

# FileMaker® 16

Guide de référence SQL



**FileMaker®**  
An Apple Subsidiary

© 2013-2017, FileMaker, Inc. Tous droits réservés.

FileMaker, Inc.  
5201 Patrick Henry Drive  
Santa Clara, Californie 95054

FileMaker, FileMaker Go et le logo en forme de dossier sont des marques de FileMaker, Inc. déposées aux États-Unis et dans d'autres pays. FileMaker WebDirect and FileMaker Cloud sont des marques de FileMaker, Inc. Toutes les autres marques sont la propriété de leurs détenteurs respectifs.

La documentation de FileMaker est protégée par la législation sur les droits d'auteur. Vous n'êtes pas autorisé à créer des copies supplémentaires ni à distribuer cette documentation sans l'accord écrit de FileMaker. Vous devez posséder une copie sous licence valide de FileMaker pour utiliser cette documentation.

Toutes les personnes, sociétés, adresses email et URL citées dans les exemples sont fictives et toute ressemblance avec des personnes, des sociétés, des adresses email ou des URL existantes ne serait que pure coïncidence. La liste des auteurs est disponible dans les documents Remerciements fournis avec ce logiciel. Les produits tiers et les adresses URL sont mentionnés à titre indicatif uniquement, et non pas à titre de recommandation. FileMaker, Inc. se dégage de toute responsabilité concernant les performances de ces produits.

Pour plus d'informations, consultez notre site Web à l'adresse <http://www.filemaker.com/fr/>.

Edition : 01

# Table des matières

## Chapitre 1

### *Introduction*

A propos de ce guide	5
A propos de SQL	5
Utilisation d'une base de données FileMaker comme source de données	5
Utilisation de la fonction ExecuterSQL	6

## Chapitre 2

### *Normes prises en charge*

Prise en charge des caractères Unicode	7
Instructions SQL	7
SELECT, instruction	8
Clauses SQL	9
Clause FROM	9
Clause WHERE	11
Clause GROUP BY	11
Clause HAVING	12
Opérateur UNION	12
Clause ORDER BY	13
Clauses OFFSET et FETCH FIRST	13
Clause FOR UPDATE	14
Instruction DELETE	17
Instruction INSERT	17
Instruction UPDATE	19
Instruction CREATE TABLE	21
Instruction TRUNCATE TABLE	22
Instruction ALTER TABLE	23
Instruction CREATE INDEX	23
Instruction DROP INDEX	24
Expressions SQL	24
Noms de rubriques	25
Constantes	25
Notification en virgule flottante/scientifique	26
Opérateurs numériques	26
Opérateurs de caractères	26
Opérateurs de dates	27
Opérateurs relationnels	27
Opérateurs logiques	29
Ordre de priorité des opérateurs	29

Fonctions SQL	30
Fonctions statistiques	30
Fonctions qui renvoient des chaînes de caractères	31
Fonctions qui renvoient des nombres	33
Fonctions qui renvoient des dates	35
Fonctions conditionnelles	36
Objets système FileMaker	37
Tables système FileMaker	37
Colonnes système FileMaker	38
Mots-clés SQL réservés	39
<i><b>Index</b></i>	42

# Chapitre 1

## Introduction

En tant que développeur de bases de données, vous pouvez utiliser FileMaker Pro pour créer des solutions de bases de données sans connaître le langage SQL. En revanche, si vous connaissez le langage SQL, vous pouvez utiliser un fichier de base de données FileMaker comme source de données ODBC ou JDBC, en partageant vos données avec d'autres applications à l'aide d'ODBC et de JDBC. Vous pouvez également utiliser la fonction ExecuterSQL de FileMaker Pro pour récupérer des données de n'importe quelle occurrence de table dans une base de données FileMaker Pro.

Le présent guide de référence décrit les instructions et les normes SQL prises en charge par FileMaker. Les pilotes clients ODBC et JDBC de FileMaker prennent en charge toutes les instructions SQL décrites dans ce guide de référence. La fonction ExecuterSQL de FileMaker Pro prend uniquement en charge l'instruction SELECT.

### A propos de ce guide

- Pour plus d'informations sur l'utilisation d'ODBC et de JDBC avec les versions antérieures de FileMaker Pro, consultez le [Centre de documentation produit](#).
- Ce guide de référence suppose que vous connaissez bien les principes d'utilisation de base des fonctions de FileMaker Pro, le codage des applications ODBC et JDBC et l'élaboration de requêtes SQL. Pour plus d'informations sur ces sujets, consultez un livre spécialisé.
- Ce guide de référence mentionne « FileMaker Pro » pour faire référence à FileMaker Pro et à FileMaker Pro Advanced, sauf quand il décrit des fonctions propres à FileMaker Pro Advanced.

### A propos de SQL

SQL, ou Structured Query Language, est un langage de programmation permettant d'interroger des données dans une base de données relationnelle. La principale instruction utilisée pour interroger une base de données est l'instruction SELECT.

Outre le langage permettant d'interroger une base de données, SQL fournit des instructions afin de manipuler des données pour ajouter, mettre à jour et supprimer des données.

SQL fournit également des instructions pour définir des données. Ces instructions vous permettent de créer et de modifier des tables et des index.

Les instructions et normes SQL prises en charge par FileMaker sont décrites dans chapitre 2, «Normes prises en charge».

### Utilisation d'une base de données FileMaker comme source de données

Lorsque vous hébergez une base de données FileMaker en tant que source de données ODBC ou JDBC, les données FileMaker peuvent être partagées avec des applications compatibles avec ODBC et JDBC. Les applications se connectent à la source de données FileMaker à l'aide du pilote client FileMaker, crée et exécute des requêtes SQL à l'aide d'ODBC ou de JDBC et traite les données récupérées dans la solution de base de données FileMaker.

Pour plus d'informations sur la manière dont vous pouvez utiliser le logiciel FileMaker en tant que source de données pour les applications ODBC et JDBC, consultez le [Guide ODBC et JDBC de FileMaker](#).

Les pilotes clients ODBC et JDBC de FileMaker prennent en charge toutes les instructions SQL décrites dans ce guide de référence.

## Utilisation de la fonction ExecuterSQL

La fonction ExecuterSQL de FileMaker Pro vous permet de récupérer des données dans les occurrences de tables dont le nom est affiché dans le graphe de liens, mais indépendamment de toute relation définie. Vous pouvez récupérer des données dans plusieurs tables sans créer de lien de table ou une quelconque relation entre les tables. Dans certains cas, vous pouvez réduire la complexité de votre graphe de liens grâce à la fonction ExecuterSQL.

Les rubriques que vous interrogez avec la fonction ExecuterSQL ne doivent pas nécessairement figurer sur un modèle, vous pouvez donc utiliser cette fonction pour récupérer des données indépendamment de tout contexte de modèle. En raison de cette indépendance du contexte, l'utilisation de la fonction ExecuterSQL dans des scripts peut améliorer la portabilité des scripts. Vous pouvez utiliser cette fonction partout où vous pouvez spécifier des calculs, notamment la création de graphiques et de rapports.

Elle prend uniquement en charge l'instruction SELECT, décrite dans la section « SELECT, instruction », page 8.

Aussi, cette fonction n'accepte que les formats de date et d'heure ISO syntaxe SQL-92 sans accolades ({}). Elle n'accepte pas les constantes de date, d'heure et d'horodatage au format ODBC/JDBC entre accolades.

Pour plus d'informations sur la syntaxe et l'utilisation de la fonction ExecuterSQL, consultez l'[aide FileMaker Pro](#).

# Chapitre 2

## Normes prises en charge

Les pilotes clients ODBC et JDBC de FileMaker permettent d'accéder à une solution de base de données FileMaker à partir d'une application compatible ODBC ou JDBC. Cette solution peut être hébergée par FileMaker Pro ou FileMaker Server

- Le pilote client ODBC prend en charge ODBC 3.0 niveau 1.
- Le pilote client JDBC prend partiellement en charge la spécification JDBC 3.0.
- Les pilotes clients ODBC et JDBC sont compatibles avec le niveau d'entrée de SQL-92, ainsi qu'avec certaines fonctions SQL-92 intermédiaires.

### Prise en charge des caractères Unicode

Les pilotes clients ODBC et JDBC prennent en charge l'API Unicode. Toutefois, si vous créez une application personnalisée qui utilise les pilotes clients, utilisez le format ASCII pour les noms de rubriques, de tables et de fichiers (si vous utilisez un outil de requête ou une application non Unicode).

**Remarque** Pour insérer et extraire des données Unicode, utilisez `SQL_C_WCHAR`.

### Instructions SQL

Les pilotes clients ODBC et JDBC prennent en charge les instructions SQL suivantes :

- SELECT (page 8)
- DELETE (page 17)
- INSERT (page 17)
- UPDATE (page 19)
- CREATE TABLE (page 21)
- TRUNCATE TABLE (page 22)
- ALTER TABLE (page 23)
- CREATE INDEX (page 23)
- DROP INDEX (page 24)

Les pilotes clients prennent aussi en charge la conversion du type de données FileMaker en types de données ODBC SQL et JDBC SQL. Pour plus d'informations sur les conversions de types de données, consultez le [Guide ODBC et JDBC de FileMaker](#). Pour plus d'informations sur la création de requêtes SQL, consultez un ouvrage spécialisé.

**Remarque** Les pilotes clients ODBC et JDBC ne prennent pas en charge les tables externes FileMaker.

## SELECT, instruction

L'instruction `SELECT` vous permet de spécifier les colonnes qui vous intéressent. Faites suivre l'instruction `SELECT` des expressions de colonne (comparables aux noms de rubriques) que vous voulez extraire (par exemple, `nom_famille`). Les expressions peuvent contenir des opérations mathématiques ou des instructions de manipulation de chaînes de texte (par exemple, `SALAIRE * 1,05`).

L'instruction `SELECT` peut être accompagnée de diverses clauses :

```
SELECT [DISTINCT] { * | expression_colonne [[AS] alias_colonne], ... }
FROM nom_table [alias_table], ...
[ WHERE expr1 opérateur_rel expr2 ]
[ GROUP BY {expression_colonne, ...} ]
[ HAVING expr1 opérateur_rel expr2 ]
[ UNION [ALL] (SELECT...) ]
[ ORDER BY {expression_tri [DESC | ASC]}, ... ]
[ OFFSET n {ROWS | ROW} ]
[ FETCH FIRST [ n [ PERCENT ] ] { ROWS | ROW } { ONLY | WITH TIES } ]
[ FOR UPDATE [OF {expression_colonne, ...}] ]
```

Les éléments entre crochets sont facultatifs.

`alias_colonne` peut servir à attribuer à la colonne un nom plus descriptif ou à abrégé un nom de colonne long.

### Exemple

Attribuer l'alias `service` à la colonne `serv`.

```
SELECT serv AS service FROM emp
```

Vous pouvez faire précéder le nom des rubriques par le nom ou de l'alias de la table. Par exemple, `EMP.NOM_FAMILLE` ou `E.NOM_FAMILLE`, où `E` est l'alias de la table `EMP`.

Vous pouvez faire précéder la première expression de colonne de l'opérateur `DISTINCT`. Cet opérateur supprime les rangées en double du résultat d'une requête.

