

FileMaker® 16

Referencia de SQL



FileMaker®
An Apple Subsidiary

© 2013-2017 FileMaker, Inc. Reservados todos los derechos.

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

FileMaker, FileMaker Go y el logotipo de la carpeta de archivos son marcas comerciales de FileMaker, Inc. registradas en los EE. UU. y en otros países. FileMaker WebDirect y FileMaker Cloud son marcas comerciales de FileMaker, Inc. Las demás marcas comerciales pertenecen a sus respectivos propietarios.

La documentación de FileMaker está protegida por derechos de autor. No está autorizado a realizar copias adicionales ni distribuir esta documentación sin el consentimiento expreso por escrito de FileMaker. Esta documentación sólo puede utilizarse con una copia del software de FileMaker que tenga una licencia válida.

Todas las personas, las empresas, las direcciones de correo electrónico y las URL incluidas en los ejemplos son ficticias y cualquier parecido con personas, empresas, direcciones de correo electrónico o URL existentes es una mera coincidencia. Los créditos aparecen en los documentos de agradecimientos que acompañan a este software. La mención de productos y direcciones URL de terceros es meramente informativa y no representa ningún tipo de garantía ni recomendación. FileMaker, Inc. no asume ninguna responsabilidad respecto al rendimiento de estos productos.

Para obtener más información, visite nuestro sitio Web en <http://www.filemaker.com/es/>.

Edición: 01

Contenido

Capítulo 1

Introducción

Acerca de esta referencia	5
Acerca de SQL	5
Uso de una base de datos de FileMaker como fuente de datos	5
Uso de la función ExecuteSQL	6

Capítulo 2

Estándares admitidos

Compatibilidad con caracteres Unicode	7
Secuencias SQL	7
Secuencia SELECT	8
Cláusulas SQL	9
Cláusula FROM	9
Cláusula WHERE	11
Cláusula GROUP BY	11
Cláusula HAVING	12
Operador UNION	12
Cláusula ORDER BY	13
Cláusulas OFFSET y FETCH FIRST	13
Cláusula FOR UPDATE	14
Secuencia DELETE	17
Secuencia INSERT	17
Secuencia UPDATE	19
Secuencia CREATE TABLE	20
Secuencia TRUNCATE TABLE	22
Secuencia ALTER TABLE	22
Secuencia CREATE INDEX	23
Secuencia DROP INDEX	23
Expresiones SQL	24
Nombres de campo	24
Constantes	24
Notación exponencial/científica	26
Operadores numéricos	26
Operadores de caracteres	26
Operadores de fecha	26
Operadores relacionales	27
Operadores lógicos	28
Prioridad de operadores	29

Funciones SQL	29
Funciones de agregación	30
Funciones que devuelven cadenas de caracteres	31
Funciones que devuelven números	33
Funciones que devuelven fechas	34
Funciones condicionales	35
Objetos del sistema de FileMaker	36
Tablas del sistema de FileMaker	36
Columnas del sistema de FileMaker	37
Palabras clave de SQL reservadas	38
<i>Índice</i>	41

Capítulo 1

Introducción

Como creador de bases de datos, puede utilizar FileMaker Pro para crear soluciones de base de datos sin tener ningún conocimiento de SQL. Pero, si cuenta con algunas nociones de SQL, puede utilizar una base de datos de FileMaker como fuente de datos de ODBC o JDBC, compartiendo sus datos con otras aplicaciones mediante ODBC y JDBC. También puede utilizar la función ExecuteSQL de FileMaker Pro para recuperar datos de cualquier instancia de tabla de una base de datos de FileMaker Pro.

Esta referencia describe las secuencias SQL y los estándares admitidos por FileMaker. Los controladores de cliente ODBC y JDBC de FileMaker admiten todas las secuencias SQL que se describen en esta referencia. La función ExecuteSQL de FileMaker Pro admite solo la secuencia SELECT.

Acerca de esta referencia

- Para obtener información sobre la utilización de ODBC y JDBC con versiones anteriores de FileMaker Pro, consulte el [Centro de documentación del producto](#).
- En esta referencia se asume que conoce los conceptos básicos del uso de las funciones de FileMaker Pro, la codificación de aplicaciones ODBC y JDBC, así como con la creación de consultas SQL. Deberá consultar otra publicación para obtener más información sobre estos temas.
- En esta documentación se utiliza "FileMaker Pro" para hacer referencia tanto a FileMaker Pro como a FileMaker Pro Advanced, a no ser que se describan funcionalidades específicas de FileMaker Pro Advanced.

Acerca de SQL

Structured Query Language o SQL (por sus siglas en inglés) es un lenguaje de programación diseñado para consultar datos de una base de datos relacional. La principal secuencia que se utiliza para consultar una base de datos es la secuencia SELECT.

Además del lenguaje para consultar una base de datos, SQL proporciona secuencias para llevar a cabo la manipulación de datos, que le permite añadir, actualizar y eliminar datos.

SQL también proporciona secuencias para la definición de datos. Estas secuencias permiten la creación y modificación de tablas e índices.

Las secuencias SQL y los estándares admitidos por FileMaker se describen en capítulo 2, "Estándares admitidos"

Uso de una base de datos de FileMaker como fuente de datos

Cuando se aloja una base de datos de FileMaker como una fuente de datos ODBC o JDBC, los datos de FileMaker se pueden compartir con aplicaciones compatibles con ODBC y JDBC. Las aplicaciones se conectan con la fuente de datos de FileMaker mediante los controladores de cliente de FileMaker, crean y ejecutan las consultas de SQL mediante ODBC o JDBC, y procesan los datos recuperados de la solución de base de datos de FileMaker.

Consulte la [Guía de ODBC y JDBC de FileMaker](#) para obtener amplia información acerca de cómo puede utilizar el software de FileMaker como fuente de datos para aplicaciones ODBC y JDBC.

Los controladores de cliente ODBC y JDBC de FileMaker admiten todas las secuencias SQL que se describen en esta referencia.

Uso de la función ExecuteSQL

La función ExecuteSQL de FileMaker Pro le permite recuperar datos de instancias de tabla que se nombran en el gráfico de relaciones pero que son independientes de cualquier relación definida. Puede recuperar datos de varias tablas sin crear uniones de tablas o relación entre tablas. En algunos casos, puede reducir la complejidad del gráfico de relaciones utilizando la función ExecuteSQL.

Los campos que consulte con la función ExecuteSQL no tienen por qué estar en una presentación, de manera que pueda utilizar la función ExecuteSQL para recuperar datos independientes de cualquier contexto de presentación. Dada esta independencia de contexto, el uso de la función ExecuteSQL en guiones puede mejorar la portabilidad de los mismos. Puede utilizar la función ExecuteSQL en cualquier situación en que se realicen cálculos, como al generar gráficos e informes.

La función ExecuteSQL admite solo la secuencia SELECT, que se describe en la sección “Secuencia SELECT” en la página 8.

Además, la función ExecuteSQL acepta solo los formatos de hora y fecha ISO de sintaxis SQL-92 sin corchetes ({}). La función ExecuteSQL no acepta las constantes de fecha y hora, hora y fecha de formato ODBC/JDBC con corchetes.

Para obtener información acerca de la sintaxis y el uso de la función ExecuteSQL, consulte la [Ayuda de FileMaker Pro](#).

Capítulo 2

Estándares admitidos

Utilice los controladores de cliente ODBC y JDBC de FileMaker para acceder una solución de base de datos de FileMaker desde una aplicación compatible con ODBC o JDBC. La solución de base de datos de FileMaker solo se puede alojar en FileMaker Pro o FileMaker Server.

- El controlador de cliente ODBC admite ODBC 3.0 Nivel 1.
- El controlador de cliente JDBC ofrece una compatibilidad parcial con la especificación JDBC 3.0.
- Los controladores de cliente ODBC y JDBC admiten la compatibilidad con el nivel de entrada SQL-92, con algunas características intermedias de SQL-92.

Compatibilidad con caracteres Unicode

Los controladores de clientes ODBC y JDBC admiten la API Unicode. Sin embargo, si está creando una aplicación personalizada que utiliza los controladores de cliente, utilice ASCII para nombres de campos, nombres de tablas y nombres de archivo (por si se utiliza una aplicación o herramienta de consulta distinta de Unicode).

Nota Para introducir y recuperar datos Unicode, utilice `SQL_C_WCHAR`.

Secuencias SQL

Los controladores de clientes ODBC y JDBC proporcionan compatibilidad para las siguientes secuencias SQL.

- SELECT (página 8)
- DELETE (página 17)
- INSERT (página 17)
- UPDATE (página 19)
- CREATE TABLE (página 20)
- TRUNCATE TABLE (página 22)
- ALTER TABLE (página 22)
- CREATE INDEX (página 23)
- DROP INDEX (página 23)

Los controladores de clientes también admiten la asignación de los tipos de datos de FileMaker a tipos de datos SQL de JDBC y SQL de ODBC. Consulte la [Guía de ODBC y JDBC de FileMaker](#) para obtener información acerca de las conversiones de tipos de datos. Para obtener más información sobre la creación de consultas SQL, deberá consultar otra publicación.

Nota Los controladores de cliente ODBC y JDBC no son compatibles con los portales de FileMaker.

Secuencia SELECT

Utilice la secuencia `SELECT` para especificar qué columnas está solicitando. Indique después de la secuencia `SELECT` las expresiones de columna (similar a nombres de campos) que desee recuperar (por ejemplo, apellidos). Las expresiones pueden incluir operaciones matemáticas o manipulaciones de cadenas (por ejemplo, `SALARIO * 1,05`).

La secuencia `SELECT` puede utilizar diversas cláusulas:

```
SELECT [DISTINCT] { * | expresión_columna [[AS] alias_columna], ... }
FROM table_name [table_alias], ...
[ WHERE expr1 operador_rel expr2 ]
[ GROUP BY {expresión_columna, ...} ]
[ HAVING expr1 operador_rel expr2 ]
[ UNION [ALL] (SELECT...) ]
[ ORDER BY {expresión_ordenación [DESC | ASC]}, ... ]
[ OFFSET n {ROWS | ROW} ]
[ FETCH FIRST [ n [ PERCENT ] ] { ROWS | ROW } { ONLY | WITH TIES } ]
[ FOR UPDATE [OF {expresión_columna, ...}] ]
```

Los elementos escritos entre corchetes son opcionales.

