FileMaker® 16

SQL-Referenzhandbuch

FileMaker®
An Apple Subsidiary

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

Weitere Informationen finden Sie auf unserer Website unter http://www.filemaker.com/de/.

Edition: 01

Inhalt

Kapitel 1
Einführung

Als Datenbankentwickler können Sie FileMaker Pro einsetzen, um Datenbanklösungen zu erstellen, ohne SQL-Kenntnisse besitzen zu müssen. Wenn Sie jedoch über einige SQL-Kenntnisse verfügen, können Sie eine FileMaker-Datenbankdatei als ODBC- bzw. JDBC-Datenquelle bereitstellen und Ihre Daten mit anderen Anwendungen austauschen, die ODBC und JDBC verwenden. Sie können zudem die FileMaker Pro-Funktion „SQLAusführen“ verwenden, um Daten aus beliebigen Tabellenauftreten innerhalb einer FileMaker Pro-Datenbank abzurufen.

Diese Referenz beschreibt die SQL-Anweisungen und -Standards, die FileMaker unterstützt. Die FileMaker ODBC- und JDBC-Client-Treiber unterstützen sämtliche in dieser Referenz beschriebenen SQL-Anweisungen. Die FileMaker Pro-Funktion „SQLAusführen“ unterstützt nur die SELECT-Anweisung.

Über diese Referenz

- Informationen über die Verwendung von ODBC und JDBC mit früheren Versionen von FileMaker Pro finden Sie im Produktdokumentationszentrum.
- Diese Referenz setzt voraus, dass Sie mit den Grundlagen der Verwendung von FileMaker Pro-Funktionen, der Codierung von ODBC- und JDBC-Anwendungen sowie der Erstellung von SQL-Abfragen vertraut sind. Zusätzliche Informationen zu diesen Themen finden Sie in Büchern von Fremdanbietern.
- In dieser Referenz bezieht sich „FileMaker Pro“ sowohl auf FileMaker Pro als auch auf FileMaker Pro Advanced. Ausgenommen ist die Beschreibung von Funktionen, die spezifisch für FileMaker Pro Advanced sind.

Über SQL

SQL bzw. Structured Query Language ist eine Programmiersprache, die für die Abfrage von Daten aus einer relationalen Datenbank konzipiert wurde. Die für die Abfrage einer Datenbank verwendete Primäranweisung ist die SELECT-Anweisung.

Neben den Befehlen für die Abfrage einer Datenbank bietet SQL Anweisungen für die Durchführung von Datenmanipulationen, mit der Sie Daten hinzufügen, aktualisieren und löschen können.

SQL bietet zudem Anweisungen für Datendefinitionen. Mit diesen Anweisungen können Sie Tabellen und Indizes erstellen und ändern.

Die SQL-Anweisungen und Standards, die FileMaker unterstützt, werden in Kapitel 2, „Unterstützte Standards“ beschrieben.

Verwenden einer FileMaker-Datenbank als Datenquelle

Wenn Sie eine FileMaker-Datenbank als ODBC- bzw. JDBC-Datenquelle bereitstellen, können FileMaker-Daten mit ODBC- und JDBC-kompatiblen Anwendungen gemeinsam genutzt werden. Die Anwendungen stellen unter Verwendung der FileMaker-Client-Treiber eine Verbindung zur FileMaker-Datenquelle her, erstellen und führen die SQL-Abfragen mittels ODBC oder JDBC aus und verarbeiten die aus der FileMaker-Datenbanklösung abgerufenen Daten.

Umfassende Informationen zur Verwendung der FileMaker-Software als Datenquelle für ODBC- bzw. JDBC-Anwendungen finden Sie im FileMaker ODBC- und JDBC-Handbuch.

Die FileMaker ODBC- und JDBC-Client-Treiber unterstützen sämtliche in dieser Referenz beschriebenen SQL-Anweisungen.

Verwenden der Funktion „SQLAusführen“

Mit der FileMaker Pro-Funktion „SQLAusführen“ können Sie Daten aus Tabellenauftreten abrufen, die im Beziehungsdiagramm benannt werden, aber unabhängig von definierten Beziehungen sind. Sie können Daten aus mehreren Tabellen abrufen, ohne Tabellenverknüpfungen oder Beziehungen zwischen Tabellen erstellen zu müssen. In einigen Fällen können Sie die Komplexität Ihrer Beziehungsdiagramme durch den Einsatz der Funktion „SQLAusführen“ verringern.

Die Felder, die Sie mit der Funktion „SQLAusführen“ abfragen, müssen sich in keinem Layout befinden, daher können Sie die Funktion „SQLAusführen“ verwenden, um Daten unabhängig vom Layoutkontext abzurufen. Aufgrund der Kontextunabhängigkeit kann der Einsatz der Funktion „SQLAusführen“ in Scripts die Portabilität der Scripts verbessern. Sie können die Funktion „SQLAusführen“ überall dort verwenden, wo Sie Formeln einschließlich Diagramm- und Berichterstellung verwenden.

Die Funktion „SQLAusführen“ unterstützt nur die SELECT-Anweisung wie im Abschnitt „SELECT-Anweisung“ auf Seite 8 beschrieben.

Die Funktion „SQLAusführen“ akzeptiert zudem nur ISO-Datums- und Zeitformate der SQL-92-Syntax ohne geschweifte Klammern ({}). Die Funktion „SQLAusführen“ akzeptiert das ODBC/JDBC-Format für Datums-, Zeit- und Zeitstempelkonstanten in geschweiften Klammern nicht.

Informationen zur Syntax und der Verwendung der Funktion „SQLAusführen“ finden Sie in der FileMaker Pro Hilfe.

Kapitel 2
Unterstützte Standards

Verwenden Sie die FileMaker ODBC- und JDBC-Client-Treiber, um von einer Anwendung, die ODBC oder JDBC unterstützt, auf eine FileMaker-Datenbanklösung zuzugreifen. Die FileMaker-Datenbanklösung kann von FileMaker Pro oder FileMaker Server bereitgestellt sein.

- Der ODBC-Client-Treiber unterstützt ODBC 3.0 Level 1.
- Der JDBC-Client-Treiber unterstützt teilweise die JDBC-3.0-Spezifikation.
- Die ODBC- und JDBC-Client-Treiber richten sich nach SQL-92-Entry-Level-Konformität mit einigen zwischengeschalteten SQL-92-Funktionen.