### Exemple

```
SELECT DISTINCT serv FROM emp
```

## Clauses SQL

Les pilotes clients ODBC et JDBC prennent en charge les clauses SQL suivantes.

Utilisez cette clause SQL	Pour
FROM (page 9)	Indiquer les tables utilisées dans l'instruction <code>SELECT</code> .
WHERE (page 11)	Indiquer les conditions que doivent remplir les enregistrements à extraire (comme dans une recherche sous FileMaker Pro).
GROUP BY (page 11)	Indiquer le nom d'une ou de plusieurs rubriques en fonction desquelles les valeurs renvoyées doivent être groupées. Cette clause sert à renvoyer un ensemble de valeurs statistiques en renvoyant une rangée par groupe (comme dans un sous-récapitulatif FileMaker Pro).
HAVING (page 12)	Indiquer les conditions s'appliquant à des groupes d'enregistrements (par exemple, afficher uniquement les services dont le total des salaires est supérieur à 200 000 euros).
UNION (page 12)	Combiner les résultats de deux instructions <code>SELECT</code> ou plus en un seul résultat.
ORDER BY (page 13)	Indiquer le mode de tri appliqué aux enregistrements.
OFFSET (page 13)	Indiquer le nombre de rangées à ignorer avant de commencer la récupération de rangées.
FETCH FIRST (page 13)	Indiquer le nombre de rangées à récupérer. Le nombre maximum de rangées spécifiées est renvoyé, il est possible que moins de rangées soient renvoyées si la requête fournit un nombre inférieur de rangées spécifiées.
FOR UPDATE (page 14)	Procéder à des mises à jour positionnées ou à des suppressions positionnées via les curseurs SQL

**Remarque** Si vous essayez de récupérer des données depuis une table sans colonne, l'instruction `SELECT` ne renvoie rien.

## Clause FROM

La clause `FROM` indique quelles tables sont utilisées dans l'instruction `SELECT`. La syntaxe de cette clause est la suivante :

```
FROM nom_table [alias_table], [, nom_table [alias_table]]
```

`nom_table` correspond au nom d'une table dans la base de données active. Le nom de la table doit commencer par un caractère alphabétique. Si le nom de la table commence par un caractère autre qu'un caractère alphabétique, placez-le entre des guillemets doubles (identifiant cité).

`alias_table` peut être utilisé pour attribuer à la table un nom plus descriptif, pour abrégé un nom de table plus long ou pour inclure la même table dans la requête à plusieurs reprises (par exemple dans les liens internes).

Les noms des rubriques commencent par un caractère alphabétique. Si le nom d'une rubrique commence par un caractère autre qu'un caractère alphabétique, placez-le entre des guillemets doubles (identifiant cité).

## Exemple

L'instruction `ExecuterSQL` de la rubrique intitulée `_NOMFAMILLE` est :

```
SELECT "_NOMFAMILLE" from emp
```

Vous pouvez faire précéder le nom des rubriques par le nom ou de l'alias de la table.

### Exemple

A partir de la spécification de table `FROM employés E`, vous pouvez faire référence à la rubrique `NOM_FAMILLE` sous la forme `E.NOM_FAMILLE`. Les alias de tables doivent être utilisés si l'instruction `SELECT` joint une table à elle-même.

```
SELECT * FROM employés E, employés F WHERE E.id_directeur = F.id_employé
```

Le signe égal (=) n'inclut que les rangées correspondantes dans le résultat.

Si vous joignez plusieurs tables et que vous souhaitez ignorer toutes les rangées qui n'ont pas de rangées correspondantes dans les deux tables sources, vous pouvez utiliser une instruction `INNER JOIN`.

### Exemple

```
SELECT *  
FROM Vendeurs INNER JOIN Informations_Ventes  
ON Vendeurs.ID_Vendeur = Informations_Ventes.ID_Vendeur
```

Si vous liez deux tables, mais ne souhaitez pas ignorer les rangées de la première table (table de «gauche»), vous pouvez utiliser `LEFT OUTER JOIN`.

### Exemple

```
SELECT *  
FROM Vendeurs LEFT OUTER JOIN Informations_Ventes  
ON Vendeurs.ID_Vendeur = Informations_Ventes.ID_Vendeur
```

Chaque rangée de la table « Vendeurs » apparaît dans la table liée.

### Remarques

- `RIGHT OUTER JOIN` n'est pas pris en charge actuellement.
- `FULL OUTER JOIN` n'est pas pris en charge actuellement.

## Clause WHERE

La clause `WHERE` indique les conditions que les enregistrements doivent remplir pour être extraits. La clause `WHERE` spécifie ces conditions sous la forme suivante :

```
WHERE expr1 opérateur_rel expr2
```

`expr1` et `expr2` peuvent être des noms de rubriques, des valeurs constantes ou des expressions.

`opérateur_rel` est l'opérateur relationnel qui lie les deux expressions.

### Exemple

Extraire le nom des employés dont le salaire est supérieur ou égal à 20 000 euros.

```
SELECT nom_famille,prénom FROM emp WHERE salaire >= 20000
```

La clause `WHERE` peut également utiliser des expressions telles que les suivantes :

```
WHERE expr1 IS NULL
```

```
WHERE NOT expr2
```

**Remarque** Si vous utilisez des noms entièrement qualifiés dans la liste (de projection) `SELECT`, vous devez également utiliser les noms entièrement qualifiés dans la clause `WHERE` liée.

## Clause GROUP BY

La clause `GROUP BY` indique les noms d'une ou de plusieurs rubriques devant servir à grouper les valeurs renvoyées. Cette clause sert à renvoyer un jeu de valeurs statistiques. La syntaxe de cette clause est la suivante :

```
GROUP BY colonnes
```

La portée de la clause `GROUP BY` est l'expression de table dans la clause `FROM`. Par conséquent, les expressions de colonne spécifiées par les `colonnes` doivent provenir des tables spécifiées dans la clause `FROM`. Une expression de colonne peut être composée d'un ou de plusieurs noms de rubriques de la table de base de données, séparés par des virgules.

### Exemple

Faire la somme des salaires de chaque service.

```
SELECT id_serv, SUM (salaire) FROM emp GROUP BY id_serv
```

Cette instruction renvoie une rangée pour chaque ID de service distinct. Chaque rangée contient l'ID de service et calcule la somme des salaires des employés du service.

## Clause HAVING

La clause `HAVING` vous permet de spécifier des conditions pour des groupes d'enregistrements (par exemple, afficher uniquement les services dont le total des salaires est supérieur à 200 000 euros). La syntaxe de cette clause est la suivante :

```
HAVING expr1 opérateur_rel expr2
```

`expr1` et `expr2` peuvent être des noms de rubriques, des valeurs constantes ou des expressions. Ces expressions n'ont pas besoin de correspondre à une expression de colonne dans la clause `SELECT`.

`opérateur_rel` est l'opérateur relationnel qui lie les deux expressions.

### Exemple

Renvoyer uniquement les services dont la somme des salaires est supérieure à 200 000 euros.

```
SELECT id_serv, SUM (salaire) FROM emp
      GROUP BY id_serv HAVING SUM (salaire) > 200000
```

## Opérateur UNION

L'opérateur `UNION` combine les résultats de deux instructions `SELECT` ou plus en un seul et même résultat. Ce résultat correspond à l'ensemble des enregistrements renvoyés par les instructions `SELECT`. Par défaut, le système ne renvoie pas les enregistrements en double. Si vous les voulez également, employez le mot-clé `ALL` (`UNION ALL`). La syntaxe de cette clause est la suivante :

```
instruction_SELECT UNION [ALL] instruction_SELECT
```

Avec l'opérateur `UNION`, les listes de sélection de chaque instruction `SELECT` doivent avoir le même nombre d'expressions de colonne, avec les mêmes types de données et le même ordre.

### Exemple

```
SELECT nom_famille, salaire, date_embauche FROM emp UNION SELECT nom, paie,
date_naissance FROM personne
```

L'exemple suivant n'est pas valide car les types de données des expressions de colonne sont différents (la rubrique `SALAIRE` de la table `EMP` n'emploie pas le même type de données que la rubrique `NOM_FAMILLE` de la table `AUGMENTATIONS`). Cet exemple utilise le même nombre d'expressions de colonne dans chaque instruction `SELECT`, mais ces expressions ne sont pas dans le même ordre par type de données.

### Exemple

```
SELECT nom_famille, salaire FROM emp UNION SELECT salaire, nom_famille FROM
augmentations
```

## Clause ORDER BY

La clause `ORDER BY` indique comment les enregistrements doivent être triés. Si votre instruction `SELECT` n'inclut pas de clause `ORDER BY`, les enregistrements peuvent être renvoyés dans n'importe quel ordre.

La syntaxe de cette clause est la suivante :

```
ORDER BY {expression_tri [DESC | ASC]}, ...
```

`expression_tri` peut être le nom de la rubrique ou le numéro de position de l'expression de colonne à utiliser. Par défaut, le tri est croissant (`ASC`).

### Exemples

Trier par `nom_famille`, puis par `prénom`.

```
SELECT id_emp, nom_famille, prénom FROM emp ORDER BY nom_famille, prénom
```

Le deuxième exemple utilise les numéros de position 2 et 3 pour obtenir le même ordre que l'exemple précédent qui spécifiait explicitement `nom_famille` et `prénom`.

```
SELECT id_emp, nom_famille, prénom FROM emp ORDER BY 2,3
```

**Remarque** FileMaker SQL utilise l'ordre de tri binaire Unicode, qui est différent de l'ordre de tri FileMaker Pro utilisé avec le tri d'après la langue ou de l'ordre de tri indépendant de la langue par défaut.

## Clauses OFFSET et FETCH FIRST

Les clauses `OFFSET` et `FETCH FIRST` permettent de renvoyer une plage de rangées en commençant par un point de départ donné dans un ensemble de résultats. La capacité à limiter les rangées récupérées dans des ensembles de résultats étendus vous permet de passer les données en revue et d'améliorer l'efficacité.

La clause `OFFSET` indique le nombre de rangées à ignorer avant de commencer à renvoyer des données. Si la clause `OFFSET` n'est pas utilisée dans une instruction `SELECT`, la rangée de départ sera 0. La clause `FETCH FIRST` indique le nombre de rangées à renvoyer, sous la forme d'un entier non signé supérieur ou égal à 1 ou sous la forme d'un pourcentage, depuis le point de départ indiqué dans la clause `OFFSET`. Si les deux clauses `OFFSET` et `FETCH FIRST` sont utilisées dans une instruction `SELECT`, la clause `OFFSET` doit être prioritaire.

Les clauses `OFFSET` et `FETCH FIRST` ne sont pas prises en charge dans les sous-requêtes.

### Format OFFSET

Le format `OFFSET` est le suivant :

```
OFFSET n {ROWS | ROW} ]
```

`n` est un entier non signé. Si `n` est supérieur au nombre de rangées renvoyées dans l'ensemble de résultats, alors rien ne sera renvoyé et aucun message d'erreur n'apparaîtra.

`ROWS` est l'équivalent de `ROW`.

## Format FETCH FIRST

Le format `FETCH FIRST` est le suivant :

```
FETCH FIRST [ n [ PERCENT ] ] { ROWS | ROW } { ONLY | WITH TIES } ]
```

`n` indique le nombre de rangées à renvoyer. La valeur par défaut est 1 si `n` est exclu .