`alias_columna` se puede utilizar para asignar a la columna un nombre más descriptivo o para abreviar un nombre de columna muy largo.

Ejemplo

Asigne el alias `departamento` a la columna `dept`.

```
SELECT dep AS departamento FROM emp
```

Los nombres de campo pueden llevar delante el nombre de la tabla o el alias de la tabla. Por ejemplo, `EMP.APELLIDOS` o `E.APELLIDOS`, siendo `E` el alias de la tabla `EMP`.

El operador `DISTINCT` puede ir delante de la primera expresión de columna. Este operador elimina las filas duplicadas del resultado de una consulta.

Ejemplo

```
SELECT DISTINCT dep FROM emp
```

Cláusulas SQL

Los controladores de clientes ODBC y JDBC proporcionan compatibilidad con las siguientes cláusulas SQL.

Utilice esta cláusula SQL	Para
FROM (página 9)	Indicar qué tablas se usan en la secuencia <code>SELECT</code> .
WHERE (página 11)	Especificar las condiciones que deben cumplir los registros para ser recuperados (como una petición de búsqueda de FileMaker Pro).
GROUP BY (página 11)	Especificar los nombres de uno o varios campos según los cuales se deben agrupar los valores devueltos. Esta cláusula se utiliza para devolver un conjunto de valores sumados devolviendo una fila para cada grupo (como un subsumario de FileMaker Pro).
HAVING (página 12)	Especificar las condiciones para los grupos de registros (por ejemplo, mostrar sólo los departamentos con salarios que sumen más de 200.000 €).
UNION (página 12)	Combine los resultados de dos o más secuencia <code>SELECT</code> en un único resultado.
ORDER BY (página 13)	Indique cómo se ordenan los registros.
OFFSET (página 13)	Determine el número de filas que se salten antes de empezar a recuperar filas.
FETCH FIRST (página 13)	Especifique el número de filas que se vayan a recuperar. No se devuelven más filas del número especificado aunque se pueden devolver menos filas si la consulta encuentra un número menor al especificado.
FOR UPDATE (página 14)	Realice actualizaciones posicionadas o eliminaciones posicionadas mediante cursores SQL.

Nota Si intenta recuperar datos desde una tabla sin columnas, la secuencia `SELECT` no devuelve nada.

Cláusula FROM

La cláusula `FROM` indica las tablas que se utilizan en la secuencia `SELECT`. El formato es:

```
FROM nombre_tabla [alias_tabla] [, nombre_tabla [alias_tabla]]
```

`nombre_tabla` es el nombre de una tabla de la base de datos actual. El nombre de la tabla debe comenzar con un carácter alfabético. Si el nombre de la tabla comienza por otro que no sea un carácter alfabético, escríbalo entre comillas dobles (identificador entre comillas).

`alias_tabla` se puede utilizar para proporcionar a la tabla un nombre más descriptivo, para abreviar un nombre de tabla demasiado largo o para incluir la misma tabla más de una vez en una consulta (por ejemplo, en autouniones).

Los nombres de campo deben comenzar con un carácter alfabético. Si el nombre del campo comienza por otro que no sea un carácter alfabético, escríbalo entre comillas dobles (identificador entre comillas).

Ejemplo

La secuencia `ExecuteSQL` para el campo con el nombre `_APELLIDOS` es:

```
SELECT "_APELLIDOS" from emp
```

Los nombres de campo pueden llevar delante el nombre de la tabla o el alias de la tabla.

Ejemplo

Dada la especificación de tabla `FROM empleado E`, puede hacer referencia al campo `APELLIDOS` como `E.APELLIDOS`. Los alias de tabla se deben utilizar si la secuencia `SELECT` une una tabla consigo misma.

```
SELECT * FROM empleado E, empleado F WHERE E.id_director = F.id_empleado
```

El signo igual (=) sólo incluye las filas coincidentes en los resultados.

Si une más de una tabla y desea desechar todas las filas que no tengan filas correspondientes en ambas tablas de origen, puede utilizar `INNER JOIN`.

Ejemplo

```
SELECT *
  FROM Vendedores INNER JOIN Datos_ventas
  ON Vendedores.ID_Vendedor = Datos_ventas.ID_Vendedor
```

Si va a unir dos tablas, pero no desea desechar filas de la primera tabla (la tabla de la izquierda), puede utilizar `LEFT OUTER JOIN`.

Ejemplo

```
SELECT *
  FROM Vendedores LEFT OUTER JOIN Datos_ventas
  ON Vendedores.ID_Vendedor = Datos_ventas.ID_Vendedor
```

Cada una de las filas de la tabla "Vendedores" aparecerá en la tabla unida.

Notas

- `RIGHT OUTER JOIN` no se admite de momento.
- `FULL OUTER JOIN` no se admite de momento.

Cláusula WHERE

La cláusula `WHERE` especifica las condiciones que deben cumplir los registros para ser recuperados. Esta cláusula contiene condiciones de la forma:

```
WHERE expr1 operador_rel expr2
```

`expr1` y `expr2` pueden ser nombres de campos, valores constantes o expresiones.

`operador_rel` es el operador relacional que enlaza las dos expresiones.

Ejemplo

Recupere los nombres de los empleados que ganan 20.000 € o más.

```
SELECT apellidos,nombre FROM emp WHERE salario >= 20000
```

La cláusula `WHERE` puede además utilizar expresiones como las siguientes:

```
WHERE expr1 IS NULL
```

```
WHERE NOT expr2
```

Nota Si selecciona nombres totalmente calificados en la lista `SELECT` (proyección), también debe utilizar nombres totalmente calificados en la cláusula `WHERE` relacionada.

Cláusula GROUP BY

La cláusula `GROUP BY` especifica los nombres de uno o varios campos según los cuales se deben agrupar los valores devueltos. Esta cláusula se utiliza para devolver un conjunto de valores sumados. Tiene el siguiente formato:

```
GROUP BY columnas
```

El ámbito de la cláusula `GROUP BY` es la expresión de tabla de la cláusula `FROM`. Por lo tanto, las expresiones de columna especificadas por `columnas` deben proceder de las tablas especificadas en la cláusula `FROM`. Una expresión de columna pueden ser uno o más nombres de campo de la tabla de base de datos separados por comas.

Ejemplo

Sume los salarios de cada departamento.

```
SELECT id_dep, SUM (salario) FROM emp GROUP BY id_dep
```

Esta secuencia devuelve una fila para cada ID de departamento distinto. Cada fila contiene el ID de departamento y la suma de los salarios de los empleados que conforman el departamento.

Cláusula HAVING

La cláusula `HAVING` le permite especificar las condiciones para los grupos de registros (por ejemplo, mostrar solo los departamentos con salarios que sumen más de 200.000 €). Tiene el siguiente formato:

```
HAVING expr1 operador_rel expr2
```

`expr1` y `expr2` pueden ser nombres de campos, valores constantes o expresiones. Estas expresiones no tienen que coincidir con una expresión de columna en la cláusula `SELECT`.

`operador_rel` es el operador relacional que enlaza las dos expresiones.

Ejemplo

Devuelva solamente los departamentos cuyas sumas de salarios son superiores a 200.000 €.

```
SELECT id_dep, SUM (salario) FROM emp
      GROUP BY id_dep HAVING SUM (salario) > 200000
```

Operador UNION

El operador `UNION` combina los resultados de dos o más secuencias `SELECT` en un único resultado. El resultado único son todos los registros devueltos desde las secuencias `SELECT`. De forma predeterminada, los registros duplicados no se devuelven. Para devolver registros duplicados, utilice la palabra clave `ALL` (`UNION ALL`). El formato es:

```
secuencia SELECT UNION [ALL] secuencia SELECT
```

Cuando se utiliza el operador `UNION`, las listas de selección de cada secuencia `SELECT` deben tener el mismo número de expresiones de columna, con los mismos tipos de datos y deben especificarse en el mismo orden.

Ejemplo

```
SELECT apellidos, salario, fecha_contratación FROM emp UNION SELECT nombre,
      paga, cumpleaños FROM persona
```

El siguiente ejemplo no es válido, pues los tipos de datos de las expresiones de columna son diferentes (`SALARIO` de `EMP` tiene un tipo de datos diferente a `APELLIDOS` de `SUBIDAS`). Este ejemplo tiene el mismo número de expresiones de columna en cada secuencia `SELECT`, pero las expresiones no tienen el mismo orden por tipo de datos.

Ejemplo

```
SELECT apellidos, salario FROM emp UNION SELECT salario, apellidos FROM
      subidas
```

Cláusula ORDER BY

La cláusula `ORDER BY` indica cómo se van a ordenar los registros. Si la secuencia `SELECT` no incluye una cláusula `ORDER BY`, es posible que los registros se devuelvan en cualquier orden.

El formato es:

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

`expresión_ordenación` puede ser el nombre de campo o el número de posición de la expresión de columna que se va a utilizar. De forma predeterminada, se realiza un ordenamiento ascendente (`ASC`).

Ejemplos

Ordene por apellidos y, a continuación, por nombre.

```
SELECT id_emp, apellidos, nombre FROM emp ORDER BY apellidos, nombre
```

En el segundo ejemplo se utilizan los números de posición 2 y 3 para obtener la misma ordenación que en el ejemplo anterior, que especificaba de forma explícita apellidos y nombre.

```
SELECT id_emp, apellidos, nombre FROM emp ORDER BY 2,3
```

Nota FileMaker SQL utiliza el tipo de ordenación binario Unicode, que es diferente al tipo de ordenación de FileMaker Pro utilizado con la ordenación en función del idioma o la ordenación predeterminada neutra respecto al idioma.