Unterstützung von Unicode-Zeichen

Die ODBC- und JDBC-Client-Treiber unterstützen die Unicode-API. Wenn Sie jedoch eine eigene Anwendung erstellen, die die Client-Treiber verwendet, verwenden Sie für Feldnamen, Tabellennamen und Dateinamen ASCII-Code (falls Nicht-Unicode-Abfrage-Tools oder -Anwendungen verwendet werden).

Hinweis Um Unicode-Daten einzufügen und abzurufen, verwenden Sie `SQL_C_WCHAR`.

SQL-Anweisungen

Die ODBC- und JDBC-Client-Treiber unterstützen folgende SQL-Anweisungen:

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

Die Client-Treiber unterstützen darüber hinaus die Zuordnung von FileMaker-Datentypen zu ODBC SQL- und JDBC SQL-Datentypen. Informationen zur Datentypkonvertierung finden Sie im FileMaker ODBC- und JDBC-Handbuch. Zusätzliche Informationen über das Erstellen von SQL-Abfragen finden Sie in Büchern von Fremdanbietern.

Hinweis Die ODBC- und JDBC-Client-Treiber unterstützen keine FileMaker-Ausschnitte.

SELECT-Anweisung

Verwenden Sie die `SELECT`-Anweisung, um anzugeben, welche Spalten Sie anfordern. Geben Sie nach der `SELECT`-Anweisung den Spaltenausdruck (ähnlich wie Feldnamen) an, den Sie abrufen möchten (z. B. `last_name`). Ausdrücke können Rechenoperationen oder Zeichenfolgen beinhalten (z. B. `SALARY * 1,05`).

Die `SELECT`-Anweisung kann verschiedene Klauseln verwenden:

```
SELECT [DISTINCT] {* | spaltenausdruck [[AS] spaltenalias],...}
FROM tabellenname [tabellenalias], ...
[ WHERE ausdr1 rel_operator ausdr2 ]
[ GROUP BY {spaltenausdruck, ...} ]
[ HAVING ausdr1 rel_operator ausdr2 ]
[ UNION [ALL] (SELECT...) ]
[ ORDER BY {sort_ausdruck [DESC | ASC]}, ... ]
[ OFFSET n {ROWS | ROW} ]
[ FETCH FIRST [ n [ PERCENT ] ] { ROWS | ROW } {ONLY | WITH TIES } ]
[ FOR UPDATE [OF {spaltenausdruck, ...}] ]
```

Objekte in Klammern sind optional.

`spaltenalias` kann verwendet werden, um der Spalte einen beschreibenden Namen zu geben oder um einen längeren Spaltennamen abzukürzen.

Beispiel:

Weisen Sie z. B. den Alias `abteilung` der Spalte `abt` zu:

```
SELECT abt AS abteilung FROM ang
```

Feldnamen kann der Tabellenname oder der Tabellenalias vorangestellt werden. Zum Beispiel `ANG.NACHNAME` oder `A.NACHNAME`, wobei `A` der Alias für die Tabelle `ANG` ist.

Der Operator `DISTINCT` kann dem ersten Spaltenausdruck vorangestellt werden. Dieser Operator eliminiert doppelte Reihen aus dem Ergebnis einer Abfrage.

Beispiel:

```
SELECT DISTINCT abt FROM ang
```

SQL-Klauseln

Die ODBC- und JDBC-Client-Treiber unterstützen folgende SQL-Klauseln:

Verwenden Sie diese SQL-Klausel	Um
FROM (Seite 9)	anzuzeigen, welche Tabellen in der SELECT-Anweisung verwendet werden.
WHERE (Seite 11)	die Bedingungen anzugeben, die Datensätze erfüllen müssen, um abgefragt zu werden (wie FileMaker Pro-Suchabfragen).
GROUP BY (Seite 11)	die Namen eines oder mehrerer Felder anzugeben, nach denen die Ergebniswerte gruppiert werden sollen. Diese Klausel wird verwendet, um ein Set von Sammelwerten zurückzugeben, indem eine Zeile für jede Gruppe zurückgegeben wird (wie ein FileMaker Pro-Zwischenergebnis).
HAVING (Seite 12)	die Bedingungen für Gruppen von Datensätzen anzugeben (z. B. um nur die Abteilungen anzuzeigen, die Gehälter von insgesamt mehr als 200.000 Euro haben).
UNION (Seite 12)	die Ergebnisse zweier oder mehrerer SELECT-Anweisungen in einem einzigen Ergebnis zu kombinieren.
ORDER BY (Seite 13)	anzuzeigen, wie die Datensätze sortiert sind.
OFFSET (Seite 13)	die Anzahl der Zeilen anzugeben, die übersprungen werden, bevor damit begonnen wird, Zeilen abzurufen.
FETCH FIRST (Seite 13)	die Anzahl an abzurufenden Zeilen anzugeben. Es werden nicht mehr als die angegebene Anzahl an Zeilen zurückgegeben. Wenn die Abfrage jedoch weniger als die angegebene Anzahl an Zeilen ergibt, werden weniger Zeilen zurückgegeben.
FOR UPDATE (Seite 14)	„Positioned Updated“ und „Positioned Deletes“ über SQL-Cursor durchzuführen.

Hinweis Wenn Sie versuchen, Daten von einer Tabelle ohne Spalten abzurufen, gibt die SELECT-Anweisung nichts zurück.

FROM-Klausel

Die FROM-Klausel zeigt die Tabellen an, die in der SELECT-Anweisung verwendet werden. Das Format ist:

```
FROM tabellenname [tabellenalias] [, tabellenname [tabellenalias]]
```

tabellenname ist der Name einer Tabelle in der aktuellen Datenbank. Der Tabellenname muss mit einem Zeichen aus dem Alphabet beginnen. Wenn der Tabellenname mit etwas anderem als einem Zeichen aus dem Alphabet beginnt, schließen Sie ihn in Anführungszeichen ein (Quoted Identifier).

tabellenalias kann verwendet werden, um der Tabelle einen beschreibenderen Namen zu geben, einen langen Tabellennamen abzukürzen oder die gleiche Tabelle mehr als einmal in die Abfrage aufzunehmen (z. B. bei Self-Joins).

