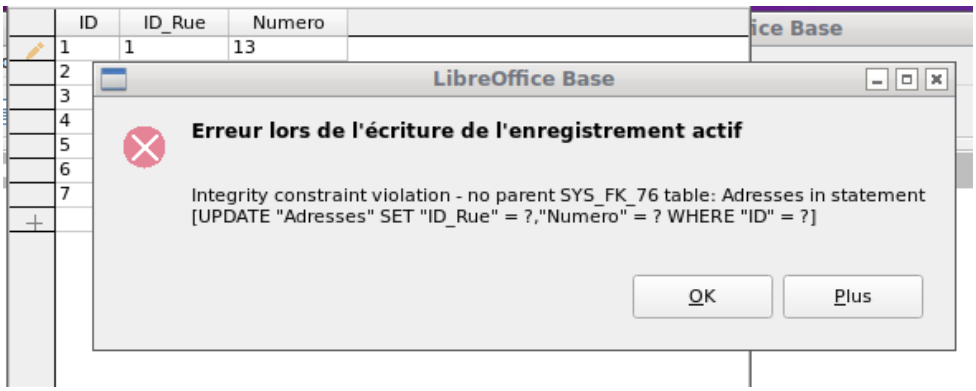


Dans ce cas, les types de colonnes correspondent. L'instruction SQL est la même que dans le premier exemple. Mais encore une fois, il y a une erreur :

`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.

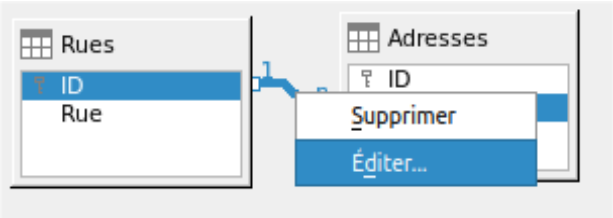


Figure 11: Lien peut être édité avec clic-droit

Les propriétés d'un lien peuvent être modifiées (Cf. Figure 11)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 12 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.

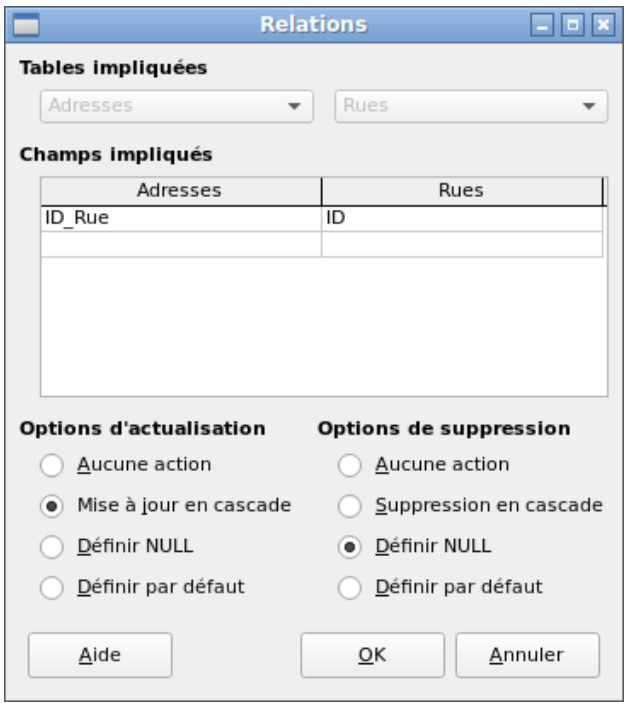


Figure 12: Edition des propritétés d'une relation

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.