Cláusulas OFFSET y FETCH FIRST

Las cláusulas `OFFSET` y `FETCH FIRST` se utilizan para devolver un rango especificado de filas que empiezan a partir de un determinado punto dentro de un conjunto de resultados. La capacidad para limitar las filas recuperadas de grandes conjuntos de resultados le permite pasar de una página de datos a otra y mejora la eficacia.

La cláusula `OFFSET` indica el número de filas que se saltan antes de empezar a devolver datos. Si no se utiliza la cláusula `OFFSET` en una secuencia `SELECT`, la fila de inicio es 0. La cláusula `FETCH FIRST` especifica el número de filas que se van a devolver, bien como un valor entero sin signo mayor que o igual a 1, bien como un porcentaje, desde el punto de inicio indicado en la cláusula `OFFSET`. Si se utilizan ambas cláusulas, `OFFSET` y `FETCH FIRST`, en una secuencia `SELECT`, la cláusula `OFFSET` debe ir primero.

Las cláusulas `OFFSET` y `FETCH FIRST` no se admiten en subconsultas.

Formato OFFSET

El formato `OFFSET` es:

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

`n` es un número entero sin signo. Si `n` es mayor que el número de filas que se devuelven en el conjunto de resultados, entonces no se devuelve nada y no aparece ningún mensaje de error.

`ROWS` es lo mismo que `ROW`.

Formato FETCH FIRST

El formato `FETCH FIRST` es:

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

`n` es el número de filas que se van a devolver. El valor predeterminado es 1 si `n` se omite.

`n` es un entero sin signo superior o igual 1, a menos que incluya `PERCENT` tras él. Si a `n` le sigue `PERCENT`, el valor puede ser tanto un valor fraccional positivo como un valor entero sin signo.

`ROWS` es lo mismo que `ROW`.

`WITH TIES` se debe utilizar con la cláusula `ORDER BY`.

`WITH TIES` permite que se devuelvan más filas de las especificadas en el valor de conteo `FETCH` dado que las filas de pares, aquellas que no son distintas en base a la cláusula `ORDER BY`, también se devuelven.

Ejemplos

Devuelva información de la fila 26 del conjunto de resultados ordenado por apellidos y, a continuación por nombre.

```
SELECT id_emp, apellidos, nombre FROM emp ORDER BY apellidos, nombre OFFSET 25 ROWS
```

Especifique que quiere que se devuelvan solo diez filas:

```
SELECT id_emp, apellidos, nombre FROM emp ORDER BY apellidos, nombre OFFSET 25 ROWS FETCH FIRST 10 ROWS ONLY
```

Devuelva las diez filas y sus filas de pares (filas que no son distintas en base a la cláusula `ORDER BY`).

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

Cláusula FOR UPDATE

La cláusula `FOR UPDATE` bloquea registros para actualizaciones posicionadas o eliminaciones posicionadas mediante los cursores SQL. El formato es:

```
FOR UPDATE [OF expresiones_columna]
```

`expresiones_columna` es una lista de nombres de campos en una tabla de la base de datos que desea actualizar, separados por una coma. `expresiones_columna` es opcional y se ignora.

Ejemplo

Devuelva todos los registros de la base de datos de empleados que tengan un valor del campo `SALARIO` superior a 20.000 €.

```
SELECT * FROM emp WHERE salario > 20000
FOR UPDATE OF apellidos, nombre, salario
```

Cuando se recupera cada registro, se bloquea. Si el registro se actualiza o elimina, el bloqueo se mantiene hasta que consigne el cambio. En caso contrario, el bloqueo se levanta al recuperar el siguiente registro.

Ejemplos

Uso	SQL de ejemplo
constante de texto	<code>SELECT 'CatDog' FROM Vendedores</code>
constante numérica	<code>SELECT 999 FROM Vendedores</code>
constante de fecha	<code>SELECT DATE '2019-06-05' FROM Vendedores</code>
constante de hora	<code>SELECT TIME '02:49:03' FROM Vendedores</code>
constante de fecha y hora	<code>SELECT TIMESTAMP '2019-06-05 02:49:03' FROM Vendedores</code>
columna de texto	<code>SELECT Nombre_Empresa FROM Datos_ventas</code> <code>SELECT DISTINCT Nombre_Empresa FROM Datos_ventas</code>
columna numérica	<code>SELECT Cantidad FROM Datos_ventas</code> <code>SELECT DISTINCT Cantidad FROM Datos_ventas</code>
columna de fecha	<code>SELECT Fecha_Venta FROM Datos_ventas</code> <code>SELECT DISTINCT Fecha_Venta FROM Datos_ventas</code>
columna de hora	<code>SELECT Hora_Venta FROM Datos_ventas</code> <code>SELECT DISTINCT Hora_Venta FROM Datos_ventas</code>
columna de fecha y hora	<code>SELECT Fecha_Hora_Venta FROM Datos_ventas</code> <code>SELECT DISTINCT Fecha_Hora_Venta FROM Datos_ventas</code>
columna BLOB ^a	<code>SELECT Empresa_folletos FROM Datos_ventas</code> <code>SELECT GETAS(Empresa_Logo, 'JPEG') FROM Datos_ventas</code>
Comodín *	<code>SELECT * FROM Vendedores</code> <code>SELECT DISTINCT * FROM Vendedores</code>

a. Un BLOB es un campo contenedor de un archivo de base de datos de FileMaker.

Notas de los ejemplos

Una `columna` es una referencia a un campo en el archivo de base de datos de FileMaker. (El campo puede contener muchos valores distintos).

El carácter comodín asterisco (*) es una forma abreviada de indicar "todo". En el ejemplo `SELECT * FROM Vendedores`, el resultado son todas las columnas de la tabla `Vendedores`. En el ejemplo `SELECT DISTINCT * FROM Vendedores`, el resultado son todas las filas no repetidas de la tabla `Vendedores`.

- FileMaker no almacena datos de cadenas vacías, de manera que las siguientes consultas siempre se devuelven sin registros:


```
SELECT * FROM prueba WHERE c = ''
SELECT * FROM prueba WHERE c <> ''
```
- Si utiliza `SELECT` con datos binarios, debe utilizar la función `GetAs()` para especificar la secuencia que se va a devolver. Para obtener más información, consulte la siguiente sección "Recuperación del contenido de un campo contenedor: Función `CAST()` y función `GetAs()`".

Recuperación del contenido de un campo contenedor: Función CAST() y función GetAs()

Puede recuperar información de referencia de archivos, datos binarios o datos de un tipo de archivo específico de un campo contenedor.

- Para recuperar la información de referencia de archivo de un campo contenedor, como la ruta a un archivo, imagen o película Quicktime, utilice la función `CAST()` con una secuencia `SELECT`.
- Si existen datos de archivo o los datos binarios JPEG, la secuencia `SELECT` con `GetAS(nombre de campo, 'JPEG')` recupera los datos en formato binario; en caso contrario, la secuencia `SELECT` con nombre de campo devuelve `NULL`.

Ejemplo

Utilice la función `CAST()` con una secuencia `SELECT` para recuperar la información de referencia de archivos.

```
SELECT CAST(Empresa_folletos AS VARCHAR) FROM Datos_ventas
```

En este ejemplo, si:

- Ha insertado un archivo en un campo contenedor mediante FileMaker Pro pero ha almacenado sólo una referencia al archivo, la secuencia `SELECT` recupera la información de la referencia del archivo como tipo `SQL_VARCHAR`.
- Ha insertado el contenido de un archivo en el campo contenedor mediante FileMaker Pro, la secuencia `SELECT` recupera el nombre del archivo.
- Ha importado un archivo en un campo contenedor desde otra aplicación, la secuencia `SELECT` muestra '?' (el archivo se muestra como **Untitled.dat** en FileMaker Pro).

Puede utilizar la secuencia `SELECT` con la función `GetAs()` para recuperar los datos en formato binario de las siguientes formas:

- Al utilizar la función `GetAs()` con la opción `DEFAULT`, recuperará la secuencia principal del contenedor sin necesidad de definir explícitamente el tipo de secuencia.

Ejemplo

```
SELECT GetAs(Empresa_Folletos, DEFAULT) FROM Datos_ventas
```

- Para recuperar un tipo de secuencia individual de un campo contenedor, utilice la función `GetAs()` con el tipo del archivo según la manera en que se insertaron los datos en el campo contenedor en FileMaker Pro.

Ejemplo

Si los datos se insertaron mediante el comando **Insertar > Archivo**, especifique `'FILE'` en la función `GetAs()`.

```
SELECT GetAs(Empresa_Folletos, 'FILE') FROM Datos_ventas
```

Ejemplo

Si los datos se insertaron mediante el comando **Insertar > Imagen**, la función arrastrar y soltar o se pegaron desde el portapapeles, especifique uno de los tipos de archivos que se muestran en la siguiente tabla, por ejemplo, 'JPEG'.

```
SELECT GetAs(Logo_Empresa, 'JPEG') FROM Iconos_Empresa
```

Tipo de archivo	Descripción
'GIFf'	Formato de intercambio de gráficos
'JPEG'	Imágenes fotográficas
'TIFF'	Formato de archivos raster para imágenes digitales
'PDF'	Formato de documento portátil
'PNGf'	Formato de imagen de mapa de bits

Secuencia DELETE

Utilice la secuencia `DELETE` para eliminar registros de una tabla de base de datos. El formato de la secuencia `DELETE` es:

```
DELETE FROM nombre_tabla [ WHERE { condiciones } ]
```

Nota La cláusula `WHERE` determina los registros que se van a eliminar. Si no incluye la palabra clave `WHERE`, se eliminan todos los registros de la tabla (pero la tabla queda intacta).

Ejemplo

Elimine un registro de la tabla `emp`.

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

Cada secuencia `DELETE` elimina todos los registros que cumplen las condiciones de la cláusula `WHERE`. En este caso, se eliminan todos los registros que tengan el ID de empleado `E10001`. Como los ID de empleado son únicos en la tabla `Empleado`, sólo se elimina un registro.