Feldnamen beginnen mit einem Zeichen aus dem Alphabet. Wenn der Feldname mit einem anderen Zeichen beginnt, umschließen Sie ihn mit Anführungszeichen (Quoted Identifier).

Beispiel:

Die „SQLAusführen“-Anweisung für das Feld namens _NACHNAME lautet zum Beispiel:

```
SELECT "_NACHNAME" from ang
```

Der Tabellenname oder der Tabellenalias kann Feldnamen vorangestellt werden.

Beispiel:

Zum Beispiel können Sie mit der Tabellenspezifikation `FROM angestellte A` das Feld `NACHNAME` als `A.NACHNAME` angeben. Tabellenaliasse müssen verwendet werden, wenn die `SELECT`-Anweisung eine Tabelle mit sich selbst verknüpft.

```
SELECT * FROM angestellte A, angestellte F WHERE A.managernr =
F.angestelltennr
```

Das Gleichheitszeichen (=) nimmt nur passende Zeilen in die Ergebnisse auf.

Wenn Sie mehr als eine Tabelle verknüpfen und alle Zeilen auslassen möchten, die nicht in beiden Quelltabellen über entsprechende Zeilen verfügen, können Sie `INNER JOIN` verwenden.

Beispiel:

```
SELECT *
   FROM Verkaeufer INNER JOIN Vertriebsdaten
   ON Verkaeufer.Verkaeufernr = Vertriebsdaten.Verkaeufernr
```

Wenn Sie zwei Tabellen verbinden, aber Zeilen der ersten Tabelle (die „linke“ Tabelle) nicht verwerfen möchten, können Sie `LEFT OUTER JOIN` verwenden.

Beispiel:

```
SELECT *
   FROM Verkaeufer LEFT OUTER JOIN Vertriebsdaten
   ON Verkaeufer.Verkaeufernr = Vertriebsdaten.Verkaeufernr
```

Jede Zeile aus der Tabelle „Verkaeufer“ erscheint in der verbundenen Tabelle.

Hinweise

- `RIGHT OUTER JOIN` wird zurzeit nicht unterstützt.
- `FULL OUTER JOIN` wird zurzeit nicht unterstützt.

WHERE-Klausel

Die WHERE-Klausel gibt die Bedingungen an, die Datensätze erfüllen müssen, um abgerufen zu werden. Die WHERE-Klausel enthält Bedingungen in der Form:

```
WHERE ausdr1 rel_operator ausdr2
```

ausdr1 und ausdr2 können Feldnamen, Konstantenwerte oder Ausdrücke sein.

rel_operator ist der relationale Operator, der die beiden Ausdrücke verbindet.

Beispiel:

Fragen Sie die Namen von Mitarbeitern ab, die 20.000 Euro oder mehr verdienen.

```
SELECT nachname,vorname FROM ang WHERE gehalt >= 20000
```

Die WHERE-Klausel kann auch Ausdrücke wie diese verwenden:

```
WHERE expr1 IS NULL
WHERE NOT expr2
```

Hinweis Wenn Sie vollständig qualifizierte Namen in der SELECT-Liste (Projektion) verwenden, müssen Sie auch vollständig qualifizierte Namen in der zugehörigen WHERE-Klausel verwenden.

GROUP BY-Klausel

Die GROUP BY-Klausel gibt die Namen eines oder mehrerer Felder an, nach denen die Ergebniswerte gruppiert werden sollen. Diese Klausel wird verwendet, um eine Menge von Aggregatwerten zurückzugeben. Sie hat folgendes Format:

```
GROUP BY Spalten
```

Der Umfang der GROUP BY-Klausel ist der Tabellenausdruck in der FROM-Klausel. Daher müssen die in Spalten angegebenen Spaltenausdrücke aus den in der FROM-Klausel angegebenen Tabellen stammen. Ein Spaltenausdruck kann ein oder mehrere Feldnamen der Datenbanktabelle, getrennt durch Kommata, sein.

Beispiel:

Summieren Sie die Gehälter in jeder Abteilung auf.

```
SELECT abtnr, SUM (gehalt) FROM ang GROUP BY abtnr
```

Diese Anweisung gibt für jede Abteilungsnummer eine Zeile zurück. Jede Zeile enthält die Abteilungsnummer und die Summe der Gehälter der Mitarbeiter in der Abteilung.

HAVING-Klausel

Die HAVING-Klausel ermöglicht Ihnen, die Bedingungen für Gruppen von Datensätzen anzugeben (z. B. um nur die Abteilungen anzuzeigen, die Gehälter von insgesamt mehr als 200.000 Euro haben). Sie hat folgendes Format:

```
HAVING ausdr1 rel_operator ausdr2
```

ausdr1 und ausdr2 können Feldnamen, Konstantenwerte oder Ausdrücke sein. Diese Ausdrücke müssen nicht mit einem Spaltenausdruck in der SELECT-Klausel übereinstimmen.

rel_operator ist der relationale Operator, der die beiden Ausdrücke verbindet.

Beispiel:

Geben Sie nur die Abteilungen zurück, deren Gehaltssummen größer als 200.000 Euro sind.

```
SELECT abtnr, SUM (gehalt) FROM ang
   GROUP BY abtnr HAVING SUM (gehalt) > 200000
```

UNION-Operator

Der UNION-Operator kombiniert die Ergebnisse von zwei oder mehr SELECT-Anweisungen in ein einziges Ergebnis. Das einzelne Ergebnis besteht aus den zurückgegebenen Datensätzen der SELECT-Anweisungen. Standardmäßig werden doppelte Datensätze nicht zurückgegeben. Um doppelte Datensätze zurückzugeben, verwenden Sie das Schlüsselwort ALL (UNION ALL). Das Format ist:

```
SELECT anweisung UNION [ALL] SELECT anweisung
```

Bei Verwendung des UNION-Operators müssen die Auswahllisten für jede SELECT-Anweisung die gleiche Anzahl an Spaltenausdrücken mit den gleichen Datentypen besitzen und in der gleichen Reihenfolge angegeben sein.