`n` est un entier non signé supérieur ou égal à 1 à moins qu'il soit suivi par `PERCENT`. Si `n` est suivi par `PERCENT`, la valeur peut être une valeur fractionnelle positive ou un entier non signé.

`ROWS` est l'équivalent de `ROW`.

`WITH TIES` doit être utilisé avec la clause `ORDER BY`.

`WITH TIES` permet de renvoyer davantage de rangées contrairement à la valeur spécifiée dans `FETCH` car les rangées homologues, ces rangées qui ne sont pas distinctes et basées sur la clause `ORDER BY`, sont également renvoyées.

## Exemples

Renvoyer des informations depuis la 26ème rangée de l'ensemble de résultats triés par `nom_famille`, puis par `prénom`.

```
SELECT id_emp, nom_famille, prénom FROM emp ORDER BY nom_famille, prénom
OFFSET 25 ROWS
```

Indiquer que vous voulez uniquement renvoyer dix rangées.

```
SELECT id_emp, nom_famille, prénom FROM emp ORDER BY nom_famille, prénom
OFFSET 25 ROWS FETCH FIRST 10 ROWS ONLY
```

Renvoyer dix rangées et leurs rangées homologues (rangées qui ne sont pas distinctes et basées sur la clause `ORDER BY`).

```
SELECT id_emp, nom_famille, prénom FROM emp ORDER BY nom_famille, prénom
OFFSET 25 ROWS FETCH FIRST 10 ROWS WITH TIES
```

## Clause FOR UPDATE

La clause `FOR UPDATE` verrouille les enregistrements pour des mises à jour positionnées ou des suppressions positionnées via les curseurs SQL. La syntaxe de cette clause est la suivante :

```
FOR UPDATE [OF expressions_colonne]
```

`expressions_colonne` est une liste de noms de rubriques de la table de base de données que vous souhaitez mettre à jour, séparés par une virgule. `expressions_colonne` est facultatif et est ignoré.

## Exemple

Renvoyer tous les enregistrements de la base de données d'employés dont la valeur de la rubrique `SALAIRE` est supérieure à 20 000 euros.

```
SELECT * FROM emp WHERE salaire > 20000
FOR UPDATE OF nom, prénom, salaire
```

Lorsque chaque enregistrement est récupéré, il est verrouillé. Si vous mettez à jour l'enregistrement ou que vous le supprimez, le verrou est maintenu jusqu'à ce que vous validiez la modification. Sinon, le verrou disparaît lorsque vous passez à l'enregistrement suivant.

## Exemples

Utilisation de	Exemple de code SQL
Constante de type texte	<code>SELECT 'ChienChat' FROM Vendeurs</code>
Constante de type numérique	<code>SELECT 999 FROM Vendeurs</code>
Constante de type date	<code>SELECT DATE '2019-06-05' FROM Vendeurs</code>
Constante de type heure	<code>SELECT TIME '02:49:03' FROM Vendeurs</code>
Constante de type horodatage	<code>SELECT TIMESTAMP '2019-06-05 02:49:03' FROM Vendeurs</code>
Colonne de texte	<code>SELECT Nom_Société FROM Informations_Ventes</code> <code>SELECT DISTINCT Nom_Société FROM Informations_Ventes</code>
colonne de type numérique	<code>SELECT Quantité FROM Informations_Ventes</code> <code>SELECT DISTINCT Quantité FROM Informations_Ventes</code>
Colonne de type date	<code>SELECT Date_Vente FROM Informations_Ventes</code> <code>SELECT DISTINCT Date_Vente FROM Informations_Ventes</code>
Colonne de type heure	<code>SELECT Heure_Vente FROM Informations_Ventes</code> <code>SELECT DISTINCT Heure_Vente FROM Informations_Ventes</code>
Colonne de type horodatage	<code>SELECT Horodatage_Vente FROM Informations_Ventes</code> <code>SELECT DISTINCT Horodatage_Vente FROM Informations_Ventes</code>
Colonne <sup>a</sup> Conteneur	<code>SELECT Brochures_Société FROM Informations_Ventes</code> <code>SELECT GETAS(Logo_Société, 'JPEG') FROM Informations_Ventes</code>
Caractère joker *	<code>SELECT * FROM Vendeurs</code> <code>SELECT DISTINCT * FROM Vendeurs</code>

a. Un conteneur de fichier de base de données est une rubrique qui contient un fichier de base de données FileMaker.

## Notes à propos des exemples

Une `colonne` est une référence à une rubrique dans un fichier de base de données FileMaker. (Cette rubrique peut contenir plusieurs valeurs distinctes.)

Le caractère générique (\*) est une manière plus courte de dire « tout ». L'exemple `SELECT * FROM Vendeurs` fait apparaître toutes les colonnes de la table `Vendeurs`. L'exemple `SELECT DISTINCT * FROM Vendeurs` fait apparaître toutes les rangées uniques de la table `Vendeurs` (sans doublons).

- FileMaker ne stocke pas de données pour les chaînes vides. Les requêtes suivantes ne renvoient donc jamais d'enregistrement :

```
SELECT * FROM test WHERE c = ''
SELECT * FROM test WHERE c <> ''
```

- Si vous utilisez `SELECT` avec des données binaires, vous devez utiliser la fonction `GetAs()` pour indiquer le flux à renvoyer. Consultez la section « Récupération du contenu d'une rubrique Conteneur : fonctions `CAST()` et `GetAs()` », pour plus d'informations.

### Récupération du contenu d'une rubrique Conteneur : fonctions CAST() et GetAs()

Depuis une rubrique Conteneur, vous pouvez récupérer des informations de référence de fichier, des données binaires ou des données appartenant à un type de fichier particulier.

- Pour récupérer des informations de référence de fichier depuis une rubrique Conteneur, comme le chemin d'accès à un fichier, à une image ou à une séquence QuickTime, utilisez la fonction `CAST()` avec une instruction `SELECT`.
- S'il existe des données de fichier ou des données binaires JPEG, l'instruction `SELECT` avec la fonction `GetAs(nom rubrique, 'JPEG')` récupère des données au format binaire ; sinon, l'instruction `SELECT` avec un nom de rubrique renvoie la valeur `NULL`.

#### Exemple

Pour récupérer des informations de référence de fichier, utilisez la fonction `CAST()` avec une instruction `SELECT`.

```
SELECT CAST(Brochures_Société AS VARCHAR) FROM Informations_Ventes
```

Dans cet exemple :

- Si vous avez inséré un fichier dans la rubrique Conteneur en utilisant FileMaker Pro, mais que vous avez enregistré uniquement une référence dans le fichier, l'instruction `SELECT` récupère les informations de référence du fichier comme étant du type `SQL_VARCHAR`.
- Si vous avez inséré le contenu d'un fichier dans la rubrique Conteneur en utilisant FileMaker Pro, l'instruction `SELECT` récupère le nom de ce fichier.
- Si vous avez importé un fichier dans la rubrique Conteneur depuis une autre application, l'instruction `SELECT` affiche « ? » (le fichier apparaît sous la forme **Untitled.dat** dans FileMaker Pro).

Pour récupérer des données binaires, vous pouvez utiliser l'instruction `SELECT` avec la fonction `GetAs()` de l'une des façons suivantes :

- Lorsque vous utilisez la fonction `GetAs()` avec l'option `DEFAULT`, vous récupérez le flux maître du conteneur sans devoir définir explicitement le type de flux.

#### Exemple

```
SELECT GetAs(Brochures_Société, DEFAULT) FROM Informations_Ventes
```

- Pour récupérer un type de flux individuel depuis un conteneur, utilisez la fonction `GetAs()` et indiquez le type du fichier en fonction du mode d'insertion des données dans la rubrique Conteneur dans FileMaker Pro.

#### Exemple

Si les données ont été insérées à l'aide de la commande **Insérer > Fichier**, indiquez `'FILE'` dans la fonction `GetAs()`.

```
SELECT GetAs(Brochures_Société, 'FILE') FROM Informations_Ventes
```

## Exemple

Si les données ont été insérées à l'aide de la commande **Insérer > Image**, faites glisser et déposez le fichier ou copiez-le depuis le Presse-papiers et indiquez l'un des types de fichiers répertoriés dans le tableau ci-dessous, par exemple, 'JPEG'.

```
SELECT GetAs(Logo_Société, 'JPEG') FROM Icônes_Société
```

Type de fichier	Description
'GIFf'	Graphics Interchange Format
'JPEG'	Photos
'TIFF'	Format de fichier trame pour images numériques
'PDF'	Portable Document Format
'PNGf'	Format d'image Bitmap

## Instruction DELETE

L'instruction `DELETE` permet de supprimer des enregistrements d'une table de base de données. La syntaxe de l'instruction `DELETE` est la suivante :

```
DELETE FROM nom_table [ WHERE { conditions } ]
```

**Remarque** La clause `WHERE` indique quels enregistrements doivent être supprimés. Si vous n'incluez pas le mot-clé `WHERE`, tous les enregistrements de la table seront supprimés (mais la table restera intacte).

## Exemple

Supprimer un enregistrement de la table `emp`.

```
DELETE FROM emp WHERE id_emp = 'E10001'
```

Chaque instruction `DELETE` supprime chaque enregistrement réunissant les conditions définies dans la clause `WHERE`. Ici, chaque enregistrement contenant l'ID d'employé `E10001` est supprimé. Les ID d'employés étant uniques dans la table `Employés`, un seul enregistrement est supprimé.

## Instruction INSERT

L'instruction `INSERT` permet de créer des enregistrements dans une table de base de données. Vous pouvez spécifier, au choix :

- Une liste de valeurs à insérer sous la forme d'un nouvel enregistrement
- Une instruction `SELECT` permettant de copier les données d'une autre table et de les insérer sous forme de nouveaux enregistrements

La syntaxe de l'instruction `INSERT` est la suivante :

```
INSERT INTO nom_table [(nom_col, ...)] VALUES (expr, ...)
```

`nom_col` est une liste facultative de noms de colonnes qui indique le nom et l'ordre des colonnes dont les valeurs sont spécifiées dans la clause `VALUES`. Si vous omettez `nom_col`, les expressions de valeur (`expr`) doivent fournir des valeurs pour toutes les colonnes définies dans la table et elles doivent figurer dans le même ordre que les colonnes définies pour la table.

`nom_col` peut également indiquer la répétition d'une rubrique, par exemple : `lastDates[4]`.

`expr` représente la liste d'expressions donnant les valeurs des colonnes du nouvel enregistrement. Généralement, les expressions sont les valeurs constantes des colonnes (mais il peut également s'agir d'une sous-requête). Les valeurs de type chaîne de caractères doivent être placées entre guillemets simples (''). Pour inclure un guillemet simple (correspondant à une apostrophe) dans une valeur de type chaîne de caractères placée entre guillemets simples, utilisez deux guillemets simples consécutifs (par exemple 'L''instance').

Les sous-requêtes doivent être entre parenthèses.

### Exemple

Insérer une liste d'expressions.

```
INSERT INTO emp (nom_famille, prénom, id_emp, salaire, date_embauche)
VALUES ('Martin', 'Robert', 'E22345', 27500, DATE '2019-06-05')
```

Chaque instruction `INSERT` ajoute un enregistrement à la table de base de données. Ici, nous avons ajouté un enregistrement à la table `emp` de la base de données. Les valeurs sont spécifiées pour cinq colonnes. Les autres colonnes de la table n'ont pas reçu de valeur. Autrement dit, leur valeur est nulle.