Secuencia INSERT

Utilice la secuencia `INSERT` para crear registros en una tabla de base de datos. Puede especificar:

- Una lista de valores para insertar como nuevo registro
- Una secuencia `SELECT` que copia datos de otra tabla para insertarlos como conjunto de registros nuevos

El formato de la secuencia `INSERT` es:

```
INSERT INTO nombre_tabla [(nombre_columna, ...)] VALUES (expr, ...)
```

`nombre_columna` es una lista opcional de nombres de columnas que proporciona el nombre y el orden de las columnas cuyos valores se han especificado en la cláusula `VALUES`. Si omite `nombre_columna`, las expresiones de valor (`expr`) deben proporcionar valores para todas las columnas definidas en la tabla y deben encontrarse en el mismo orden en que se definen las columnas para la tabla. `nombre_columna` también puede especificar una repetición de campo, por ejemplo `lastDates[4]`.

`expr` es la lista de expresiones que proporcionan los valores para las columnas del nuevo registro. Normalmente, las expresiones son valores constantes para las columnas (pero también pueden ser subconsultas). Debe escribir los valores de las cadenas de caracteres entre comillas sencillas (`'`). Para incluir un signo de comillas sencillas en un valor de cadena de caracteres escrito entre comillas sencillas, utilice un signo de comillas dobles (por ejemplo, `'O' 'Neal'`).

Las subconsultas deben escribirse entre paréntesis.

Ejemplo

Inserte una lista de expresiones.

```
INSERT INTO emp (apellidos, nombre, id_emp, salario, fecha_contratación)
VALUES ('Smith', 'John', 'E22345', 27500, DATE '2019-06-05')
```

Cada secuencia `INSERT` añade un registro a la tabla de la base de datos. En este caso, se ha añadido un registro a la tabla de base de datos de empleados, `emp`. Se han especificado valores para cinco columnas. A las demás columnas de la tabla se les asigna un valor en blanco, que significa Nulo.

Nota En los campos contenedores, puede insertar sólo texto (`INSERT`), a menos que cree una secuencia con parámetros y envíe los datos desde su aplicación. Para usar datos binarios, puede asignar simplemente el nombre de archivo escribiéndolo entre comillas simples o utilizar la función `PutAs()`. A la hora de especificar el nombre de archivo, el tipo de archivo se deduce de la extensión del mismo:

```
INSERT INTO nombre_tabla (nombre_contenedor) VALUES
(? AS 'nombrearchivo.extensión archivo')
```

Los tipos de archivo que no se admitan se insertarán como `FILE`.

A la hora de utilizar la función `PutAs()`, especifique el tipo: `PutAs(col, 'tipo')`, donde el valor `tipo` es un tipo de archivo admitido tal y como se describe en “Recuperación del contenido de un campo contenedor: Función `CAST()` y función `GetAs()`” en la página 16.

La secuencia `SELECT` es una consulta que devuelve valores para cada valor de `nombre_columna` especificado en la lista de nombres de columnas. El uso de una secuencia `SELECT` en lugar de una lista de expresiones de valores le permite seleccionar un conjunto de filas de una tabla e insertarlo en otra tabla utilizando una única secuencia `INSERT`.

Ejemplo

Realice la inserción mediante la secuencia `SELECT`.

```
INSERT INTO emp1 (nombre, apellidos, id_emp, dep, salario)
SELECT nombre, apellidos, id_emp, dep, salario from emp
WHERE dep = 'D050'
```

En este tipo de secuencia `INSERT`, el número de columnas para insertar debe coincidir con el número de columnas de la secuencia `SELECT`. La lista de columnas que se van a insertar debe corresponder con las columnas de la secuencia `SELECT` del mismo modo que lo haría con una lista de expresiones de valores en el otro tipo de secuencia `INSERT`. Por ejemplo, la primera columna insertada corresponde con la primera columna seleccionada; la segunda insertada con la segunda seleccionada, etc.

El tamaño y el tipo de datos de estas columnas correspondientes deben ser compatibles. Cada columna de la lista `SELECT` debe tener un tipo de datos que acepte el controlador de cliente ODBC o JDBC en una secuencia `INSERT/UPDATE` normal de la columna correspondiente de la lista `INSERT`. Si el tamaño del valor de la columna de lista `SELECT` es mayor que el de la columna de lista `INSERT` correspondiente, los valores se truncan.

La secuencia `SELECT` se evalúa antes de que se inserten los valores.

Secuencia `UPDATE`

Utilice la secuencia `UPDATE` para cambiar los registros de una tabla de base de datos. El formato de la secuencia `UPDATE` es:

```
UPDATE nombre_tabla SET nombre_columna = expr, ... [ WHERE { condiciones } ]
```

`nombre_columna` es el nombre de la columna cuyo valor va a cambiar. En una secuencia se pueden cambiar varias columnas.

`expr` es el nuevo valor para la columna.

Normalmente, las expresiones son valores constantes para las columnas (pero también pueden ser subconsultas). Debe escribir los valores de las cadenas de caracteres entre comillas sencillas ('). Para incluir un signo de comillas sencillas en un valor de cadena de caracteres escrito entre comillas sencillas, utilice un signo de comillas dobles (por ejemplo, 'O' 'Neal').

Las subconsultas deben escribirse entre paréntesis.

La cláusula `WHERE` es cualquier cláusula válida. Determina qué registros se actualizan.

Ejemplo

La secuencia `UPDATE` en la tabla `emp`.

```
UPDATE emp SET salario=32000, exenc=1 WHERE id_emp = 'E10001'
```

La secuencia `UPDATE` cambia todos los registros que cumplen las condiciones de la cláusula `WHERE`. En este caso se cambian el salario y el estado de exención para todos los empleados que tengan el ID de empleado `E10001`. Como los ID de empleado son únicos en la tabla `Empleado`, sólo se actualiza un registro.

Ejemplo

La secuencia `UPDATE` en la tabla `emp` con una subconsulta.

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

En este caso, el salario se cambia al salario medio de la empresa para el empleado con el ID de empleado `E10001`.

Nota En los campos contenedores, puede actualizar sólo con texto (`UPDATE`), a menos que cree una secuencia con parámetros y envíe los datos desde su aplicación. Para usar datos binarios, puede asignar simplemente el nombre de archivo escribiéndolo entre comillas simples o utilizar la función `PutAs()`. A la hora de especificar el nombre de archivo, el tipo de archivo se deduce de la extensión del mismo:

```
UPDATE nombre_tabla SET (nombre_contenedor) = ? AS 'nombrearchivo.extensión archivo'
```

Los tipos de archivo que no se admitan se insertarán como `FILE`.

A la hora de utilizar la función `PutAs()`, especifique el tipo: `PutAs(col, 'tipo')`, donde el valor `tipo` es un tipo de archivo admitido tal y como se describe en “Recuperación del contenido de un campo contenedor: Función `CAST()` y función `GetAs()`” en la página 16.

Secuencia `CREATE TABLE`

Utilice la secuencia `CREATE TABLE` para crear una tabla en un archivo de base de datos. El formato de la secuencia `CREATE TABLE` es:

```
CREATE TABLE nombre_tabla ( lista_elemento_tabla [, lista_elemento_tabla...] )
```

Dentro de esta secuencia, debe especificar el nombre y el tipo de datos de cada columna.

- `nombre_tabla` es el nombre de la tabla. `nombre_tabla` tiene un límite de 100 caracteres. No debe haber una tabla definida con el mismo nombre. El nombre de la tabla debe comenzar con un carácter alfabético. Si el nombre de la tabla comienza por otro que no sea un carácter alfabético, escríbalo entre comillas dobles (identificador entre comillas).
- El formato de `lista_elemento_tabla` es:

```
nombre_campo tipo_campo [[repeticiones]]
[DEFAULT expr] [UNIQUE | NOT NULL | PRIMARY KEY | GLOBAL]
[EXTERNAL cadena_ruta_relativa [SECURE | OPEN cadena_ruta_calc]]
```

- `nombre_campo` es el nombre del campo. Los nombres de campo deben ser exclusivos y Los nombres de campo deben comenzar con un carácter alfabético. Si el nombre del campo comienza por otro que no sea un carácter alfabético, escríbalo entre comillas dobles (identificador entre comillas).

Ejemplo

La secuencia `CREATE TABLE` para el campo con el nombre `_APELLIDOS` es:

```
CREATE TABLE "_EMPLEADO" (ID INT PRIMARY KEY, "_NOMBRE" VARCHAR(20),
  "_APELLIDOSE" VARCHAR(20))
```

- Para el elemento `repeticiones` de la secuencia `CREATE TABLE`, especifique una repetición de campo incluyendo un número entre 1 y 32000 entre corchetes después del campo.

Ejemplo

```
EMPLEADO_ID INT[4]
APELLIDOS VARCHAR(20)[4]
```

- `tipo_campo` puede ser cualquiera de los siguientes: `NUMERIC`, `DECIMAL`, `INT`, `DATE`, `TIME`, `TIMESTAMP`, `VARCHAR`, `CHARACTER VARYING`, `BLOB`, `VARBINARY`, `LONGVARBINARY` o `BINARY VARYING`. En `NUMERIC` y `DECIMAL`, puede especificar la precisión y la escala. Por ejemplo: `DECIMAL(10,0)`. En `TIME` y `TIMESTAMP`, puede especificar la precisión. Por ejemplo: `TIMESTAMP(6)`. En `VARCHAR` y `CHARACTER VARYING`, puede especificar la longitud de la cadena.