Beispiel:

```
SELECT nachname, gehalt, einst_datum FROM ang UNION SELECT name, zahlung,
geburtsdatum FROM person
```

Das folgende Beispiel ist nicht gültig, da sich die Datentypen der Spaltenausdrücke unterscheiden (GEHALT von ANG hat einen anderen Datentyp als NACHNAME von ERHOEHUNGEN). Dieses Beispiel hat die gleiche Anzahl an Spaltenausdrücken in jeder SELECT-Anweisung, aber die Ausdrücke erscheinen nach Datentyp nicht in der gleichen Reihenfolge.

Beispiel:

```
SELECT nachname, gehalt FROM ang UNION SELECT gehalt, nachname FROM
erhoehungen
```

ORDER BY-Klausel

Die ORDER BY-Klausel zeigt an, wie die Datensätze zu sortieren sind. Wenn Ihre SELECT-Anweisung keine ORDER BY-Klausel enthält, können die Datensätze in beliebiger Reihenfolge zurückgegeben werden.

Das Format ist:

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

sort_ausdruck können der Feldname oder die Positionsnummer des zu verwendenden Spaltenausdrucks sein. Standard ist die Durchführung einer aufsteigenden (ASC) Sortierung.

Beispiele

Sortieren Sie nach nachname und dann nach vorname.

```
SELECT angnr, nachname, vorname FROM ang ORDER BY nachname, vorname
```

Das zweite Beispiel verwendet die Positionsnummern 2 und 3, um die gleiche Sortierfolge wie im vorherigen Beispiel zu erhalten, das nachname und vorname explizit angegeben hat.

```
SELECT angnr, nachname, vorname FROM ang ORDER BY 2,3
```

Hinweis FileMaker SQL verwendet die binäre Unicode-Sortierfolge, die sich von der von FileMaker Pro verwendeten Sortierfolge mit Sprachsortierung oder mit sprachneutraler Standardsortierfolge unterscheidet.

OFFSET- und FETCH FIRST-Klauseln

Die Klauseln OFFSET und FETCH FIRST werden verwendet, um einen angegebenen Zeilenbereich ab einem bestimmten Startpunkt in einer Ergebnismenge zurückzugeben. Die Möglichkeit, die aus großen Ergebnismengen abgerufenen Zeilen zu beschränken, ermöglicht Ihnen, durch die Daten „zu blättern“, und verbessert die Effizienz.

Die Klausel OFFSET gibt die Anzahl an Zeilen an, die zu überspringen sind, bevor Daten zurückzugeben sind. Wenn die Klausel OFFSET in einer SELECT-Anweisung nicht verwendet wird, ist die Startzeile 0. Die Klausel FETCH FIRST gibt die Anzahl der Zeilen an, die zurückgegeben werden, entweder als Ganzzahl ohne Vorzeichen größer gleich 1 oder als Prozentsatz, ab dem Startpunkt in der Klausel OFFSET. Wenn sowohl OFFSET als auch FETCH FIRST in einer SELECT-Anweisung verwendet werden, muss die OFFSET-Klausel zuerst stehen.

Die Klauseln OFFSET und FETCH FIRST werden in Unterabfragen nicht unterstützt.

OFFSET-Format

Das Format von OFFSET ist:

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

n ist eine Ganzzahl ohne Vorzeichen. Wenn n größer als die Anzahl der in der Ergebnismenge zurückgegebenen Zeilen ist, wird nichts zurückgegeben und keine Fehlermeldung angezeigt.

ROWS ist identisch mit ROW.

FETCH FIRST-Format

Das `FETCH FIRST`-Format ist:

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

`n` die Anzahl an zurückzugebenden Zeilen an. Der Standardwert ist 1, wenn `n` ausgeschlossen wird.

`n` ist eine Ganzzahl ohne Vorzeichen größer oder gleich 1, wenn kein `PERCENT` folgt. Wenn nach `n` ein `PERCENT` folgt, kann der Wert entweder ein positiver Teilwert oder eine Ganzzahl ohne Vorzeichen sein.

`ROWS` ist identisch mit `ROW`.

`WITH TIES` muss zusammen mit der `ORDER BY`-Klausel verwendet werden.

`WITH TIES` erlaubt die Rückgabe von mehr Zeilen als im `FETCH`-Wert angegeben, da auch Peer-Zeilen zurückgegeben werden. Das sind Zeilen, die aufgrund der `ORDER BY`-Klausel nicht eindeutig sind.

Beispiele

Geben Sie Informationen aus der sechsundzwanzigsten Zeile der Ergebnismenge, sortiert nach `nachname` und dann nach `vorname` zurück.

```
SELECT angnr, nachname, vorname FROM ang ORDER BY nachname, vorname OFFSET
25 ROWS
```

Geben Sie an, dass Sie nur zehn Zeilen zurückgeben möchten.

```
SELECT angnr, nachname, vorname FROM ang ORDER BY nachname, vorname OFFSET
25 ROWS FETCH FIRST 10 ROWS ONLY
```

Geben Sie die zehn Zeilen und ihre Peer-Zeilen (Zeilen, die basierend auf der `ORDER BY`-Klausel nicht eindeutig sind) zurück.

```
SELECT angnr, nachname, vorname FROM ang ORDER BY nachname, vorname OFFSET
25 ROWS FETCH FIRST 10 ROWS WITH TIES
```

FOR UPDATE-Klausel

Die `FOR UPDATE`-Klausel sperrt Datensätze für „Positioned Updates“ oder „Positioned Deletes“ über SQL-Cursor. Das Format ist:

```
FOR UPDATE [OF spaltenausdruecke]
```

`spaltenausdruecke` ist eine Liste von Feldnamen in der Datenbanktabelle, die Sie aktualisieren möchten, getrennt durch ein Komma. `spaltenausdruecke` ist optional und wird ignoriert.

Beispiel:

Geben Sie alle Datensätze in der Angestellten-Datenbank zurück, die einen `GEHALT`-Feldwert von mehr als 20.000 Euro besitzen.

```
SELECT * FROM ang WHERE gehalt > 20000
   FOR UPDATE OF nachname, vorname, gehalt
```

Wenn jeder Datensatz abgerufen wird, wird er gesperrt. Wird der Datensatz aktualisiert oder gelöscht, wird die Sperre gehalten, bis Sie die Änderung bestätigen. Ansonsten wird die Sperre freigegeben, wenn Sie den nächsten Datensatz abrufen.