**Remarque** Dans les types de données de conteneur, vous ne pouvez utiliser `INSERT` que pour du texte, à moins de préparer une déclaration paramétrée et de générer le flux de données depuis votre application. Pour utiliser des données binaires, vous pouvez simplement assigner le nom de fichier en l'insérant entre des guillemets simples ou utiliser la fonction `PutAs()`. Lorsque vous spécifiez un nom de fichier, le type de fichier est déduit de l'extension de fichier :

```
INSERT INTO nom_table (nom_conteneur) VALUES(? AS 'nomfichier.extension
fichier')
```

Les types de fichiers non pris en charge seront insérés en tant que type `FILE`.

Lorsque vous utilisez la fonction `PutAs()`, indiquez le type : `PutAs(col, 'type')`, où la valeur du type est un type de fichier pris en charge comme décrit dans la section « Récupération du contenu d'une rubrique Conteneur : fonctions `CAST()` et `GetAs()` », page 16.

L'instruction `SELECT` est une requête qui renvoie des valeurs pour chaque valeur `nom_col` spécifiée dans la liste de noms de colonnes. L'utilisation d'une instruction `SELECT` plutôt qu'une liste d'expressions de valeurs vous permet de sélectionner un ensemble de rangées dans une table et de l'insérer dans une autre table à l'aide d'une instruction `INSERT` unique.

### Exemple

Utiliser une instruction `SELECT` pour l'insertion.

```
INSERT INTO emp1 (prénom, nom_famille, id_emp, service, salaire)
SELECT prénom, nom_famille, id_emp, service, salaire from emp
WHERE service = 'D050'
```

Dans ce type d'instruction `INSERT`, le nombre de colonnes à insérer doit être le même que celui des colonnes spécifiées dans l'instruction `SELECT`. La liste de colonnes à insérer doit être identique aux colonnes de l'instruction `SELECT` comme c'est le cas avec une liste d'expressions de valeurs dans l'autre type d'instruction `INSERT`. Par exemple, la première colonne insérée correspond à la première colonne sélectionnée, la deuxième à la deuxième colonne sélectionnée, et ainsi de suite.

La taille et le type de données de ces colonnes correspondantes doivent être compatibles. Chaque colonne de la liste `SELECT` doit avoir un type de données accepté par le pilote client ODBC ou JDBC pour une opération `INSERT/UPDATE` habituelle sur la colonne correspondante de la liste `INSERT`. Les valeurs sont tronquées si la taille de la valeur de la colonne `SELECT` est supérieure à celle de la colonne `INSERT` correspondante.

L'instruction `SELECT` est évaluée avant l'insertion de valeurs.

## Instruction UPDATE

L'instruction `UPDATE` permet de modifier les enregistrements d'une table de base de données. La syntaxe de l'instruction `UPDATE` est la suivante :

```
UPDATE nom_table SET nom_col = expr, ... [ WHERE { conditions } ]
```

`nom_col` est le nom d'une colonne dont la valeur doit être modifiée. Vous pouvez modifier plusieurs colonnes dans la même instruction.

`expr` est la nouvelle valeur de la colonne.

Généralement, les expressions sont les valeurs constantes des colonnes (mais il peut également s'agir d'une sous-requête). Les valeurs de type chaîne de caractères doivent être placées entre guillemets simples (`'`). Pour inclure un guillemet simple (correspondant à une apostrophe) dans une valeur de type chaîne de caractères placée entre guillemets simples, utilisez deux guillemets simples consécutifs (par exemple `'L' 'instance'`).

Les sous-requêtes doivent être entre parenthèses.

La clause `WHERE` peut être toute clause valide. Elle détermine les enregistrements à modifier.

### Exemple

Appliquer l'instruction `UPDATE` à la table `emp`.

```
UPDATE emp SET salaire=32000, exempt=1 WHERE id_emp = 'E10001'
```

Cette instruction `UPDATE` a pour effet de modifier chaque enregistrement réunissant les conditions définies dans la clause `WHERE`. Ici, il s'agit de changer le salaire et le statut d'exemption pour tous les employés dont l'ID est `E10001`. Les ID des employés étant uniques dans la table `Employés`, un seul enregistrement est mis à jour.

## Exemple

Appliquer l'instruction UPDATE à la table emp avec une sous-requête.

```
UPDATE emp SET salaire = (SELECT avg(salaire) from emp) WHERE id_emp = 'E10001'
```

Ici, le salaire dont l'ID d'employé est E10001 est remplacé par le salaire moyen de la société.

**Remarque** Dans les types de données de conteneur, vous ne pouvez utiliser UPDATE qu'avec du texte, à moins de préparer une déclaration paramétrée et de générer le flux de données depuis votre application. Pour utiliser des données binaires, vous pouvez simplement assigner le nom de fichier en l'insérant entre des guillemets simples ou utiliser la fonction PutAs(). Lorsque vous spécifiez un nom de fichier, le type de fichier est déduit de l'extension de fichier :

```
UPDATE nom_table SET (nom_conteneur) = ? AS 'nomfichier.extension fichier'
```

Les types de fichiers non pris en charge seront insérés en tant que type FILE.

Lorsque vous utilisez la fonction PutAs(), indiquez le type : PutAs(col, 'type'), où la valeur du type est un type de fichier pris en charge comme décrit dans la section « Récupération du contenu d'une rubrique Conteneur : fonctions CAST() et GetAs() », page 16.

## Instruction CREATE TABLE

L'instruction `CREATE TABLE` permet de créer une table dans un fichier de base de données. La syntaxe de l'instruction `CREATE TABLE` est la suivante :

```
CREATE TABLE nom_table ( liste_éléments_de_table [, liste_éléments_de_table... ] )
```

Dans cette instruction, vous devez indiquer le nom et le type de données de chaque colonne.

- `nom_table` correspond au nom de la table. `nom_table` est limité à 100 caractères. Aucune table portant ce nom ne doit avoir été définie. Le nom de la table doit commencer par un caractère alphabétique. Si le nom de la table commence par un caractère autre qu'un caractère alphabétique, placez-le entre des guillemets doubles (identifiant cité).
- Le format de `liste_éléments_table` est le suivant :

```
nom_rubrique type_rubrique [[répétitions]]
[DEFAULT expr] [UNIQUE | NOT NULL | PRIMARY KEY | GLOBAL]
[EXTERNAL chaîne_chemin_relatif [SECURE | OPEN chaîne_chemin_calc]]
```

- `nom_rubrique` correspond au nom de la rubrique. Les noms de rubriques doivent être uniques. Les noms des rubriques commencent par un caractère alphabétique. Si le nom d'une rubrique commence par un caractère autre qu'un caractère alphabétique, placez-le entre des guillemets doubles (identifiant cité).

### Exemple

L'instruction `CREATE TABLE` de la rubrique intitulée `_NOMFAMILLE` est :

```
CREATE TABLE "_EMPLOYES" (ID INT PRIMARY KEY, "_PRENOM" VARCHAR(20),
"_NOMFAMILLE" VARCHAR(20))
```

- Pour les répétitions de l'instruction `CREATE TABLE`, spécifiez une rubrique multivaluée en utilisant un nombre entre 1 et 32 000 entre crochets après le type de rubrique.

### Exemple

```
ID_EMPLOYE INT[4]
NOMFAMILLE VARCHAR(20)[4]
```

- `type_rubrique` peut être l'un des éléments suivants : `NUMERIC`, `DECIMAL`, `INT`, `DATE`, `TIME`, `TIMESTAMP`, `VARCHAR`, `CHARACTER VARYING`, `BLOB`, `VARBINARY`, `LONGVARBINARY` ou `BINARY VARYING`. Pour `NUMERIC` et `DECIMAL`, vous pouvez définir la précision et l'échelle. Par exemple : `DECIMAL(10, 0)`. Pour `TIME` et `TIMESTAMP`, vous pouvez définir la précision. Par exemple : `TIMESTAMP(6)`. Pour `VARCHAR` et `CHARACTER VARYING`, vous pouvez définir la longueur de la chaîne.

### Exemple

```
VARCHAR(255)
```

- Le mot-clé `DEFAULT` vous permet de définir une valeur par défaut pour une colonne. Pour `expr`, vous pouvez utiliser une valeur constante ou une expression. Les expressions autorisées sont les suivantes : `USER`, `USERNAME`, `CURRENT_USER`, `CURRENT_DATE`, `CURDATE`, `CURRENT_TIME`, `CURTIME`, `CURRENT_TIMESTAMP`, `CURTIMESTAMP` et `NULL`.

- Quand vous définissez une colonne comme étant `UNIQUE`, l'option de validation **Unique** est sélectionnée automatiquement pour la rubrique correspondante dans le fichier de base de données FileMaker.
- Quand vous définissez une colonne en tant que `NOT NULL`, l'option de validation **Non vide** est sélectionnée automatiquement pour la rubrique correspondante dans le fichier de base de données FileMaker. La rubrique est marquée en tant que **Valeur requise** dans l'onglet **Rubriques** de la boîte de dialogue Gérer la base de données de FileMaker Pro.
- Pour définir une colonne en tant que rubrique Conteneur, utilisez `BLOB`, `VARBINARY` ou `BINARY VARYING` en guise de `type_rubrique`.
- Pour définir une colonne en tant que rubrique Conteneur stockant les données en externe, utilisez le mot-clé `EXTERNAL`. L'élément `chaîne_chemin_relatif` définit le dossier dans lequel les données sont stockées en externe, par rapport à l'emplacement de la base de données FileMaker. Ce chemin doit être défini en tant que répertoire de base dans la boîte de dialogue Gérer les conteneurs de FileMaker Pro. Vous devez préciser `SECURE` pour un stockage sécurisé ou `OPEN` pour un stockage ouvert. Si vous utilisez un stockage ouvert, l'élément `chaîne_chemin_calc` correspond au dossier figurant dans le dossier `chaîne_chemin_relatif` où les objets Conteneur doivent être stockés. Le chemin doit utiliser des barres obliques (`/`) dans le nom de dossier.

## Exemples

Utilisation de	Exemple de code SQL
Colonne de texte	<code>CREATE TABLE T1 (C1 VARCHAR, C2 VARCHAR (50), C3 VARCHAR (1001), C4 VARCHAR (500276))</code>
colonne de type texte, NOT NULL	<code>CREATE TABLE T1NN (C1 VARCHAR NOT NULL, C2 VARCHAR (50) NOT NULL, C3 VARCHAR (1001) NOT NULL, C4 VARCHAR (500276) NOT NULL)</code>
colonne de type numérique	<code>CREATE TABLE T2 (C1 DECIMAL, C2 DECIMAL (10,0), C3 DECIMAL (7539,2), C4 DECIMAL (497925,301))</code>
Colonne de type date	<code>CREATE TABLE T3 (C1 DATE, C2 DATE, C3 DATE, C4 DATE)</code>
Colonne de type heure	<code>CREATE TABLE T4 (C1 HEURE, C2 HEURE, C3 HEURE, C4 HEURE)</code>
Colonne de type horodatage	<code>CREATE TABLE T5 (C1 TIMESTAMP, C2 TIMESTAMP, C3 TIMESTAMP, C4 TIMESTAMP)</code>
colonne pour rubrique Conteneur	<code>CREATE TABLE T6 (C1 CONTENEUR, C2 CONTENEUR, C3 CONTENEUR, C4 CONTENEUR)</code>
colonne pour rubrique Conteneur de stockage externe	<code>CREATE TABLE T7 (C1 BLOB EXTERNAL 'Files/MyDatabase/' SECURE)</code> <code>CREATE TABLE T8 (C1 BLOB EXTERNAL 'Files/MyDatabase/' OPEN 'Objects')</code>

## Instruction TRUNCATE TABLE

L'instruction `TRUNCATE TABLE` permet de supprimer rapidement tous les enregistrements d'une table spécifiée, vidant la table de toutes ses données.

```
TRUNCATE TABLE nom_table
```

Vous ne pouvez pas spécifier de clause `WHERE` avec l'instruction `TRUNCATE TABLE`. L'instruction `TRUNCATE TABLE` supprime tous les enregistrements.