Ejemplo

```
VARCHAR(255)
```

- La palabra clave `DEFAULT` le permite configurar un valor predeterminado para una columna. En `expr`, puede utilizar una expresión o un valor constante. Las expresiones permitidas son `USER`, `USERNAME`, `CURRENT_USER`, `CURRENT_DATE`, `CURDATE`, `CURRENT_TIME`, `CURTIME`, `CURRENT_TIMESTAMP`, `CURTIMESTAMP` y `NULL`.
- Al definir una columna como `UNIQUE`, se selecciona automáticamente la opción de validación **Único** para el campo correspondiente del archivo de base de datos de FileMaker.
- Al definir una columna como `NOT NULL`, se selecciona automáticamente la opción de validación **No vacío** para el campo correspondiente del archivo de base de datos de FileMaker. El campo se marca como **Valor requerido** en la pestaña **Campos** del cuadro de diálogo Gestionar base de datos de FileMaker Pro.
- Para definir una columna como un campo contenedor, utilice `BLOB`, `VARBINARY` o `BINARY VARYING` como `tipo_campo`.
- Para definir una columna como un campo contenedor que almacena datos de forma externa, utilice la palabra clave `EXTERNAL`. `cadena_ruta_relativa` define la carpeta en la que se almacenan de forma externa los datos en relación con la ubicación de la base de datos de FileMaker. Esta ruta debe especificarse como directorio base en el cuadro de diálogo Administrar contenedores de FileMaker Pro. Debe especificar `SECURE` para el almacenamiento seguro u `OPEN` para el almacenamiento abierto. Si utiliza el almacenamiento abierto, `cadena_ruta_calc` hace referencia a la carpeta incluida dentro de la carpeta `cadena_ruta_relativa` en la que se almacenarán los objetos del contenedor. En la ruta se deben utilizar barras (/) para el nombre de carpeta.

Ejemplos

Uso	SQL de ejemplo
columna de texto	CREATE TABLE T1 (C1 VARCHAR, C2 VARCHAR (50), C3 VARCHAR (1001), C4 VARCHAR (500276))
columna de texto, NOT NULL	CREATE TABLE T1NN (C1 VARCHAR NOT NULL, C2 VARCHAR (50) NOT NULL, C3 VARCHAR (1001) NOT NULL, C4 VARCHAR (500276) NOT NULL)
columna numérica	CREATE TABLE T2 (C1 DECIMAL, C2 DECIMAL (10,0), C3 DECIMAL (7539,2), C4 DECIMAL (497925,301))
columna de fecha	CREATE TABLE T3 (C1 DATE, C2 DATE, C3 DATE, C4 DATE)
columna de hora	CREATE TABLE T4 (C1 TIME, C2 TIME, C3 TIME, C4 TIME)
columna de fecha y hora	CREATE TABLE T5 (C1 TIMESTAMP, C2 TIMESTAMP, C3 TIMESTAMP, C4 TIMESTAMP)
columna del campo contenedor	CREATE TABLE T6 (C1 BLOB, C2 BLOB, C3 BLOB, C4 BLOB)
columna del campo contenedor de almacenamiento externo	CREATE TABLE T7 (C1 BLOB EXTERNAL 'Files/MyDatabase/' SECURE) CREATE TABLE T8 (C1 BLOB EXTERNAL 'Files/MyDatabase/' OPEN 'Objects')

Secuencia TRUNCATE TABLE

Utilice la secuencia `TRUNCATE TABLE` para eliminar rápidamente todos los registros de la tabla especificada, vaciando la tabla de todos los datos.

```
TRUNCATE TABLE nombre_tabla
```

No se puede especificar una cláusula `WHERE` con la secuencia `TRUNCATE TABLE`. La secuencia `TRUNCATE TABLE` elimina todos los registros.

Solo se eliminan los registros de la tabla especificados por `nombre_tabla`. Los registros de las tablas relacionadas no se ven afectados.

La secuencia `TRUNCATE TABLE` debe poder bloquear todos los registros de la tabla para eliminar los datos de registros. Si cualquier registro de la tabla está bloqueado por otro usuario, FileMaker devuelve el código de error 301 ("Otro usuario está utilizando el registro").

Secuencia ALTER TABLE

Utilice la secuencia `ALTER TABLE` para cambiar la estructura de una tabla existente en un archivo de base de datos. Sólo puede modificar una columna en cada secuencia. Los formatos de la secuencia `ALTER TABLE` son:

```
ALTER TABLE nombre_tabla ADD [COLUMN] definición_columna
```

```
ALTER TABLE nombre_tabla DROP [COLUMN] nombre_columna_no_cualificado
```

```
ALTER TABLE nombre_tabla ALTER [COLUMN] definición_columna SET DEFAULT expr
```

```
ALTER TABLE nombre_tabla ALTER [COLUMN] definición_columna DROP DEFAULT
```

Debe conocer la estructura de la tabla y el modo en que desea modificarla antes de utilizar la secuencia `ALTER TABLE`.

Ejemplos

Para	SQL de ejemplo
añadir columnas	<code>ALTER TABLE Vendedores ADD C1 VARCHAR</code>
eliminar columnas	<code>ALTER TABLE Vendedores DROP C1</code>
establecer el valor predeterminado para una columna	<code>ALTER TABLE Vendedores ALTER Empresa SET DEFAULT 'FileMaker'</code>
eliminar el valor predeterminado para una columna	<code>ALTER TABLE Vendedores ALTER Empresa DROP DEFAULT</code>

Nota `SET DEFAULT` y `DROP DEFAULT` no afectan a las filas existentes de la tabla, pero cambian el valor predeterminado de las filas que se añadan posteriormente a la tabla.

Secuencia CREATE INDEX

Utilice la secuencia `CREATE INDEX` para acelerar las búsquedas en el archivo de base de datos. El formato de la secuencia `CREATE INDEX` es:

```
CREATE INDEX ON nombre_tabla.nombre_columna
CREATE INDEX ON nombre_tabla (nombre_columna)
```

`CREATE INDEX` se admite para una única columna (no se admiten índices de varias columnas). No se permiten índices en columnas que correspondan con tipos de campo contenedor, campos sumario, campos que tengan la opción de almacenamiento global o campos de cálculo sin almacenar en un archivo de base de datos de FileMaker.

Al crear un índice para una columna de texto se selecciona automáticamente la Opción de almacenamiento **Mínimo** en **Indexación** para el campo correspondiente del archivo de base de datos de FileMaker. Al crear un índice para una columna que no sea de texto (o que tenga el formato de texto en japonés) se selecciona automáticamente la Opción de almacenamiento **Todo** en **Indexación** para el campo correspondiente del archivo de base de datos de FileMaker.

Al crear un índice para cualquier columna se selecciona automáticamente la Opción de almacenamiento **Crear índices automáticamente según sea necesario** en **Indexación** para el campo correspondiente del archivo de base de datos de FileMaker.

FileMaker crea automáticamente índices según sea necesario. Cuando se utiliza `CREATE INDEX`, el índice se crea inmediatamente en lugar de tener que solicitarlo.

Ejemplo

```
CREATE INDEX ON Vendedores.ID_Vendedor
```

Secuencia DROP INDEX

Utilice la secuencia `DROP INDEX` para quitar un índice de un archivo de base de datos. El formato de la secuencia `DROP INDEX` es:

```
DROP INDEX ON nombre_tabla.nombre_columna
DROP INDEX ON nombre_tabla (nombre_columna)
```

Puede quitar un índice si el archivo de base de datos es demasiado grande o si no utiliza a menudo un campo en consultas.

Si las consultas van muy lentas y está trabajando con un archivo de base de datos de FileMaker excesivamente grande con muchos campos de texto indexados, considere la posibilidad de eliminar los índices de algunos campos. Considere también la posibilidad de quitar los índices de los campos que utilice con poca frecuencia en secuencias `SELECT`.

Al quitar un índice para cualquier columna se selecciona automáticamente la Opción de almacenamiento **Ninguno** y se borra **Crear índices automáticamente según sea necesario en Indexación** para el campo correspondiente del archivo de base de datos de FileMaker.

No se admite el atributo `PREVENT INDEX CREATION`.

Ejemplo

```
DROP INDEX ON Vendedores.ID_Vendedor
```

Expresiones SQL

Utilice expresiones en las cláusulas `WHERE`, `HAVING` y `ORDER BY` de las secuencias `SELECT` para crear consultas de base de datos detalladas y sofisticadas. Los elementos de expresión válidos son:

- Nombres de campo
- Constantes
- Notación exponencial/científica
- Operadores numéricos
- Operadores de caracteres
- Operadores de fecha
- Operadores relacionales
- Operadores lógicos
- Funciones

Nombres de campo

La expresión más común es un nombre de campo sencillo, como `calc o Datos_ventas.ID_Factura`.

Constantes

Las constantes son valores que no cambian. Por ejemplo, en la expresión `PRICE * 1,05`, el valor `1,05` es una constante. También puede asignar el valor `30` a la constante `Número_De_Días_De_Junio`.

Debe escribir las constantes de caracteres entre comillas sencillas (`'`). Para incluir un signo de comillas sencillas en una constante con caracteres escrita entre comillas sencillas, utilice un signo de comillas dobles (por ejemplo `'O' 'Neal'`).

Para las aplicaciones ODBC y JDBC, FileMaker admite las constantes de fecha, hora y fecha y hora en formato ODBC/JDBC entre corchetes ({}).

Ejemplos

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

FileMaker permite escribir el especificador de tipo (D, T o TS) en mayúsculas o minúsculas. Puede insertar los espacios que desee tras el especificador de tipo, o incluso no insertar ningún espacio.

FileMaker también admite los formatos de hora y fecha ISO de sintaxis SQL-92 sin corchetes.

Ejemplos

- DATE 'AAAA-MM-DD'
- TIME 'HH:MM:SS'
- TIMESTAMP 'AAAA-MM-DD HH:MM:SS'

Además, la función ExecuteSQL de FileMaker Pro acepta solo los formatos de hora y fecha ISO de sintaxis SQL-92 sin corchetes ({}).