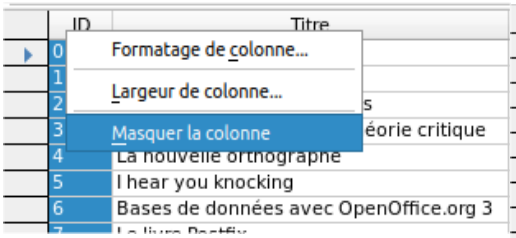


Figure 13: Saisie de table – Masquer colonne

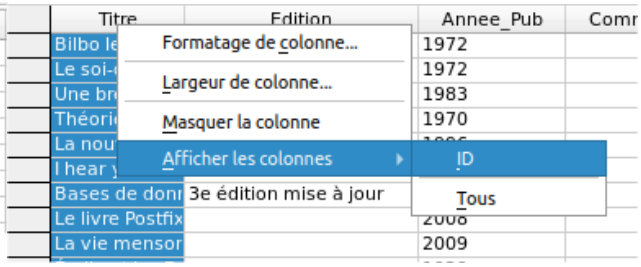


Figure 14: Saisie de table – Afficher colonne

Les colonnes individuelles de la vue des données de table peuvent être masquées (Cf.figure 13). 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 14). Il est également possible, comme avec Calc, de modifier la largeur d'une colonne avec les séparateurs de colonne.

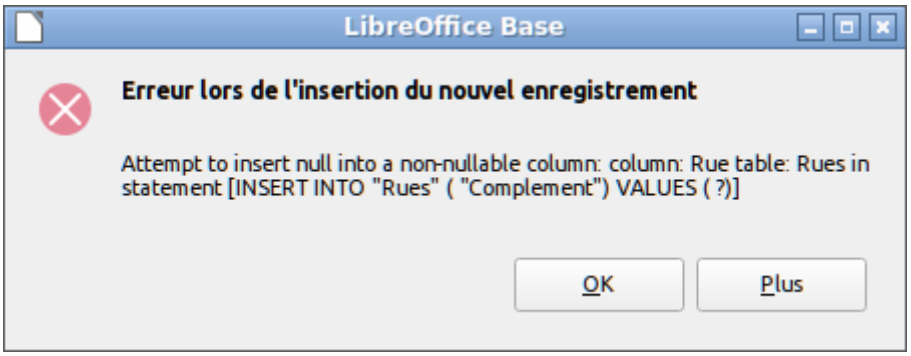
L'entrée dans le tableau s'effectue généralement de gauche à droite à l'aide du clavier avec les touches Tab ou Entrée. Vous pouvez également utiliser la souris.

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

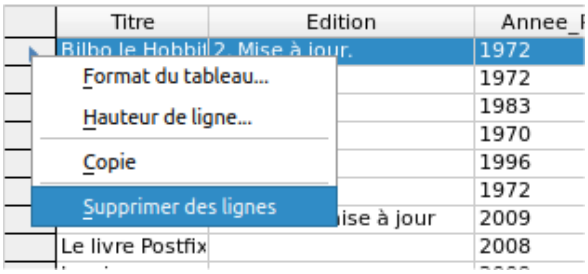
Pour HSQLDB, les données sont en mémoire de travail. elles ne seront transférée sur le disque dur qu'à la fermeture de Base, ou à l'enregistrement du fichier sur disque (malheureusement du point de vue de la sécurité des données). Si Base, pour une raison quelconque, ne ferme pas de la bonne manière,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



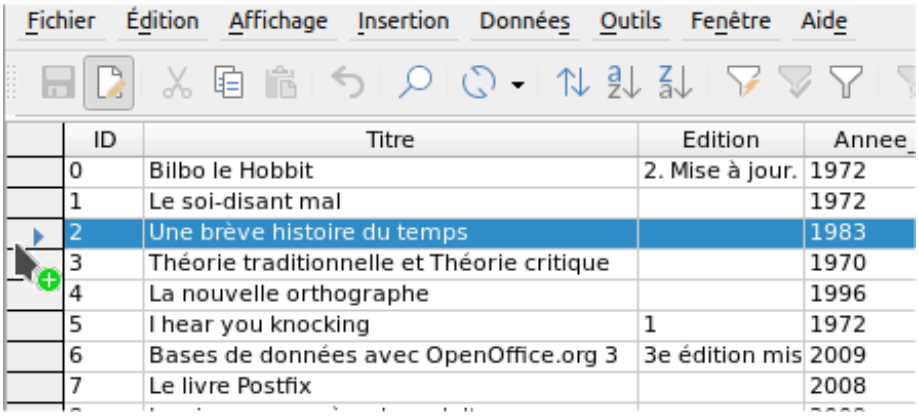
La colonne correspondante, la table et la commande SQL (telle que traduite par l'interface graphique) sont également affichées.