Seuls les enregistrements de la table spécifiée par `nom_table` sont supprimés. Les enregistrements des tables liées ne sont pas affectés.

L'instruction `TRUNCATE TABLE` doit verrouiller tous les enregistrements de la table afin de pouvoir supprimer leurs données. Si un enregistrement de la table est verrouillé par un autre utilisateur, FileMaker renvoie le code d'erreur 301 (« L'enregistrement est utilisé par un autre utilisateur »).

## Instruction ALTER TABLE

L'instruction `ALTER TABLE` permet de modifier la structure d'une table existante dans un fichier de base de données. Chaque instruction ne permet de modifier qu'une seule colonne. Les formats de l'instruction `ALTER TABLE` sont les suivants :

```
ALTER TABLE nom_table ADD [COLUMN] définition_colonne
ALTER TABLE nom_table DROP [COLUMN] nom_colonne_non_qualifié
ALTER TABLE nom_table ALTER [COLUMN] définition_colonne SET DEFAULT expr
ALTER TABLE nom_table ALTER [COLUMN] définition_colonne DROP DEFAULT
```

Vous devez connaître la structure de la table et savoir comment vous souhaitez la modifier avant d'utiliser l'instruction `ALTER TABLE`.

### Exemples

Pour	Exemple de code SQL
Ajouter des colonnes	<code>ALTER TABLE Vendeurs ADD (C1 VARCHAR)</code>
Supprimer des colonnes	<code>ALTER TABLE Vendeurs DROP (C1)</code>
Définir la valeur par défaut d'une colonne	<code>ALTER TABLE Vendeurs ALTER Société SET DEFAULT 'FileMaker'</code>
Supprimer la valeur par défaut d'une colonne	<code>ALTER TABLE Vendeurs ALTER Société DROP DEFAULT</code>

**Remarque** `SET DEFAULT` et `DROP DEFAULT` n'affectent pas les rangées existantes de la table, mais modifient la valeur par défaut des rangées ajoutées par la suite à la table.

## Instruction CREATE INDEX

L'instruction `CREATE INDEX` permet d'accélérer les recherches dans votre fichier de base de données. La syntaxe de l'instruction `CREATE INDEX` est la suivante :

```
CREATE INDEX ON nom_table.nom_col
CREATE INDEX ON nom_table (nom_col)
```

`CREATE INDEX` ne fonctionne qu'avec une colonne unique (les index multi-colonne ne sont pas pris en charge). Les index ne sont pas autorisés sur les colonnes correspondant à des rubriques de données de conteneur ou statistique, à des rubriques employant des options de stockage global ou à des rubriques de type calcul non stockées.

Quand vous créez un index pour une colonne de type texte, l'option de stockage **Minimal** est sélectionnée automatiquement dans **Indexation** pour la rubrique correspondante du fichier de base de données FileMaker. Quand vous créez un index pour une colonne qui n'est pas de type texte (ou pour une colonne de type texte au format japonais), l'option de stockage **Tout** est sélectionnée automatiquement dans **Indexation** pour la rubrique correspondante du fichier de base de données FileMaker.

Quand vous créez un index pour une colonne, l'option de stockage **Indexation automatique si nécessaire** est sélectionnée automatiquement dans **Indexation** pour la rubrique correspondante du fichier de base de données FileMaker.

FileMaker crée automatiquement les index selon les besoins. L'utilisation de `CREATE INDEX` entraîne la génération immédiate de l'index plutôt qu'à la demande.

### Exemple

```
CREATE INDEX ON Vendeurs.ID_vendeur
```

### Instruction DROP INDEX

L'instruction `DROP INDEX` permet de supprimer un index d'un fichier de base de données. La syntaxe de l'instruction `DROP INDEX` est la suivante :

```
DROP INDEX ON nom_table.nom_col  
DROP INDEX ON nom_table (nom_col)
```

Vous pouvez supprimer un index quand votre fichier de base de données est trop volumineux ou quand vous n'employez pas souvent une rubrique dans les requêtes.

Si les performances de vos requêtes ne sont pas satisfaisantes et que vous travaillez sur un fichier de base de données FileMaker extrêmement volumineux contenant un grand nombre de rubriques de type texte indexées, pensez à supprimer les index de certaines rubriques. Pensez également à supprimer les index des rubriques que vous utilisez rarement dans des instructions `SELECT`.

Quand vous supprimez un index pour une colonne, l'option de stockage **Aucun** est sélectionnée et l'option **Indexation automatique si nécessaire** est désélectionnée automatiquement dans **Indexation** pour la rubrique correspondante du fichier de base de données FileMaker.

L'attribut `PREVENT INDEX CREATION` n'est pas pris en charge.

### Exemple

```
DROP INDEX ON Vendeurs.ID_Vendeur
```

## Expressions SQL

Vous pouvez utiliser des expressions dans les clauses `WHERE`, `HAVING` et `ORDER BY` des instructions `SELECT` pour créer des requêtes de base de données complexes et détaillées. Les éléments d'expression valides sont les suivants :

- Noms de rubriques
- Constantes
- Notification en virgule flottante/scientifique
- Opérateurs numériques
- Opérateurs de caractères
- Opérateurs de dates
- Opérateurs relationnels
- Opérateurs logiques
- Fonctions

## Noms de rubriques

L'expression la plus courante est un simple nom de rubrique, tel que `calc` ou `Informations_Ventes.ID_facture`.

## Constantes

Les constantes sont des valeurs qui ne changent pas. Par exemple, dans l'expression `RIX * 1,05`, la valeur `1,05` est une constante. Vous pouvez également affecter la valeur `30` à la constante `Nombre_de_jours_en_juin`.

Les valeurs de type constante doivent être placées entre guillemets simples (`'`). Pour inclure un guillemet simple (correspondant à une apostrophe) dans une constante de type chaîne de caractères placée entre guillemets simples, utilisez deux guillemets simples consécutifs (par exemple `'L' instance'`).

Dans les applications ODBC et JDBC, FileMaker accepte les constantes de date, d'heure et d'horodatage au format ODBC/JDBC entre accolades (`{}`).

### Exemples

- `{D '2019-06-05' }`
- `{T '14:35:10' }`
- `{TS '2019-06-05 14:35:10' }`

FileMaker permet d'utiliser la version en majuscules ou en minuscules des spécificateurs de type (`D`, `T`, `TS`). Vous pouvez utiliser le nombre d'espaces souhaité après le spécificateur de type ou bien n'inclure aucun espace.

FileMaker accepte également les formats de date et d'heure ISO syntaxe SQL-92 sans accolades.

### Exemples

- `DATE 'AAAA-MM-JJ'`
- `TIME 'HH:MM:SS'`
- `TIMESTAMP 'AAAA-MM-JJ HH:MM:SS'`

La fonction `ExecuterSQL` de FileMaker Pro n'accepte que les formats de date et d'heure ISO syntaxe SQL-92 sans accolades.

Constante	Syntaxe acceptable (exemples)
Texte	<code>'Paris'</code>
Nombre	<code>1.05</code>
Date	<code>DATE '2019-06-05'</code> <code>{D '2019-06-05' }</code> <code>{06/05/2019}</code> <code>{06/05/19}</code>
	<b>Remarque</b> La syntaxe d'année à deux chiffres n'est pas prise en charge pour le format ODBC/JDBC ou le format SQL-92.

Constante	Syntaxe acceptable (exemples)
Heure	<pre>TIME '14:35:10'</pre> <pre>{ T '14:35:10' }</pre> <pre>{14:35:10}</pre>
Horodatage	<pre>TIMESTAMP '2019-06-05 14:35:10'</pre> <pre>{ TS '2019-06-05 14:35:10' }</pre> <pre>{06/05/2019 14:35:10}</pre> <pre>{06/05/19 14:35:10}</pre> <p>Vérifiez que <b>type de données strictes : Année à 4 chiffres</b> n'est pas sélectionnée comme option de validation dans le fichier de base de données FileMaker pour une rubrique qui utilise cette syntaxe d'année à deux chiffres.</p> <p><b>Remarque</b> La syntaxe d'année à deux chiffres n'est pas prise en charge pour le format ODBC/JDBC ou le format SQL-92.</p>

Lors de la saisie de valeurs de date et d'heure, faites correspondre le format de la configuration locale du fichier de base de données. Par exemple, si la base de données a été créée sur un système italien, utilisez les formats italiens de date et d'heure.

### Notification en virgule flottante/scientifique

Les nombres peuvent être exprimés à l'aide d'une notation scientifique.

#### Exemple

```
SELECT colonne1 / 3.4E+7 FROM table1 WHERE calc < 3.4E-6 * colonne2
```

### Opérateurs numériques

Vous pouvez inclure les opérateurs suivants dans une expression numérique : +, -, \*, /, et ^ ou \*\* (puissance).

Vous pouvez faire précéder des expressions numériques par un signe plus (+) ou moins (-) unaire.

### Opérateurs de caractères

Vous pouvez concaténer des caractères. Dans les exemples suivants, `nom_famille` a pour valeur 'MARTIN ' et `prénom` a pour valeur 'ROBERT ' :

Opérateur	Concaténation	Exemple	Résultat
+	Laisse les espaces finaux à leur place	prénom + nom_famille	'ROBERT MARTIN '
-	Déplace les espaces finaux à la fin	prénom - nom_famille	'ROBERTMARTIN '

## Opérateurs de dates

Vous pouvez modifier des dates. Dans les exemples suivants, `date_embauche` a pour valeur `DATE '2019-01-30'`.