Beispiele

Verwenden von	Beispiel-SQL
Textkonstante	`SELECT 'KatzeHund' FROM Verkaeufer`
Zahlenkonstante	`SELECT 999 FROM Verkaeufer`
Datumskonstante	`SELECT DATE '2019-06-05' FROM Verkaeufer`
Zeitkonstante	`SELECT TIME '02:49:03' FROM Verkaeufer`
Zeitstempelkonstante	`SELECT TIMESTAMP '2019-06-05 02:49:03' FROM Verkaeufer`
Textspalte	`SELECT Firmenname FROM Vertriebsdaten` `SELECT DISTINCT Firmenname FROM Vertriebsdaten`
Zahlenspalte	`SELECT Betrag FROM Vertriebsdaten` `SELECT DISTINCT Betrag FROM Vertriebsdaten`
Datumsspalte	`SELECT Verkaufsdatum FROM Vertriebsdaten` `SELECT DISTINCT Verkaufsdatum FROM Vertriebsdaten`
Zeitspalte	`SELECT Verkaufszeit FROM Vertriebsdaten` `SELECT DISTINCT Verkaufszeit FROM Vertriebsdaten`
Zeitstempelspalte	`SELECT Verkaufszeitstempel FROM Vertriebsdaten` `SELECT DISTINCT Verkaufszeitstempel FROM Vertriebsdaten`
BLOB[a]-Spalte	`SELECT Firmenbroschueren FROM Vertriebsdaten` `SELECT GETAS(Firmenlogo, 'JPEG') FROM Vertriebsdaten`
Jokerzeichen *	`SELECT * FROM Verkaeufer` `SELECT DISTINCT * FROM Verkaeufer`

a. Ein BLOB ist ein FileMaker-Datenbankdatei-Containerfeld.

Hinweise zu den Beispielen

Eine `Spalte` ist ein Verweis auf ein Feld in der FileMaker-Datenbankdatei. (Das Feld kann viele unterschiedliche Werte enthalten.)

Das Jokerzeichen (*) ist eine Abkürzung für „Alles“. Für das Beispiel `SELECT * FROM Verkaeufer` ist das Ergebnis alle Spalten in der Tabelle `Verkaeufer`. Für das Beispiel `SELECT DISTINCT * FROM Verkaeufer` ist das Ergebnis alle eindeutigen Zeilen in der Tabelle `Verkaeufer` (keine doppelten Werte).

- FileMaker speichert keine Daten für leere Zeichenfolgen, so dass die folgenden Abfragen immer keine Datensätze zurückgeben:

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

- Wenn Sie `SELECT` mit Binärdaten verwenden, müssen Sie die Funktion `GetAs()` verwenden, um den zurückzugebenden Stream anzugeben. Weitere Informationen finden Sie im folgenden Abschnitt, „Abrufen des Inhalts eines Containerfelds: CAST()-Funktion und GetAs()-Funktion“.

Abrufen des Inhalts eines Containerfelds: CAST()-Funktion und GetAs()-Funktion

Sie können Dateiverweisinformationen, Binärdaten oder Daten eines angegebenen Dateityps von einem Containerfeld abrufen.

- Um Dateiverweisinformationen von einem Containerfeld wie den Dateipfad zu einer Datei, einem Bild oder einem QuickTime-Film abzurufen, verwenden Sie die `CAST`-Funktion mit einer `SELECT`-Anweisung.
- Wenn Dateidaten oder JPEG-Binärdaten existieren, ruft die `SELECT`-Anweisung mit `GetAS(feldname, 'JPEG')` die Daten in Binärform ab. Ansonsten gibt die `SELECT`-Anweisung mit Feldname `NULL` zurück.

Beispiel:

Verwenden Sie die Funktion `CAST()` zusammen mit einer `SELECT`-Anweisung, um Dateiverweisinformationen abzurufen.

```
SELECT CAST(Firmenbroschueren AS VARCHAR) FROM Vertriebsdaten
```

Wenn Sie in diesem Beispiel:

- eine Datei in das Containerfeld mithilfe von FileMaker Pro eingefügt haben, aber nur einen Verweis auf die Datei gespeichert haben, ruft die `SELECT`-Anweisung die Dateiverweisinformationen als Typ `SQL_VARCHAR` ab.
- den Inhalt einer Datei in das Containerfeld mithilfe von FileMaker Pro eingefügt haben, ruft die `SELECT`-Anweisung den Namen der Datei ab.
- eine Datei in das Containerfeld von einer anderen Anwendung importiert haben, zeigt die `SELECT`-Anweisung '?' an (die Datei wird als **Untitled.dat** in FileMaker Pro angezeigt).

Sie können die `SELECT`-Anweisung mit der Funktion `GetAs()` auf folgende Arten verwenden, um die Daten in Binärform abzurufen:

- Wenn Sie die Funktion `GetAs()` mit der Option `DEFAULT` verwenden, rufen Sie den Masterstream für den Container ab, ohne den Streamtyp exakt definieren zu müssen.

  Beispiel:

  ```
  SELECT GetAs(Firmenprospekte, DEFAULT) FROM Vertriebsdaten
  ```

- Um einen einzelnen Streamtyp aus einem Container abzurufen, verwenden Sie die Funktion `GetAs()` mit dem Typ der Datei, je nachdem, wie die Daten in das Containerfeld in FileMaker Pro eingegeben wurden.

  Beispiel:

  Wenn die Daten mit dem Befehl **Einfügen** > **Datei** eingefügt wurden, geben Sie `'FILE'` in der Funktion `GetAs()` an.

  ```
  SELECT GetAs(Firmenprospekte, 'FILE') FROM Vertriebsdaten
  ```

Beispiel:

Wenn die Daten mit dem **Befehl Einfügen** > **Bild**, per Drag & Drop oder aus der Zwischenablage eingefügt wurden, geben Sie einen der Dateitypen ein, die in der folgenden Tabelle genannt werden, z. B. `'JPEG'`.

```
SELECT GetAs(Firmenlogo, 'JPEG') FROM Firmensymbole
```

Dateityp	Beschreibung
`'GIFf'`	Graphics Interchange Format
`'JPEG'`	Fotografische Bilder
`'TIFF'`	Raster-Dateiformat für digitale Bilder
`'PDF'`	Portable Document Format
`'PNGf'`	Bitmap-Bildformat

DELETE-Anweisung

Verwenden Sie die `DELETE`-Anweisung, um Datensätze aus einer Datenbanktabelle zu löschen. Das Format der `DELETE`-Anweisung ist:

```
DELETE FROM tabellenname [ WHERE { bedingungen } ]
```

Hinweis Die `WHERE`-Klausel legt fest, welche Datensätze gelöscht werden. Wenn Sie das Schlüsselwort `WHERE` nicht verwenden, werden alle Datensätze in der Tabelle gelöscht (aber die Tabelle bleibt intakt).