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.



	ID	Titre	Edition	Annee_
	0	Bilbo le Hobbit	2. Mise à jour.	1972
	1	Le soi-disant mal		1972
▸	2	Une brève histoire du temps		1983
	3	Théorie traditionnelle et Théorie critique		1970
	4	La nouvelle orthographe		1996
	5	I hear you knocking	1	1972
	6	Bases de données avec OpenOffice.org 3	3e édition mis	2009
	7	Le livre Postfix		2008

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.

	ID	Titre	Edition	Annee_F
	0	Bilbo le Hobbit	2. Mise à jour.	1972
	1	Le soi-disant mal		1972
▸	2	Une brève histoire du temps		1983
	3	Théorie traditionnelle et Théorie critique		1970
	4	La nouvelle orthographe		1996
	5	I hear you knocking	1	1972
	6	Bases de données avec OpenOffice.org 3	3e édition mis	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
	12	Une brève histoire du temps		1983
	<Auto			

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

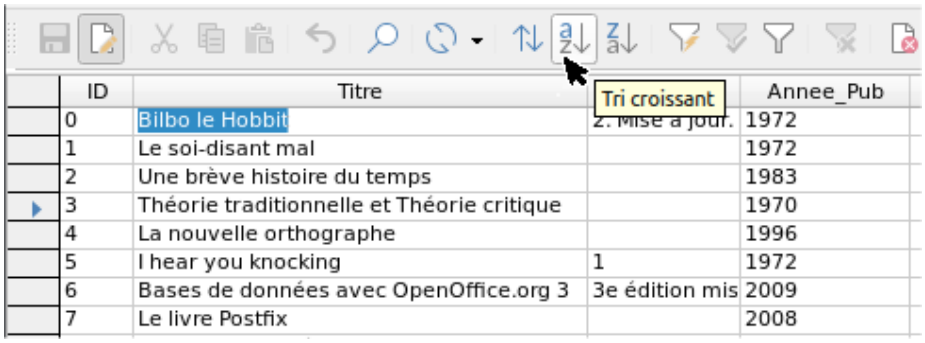


Figure 15: Tri rapide

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 15 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 16):

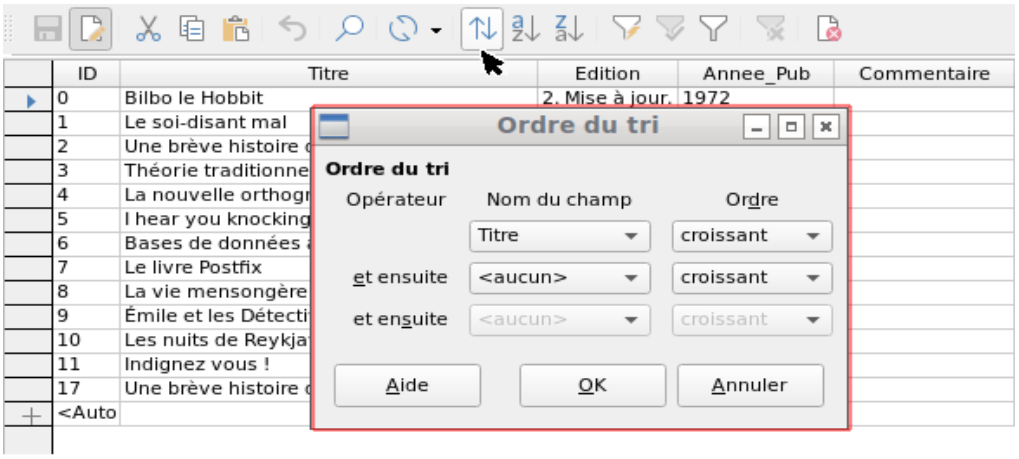
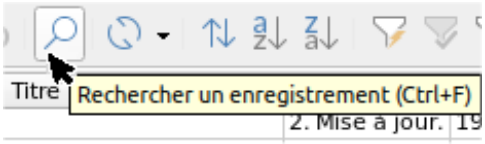


Figure 16: Tri sur plus d'une colonne

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 17 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.