Constante	Sintaxis aceptable (ejemplos)
Texto	'París'
Numérico	1.05
Fecha	DATE '2019-06-05' { D '2019-06-05' } {06/05/2019} {06/05/19} Nota La sintaxis de año de dos dígitos no es compatible con el formato ODBC/JDBC ni con el formato SQL-92.
Hora	TIME '14:35:10' { T '14:35:10' } {14:35:10}
Fecha y hora	TIMESTAMP '2019-06-05 14:35:10' { TS '2019-06-05 14:35:10' } {06/05/2019 14:35:10} {06/05/19 14:35:10} Compruebe que el Tipo de datos: 4-Digit Year Date no está seleccionado como opción de validación en el archivo de base de datos de FileMaker para un campo que utilice una sintaxis con años de 2 dígitos. Nota La sintaxis de año de dos dígitos no es compatible con el formato ODBC/JDBC ni con el formato SQL-92.

Cuando se introducen valores de fecha y hora, haga coincidir el formato de la configuración regional de los archivos de base de datos. Por ejemplo, si la base de datos se ha creado en un sistema de idioma italiano, utilice los formatos de fecha y hora italianos.

Notación exponencial/científica

Los números se pueden expresar mediante una notación científica.

Ejemplo

```
SELECT column1 / 3.4E+7 FROM tabla1 WHERE calc < 3.4E-6 * columna2
```

Operadores numéricos

Puede incluir los siguientes operadores en expresiones numéricas: +, -, *, / y ^ o ** (potencias).

Puede escribir delante de las expresiones numéricas un signo más (+) o menos (-).

Operadores de caracteres

Puede concatenar los caracteres. A continuación, `apellidos` es 'JONES ' y `nombre` es 'ROBERT ':

Operador	Concatenación	Ejemplo	Resultado
+	Mantener los caracteres en blanco posteriores	nombre + apellidos	'ROBERT JONES '
-	Mover los caracteres en blanco posteriores al final	nombre - apellidos	'ROBERTJONES '

Operadores de fecha

Puede modificar las fechas. A continuación, `fecha_contratación` es DATE '2019-01-30'.

Operador	Efecto sobre fecha	Ejemplo	Resultado
+	Añade un número de días a una fecha	<code>fecha_contratación + 5</code>	DATE '2019-02-04'
-	Hallar el número de días entre dos fechas	<code>fecha_contratación - DATE '2019-01-01'</code>	29
	Restar un número de días a una fecha	<code>fecha_contratación - 10</code>	DATE '2019-01-20'

Ejemplos adicionales

```
SELECT Fecha_Venta, Fecha_Venta + 30 AS agg FROM Datos_ventas
SELECT Fecha_Venta, Fecha_Venta - 30 AS agg FROM Datos_ventas
```

Operadores relacionales

Operador	Significado
=	Igual a
<>	No es igual a
>	Mayor que
>=	Mayor o igual que
<	Es menor que
<=	Menor o igual que
LIKE	Coincide con un patrón
NOT LIKE	No coincide con un patrón
IS NULL	Igual a Nulo
IS NOT NULL	No igual a Nulo
BETWEEN	Rango de valores entre un límite inferior y uno superior
IN	Miembro de un conjunto de valores especificados o miembro de una subconsulta
NOT IN	No es miembro de un conjunto de valores especificados ni miembro de una subconsulta
EXISTS	'Verdadero' si una subconsulta ha devuelto al menos un registro
ANY	Compara un valor con cada valor devuelto por una subconsulta (el operador debe llevar delante =, <>, >, >=, < o <=); =Any es equivalente a In
ALL	Compara un valor con cada valor devuelto por una subconsulta (el operador debe llevar delante =, <>, >, >=, < o <=)

Ejemplo

```

SELECT Datos_ventas.ID_Factura FROM Datos_ventas
  WHERE Datos_ventas.ID_Vendedor = 'SP-1'
SELECT Datos_ventas.Cantidad FROM Datos_ventas
  WHERE Datos_ventas.ID_Factura <> 125
SELECT Datos_ventas.Cantidad FROM Datos_ventas
  WHERE Datos_ventas.Cantidad > 3000
SELECT Datos_ventas.Hora_Venta FROM Datos_ventas
  WHERE Datos_ventas.Hora_Venta < '12:00:00'
SELECT Datos_ventas.Empresa_nombre FROM Datos_ventas
  WHERE Datos_ventas.Empresa_nombre LIKE '%Universidad'
SELECT Datos_ventas.Empresa_nombre FROM Datos_ventas
  WHERE Datos_ventas.Empresa_nombre NOT LIKE '%Universidad'
SELECT Datos_ventas.Cantidad FROM Datos_ventas
  WHERE Datos_ventas.Cantidad IS NULL
SELECT Datos_ventas.Cantidad FROM Datos_ventas
  WHERE Datos_ventas.Cantidad IS NOT NULL
SELECT Datos_ventas.ID_Factura FROM Datos_ventas
  WHERE Datos_ventas.ID_Factura BETWEEN 1 AND 10
SELECT COUNT(Datos_ventas.ID_Factura) AS agg
  FROM Datos_ventas WHERE Datos_ventas.ID_Factura IN (50,250,100)
SELECT COUNT(Datos_ventas.ID_Factura) AS agg
  FROM Datos_ventas WHERE Datos_ventas.ID_Factura NOT IN (50,250,100)
SELECT COUNT(Datos_ventas.ID_Factura) AS agg FROM Datos_ventas
  WHERE Datos_ventas.ID_Factura NOT IN (SELECT Datos_ventas.ID_Factura
  FROM Datos_ventas WHERE Datos_ventas.ID_Vendedor = 'SP-4')
SELECT *
  FROM Datos_ventas WHERE EXISTS (SELECT Datos_ventas.Cantidad
  FROM Datos_ventas WHERE Datos_ventas.ID_Vendedor IS NOT NULL)
SELECT *
  FROM Datos_ventas WHERE Datos_ventas.Cantidad = ANY (SELECT
  Datos_ventas.Cantidad
  FROM Datos_ventas WHERE Datos_ventas.ID_Vendedor = 'SP-1')
SELECT *
  FROM Datos_ventas WHERE Datos_ventas.Cantidad = ALL (SELECT
  Datos_ventas.Cantidad
  FROM Datos_ventas WHERE Datos_ventas.ID_Vendedor IS NULL)

```

Operadores lógicos

Puede combinar dos o más condiciones. Las condiciones deben estar relacionadas por AND u OR, como:

```
salario = 40000 AND exenc = 1
```

El operador lógico NOT se utiliza para invertir el significado, como:

```
NOT (salario = 40000 AND exenc = 1)
```

Ejemplo

```
SELECT * FROM Datos_ventas WHERE Datos_ventas.Empresa_nombre
NOT LIKE '%Universidad' AND Datos_ventas.Cantidad > 3000
SELECT * FROM Datos_ventas WHERE (Datos_ventas.Empresa_nombre
LIKE '%Universidad' OR Datos_ventas.Cantidad > 3000)
AND Datos_ventas.ID_Vendedor = 'SP-1'
```

Prioridad de operadores

A medida que las expresiones se hacen más complejas, es importante el orden en que éstas se evalúan. Esta tabla muestra el orden en que se evalúan los operadores. Los operadores de la primera línea se evalúan primero, y así sucesivamente. Los operadores de la misma línea se evalúan de izquierda a derecha en la expresión.

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

Ejemplos

```
WHERE salario > 40000 OR fecha_contratación > (DATE '2008-01-30') AND
dep = 'D101'
```

Como se evalúa AND en primer lugar, esta consulta recupera los empleados del departamento D101 contratados después del 30.01.08, así como todos los empleados que ganen más de 40.000 €, independientemente del departamento o la fecha de contratación.

Para hacer que la cláusula se evalúe en un orden diferente, escriba entre paréntesis las condiciones que se deban evaluar primero.

```
WHERE (salario > 40000 OR fecha_contratación > DATE '2008-01-30') AND
dep = 'D101'
```

En este ejemplo, se recuperan los empleados del departamento D101 que ganan más de 40.000 € o fueron contratados después del 30.01.08.

Funciones SQL

FileMaker SQL admite muchas funciones que puede utilizar en expresiones. Algunas funciones devuelven cadenas de caracteres, algunas devuelven números, algunas devuelven fechas y algunas devuelven valores que dependen de las condiciones que cumplan los argumentos de la función.

Funciones de agregación

Las funciones de agregación devuelven un valor único de un conjunto de registros. Puede utilizar una función de agregación como parte de una secuencia `SELECT`, con un nombre de campo (por ejemplo, `AVG (SALARIO)`) o en combinación con una expresión de columna (por ejemplo, `AVG (SALARIO * 1,07)`).

Puede escribir delante de la expresión de columna el operador `DISTINCT` para eliminar los valores duplicados.

Ejemplo

```
COUNT (DISTINCT apellidos)
```

En este ejemplo, sólo se cuentan los valores de apellidos únicos.

Función de agregación	Devuelve
SUM	El total de los valores de una expresión de campo numérico. Por ejemplo, <code>SUM (SALARIO)</code> devuelve la suma de todos los valores de campos de salario.
AVG	La media de los valores de una expresión de campo numérico. Por ejemplo, <code>AVG (SALARIO)</code> devuelve la media de todos los valores de campos de salario.
COUNT	El número de valores de cualquier expresión de campo. Por ejemplo, <code>COUNT (NOMBRE)</code> devuelve el número de valores de nombres. Cuando se utiliza <code>COUNT</code> con un nombre de campo, <code>COUNT</code> devuelve el número de valores de campos no nulos. Un ejemplo especial es <code>COUNT (*)</code> , que devuelve el número de registros del conjunto, incluidos los registros con valores nulos.
MAX	El valor máximo de cualquier expresión de campo. Por ejemplo, <code>MAX (SALARIO)</code> devuelve el valor máximo de los campos de salario.
MIN	El valor mínimo de cualquier expresión de campo. Por ejemplo, <code>MIN (SALARIO)</code> devuelve el valor mínimo de los campos de salario.