Beispiel:

Löschen Sie einen Datensatz aus der Tabelle `ang`.

```
DELETE FROM ang WHERE angnr = 'A10001'
```

Jede `DELETE`-Anweisung entfernt jeden Datensatz, der die Bedingungen der `WHERE`-Klausel erfüllt. In diesem Fall wird jeder Datensatz gelöscht, der die Angestelltennummer `A10001` hat. Da in der Angestellten-Tabelle Angestelltennummern eindeutig sind, wird nur ein Datensatz gelöscht.

INSERT-Anweisung

Verwenden Sie die `INSERT`-Anweisung, um Datensätze in einer Datenbanktabelle zu erstellen. Sie können angeben:

- Eine Liste von Werten, die als neuer Datensatz eingefügt werden
- Eine `SELECT`-Anweisung, die als neuen Satz von Datensätzen einzufügende Daten aus einer anderen Tabelle kopiert

Das Format der `INSERT`-Anweisung ist:

```
INSERT INTO tabellenname [(spaltenname, ...)] VALUES (ausdr, ...)
```

`spaltenname` ist eine optionale Liste von Spaltennamen, die den Namen und die Reihenfolge der Spalten angibt, deren Werte in der `VALUES`-Klausel angegeben sind. Wenn Sie `spaltenname` nicht angeben, müssen die Werteausdrücke (`ausdr`) Werte für alle in der Tabelle definierten Spalten angeben und in der gleichen Reihenfolge sein wie die für die Tabelle definierten Spalten. `spaltenname` kann auch eine Feldwiederholung, zum Beispiel `lastDates[4]`, angeben.

`ausdr` ist die Liste der Ausdrücke, die Werte für die Spalten des neuen Datensatzes zur Verfügung stellt. Gewöhnlich sind die Ausdrücke konstante Werte für die Spalten (sie können aber auch Unterabfragen sein). Sie müssen Zeichenfolgenwerte in einfachen Anführungszeichen (') angeben. Um ein einfaches Anführungszeichen in einer Zeichenfolge, die durch einfache Anführungszeichen eingeschlossen ist, aufzunehmen, verwenden Sie zwei einfache Anführungszeichen (z. B. `'ist''s'`).

Unterabfragen müssen in Klammern angegeben werden.

Beispiel:

Fügen Sie eine Liste von Ausdrücken ein.

```
INSERT INTO ang (nachname, vorname, angnr, gehalt, einst_datum)
   VALUES ('Schmidt, 'Johann', 'E22345', 27500, DATE '2019-06-05')
```

Jede `INSERT`-Anweisung fügt der Datenbanktabelle einen Datensatz hinzu. In diesem Fall wurde der Angestellten-Datenbanktabelle `ang` ein Datensatz hinzugefügt. Werte werden für fünf Spalten angegeben. Den restlichen Spalten in der Tabelle wird ein leerer Wert, also null, zugeordnet.

Hinweis In Containerfeldern können Sie `INSERT` nur Text, wenn Sie keine parametrisierte Anweisung vorbereiten und die Daten aus Ihrer Anwendung streamen. Um Binärdaten zu verwenden, können Sie einfach den Dateinamen zuordnen, indem Sie ihn in einfachen Anführungszeichen angeben, oder Sie verwenden die Funktion `PutAs()`. Wenn Sie den Dateinamen angeben, wird der Dateityp aus der Dateierweiterung abgeleitet:

```
INSERT INTO tabellenname (containername) VALUES(? AS 'dateiname.dateierweiterung')
```

Nicht unterstützte Dateitypen werden als Typ FILE eingefügt.

Wenn Sie die Funktion `PutAs()` verwenden, geben Sie den Typ an: `PutAs(col, 'typ')`, wobei der Typwert ein Typ ist, der unter „Abrufen des Inhalts eines Containerfelds: CAST()-Funktion und GetAs()-Funktion“ auf Seite 16 beschrieben wird.

Die `SELECT`-Anweisung ist eine Abfrage, die Werte für jeden in der Spaltenname-Liste angegebenen Wert `spaltenname` zurückgibt. Die Verwendung einer `SELECT`-Anweisung anstelle einer Liste von Wertausdrücken ermöglicht Ihnen, eine Menge von Zeilen aus einer Tabelle auszuwählen und sie in eine andere Tabelle mit einer einzelnen `INSERT`-Anweisung einzufügen.

Beispiel:

Fügen Sie mit einer `SELECT`-Anweisung ein.

```
INSERT INTO ang1 (vorname, nachname, angnr, abt, gehalt)
   SELECT vorname, nachname, angnr, abt, gehalt from ang
   WHERE abt = 'D050'
```

In dieser Art von INSERT-Anweisung muss die Anzahl der einzufügenden Spalten der Anzahl der Spalten in der SELECT-Anweisung entsprechen. Die Liste der einzufügenden Spalten muss den Spalten in der SELECT-Anweisung so entsprechen, wie sie einer Liste von Wertausdrücken in einer anderen Art von INSERT-Anweisung entsprechen würde. Zum Beispiel entspricht die erste eingefügte Spalte der ersten ausgewählten Spalte, die zweite eingefügte der zweiten usw.

Größe und Datentyp dieser entsprechenden Spalten müssen kompatibel sein. Jede Spalte in der SELECT-Liste sollte über einen Datentyp verfügen, den der ODBC- bzw. JDBC-Treiber bei einem regulären INSERT/UPDATE der entsprechenden Spalte in der INSERT-Liste akzeptiert. Werte werden abgeschnitten, wenn die Größe des Werts in der SELECT-Listenspalte größer als die Größe der entsprechenden INSERT-Listenspalte ist.

Die SELECT-Anweisung wird vor allen eingefügten Werten ausgewertet.