Opérateur	Effet sur la date	Exemple	Résultat
+	Ajoute un nombre de jours à une date	<code>date_embauche + 5</code>	<code>DATE '2019-02-04'</code>
-	Recherche le nombre de jours entre deux dates	<code>date_embauche - DATE '2019-01-01'</code>	29
	Enlève un nombre de jours à une date	<code>date_embauche - 10</code>	<code>DATE '2019-01-20'</code>

## Exemples supplémentaires

```
SELECT Date_Vente, Date_Vente + 30 AS stat FROM Informations_Ventes
SELECT Date_Vente, Date_Vente - 30 AS stat FROM Informations_Ventes
```

## Opérateurs relationnels

Opérateur	Signification
=	Egal
<>	N'est pas égal à
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à
LIKE	Identique à une structure
NOT LIKE	Pas identique à une structure
IS NULL	A une valeur nulle
IS NOT NULL	N'a pas de valeur nulle
BETWEEN	Plage de valeurs entre une limite inférieure et une limite supérieure
IN	Membre d'un jeu de valeurs indiquées ou d'une sous-requête
NOT IN	Non membre d'un jeu de valeurs indiquées ou d'une sous-requête
EXISTS	Renvoie 'True' (Vrai) si une sous-requête a renvoyé au moins un enregistrement
ANY	Compare une valeur à chaque valeur renvoyée par une sous-requête (l'opérateur doit être précédé de =, <>, >, >=, <, ou <=) ; =Any est équivalent à In
ALL	Compare une valeur à chaque valeur renvoyée par une sous-requête (l'opérateur doit être précédé de =, <>, >, >=, <, ou <=)

## Exemple

```

SELECT Informations_Ventes.ID_Facture FROM Informations_Ventes
  WHERE Informations_Ventes.ID_Vendeur = 'SP-1'
SELECT Informations_Ventes.Quantité FROM Informations_Ventes
  WHERE Informations_Ventes.ID_Facture <> 125
SELECT Informations_Ventes.Quantité FROM Informations_Ventes
  WHERE Informations_Ventes.Quantité > 3000
SELECT Informations_Ventes.Heure_Vente FROM Informations_Ventes
  WHERE Informations_Ventes.Heure_Vente < '12:00:00'
SELECT Informations_Ventes.Nom_Société FROM Informations_Ventes
  WHERE Informations_Ventes.Nom_Société LIKE '%Université'
SELECT Informations_Ventes.Nom_Société FROM Informations_Ventes
  WHERE Informations_Ventes.Nom_Société NOT LIKE '%Université'
SELECT Informations_Ventes.Montant FROM Informations_Ventes
  WHERE Informations_Ventes.Montant IS NULL
SELECT Informations_Ventes.Montant FROM Informations_Ventes
  WHERE Informations_Ventes.Montant IS NOT NULL
SELECT Informations_Ventes.ID_Facture FROM Informations_Ventes
  WHERE Informations_Ventes.ID_Facture BETWEEN 1 AND 10
SELECT COUNT(Informations_Ventes.ID_Facture) AS stat
  FROM Informations_Ventes WHERE Informations_Ventes.ID_FACTURE IN
(50,250,100)
SELECT COUNT(Informations_Ventes.ID_Facture) AS stat
  FROM Informations_Ventes WHERE Informations_Ventes.ID_FACTURE NOT IN
(50,250,100)
SELECT COUNT(Informations_Ventes.ID_Facture) AS stat FROM
Informations_Ventes
  WHERE Informations_Ventes.ID_FACTURE NOT IN
(SELECT Informations_Ventes.ID_Facture
  FROM Informations_Ventes WHERE Informations_Ventes.ID_Vendeur = 'SP-4')
SELECT *
  FROM Informations_Ventes WHERE EXISTS
(SELECT Informations_Ventes.Quantité
  FROM Informations_Ventes WHERE Informations_Ventes.ID_Vendeur IS NOT
NULL)
SELECT *
  FROM Informations_Ventes WHERE Informations_Ventes.Quantité = ANY
(SELECT Informations_Ventes.Quantité
  FROM Informations_Ventes WHERE Informations_Ventes.ID_Vendeur = 'SP-1')
SELECT *
  FROM Informations_Ventes WHERE Informations_Ventes.Quantité = ALL
(SELECT Informations_Ventes.Quantité
  FROM Informations_Ventes WHERE Informations_Ventes.ID_Vendeur IS NULL)

```

## Opérateurs logiques

Vous pouvez combiner deux conditions ou davantage. Ces conditions doivent être liées par AND ou OR, sous la forme :

```
salaire = 40000 AND exempt = 1
```

L'opérateur logique NOT sert à inverser la signification comme suit :

```
NOT (salaires = 40000 AND exempt = 1)
```

### Exemple

```
SELECT * FROM Informations_Ventes WHERE Informations_Ventes.Nom_Société
  NOT LIKE '%Université' AND Informations_Ventes.Montant > 3000
SELECT * FROM Informations_Ventes WHERE
  (Informations_Ventes.Nom_Société
    LIKE '%Université' OR Informations_Ventes.Montant > 3000)
  AND Informations_Ventes.ID_Vendeur = 'SP-1'
```

## Ordre de priorité des opérateurs

Au fur et à mesure que les expressions se compliquent, l'ordre dans lequel elles sont évaluées devient important. Le tableau suivant montre dans quel ordre les opérateurs sont évalués. Les opérateurs de la première ligne sont évalués en premier et ainsi de suite. Les opérateurs qui figurent sur une même ligne sont évalués de gauche à droite dans l'expression.

Ordre de priorité	Opérateur
1	Unaire '-', Unaire '+'
2	^, **
3	*, /
4	+, -
5	=, <>, <, <=, >, >=, Like, Not Like, Is Null, Is Not Null, Between, In, Exists, Any, All
6	Not
7	AND
8	OR

## Exemples

```
WHERE salaire > 4000 OR date_embauche > (DATE '2008-01-30') AND serv = 'D101'
```

Comme l'opérateur AND est évalué en premier, cette requête extrait les employés du service D101 embauchés après le 30.01.08, ainsi que les employés dont le salaire est supérieur à 40 000 euros quel que soit leur service ou leur date d'embauche.

Pour forcer la clause à être évaluée dans un ordre différent, utilisez des parenthèses pour spécifier les conditions à évaluer en premier.

```
WHERE (salaire > 40000 OR date_embauche > DATE '2008-01-30') AND serv = 'D101'
```

Cet exemple extrait les employés du service D101 dont le salaire est supérieur à 40 000 euros ou qui ont été embauchés après le mercredi 30 janvier 2008.

## Fonctions SQL

Le langage SQL de FileMaker prend en charge un grand nombre de fonctions à utiliser dans des expressions. Certaines des fonctions renvoient des chaînes de caractères, des nombres, des dates de retour et d'autres renvoient des valeurs qui varient selon les conditions remplies par les arguments de fonctions.

### Fonctions statistiques

Les fonctions statistiques renvoient une valeur unique à partir d'un jeu d'enregistrements. Vous pouvez employer une fonction statistique dans une instruction `SELECT` avec un nom de rubrique (par exemple, `AVG(SALAIRE)`) ou en combinaison avec une expression de colonne (par exemple, `AVG(SALAIRE * 1,07)`).

Vous pouvez faire précéder l'expression de colonne par l'opérateur `DISTINCT` pour éliminer les doublons.

### Exemple

```
COUNT (DISTINCT nom_famille)
```

Dans cet exemple, seules les valeurs de nom de famille uniques sont comptées.

Fonctions statistiques	Résultat
SUM	Total des valeurs d'une expression de rubrique de type numérique. Par exemple, <code>SUM(SALAIRE)</code> renvoie la somme de toutes les valeurs de la rubrique SALAIRE.
AVG	Moyenne des valeurs d'une expression de rubrique de type numérique. Par exemple, <code>AVG(SALAIRE)</code> renvoie la moyenne de toutes les valeurs de la rubrique SALAIRE.
COUNT	Nombre de valeurs d'une expression de rubrique. Par exemple, <code>COUNT(NOM)</code> renvoie le nombre de valeurs de la rubrique NOM. Utilisé avec un nom de rubrique, <code>COUNT</code> renvoie le nombre de valeurs non nulles de la rubrique. <code>COUNT(*)</code> est une forme spéciale qui renvoie le nombre d'enregistrements d'un jeu, en incluant ceux qui contiennent des valeurs nulles.
MAX	Valeur maximale d'une expression de rubrique. Par exemple, <code>MAX(SALAIRE)</code> renvoie la valeur maximale de la rubrique SALAIRE.
MIN	Valeur minimale d'une expression de rubrique. Par exemple, <code>MIN(SALAIRE)</code> renvoie la valeur minimale de la rubrique SALAIRE.

## Exemple

```
SELECT SUM (Informations_Ventes.Quantité) AS stat FROM Informations_Ventes
SELECT AVG (Informations_Ventes.Quantité) AS stat FROM Informations_Ventes
SELECT COUNT (Informations_Ventes.Quantité) AS stat FROM Informations_Ventes
SELECT MAX (Informations_Ventes.Quantité) AS stat FROM Informations_Ventes
WHERE Informations_Ventes.Quantité < 3000
SELECT MIN (Informations_Ventes.Quantité) AS stat
FROM Informations_Ventes
WHERE Informations_Ventes.Quantité > 3000
```

Vous ne pouvez pas utiliser de fonction statistique comme argument d'autres fonctions. Si vous le faites, FileMaker renvoie le code d'erreur 8309 (« Les expressions contenant des statistiques ne sont pas prises en charge »). Par exemple, l'instruction suivante n'est pas valide car la fonction statistique `SUM` ne peut pas être utilisée comme argument de la fonction `ROUND` :

## Exemple

```
SELECT ROUND(SUM(salaire), 0) FROM paie
```

Cependant, les fonctions statistiques peuvent utiliser des fonctions qui renvoient des nombres comme arguments. L'instruction suivante est valide :

## Exemple

```
SELECT SUM(ROUND(salaire, 0)) FROM paie
```

## Fonctions qui renvoient des chaînes de caractères

### Fonctions qui renvoient des chaînes de caractères

	Description	Exemple
CHR	Convertit un code ASCII en chaîne d'un caractère	CHR(67) renvoie <b>C</b>
CURRENT_USER	Renvoie l'ID de connexion indiqué pour établir la connexion	
DAYNAME	Renvoie le nom du jour correspondant à une date précise	
RTRIM	Supprime les espaces finaux d'une chaîne	RTRIM(' ABC ') renvoie ' ABC'
TRIM	Supprime les espaces de début et finaux d'une chaîne	TRIM(' ABC ') renvoie 'ABC'
LTRIM	Supprime les espaces de début d'une chaîne	LTRIM(' ABC ') renvoie 'ABC'
UPPER	Transforme chaque lettre d'une chaîne en majuscule	UPPER('Alain') renvoie 'ALAIN'
LOWER	Transforme chaque lettre d'une chaîne en minuscule	LOWER('Alain') renvoie 'alain'

<b>Fonctions qui renvoient des chaînes de caractères</b>	<b>Description</b>	<b>Exemple</b>
LEFT	Renvoie les caractères les plus à gauche d'une chaîne	LEFT('Mathieu',3) renvoie 'Mat'
MONTHNAME	Renvoie le nom du mois calendaire	
RIGHT	Renvoie les caractères les plus à droites d'une chaîne	RIGHT('Mathieu',4) renvoie 'hieu'
SUBSTR SUBSTRING	Renvoie une sous-chaîne de chaîne avec les paramètres de la chaîne, le premier caractère à extraire et le nombre de caractères à extraire (facultatif)	SUBSTR('Conrad',2,3) renvoie 'onr' SUBSTR('Conrad',2) renvoie 'onrad'
SPACE	Génère une chaîne d'espaces	SPACE(5) renvoie ' '
STRVAL	Convertit une valeur de n'importe quel type en chaîne de caractères	STRVAL('Woltman') renvoie 'Woltman' STRVAL(5 * 3) renvoie '15' STRVAL(4 = 5) renvoie 'False' STRVAL(DATE '2019-12-25') renvoie '2019-12-25'
TIME TIMEVAL	Renvoie l'heure du jour sous forme de chaîne	A 21 h 49, TIME() renvoie <b>21:49:00</b>
USERNAME USER	Renvoie l'ID de connexion indiqué pour établir la connexion	