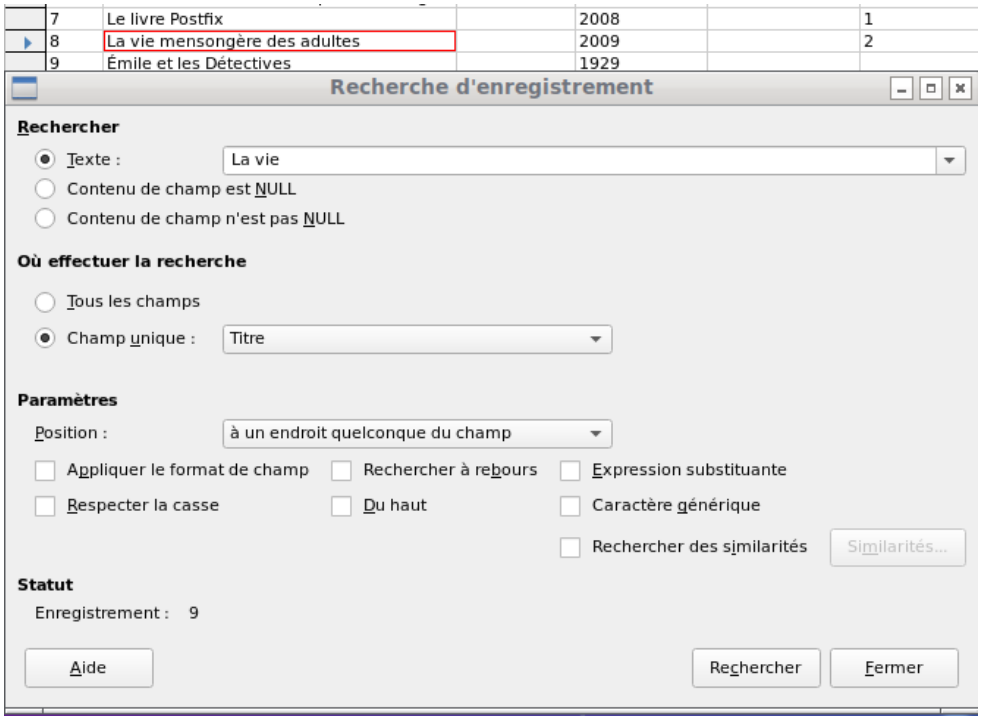


Figure 17: Masque d'entrée pour une recherche d'enregistrement

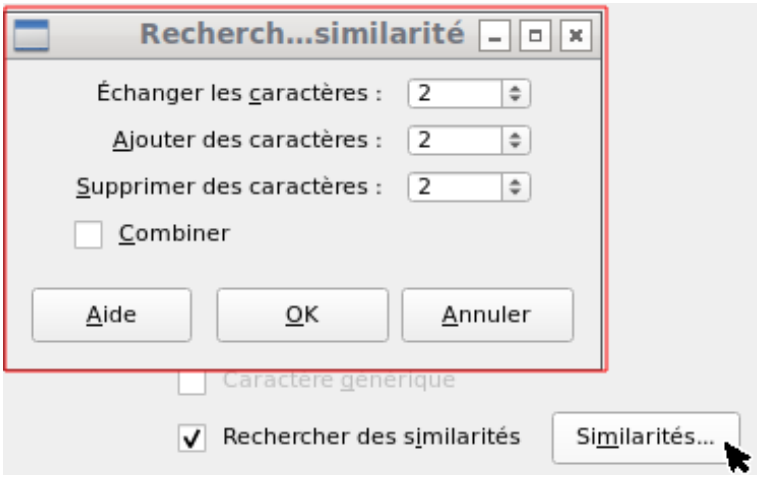


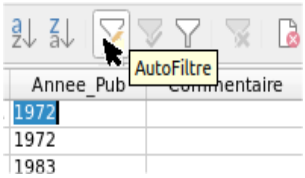
Figure 18: Limiter la recherche de similarité

La fonction de recherche de similarité figure 18est 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.