Ejemplo

```
SELECT SUM (Datos_ventas.Cantidad) AS agg FROM Datos_ventas
SELECT AVG (Datos_ventas.Cantidad) AS agg FROM Datos_ventas
SELECT COUNT (Datos_ventas.Cantidad) AS agg FROM Datos_ventas
SELECT MAX (Datos_ventas.Cantidad) AS agg FROM Datos_ventas
WHERE Datos_ventas.Cantidad < 3000
SELECT MIN (Datos_ventas.Cantidad) AS agg FROM Datos_ventas
WHERE Datos_ventas.Cantidad > 3000
```

No se pueden utilizar funciones de agregación como argumento para otras funciones. En ese caso, FileMaker devuelve el código de error 8309 ("No se admiten las expresiones con agregaciones"). Por ejemplo, la siguiente secuencia no es válida porque la función de agregación `SUM` no se puede utilizar como argumento de la función `ROUND`:

Ejemplo

```
SELECT ROUND (SUM (Salario), 0) FROM Nómina
```

Sin embargo, las funciones de agregación pueden utilizar funciones que devuelvan números como argumentos. La siguiente secuencia es válida:

Ejemplo

```
SELECT SUM(ROUND(Salario, 0)) FROM Nómina
```

Funciones que devuelven cadenas de caracteres

Funciones que devuelven cadenas de caracteres	Descripción	Ejemplo
CHR	Convierte un código ASCII en una cadena de un carácter	CHR(67) devuelve C
CURRENT_USER	Devuelve el ID de inicio de sesión especificado en el momento de la conexión	
DAYNAME	Devuelve el nombre del día que corresponde a una fecha determinada	
RTRIM	Elimina los espacios en blanco situados detrás de una cadena	RTRIM(' ABC ') devuelve ' ABC'
TRIM	Elimina los espacios en blanco situados delante y detrás de una cadena	TRIM(' ABC ') devuelve 'ABC'
LTRIM	Elimina los espacios en blanco situados delante de una cadena	LTRIM(' ABC') devuelve 'ABC'
UPPER	Pone en mayúsculas cada letra de una cadena	UPPER('Allen') devuelve 'ALLEN'
LOWER	Pone en minúsculas cada letra de una cadena	LOWER('Allen') devuelve 'allen'
LEFT	Devuelve los caracteres situados más a la izquierda de una cadena	LEFT('Mattson', 3) devuelve 'Mat'
MONTHNAME	Devuelve los nombres de los meses del calendario	
RIGHT	Devuelve los caracteres situados más a la derecha de una cadena	RIGHT('Mattson', 4) devuelve 'tson'
SUBSTR SUBSTRING	Devuelve una subcadena de una cadena y tiene como parámetros la cadena, el primer carácter de la extracción y el número de caracteres que extraer (opcional)	SUBSTR('Conrad', 2, 3) devuelve 'onr' SUBSTR('Conrad', 2) devuelve 'onrad'
SPACE	Genera una cadena de espacios en blanco	SPACE(5) devuelve ' '
STRVAL	Convierte un valor de cualquier tipo en una cadena de caracteres	STRVAL('Woltman') devuelve 'Woltman' STRVAL(5 * 3) devuelve '15' STRVAL(4 = 5) devuelve 'Falso' STRVAL(DATE '2019-12-25') devuelve '2019-12-25'
TIME TIMEVAL	Devuelve la hora del día como cadena	A las 9:49 PM, TIME() devuelve 21:49:00
USERNAME USER	Devuelve el ID de inicio de sesión especificado en el momento de la conexión	

Nota La función `TIME ()` está en desuso. Utilice en su lugar el estándar SQL `CURRENT_TIME`.

Ejemplo

```
SELECT CHR(67) + SPACE(1) + CHR(70) FROM Vendedores

SELECT RTRIM(' ' + Vendedores.ID_Vendedor) AS agg FROM Vendedores

SELECT TRIM(SPACE(1) + Vendedores.ID_Vendedor) AS agg FROM Vendedores

SELECT LTRIM(' ' + Vendedores.ID_Vendedor) AS agg FROM Vendedores

SELECT UPPER(Vendedores.Vendedor) AS agg FROM Vendedores

SELECT LOWER(Vendedores.Vendedores) AS agg FROM Vendedores

SELECT LEFT(Vendedores.Vendedor, 5) AS agg FROM Vendedores

SELECT RIGHT(Vendedores.Vendedor, 7) AS agg FROM Vendedores

SELECT SUBSTR(Vendedores.ID_Vendedor, 2, 2) +
SUBSTR(Vendedores.ID_Vendedor, 4, 2) AS agg FROM Vendedores

SELECT SUBSTR(Vendedores.ID_Vendedor, 2) +
SUBSTR(Vendedores.ID_Vendedor, 4) AS agg FROM Vendedores

SELECT SPACE(2) + Vendedores.ID_Vendedor AS ID_Vendedor FROM Vendedores

SELECT STRVAL('60506') AS agg FROM Datos_ventas WHERE
Datos_ventas.Factura = 1
```

Funciones que devuelven números

Funciones que devuelven números	Descripción	Ejemplo
ABS	Devuelve el valor absoluto de la expresión numérica	
ATAN	Devuelve la arcotangente del argumento como un ángulo expresado en radianes	
ATAN2	Devuelve el arcotangente de las coordenadas x e y como un ángulo expresado en radianes	
CEIL CEILING	Devuelve el menor valor entero que es mayor o igual que el argumento	
DEG DEGREES	Devuelve el número de grados del argumento, es decir, un ángulo expresado en radianes	
DAY	Devuelve el día de una fecha	DAY (DATE '2019-01-30') devuelve 30
DAYOFWEEK	Devuelve el día de la semana (1-7) de una expresión de fecha	DAYOFWEEK (DATE '2004-05-01') devuelve 7
MOD	Divide dos números y devuelve el resto de la división	MOD (10, 3) devuelve 1
EXP	Devuelve un valor que es la base del logaritmo natural (e) elevado a una potencia especificada por el argumento	
FLOOR	Devuelve el mayor valor entero que es menor o igual que el argumento	
HOUR	Devuelve la parte de horas de un valor	
INT	Devuelve la parte entera de un número	INT (6.4321) devuelve 6
LENGTH	Devuelve la longitud de una cadena	LENGTH ('ABC') devuelve 3
MONTH	Devuelve el mes de una fecha	MONTH (DATE '2019-01-30') devuelve 1
LN	Devuelve el logaritmo natural de un argumento	
LOG	Devuelve el logaritmo decimal de un argumento	
MAX	Devuelve el mayor de dos números	MAX (66, 89) devuelve 89
MIN	Devuelve el menor de dos números	MIN (66, 89) devuelve 66
MINUTE	Devuelve la parte de minutos de un valor	
NUMVAL	Convierte una cadena de caracteres en un número; la función falla si la cadena de caracteres no es un número válido	NUMVAL ('123') devuelve 123
PI	Devuelve el valor constante de la constante matemática pi	
RADIANS	Devuelve el número de radianes de un argumento que se expresa en grados	
ROUND	Redondea un número	ROUND (123.456, 0) devuelve 123 ROUND (123.456, 2) devuelve 123,46 ROUND (123.456, -2) devuelve 100
SECOND	Devuelve la parte de segundos de un valor	

Funciones que devuelven números	Descripción	Ejemplo
SIGN	Un indicador del signo del argumento: -1 para negativo, 0 para 0 y 1 para positivo	
SIN	Devuelve el seno del argumento	
SQRT	Devuelve la raíz cuadrada del argumento	
TAN	Devuelve la tangente del argumento	
YEAR	Devuelve el año de una fecha	YEAR (DATE '2019-01-30') devuelve 2019

Funciones que devuelven fechas

Funciones que devuelven fechas	Descripción	Ejemplo
CURDATE CURRENT_DATE	Devuelve la fecha de hoy	
CURTIME CURRENT_TIME	Devuelve la hora actual	
CURTIMESTAMP CURRENT_TIMESTAMP	Devuelve la fecha y la hora actuales	
TIMESTAMPVAL	Convierte una cadena de caracteres en una fecha y hora	TIMESTAMPVAL ('2019-01-30 14:00:00') devuelve su valor de fecha y hora
DATE TODAY	Devuelve la fecha de hoy	Si hoy es 21/11/2019, DATE () devuelve 2019-11-21
DATEVAL	Convierte una cadena de caracteres en una fecha	DATEVAL ('2019-01-30') devuelve 30/01/2019

Nota La función DATE () está en desuso. Utilice en su lugar el estándar SQL CURRENT_DATE.

Funciones condicionales

Funciones condicionales	Descripción	Ejemplo
CASE WHEN	<p>Formato CASE simple</p> <p>Compara el valor de <i>exp_entrada</i> con los valores de los argumentos <i>exp_valor</i> para determinar el resultado</p> <pre> CASE <i>exp_entrada</i> {WHEN <i>exp_valor</i> THEN <i>resultado...</i>} [ELSE <i>resultado</i>] END </pre>	<pre> SELECT ID_Factura, CASE Nombre_Empresa WHEN 'Exportaciones RU' THEN 'Exportaciones RU encontradas' WHEN 'Proveedores de mobiliario doméstico' THEN 'Proveedores de mobiliario doméstico encontrados' ELSE 'Ninguna exportación RU ni proveedores de mobiliario doméstico' END, ID_Vendedor FROM Datos_ventas </pre>
	<p>Formato CASE buscado</p> <p>Devuelve un resultado en función de si la condición especificada por una expresión WHEN es verdadera</p> <pre> CASE {WHEN <i>exp_booleana</i> THEN <i>resultado...</i>} [ELSE <i>resultado</i>] END </pre>	<pre> SELECT ID_Factura, Cantidad, CASE WHEN Cantidad > 3000 THEN 'Por encima de 3000' WHEN Cantidad < 1000 THEN 'Por debajo de 3000' ELSE 'Entre 1000 y 3000' END, ID_Vendedor FROM Datos_ventas </pre>
COALESCE	<p>Devuelve el primer valor que no es NULL</p>	<pre> SELECT ID_Vendedor, COALESCE(Gestor_Ventas, Vendedor) FROM Vendedores </pre>
NULLIF	<p>Compara dos valores y devuelve NULL si los dos valores son iguales; si no lo son, devuelve el primer valor</p>	<pre> SELECT ID_Factura, NULLIF(Cantidad, -1), ID_Vendedor FROM Datos_ventas </pre>

Objetos del sistema de FileMaker

Los archivos de base de datos de FileMaker incluyen los siguientes objetos del sistema a los que se puede acceder mediante consultas SQL.