**Remarque** La fonction TIME() n'est plus utilisée. Utilisez le standard SQL CURRENT\_TIME pour la remplacer.

### Exemple

```

SELECT CHR(67) + SPACE(1) + CHR(70) FROM Vendeurs

SELECT RTRIM(' ' + Vendeurs.ID_Vendeur) AS agg FROM Vendeurs

SELECT TRIM(SPACE(1) + Vendeurs.ID_Vendeur) AS stat FROM Vendeurs

SELECT LTRIM(' ' + Vendeurs.ID_Vendeur) AS agg FROM Vendeurs

SELECT UPPER(Vendeurs.Vendeur) AS stat FROM Vendeurs

SELECT LOWER(Vendeurs.Vendeur) AS stat FROM Vendeurs

SELECT LEFT(Vendeurs.Vendeur, 5) AS stat FROM Vendeurs

SELECT RIGHT(Vendeurs.Vendeur, 7) AS stat FROM Vendeurs

SELECT SUBSTR(Vendeurs.ID_Vendeur, 2, 2) + SUBSTR(Vendeurs.ID_Vendeur, 4, 2)
AS stat FROM Vendeurs

SELECT SUBSTR(Vendeurs.ID_Vendeur, 2) + SUBSTR(Vendeurs.ID_Vendeur, 4) AS
stat FROM Vendeurs

SELECT SPACE(2) + Vendeurs.ID_Vendeur AS ID_Vendeur FROM Vendeurs

SELECT STRVAL('60506') AS stat FROM Informations_Ventes WHERE
Informations_Ventes.Facture =1
    
```

### Fonctions qui renvoient des nombres

Fonctions qui renvoient des nombres	Description	Exemple
ABS	Renvoie la valeur absolue de l'expression numérique	
ATAN	Renvoie la tangente inverse de l'argument sous la forme d'un angle exprimé en radians	
ATAN2	Renvoie la tangente inverse des coordonnées x et y sous la forme d'un angle exprimé en radians	
CEIL CEILING	Renvoie le plus petit entier supérieur ou égal à l'argument	
DEG DEGREES	Renvoie le nombre de degrés de l'argument, qui est un angle exprimé en radians	
DAY	Renvoie la partie d'une date correspondant au jour	DAY (DATE '2019-01-30') renvoie <b>30</b>
DAYOFWEEK	Renvoie le jour de la semaine (de 1 à 7) d'une expression de date	DAYOFWEEK (DATE '2004-05-01') renvoie <b>7</b>
MOD	Divise deux nombres et renvoie le reste de la division	MOD (10, 3) renvoie <b>1</b>
EXP	Renvoie une valeur qui est la base du logarithme népérien (e) élevé à la puissance spécifiée par l'argument	

Fonctions qui renvoient des nombres	Description	Exemple
FLOOR	Renvoie le plus grand entier inférieur ou égal à l'argument	
HOUR	Renvoie la partie d'une valeur correspondant à l'heure	
INT	Renvoie la partie entière d'un nombre	INT (6.4321) renvoie <b>6</b>
LENGTH	Renvoie la longueur d'une chaîne	LENGTH ('ABC') renvoie <b>3</b>
MONTH	Renvoie la partie d'une date correspondant au mois	MONTH (DATE '2019-01-30') renvoie <b>1</b>
LN	Renvoie le logarithme népérien de l'argument	
LOG	Renvoie le logarithme courant de l'argument	
MAX	Renvoie le plus grand de deux nombres	MAX (66, 89) renvoie <b>89</b>
MIN	Renvoie le plus petit de deux nombres	MIN (66, 89) renvoie <b>66</b>
MINUTE	Renvoie la partie d'une valeur correspondant aux minutes	
NUMVAL	Convertit une chaîne de caractères en nombre. La fonction échoue si la chaîne de caractère n'est pas un nombre valide	NUMVAL ('123') renvoie <b>123</b>
PI	Renvoie la valeur de constante de la constante mathématique pi	
RADIANS	Renvoie le nombre de radians pour un argument exprimé en degrés	
ROUND	Arrondit un nombre	ROUND (123.456, 0) renvoie <b>123</b> ROUND (123.456, 2) renvoie <b>123,46</b> ROUND (123.456, - 2) renvoie <b>100</b>
SECOND	Renvoie la partie d'une valeur correspondant aux secondes	
SIGN	Indicateur du signe de l'argument : - 1 pour négatif, 0 pour 0 et 1 pour positif	
SIN	Renvoie le sinus de l'argument	
SQRT	Renvoie la racine carrée de l'argument	
TAN	Renvoie la tangente de l'argument	
YEAR	Renvoie la partie d'une date correspondant à l'année	YEAR (DATE '2019-01-30') renvoie <b>2019</b>

## Fonctions qui renvoient des dates

<b>Fonctions qui renvoient des dates</b>	<b>Description</b>	<b>Exemple</b>
CURDATE CURRENT_DATE	Renvoie la date du jour	
CURTIME CURRENT_TIME	Renvoie l'heure actuelle	
CURTIMESTAMP CURRENT_TIMESTAMP	Renvoie la valeur d'horodatage actuelle	
TIMESTAMPVAL	Convertit une chaîne de caractères en horodatage	TIMESTAMPVAL ('2019-01-30 14:00:00') renvoie sa valeur d'horodatage
DATE TODAY	Renvoie la date du jour	Si la date du jour est le 21/11/2019, DATE () renvoie <b>2019-11-21</b>
DATEVAL	Convertit une chaîne de caractères en date	DATEVAL ('2019-01-30') renvoie <b>30/01/2019</b>

**Remarque** La fonction DATE () n'est plus utilisée. Utilisez le standard SQL CURRENT\_DATE pour la remplacer.

## Fonctions conditionnelles

Fonctions conditionnelles	Description	Exemple
CASE WHEN	<p><b>Format CASE simple</b></p> <p>Compare la valeur de <i>exp_entrée</i> aux valeurs des arguments <i>exp_valeur</i> pour déterminer le résultat.</p> <pre>CASE exp_entrée {WHEN exp_valeur THEN résultat...} [ELSE résultat] END</pre>	<pre>SELECT     ID_Facture,     CASE Nom_Société         WHEN 'Exportations Royaume-Uni' THEN 'Exportations Royaume-Uni trouvées'         WHEN 'Fournisseurs d'ameublement' THEN 'Fournisseurs d'ameublement trouvés'         ELSE 'Aucune exportation Royaume-Uni ou fournisseur d'ameublement'     END,     ID_Vendeur FROM     Informations_Ventes</pre>
	<p><b>Format CASE recherché</b></p> <p>Renvoie un résultat selon que la condition spécifiée par une expression WHEN affiche la valeur True (Vrai).</p> <pre>CASE {WHEN exp_boléen THEN résultat...} [ELSE résultat] END</pre>	<pre>SELECT     ID_Facture,     Montant,     CASE         WHEN Montant &gt; 3000 THEN 'Au- dessus de 3000'         WHEN Montant &gt; 1000 THEN 'En- dessus de 3000'         ELSE 'Entre 1000 et 3000'     END,     ID_Vendeur FROM     Informations_Ventes</pre>
COALESCE	Renvoie la première valeur qui n'est pas nulle	<pre>SELECT     ID_Vendeur,     COALESCE(Directeur_Ventes, Vendeur) FROM     Vendeurs</pre>
NULLIF	Compare deux valeurs et renvoie NULL si ces deux valeurs sont égales ; sinon, renvoie la première valeur.	<pre>SELECT     ID_Facture,     NULLIF(Montant, - 1),     ID_Vendeur FROM     Informations_Ventes</pre>

## Objets système FileMaker

Les fichiers de base de données FileMaker incluent les objets système suivants, auxquels vous pouvez accéder à l'aide de requêtes SQL.

### Tables système FileMaker

Chaque fichier de base de données FileMaker inclut deux tables système : FileMaker\_Tables et FileMaker\_Fields. Pour les applications ODBC, ces tables sont incluses dans les informations renvoyées par la fonction de catalogue SQLTables. Pour les applications JDBC, ces tables sont incluses dans les informations renvoyées par la méthode getTables de l'objet DatabaseMetaData. Les tables peuvent également être utilisées dans les fonctions ExecuterSQL.

#### Table FileMaker\_Tables

La table FileMaker\_Tables contient des informations sur les tables de base de données définies dans le fichier FileMaker.

La table FileMaker\_Tables inclut une rangée pour chaque occurrence de table du graphe de liens avec les colonnes suivantes :

- TableName : nom de l'occurrence de table.
- TableId : ID unique de l'occurrence de table.
- BaseTableName : nom de la table de base de données à partir de laquelle l'occurrence de table a été créée.
- BaseFileName : nom de fichier FileMaker du fichier de base de données qui contient la table de base de données.
- ModCount : nombre total de modifications effectuées dans cette table.

#### Exemple

```
SELECT TableName FROM FileMaker_Tables WHERE TableName LIKE 'Sales%'
```

#### Table FileMaker\_Fields

La table FileMaker\_Fields contient des informations sur les rubriques définies dans le fichier FileMaker.

La table FileMaker\_Fields inclut les colonnes suivantes :

- TableName : nom de la table qui contient la rubrique.
- FieldName : nom de la rubrique.
- FieldType : type de données SQL de la rubrique.
- FieldId : ID unique de la rubrique.
- FieldClass : l'une des trois valeurs suivantes : Summary pour les rubriques Récapitulatif ; Calculated pour les résultats calculés ou Normal.
- FieldReps : nombre de répétitions de la rubrique.
- ModCount : nombre total de modifications effectuées dans cette table.

### Exemple

```
SELECT * FROM FileMaker_Fields WHERE TableName='Sales'
```

## Colonnes système FileMaker

FileMaker ajoute des colonnes système (rubriques) à toutes les rangées (enregistrements) de toutes les tables définies dans le fichier FileMaker. Pour les applications ODBC, ces colonnes sont incluses dans les informations renvoyées par la fonction de catalogue `SQLSpecialColumns`. Pour les applications JDBC, ces colonnes sont incluses dans les informations renvoyées par la méthode `getVersionColumns` de l'objet `DatabaseMetaData`. Les colonnes peuvent également être utilisées dans les fonctions `ExecuterSQL`.

### Colonne ROWID

La colonne système `ROWID` contient l'ID unique de l'enregistrement. Il s'agit de la même valeur que celle renvoyée par la fonction `Obtenir(IDEnreg)` de FileMaker Pro.

### Colonne ROWMODID

La colonne système `ROWMODID` contient le nombre total de modifications effectuées dans l'enregistrement actif. Il s'agit de la même valeur que celle renvoyée par la fonction `Obtenir(NombreModificationsEnreg)` de FileMaker Pro.

### Exemple

```
SELECT ROWID, ROWMODID FROM MaTable WHERE ROWMODID > 3
```

## Mots-clés SQL réservés

Cette section répertorie les mots-clés réservés qui ne doivent pas être utilisés comme noms de colonnes, de tables, d'alias ou d'autres objets définis par l'utilisateur. Si vous obtenez des erreurs de syntaxe, ces erreurs peuvent être dues à l'utilisation de l'un de ces mots-clés réservés. Si vous souhaitez utiliser l'un d'entre eux, utilisez des guillemets pour éviter que le mot ne soit traité comme un mot-clé.