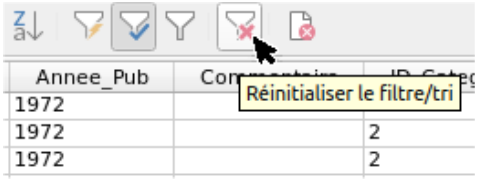
Ce module de recherche est le plus adapté aux personnes qui savent, grâce à une utilisation régulière, exactement comment obtenir un résultat donné. La plupart des utilisateurs sont plus susceptibles de réussir à trouver des enregistrements à l'aide d'un filtre.

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**.

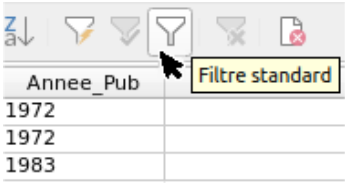


Le filtre est actif, comme indiqué par l'icône de filtre avec une coche verte. Le symbole du filtre est affiché enfoncé. Ce bouton est une bascule, donc si vous cliquez à nouveau, le filtre continue d'exister, mais tous les enregistrements sont maintenant affichés. Donc, si vous le souhaitez, vous pouvez 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 19: Filtrage par données multiples à l'aide du filtre standard

Le filtre standard fournit de nombreuses fonctions de filtrage des données SQL (Cf.figure 19). 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 19, 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 ;
```

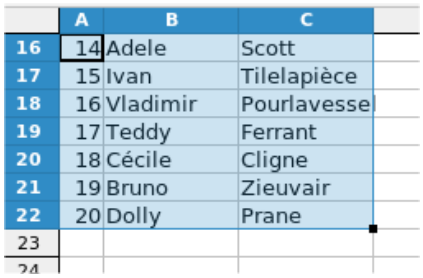
Importer des données à partir d'autres sources

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

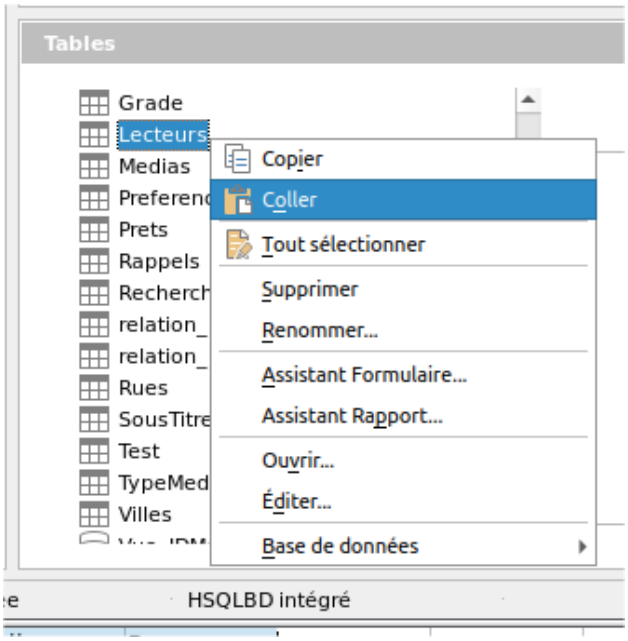
Pour importer des données à l'aide du presse-papiers, le format de données doit être lisible par Base. Ce sera toujours le cas pour les fichiers de données ouverts dans LibreOffice.

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			

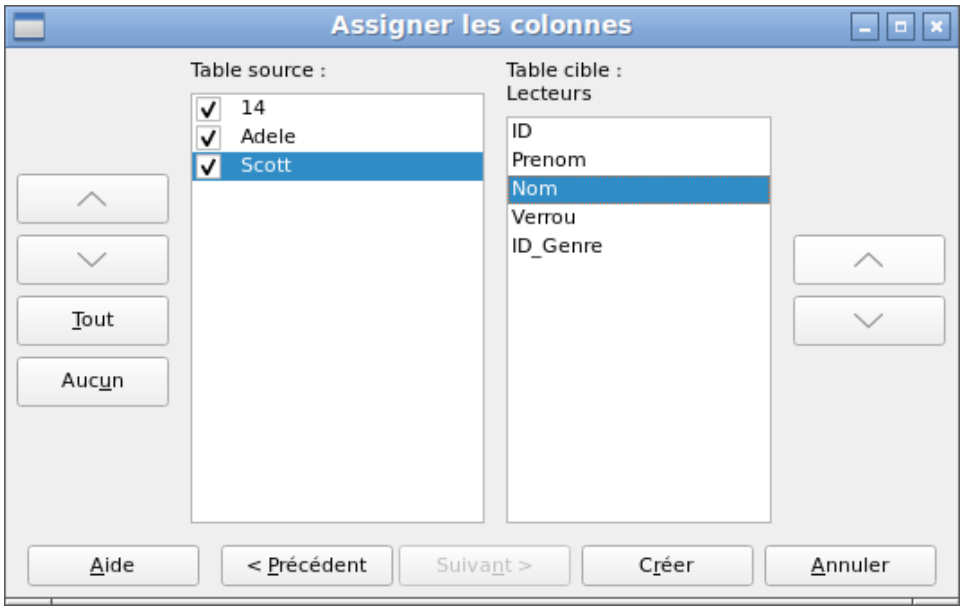
Ici, un petit exemple de tableau a été copié du tableur Calc dans le presse-papiers. Ensuite, il est collé dans le conteneur Table de la base. Bien sûr, cela aurait également pu être fait en le sélectionnant avec le bouton gauche de la souris, puis en le faisant glisser.