UPDATE-Anweisung

Verwenden Sie die UPDATE-Anweisung, um Datensätze in einer Datenbanktabelle zu ändern. Das Format der UPDATE-Anweisung ist:

```
UPDATE tabellenname SET spaltenname = ausdr, ... [ WHERE { bedingungen } ]
```

spaltenname ist der Name einer Spalte, deren Wert zu ändern ist. Mehrere Spalten können in einer Anweisung geändert werden.

ausdr ist der neue Wert für die Spalte.

Gewöhnlich sind die Ausdrücke konstante Werte für die Spalten (sie können aber auch Unterabfragen sein). Sie müssen Zeichenfolgenwerte in einfachen Anführungszeichen (') angeben. Um ein einfaches Anführungszeichen in einer Zeichenfolge, die durch einfache Anführungszeichen eingeschlossen ist, aufzunehmen, verwenden Sie zwei einfache Anführungszeichen (z. B. 'ist''s').

Unterabfragen müssen in Klammern angegeben werden.

Die WHERE-Klausel ist jede gültige Klausel. Sie bestimmt, welche Datensätze aktualisiert werden.

Beispiel:

UPDATE-Anweisung für die Tabelle ang.

```
UPDATE ang SET gehalt=32000, steuerfrei=1 WHERE angnr = 'A10001'
```

Die UPDATE-Anweisung ändert jeden Datensatz, der die Bedingungen der WHERE-Klausel erfüllt. In diesem Fall werden Gehalt und Steuerfreiheit für alle Angestellten mit der Angestelltennummer A10001 geändert. Da in der Angestellten-Tabelle Angestelltennummern eindeutig sind, wird nur ein Datensatz aktualisiert.

Beispiel:

UPDATE-Anweisung für die Tabelle ang mit einer Unterabfrage.

```
UPDATE ang SET gehalt = (SELECT avg(gehalt) from ang ) WHERE angnr = 'A10001'
```

In diesem Fall wird das Gehalt für jeden Angestellten mit der Angestelltennummer A10001 auf den Gehaltsmittelwert des Unternehmens geändert.

Hinweis In Containerfeldern können Sie `UPDATE` mit nur Text, wenn Sie keine parametrisierte Anweisung vorbereiten und die Daten aus Ihrer Anwendung streamen. Um Binärdaten zu verwenden, können Sie einfach den Dateinamen zuordnen, indem Sie ihn in einfachen Anführungszeichen angeben, oder Sie verwenden die Funktion `PutAs()`. Wenn Sie den Dateinamen angeben, wird der Dateityp aus der Dateierweiterung abgeleitet:

```
UPDATE tabellenname SET (containername) = ? AS 'dateiname.dateierweiterung'
```

Nicht unterstützte Dateitypen werden als Typ FILE eingefügt.

Wenn Sie die Funktion `PutAs()` verwenden, geben Sie den Typ an: `PutAs(col, 'typ')`, wobei der Typwert ein Typ ist, der unter „Abrufen des Inhalts eines Containerfelds: CAST()-Funktion und GetAs()-Funktion“ auf Seite 16 beschrieben wird.

CREATE TABLE-Anweisung

Verwenden Sie die `CREATE TABLE`-Anweisung, um eine Tabelle in einer Datenbankdatei zu erstellen. Das Format der `CREATE TABLE`-Anweisung ist:

```
CREATE TABLE tabellenname ( tabellenelementliste [, tabellenelementliste...] )
```

In der Anweisung geben Sie Name und Datentyp jeder Spalte an.

- `tabellenname` ist der Name der Tabelle. `tabellenname` ist auf 100 Zeichen beschränkt. Eine Tabelle mit dem gleichen Namen darf nicht bereits definiert sein. Der Tabellenname muss mit einem Zeichen aus dem Alphabet beginnen. Wenn der Tabellenname mit etwas anderem als einem Zeichen aus dem Alphabet beginnt, schließen Sie ihn in Anführungszeichen ein (Quoted Identifier).
- Das Format für `tabellenelementliste` ist:

  ```
  feld_name feld_typ [[wiederholungen]]
  [DEFAULT ausdr] [UNIQUE | NOT NULL | PRIMARY KEY | GLOBAL]
  [EXTERNAL relativer_pfad_zeichenfolge [SECURE | OPEN
  formel_pfad_zeichenfolge]]
  ```

  - `feld_name` ist der Name des Felds. Feldnamen müssen eindeutig sein. Feldnamen beginnen mit einem Zeichen aus dem Alphabet. Wenn der Feldname mit einem anderen Zeichen beginnt, umschließen Sie ihn mit Anführungszeichen (Quoted Identifier).

    Beispiel:

    Die `CREATE TABLE`-Anweisung für das Feld namens `_NACHNAME` lautet:

    ```
    CREATE TABLE "_ANGESTELLTER" (ID INT PRIMARY KEY, "_VORNAME"
    VARCHAR(20), "_NACHNAME" VARCHAR(20))
    ```

  - Geben Sie für die `CREATE TABLE`-Anweisung `wiederholungen` eine Feldwiederholung in Form einer Zahl zwischen 1 und 32000 in Klammern nach dem Feldtyp an.

Beispiel:

```
MITARBEITERNR INT[4]
NACHNAME VARCHAR(20)[4]
```

- Für `Feldtyp` sind folgende Optionen möglich: `NUMERIC`, `DECIMAL`, `INT`, `DATE`, `TIME`, `TIMESTAMP`, `VARCHAR`, `CHARACTER VARYING`, `BLOB`, `VARBINARY`, `LONGVARBINARY` und `BINARY VARYING`. Für `NUMERIC` und `DECIMAL` können Sie Genauigkeit und Skala angeben. Beispiel: `DECIMAL(10,0)`. Für `TIME` und `TIMESTAMP` können Sie die Genauigkeit angeben. Beispiel: `TIMESTAMP(6)`. Für `VARCHAR` und `CHARACTER VARYING` können Sie die Länge der Zeichenfolge angeben.