Tablas del sistema de FileMaker

Todos los archivos de base de datos de FileMaker incluyen dos tablas del sistema: FileMaker_Tables y FileMaker_Fields. En las aplicaciones ODBC, estas tablas se incluyen en la información devuelta por la función de catálogo SQLTables. En las aplicaciones JDBC, estas tablas se incluyen en la información devuelta por la función de DatabaseMetaData getTables. Las tablas también se pueden utilizar en las funciones ExecuteSQL.

FileMaker_Tables

La tabla FileMaker_Tables contiene información sobre las tablas de base de datos definidas en el archivo de FileMaker.

La tabla FileMaker_Tables incluye una fila para cada instancia de tabla del gráfico de relaciones con las siguientes columnas:

- TableName: el nombre de la instancia de tabla.
- TableId: el ID exclusivo de la instancia de tabla.
- BaseTableName: el nombre de la tabla base desde la que se ha creado la instancia de tabla.
- BaseFileName: el nombre de archivo de FileMaker del archivo de base de datos que contiene la tabla base.
- ModCount: el número total de veces que se han consignado cambios realizados en la definición de esta tabla.

Ejemplo

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

Tabla FileMaker_Fields

La tabla FileMaker_Fields contiene información sobre los campos definidos en el archivo de FileMaker.

La tabla FileMaker_Fields incluye las siguientes columnas:

- TableName: el nombre de la tabla que contiene el campo.
- FieldName: el nombre del campo.
- FieldType: el tipo de datos SQL del campo.
- FieldId: el ID exclusivo del campo.
- FieldClass: uno de estos tres valores: "Summary" para campos de sumario; "Calculated" para resultados calculados o "Normal".
- FieldReps: el número de repeticiones del campo.
- ModCount: el número total de veces que se han consignado cambios realizados en la definición de esta tabla.

Ejemplo

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

Columnas del sistema de FileMaker

FileMaker añade columnas del sistema (campos) a todas las filas (registros) de todas las tablas definidas en el archivo de FileMaker. En las aplicaciones ODBC, estas columnas se incluyen en la información devuelta por la función de catálogo `SQLSpecialColumns`. En las aplicaciones JDBC, estas columnas se incluyen en la información devuelta por la función de `DatabaseMetaData.getVersionColumns`. Las columnas también se pueden utilizar en las funciones `ExecuteSQL`.

Columna ROWID

La columna del sistema `ROWID` contiene el número de ID exclusivo del registro. Se trata del mismo valor que devuelve la función de FileMaker Pro "Get(RecordID)".

Columna ROWMODID

La columna del sistema `ROWMODID` contiene el número total de veces que se han consignado cambios realizados en el registro actual. Se trata del mismo valor que devuelve la función de FileMaker Pro "Get(RecordModificationCount)".

Ejemplo

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

Palabras clave de SQL reservadas

En esta sección se muestran las palabras clave reservadas que no deben utilizarse como nombre de columnas, tablas, alias u otros objetos definidos por el usuario. Si se producen errores de sintaxis, pueden deberse a que está utilizando una de estas palabras clave reservadas. Si desea utilizar una de estas palabras clave, tiene que usar comillas dobles para que no se considere una palabra clave.

Ejemplo

Utilice la palabra clave DEC como nombre de elemento de datos.

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

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	DOBLE
CASCADED	CURRENT_DATE	DROP
CASE	CURRENT_TIME	ELSE
CAST	CURRENT_TIMESTAMP	END

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

SECOND	UNIQUE
SECTION	UNKNOWN
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	

Índice

A

actualizaciones y eliminaciones posicionadas 14
alias de columna 8
alias de tabla 8, 9
ALTER TABLE (secuencia SQL) 22

B

BaseFileName 36
BaseTableName 36

C

cadena vacía, uso en SELECT 15
cadenas de funciones 31
campo contenedor
 almacenado externamente 21
 con función PutAs 18
 con secuencia CREATE TABLE 21, 22
 con secuencia INSERT 18
 con secuencia SELECT 16
 con secuencia UPDATE 20
caracteres en blanco 26
Columna del sistema ROWID 37
Columna del sistema ROWMODID 37
constantes en expresiones SQL 24
Controlador de cliente JDBC
 portales 7
 Unicode, compatibilidad 7
Controlador de cliente ODBC
 portales 7
 Unicode, compatibilidad 7
CREATE INDEX (secuencia SQL) 23
CREATE TABLE (secuencia SQL) 20
cumplimiento con los estándares 7
cumplimiento con los estándares de ODBC 7
cumplimiento con los estándares de SQL 7
cursores en ODBC 14

D

datos binarios, uso en SELECT 15
DEFAULT (cláusula SQL) 21
DELETE (secuencia SQL) 17
DROP INDEX (secuencia SQL) 23

E

errores de sintaxis 38
EVITAR CREACIÓN DE ÍNDICE 24
expresiones en SQL 24

Expresiones SQL
 constantes 24
 funciones 29
 nombres de campos 24
 notación exponencial o científica 26
 operadores de caracteres 26
 operadores de fecha 26
 operadores lógicos 28
 operadores numéricos 26
 operadores relacionales 27
 prioridad de operadores 29
EXTERNAL (cláusula SQL) 21

F

FETCH FIRST (cláusula SQL) 14
FieldClass 36
FieldId 36
FieldName 36
FieldReps 36
FieldType 36
filas de pares 14
FileMaker_Fields 36
FileMaker_Tables 36
FOR UPDATE (cláusula SQL) 14
formatos de fecha 25
formatos de fecha y hora 25
formatos de hora 25
FROM (cláusula SQL) 9
FULL OUTER JOIN 10
función ABS 33
función ATAN 33
función ATAN2 33
función CASE WHEN 35
función CAST 16
función CEIL 33
función CEILING 33
función CHR 31
función COALESCE 35
función CURDATE 34
función CURRENT_DATE 34
función CURRENT_TIME 34
función CURRENT_TIMESTAMP 34
función CURRENT_USER 31
función CURTIME 34
función CURTIMESTAMP 34
función DATE 34
función DATEVAL 34
función DAY 33
función DAYNAME 31
función DAYOFWEEK 33
función DEG 33
función DEGREES 33
Función ExecuteSQL 6

función EXP 33
función FLOOR 33
función GetAs 16
función HOUR 33
función INT 33
función LEFT 31
Función LENGTH 33
función LN 33
función LOG 33
función LOWER 31
función LTRIM 31
función MAX 33
función MIN 33
función MINUTE 33
función MOD 33
función MONTH 33
función MONTHNAME 31
función NULLIF 35
función NUMVAL 33
función PI 33
función PutAs 18, 20
función RADIANS 33
función RIGHT 31
función ROUND 33
función RTRIM 31
función SECOND 33
función SIGN 34
función SIN 34
función SPACE 31
función SQRT 34
función STRVAL 31
función SUBSTR 31
función SUBSTRING 31
función TAN 34
función TIME 31
función TIMESTAMPVAL 34
función TIMEVAL 31
función TODAY 34
función TRIM 31
Función UPPER 31
función USERNAME 31
función YEAR 34
funciones de agregación en SQL 30
funciones en expresiones SQL 29

G

GROUP BY (cláusula SQL) 11

H

HAVING (cláusula SQL) 12

I

INNER JOIN 10

INSERT (secuencia SQL) 17

J

join 10

L

LEFT OUTER JOIN 10

M

ModCount 36

N

nombres de campos en expresiones SQL 24

NOT NULL (cláusula SQL) 21

notación científica en expresiones SQL 26

notación exponencial en expresiones SQL 26

O

OFFSET (cláusula SQL) 13

operador ALL 27

operador AND 28

operador ANY 27

operador BETWEEN 27

operador DISTINCT 8

operador EXISTS 27

operador IN 27

operador IS NOT NULL 27

operador IS NULL 27

operador LIKE 27

operador NOT 28

operador NOT IN 27

operador NOT LIKE 27

operador OR 28

operadores de caracteres en expresiones SQL 26

operadores de fecha en expresiones SQL 26

operadores lógicos en expresiones SQL 28

operadores numéricos en expresiones SQL 26

operadores relacionales en expresiones SQL 27

ORDENAR POR (cláusula SQL) 13

OUTER JOIN 10

P

Palabras clave de SQL reservadas 38

palabras clave, SQL reservadas 38

portales 7

prioridad de operadores en expresiones SQL 29

R

repeticiones de campos 18, 21

RIGHT OUTER JOIN 10

S

Secuencias SQL

- admitidas por los controladores de cliente 7
 - ALTER TABLE 22
 - CREATE INDEX 23
 - CREATE TABLE 20
 - DELETE 17
 - DROP INDEX 23
 - INSERT 17
 - palabras clave reservadas 38
 - SELECT 8
 - TRUNCATE TABLE 22
 - UPDATE 19
- SELECT (secuencia SQL) 8
- cadena vacía 15
 - datos binarios 15
 - tipo de datos BLOB 15
- SQL, expresiones 24
- SQL, funciones de agregación 30
- SQL-92 7
- subconsultas 18

T

- tablas del sistema 36
- TableId 36
- TableName 36
- tipo de datos BLOB, uso en SELECT 15
- tipo de datos SQL_C_WCHAR 7
- tipo de ordenación 13
- TRUNCATE TABLE (secuencia SQL) 22

U

- Unicode, compatibilidad 7
- UNION (operador SQL) 12
- UNIQUE (cláusula SQL) 21
- UPDATE (secuencia SQL) 19

V

- valor null 18
- valor vacío en columnas 18
- VALUES (cláusula SQL) 18

W

- WHERE (cláusula SQL) 11
- WITH TIES (cláusula SQL) 14