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 :

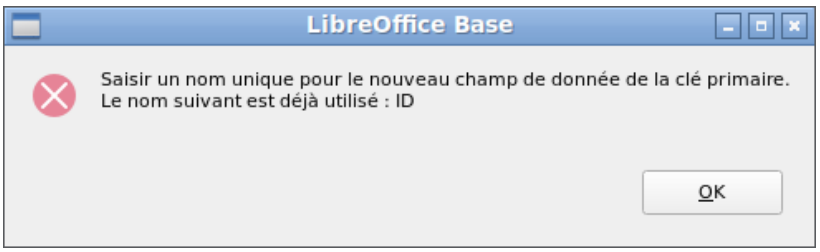
- Les champs de la table de destination nécessitent une entrée obligatoire, mais que la table source ne fournit aucune donnée pour eux.
- Les définitions de champ dans la table de destination ne sont pas cohérentes avec celles de la table source (par exemple, un nom doit être entré dans un champ numérique, 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.

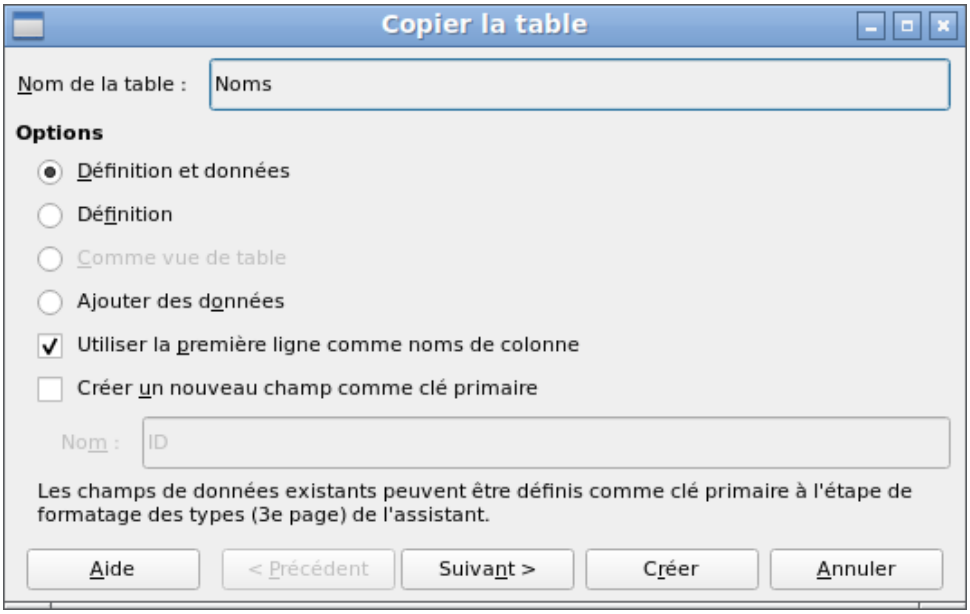
À ce stade, vous pouvez créer un nouveau champ supplémentaire pour une clé primaire. Le nom de ce champ ne doit pas exister en tant qu'en-tête de colonne dans la table Calc. Sinon, vous obtenez le message d'erreur :