### Exemple

Utilisez le mot-clé `DEC` comme nom d'élément de données.

```
create table t ("dec" numeric)
```

ABSOLUTE	CATALOG	CURRENT_USER
ACTION	CHAR	CURSOR
ADD	CHARACTER	CURTIME
ALL	CHARACTER_LENGTH	CURTIMESTAMP
ALLOCATE	CHAR_LENGTH	DATE
ALTER	CHECK	DATEVAL
AND	CHR	DAY
ANY	CLOSE	DAYNAME
ARE	COALESCE	DAYOFWEEK
AS	COLLATE	DEALLOCATE
ASC	COLLATION	DEC
ASSERTION	COLUMN	DECIMAL
AT	COMMIT	DECLARE
AUTHORIZATION	CONNECT	DEFAULT
AVG	CONNECTION	DEFERRABLE
BEGIN	CONSTRAINT	DEFERRED
BETWEEN	CONSTRAINTS	DELETE
BINARY	CONTINUE	DESC
BIT	CONVERT	DESCRIBE
BIT_LENGTH	CORRESPONDING	DESCRIPTOR
BLOB	COUNT	DIAGNOSTICS
BOOLEAN	CREATE	DISCONNECT
BOTH	CROSS	DISTINCT
BY	CURDATE	DOMAIN
CASCADE	CURRENT	DOUBLE
CASCADED	CURRENT_DATE	DROP
CASE	CURRENT_TIME	ELSE
CAST	CURRENT_TIMESTAMP	END

END_EXEC	INTERSECT	ON
ESCAPE	INTERVAL	ONLY
EVERY	INTO	OPEN
EXCEPT	IS	OPTION
EXCEPTION	ISOLATION	OR
EXEC	JOIN	ORDER
EXECUTE	KEY	OUTER
EXISTS	LANGUAGE	OUTPUT
EXTERNAL	LAST	OVERLAPS
EXTRACT	LEADING	PAD
FALSE	LEFT	PART
FETCH	LENGTH	PARTIAL
FIRST	LEVEL	PERCENT
FLOAT	LIKE	POSITION
FOR	LOCAL	PRECISION
FOREIGN	LONGVARBINARY	PREPARE
FOUND	LOWER	PRESERVE
FROM	LTRIM	PRIMARY
FULL	MATCH	PRIOR
GET	MAX	PRIVILEGES
GLOBAL	MIN	PROCEDURE
GO	MINUTE	PUBLIC
GOTO	MODULE	READ
GRANT	MONTH	REAL
GROUP	MONTHNAME	REFERENCES
HAVING	NAMES	RELATIVE
HOUR	NATIONAL	RESTRICT
IDENTITY	NATURAL	REVOKE
IMMEDIATE	NCHAR	RIGHT
IN	NEXT	ROLLBACK
INDEX	NO	ROUND
INDICATOR	NOT	ROW
INITIALLY	NULL	ROWID
INNER	NULLIF	ROWS
INPUT	NUMERIC	RTRIM
INSENSITIVE	NUMVAL	SCHEMA
INSERT	OCTET_LENGTH	SCROLL
INT	OF	SECOND
INTEGER	OFFSET	SECTION

SELECT	UPDATE
SESSION	UPPER
SESSION_USER	USAGE
SET	USER
SIZE	USERNAME
SMALLINT	USING
SOME	VALUE
SPACE	VALUES
SQL	VARBINARY
SQLCODE	VARCHAR
SQLERROR	VARYING
SQLSTATE	VIEW
STRVAL	WHEN
SUBSTRING	WHENEVER
SUM	WHERE
SYSTEM_USER	WITH
TABLE	WORK
TEMPORARY	WRITE
THEN	YEAR
TIES	ZONE
TIME	
TIMESTAMP	
TIMESTAMPVAL	
TIMEVAL	
TIMEZONE_HOUR	
TIMEZONE_MINUTE	
TO	
TODAY	
TRAILING	
TRANSACTION	
TRANSLATE	
TRANSLATION	
TRIM	
TRUE	
TRUNCATE	
UNION	
UNIQUE	
UNKNOWN	

# Index

## A

alias de colonne 8  
alias de table 8, 9  
ALTER TABLE (instruction SQL) 23

## B

BaseFileName 37  
BaseTableName 37

## C

chaîne vide, utilisation dans SELECT 15  
colonne système ROWID 38  
colonne système ROWMODID 38  
conformité avec la norme SQL 7  
conformité avec les normes 7  
conformité avec les normes ODBC 7  
constantes dans les expressions SQL 25  
CREATE INDEX, instruction SQL 23  
CREATE TABLE, instruction SQL 21  
curseurs dans ODBC 14

## D

DEFAULT (clause SQL) 21  
DELETE, instruction SQL 17  
données binaires, utilisation dans SELECT 15  
DROP INDEX (instruction SQL) 24

## E

erreurs de syntaxe 39  
espaces finaux 26  
expressions SQL 24  
    constantes 25  
    fonctions 30  
    noms de rubriques 25  
    notation en virgule flottante ou scientifique 26  
    opérateurs de caractères 26  
    opérateurs de dates 27  
    opérateurs logiques 29  
    opérateurs numériques 26  
    opérateurs relationnels 27  
    ordre de priorité des opérateurs 29  
EXTERNAL (clause SQL) 22

## F

FETCH FIRST (clause SQL) 14  
FieldClass 37  
FieldId 37  
FieldName 37  
FieldReps 37  
FieldType 37

FileMaker\_Fields 37  
FileMaker\_Tables 37  
fonction ABS 33  
fonction ATAN 33  
fonction ATAN2 33  
fonction CASE WHEN 36  
fonction CAST 16  
fonction CEIL 33  
fonction CEILING 33  
fonction CHR 31  
fonction COALESCE 36  
fonction CURDATE 35  
fonction CURRENT\_DATE 35  
fonction CURRENT\_TIME 35  
fonction CURRENT\_TIMESTAMP 35  
fonction CURRENT\_USER 31  
fonction CURTIME 35  
fonction CURTIMESTAMP 35  
fonction DATE 35  
fonction DATEVAL 35  
fonction DAY 33  
fonction DAYNAME 31  
fonction DAYOFWEEK 33  
fonction DEG 33  
fonction DEGREES 33  
fonction ExecuterSQL 6  
fonction EXP 33  
fonction FLOOR 34  
fonction GetAs 16  
fonction HOUR 34  
fonction INT 34  
fonction LEFT 32  
fonction LENGTH 34  
fonction LN 34  
fonction LOG 34  
fonction LOWER 31  
fonction LTRIM 31  
fonction MAX 34  
fonction MIN 34  
fonction MINUTE 34  
fonction MOD 33  
fonction MONTH 34  
fonction MONTHNAME 32  
fonction NULLIF 36  
fonction NUMVAL 34  
fonction PI 34  
fonction PutAs 18, 20  
fonction RADIANS 34  
fonction RIGHT 32  
fonction ROUND 34  
fonction RTRIM 31  
fonction SECOND 34

fonction SIGN 34  
 fonction SIN 34  
 fonction SPACE 32  
 fonction SQRT 34  
 fonction STRVAL 32  
 fonction SUBSTR 32  
 fonction SUBSTRING 32  
 fonction TAN 34  
 fonction TIME 32  
 fonction TIMESTAMPVAL 35  
 fonction TIMEVAL 32  
 fonction TODAY 35  
 fonction TRIM 31  
 fonction UPPER 31  
 fonction USERNAME 32  
 fonction YEAR 34  
 fonctions dans les expressions SQL 30  
 fonctions de chaînes 31  
 fonctions statistiques SQL 30  
 FOR UPDATE (clause SQL) 14  
 formats d'heure 25  
 formats d'horodatage 25  
 formats de date 25  
 FROM (clause SQL) 9  
 FULL OUTER JOIN 10

## G

GROUP BY (clause SQL) 11

## H

HAVING (clause SQL) 12

## I

INNER JOIN 10  
 INSERT (instruction SQL) 17  
 instructions SQL  
   ALTER TABLE 23  
   CREATE INDEX 23  
   CREATE TABLE 21  
   DELETE 17  
   DROP INDEX 24  
   INSERT 17  
   mots-clés réservés 39  
   prises en charge par les pilotes clients 7  
   SELECT 8  
   TRUNCATE TABLE 22  
   UPDATE 19

## J

jointure 10

## L

LEFT OUTER JOIN 10

## M

mises à jour et suppressions positionnées 14  
 ModCount 37  
 mots-clés SQL réservés 39  
 mots-clés, SQL réservés 39

## N

noms de rubriques dans les expressions SQL 25  
 NOT NULL (clause SQL) 22  
 notation en virgule flottante dans les expressions SQL 26  
 notation scientifique dans les expressions SQL 26

## O

OFFSET (clause SQL) 13  
 opérateur ALL 27  
 opérateur AND 29  
 opérateur ANY 27  
 opérateur BETWEEN 27  
 opérateur DISTINCT 8  
 opérateur EXISTS 27  
 opérateur IN 27  
 opérateur IS NOT NULL 27  
 opérateur IS NULL 27  
 opérateur LIKE 27  
 opérateur NOT 29  
 opérateur NOT IN 27  
 opérateur NOT LIKE 27  
 opérateur OR 29  
 opérateurs de caractères dans les expressions SQL 26  
 opérateurs de dates dans les expressions SQL 27  
 opérateurs logiques dans les expressions SQL 29  
 opérateurs numériques dans les expressions SQL 26  
 opérateurs relationnels dans les expressions SQL 27  
 ORDER BY (clause SQL) 13  
 ordre de priorité des opérateurs dans les expressions SQL 29  
 ordre de tri 13  
 OUTER JOIN 10

## P

pilote client JDBC  
   prise en charge Unicode 7  
   tables externes 7  
 pilote client ODBC  
   prise en charge Unicode 7  
   tables externes 7  
 PREVENT INDEX CREATION 24  
 prise en charge Unicode 7

## R

rangées homologues 14  
 RIGHT OUTER JOIN 10

rubrique Conteneur  
  avec instruction CREATE TABLE 22  
  avec instruction INSERT 18  
  avec instruction SELECT 16  
  avec instruction UPDATE 20  
  avec la fonction PutAs 18  
  stockée en externe 22  
rubriques multivaluées 18, 21

## S

SELECT (instruction SQL) 8  
  chaîne vide 15  
  type de données 15  
  type de données BLOB 15  
sous-requêtes 18  
SQL, expressions 24  
SQL, fonctions statistiques 30  
SQL-92 7

## T

TableId 37  
TableName 37  
tables externes 7  
tables système 37  
TRUNCATETABLE (instruction SQL) 22  
type de données BLOB, utilisation dans SELECT 15  
type de données SQL\_C\_WCHAR 7

## U

UNION (opérateur SQL) 12  
UNIQUE (clause SQL) 22  
UPDATE (instruction SQL) 19

## V

valeur non nulle dans les colonnes 18  
valeur nulle 18  
VALUES (clause SQL) 18

## W

WHERE, clause SQL 11  
WITH TIES (clause SQL) 14