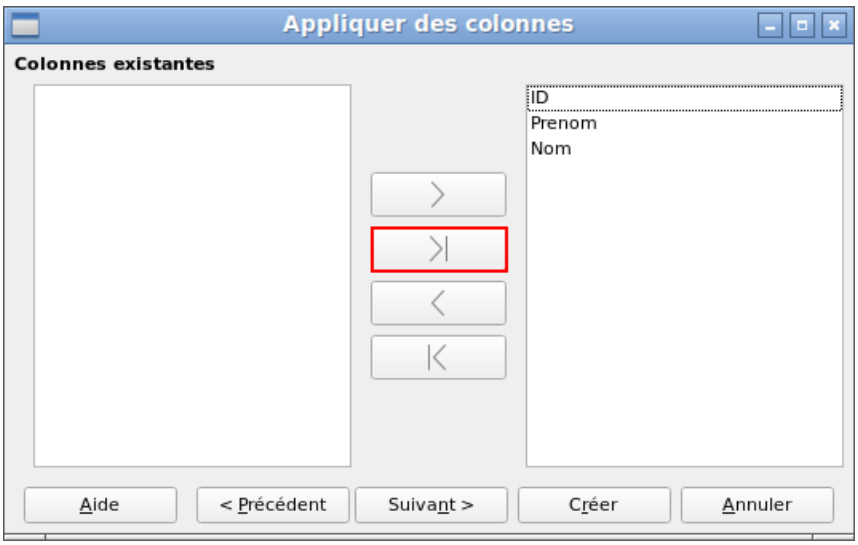
Malheureusement, ce message n'explique pas assez correctement la situation.

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.

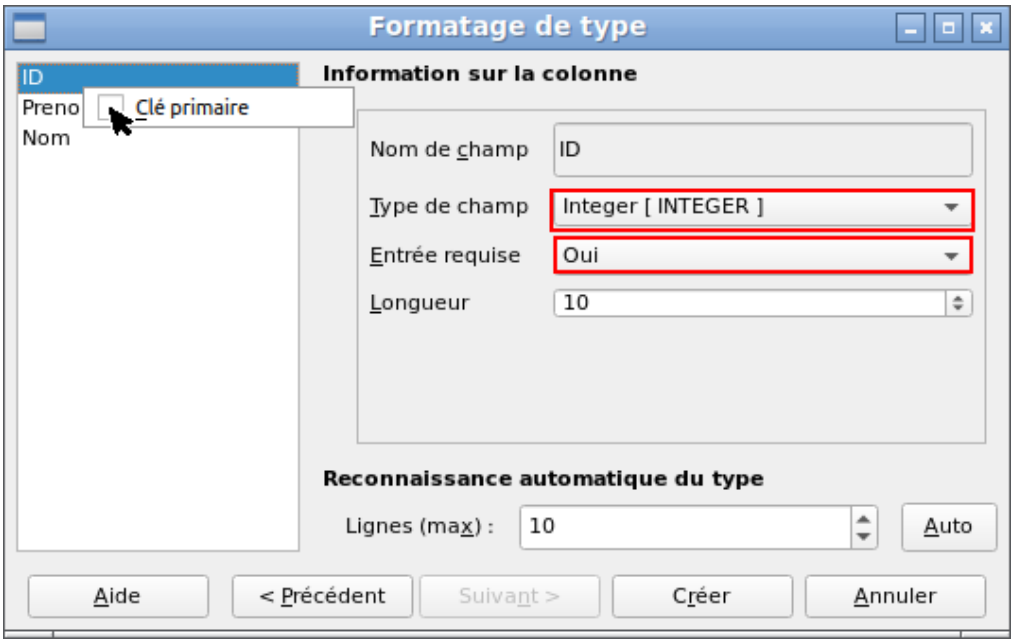
Lors de l'importation, la définition de la table et les données sont transférées.



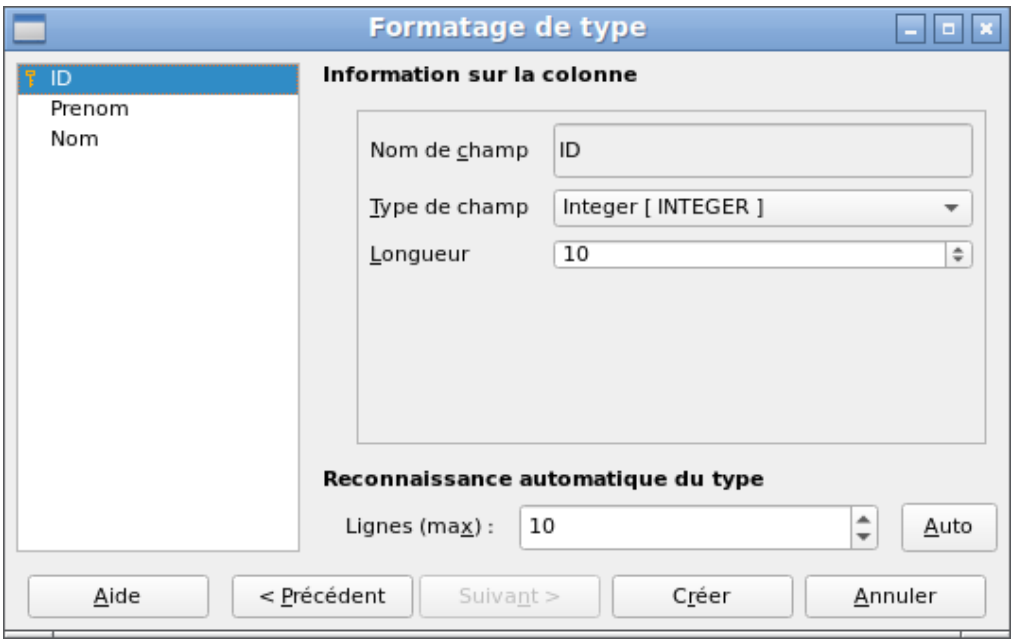
Toutes les colonnes disponibles sont transférées.



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.

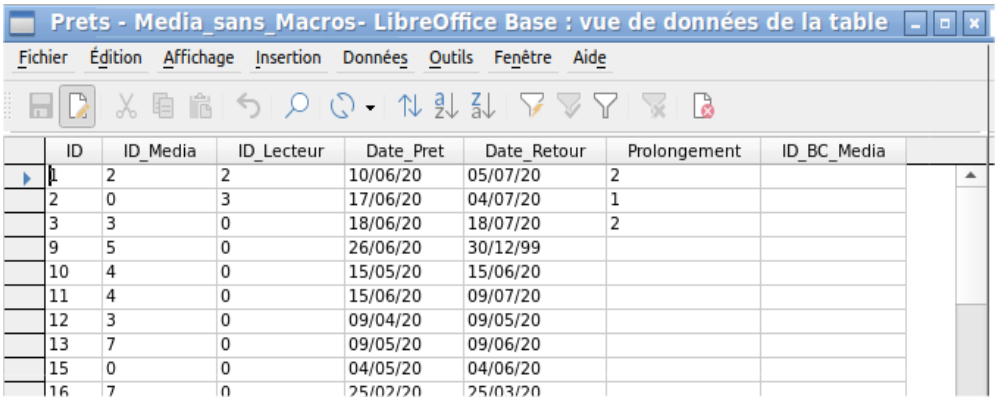
   Voici la requête:
   ```
   SELECT DISTINCT "CodePostal", "Ville" FROM "Adresses"
   ```
3) Un nouveau champ appelé ID_CodePostal est ajouté à la table Adresses.
4) À l'aide du menu **Outils > SQL**, une mise à jour est effectuée pour cette table:
   ```
   UPDATE "Adresses" AS "a" SET "a"."ID_CodePostal" = (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.