Beispiel:

```
VARCHAR(255)
```

- Über das Schlüsselwort `DEFAULT` können Sie einen Standardwert für eine Spalte festlegen. Für `ausdr` können Sie einen konstanten Wert oder einen Ausdruck verwenden. Zulässige Ausdrücke sind `USER`, `USERNAME`, `CURRENT_USER`, `CURRENT_DATE`, `CURDATE`, `CURRENT_TIME`, `CURTIME`, `CURRENT_TIMESTAMP`, `CURTIMESTAMP` und `NULL`.
- Die Definition einer Spalte als `UNIQUE` wählt automatisch die Überprüfungsoption **Eindeutig** für das entsprechende Feld in der FileMaker-Datenbankdatei aus.
- Die Definition einer Spalte als `NOT NULL` wählt automatisch die Überprüfungsoption **Nicht leer** für das entsprechende Feld in der FileMaker-Datenbankdatei aus. Das Feld wird als **Wert erforderlich** im Register **Felder** des Dialogfelds „Datenbank verwalten“ in FileMaker Pro markiert.
- Um eine Spalte als Containerfeld zu definieren, verwenden Sie `BLOB`, `VARBINARY` oder `BINARY VARYING` für den `feldtyp`.
- Um eine Spalte als Containerfeld zu definieren, das Daten extern speichert, verwenden Sie das Schlüsselwort `EXTERNAL`. `relativer_pfad_string` definiert den Ordner, in dem Daten extern, relativ zum Speicherort der FileMaker-Datenbank gespeichert werden. Dieser Pfad muss als Basisverzeichnis im FileMaker Pro-Dialogfeld „Container verwalten“ angegeben werden. Sie müssen entweder `SECURE` für einen sicheren Speicher oder `OPEN` für einen offenen Speicher angeben. Wenn Sie einen offenen Speicher verwenden, ist der `berechn_pfad_string` der Ordner in dem Ordner `relativer_pfad_string`, in dem Containerobjekte gespeichert werden sollen. Der Pfad muss Schrägstriche (/) im Ordnernamen verwenden.

Beispiele

Verwenden von	Beispiel-SQL
Textspalte	`CREATE TABLE T1 (C1 VARCHAR, C2 VARCHAR (50), C3 VARCHAR (1001), C4 VARCHAR (500276))`
Textspalte, `NOT NULL`	`CREATE TABLE T1NN (C1 VARCHAR NOT NULL, C2 VARCHAR (50) NOT NULL, C3 VARCHAR (1001) NOT NULL, C4 VARCHAR (500276) NOT NULL)`
Zahlenspalte	`CREATE TABLE T2 (C1 DECIMAL, C2 DECIMAL (10,0), C3 DECIMAL (7539,2), C4 DECIMAL (497925,301))`

Verwenden von	Beispiel-SQL
Datumsspalte	`CREATE TABLE T3 (C1 DATE, C2 DATE, C3 DATE, C4 DATE)`
Zeitspalte	`CREATE TABLE T4 (C1 TIME, C2 TIME, C3 TIME, C4 TIME)`
Zeitstempelspalte	`CREATE TABLE T5 (C1 TIMESTAMP, C2 TIMESTAMP, C3 TIMESTAMP, C4 TIMESTAMP)`
Spalte für Containerfeld	`CREATE TABLE T6 (C1 BLOB, C2 BLOB, C3 BLOB, C4 BLOB)`
Spalte für extern gespeichertes Containerfeld	`CREATE TABLE T7 (C1 BLOB EXTERNAL 'Dateien/MeineDatenbank/' SECURE)` `CREATE TABLE T8 (C1 BLOB EXTERNAL 'Dateien/MeineDatenbank/' OPEN 'Objects')`

TRUNCATE TABLE-Anweisung

Verwenden Sie die `TRUNCATE TABLE`-Anweisung, um rasch alle Datensätze in der angegebenen Tabelle zu löschen und alle Daten aus der Tabelle zu entfernen.

```
TRUNCATE TABLE tabellenname
```

Sie können keine `WHERE`-Klausel zusammen mit der `TRUNCATE TABLE`-Anweisung angeben. Die `TRUNCATE TABLE`-Anweisung löscht alle Datensätze.

Nur die Datensätze in der durch `tabellenname` angegebenen Tabelle werden gelöscht. Datensätze in Bezugstabellen sind davon nicht betroffen.

Die `TRUNCATE TABLE`-Anweisung muss alle Datensätze in der Tabelle sperren können, um die Datensatzdaten zu löschen. Falls ein Datensatz von einem anderen Anwender gesperrt ist, gibt FileMaker den Fehlercode 301 („Datensatz ist blockiert durch anderen Anwender“) zurück.

ALTER TABLE-Anweisung

Verwenden Sie die `ALTER TABLE`-Anweisung, um die Struktur einer bestehenden Tabelle in einer Datenbankdatei zu ändern. Sie können in jeder Anweisung nur eine Spalte ändern. Die Formate der `ALTER TABLE`-Anweisung sind:

```
ALTER TABLE tabellenname ADD [COLUMN] spaltendefinition
ALTER TABLE tabellenname DROP [COLUMN] unqualifizierter_spaltenname
ALTER TABLE tabellenname ALTER [COLUMN] spaltendefinition SET DEFAULT ausdr
ALTER TABLE tabellenname ALTER [COLUMN] spaltendefinition DROP DEFAULT
```

Sie müssen die Struktur der Tabelle kennen und wissen, wie Sie sie ändern, bevor Sie die `ALTER TABLE`-Anweisung verwenden.

Beispiele

Für	Beispiel-SQL
Spalten hinzufügen	`ALTER TABLE Verkaeufer ADD C1 VARCHAR`
Spalten entfernen	`ALTER TABLE Verkaeufer DROP C1`
Den Standardwert für eine Spalte festlegen	`ALTER TABLE Verkaeufer ALTER Firma SET DEFAULT 'FileMaker'`
Den Standardwert für eine Spalte entfernen	`ALTER TABLE Verkaeufer ALTER Firma DROP DEFAULT`

Hinweis `SET DEFAULT` und `DROP DEFAULT` wirken sich nicht auf vorhandene Zeilen in der Tabelle aus, aber ändern den Standardwert für Zeilen, die später der Tabelle hinzugefügt werden.

CREATE INDEX-Anweisung

Verwenden Sie die `CREATE INDEX`-Anweisung, um Suchen in einer Datenbankdatei zu beschleunigen. Das Format der `CREATE INDEX`-Anweisung ist:

```
CREATE INDEX ON tabellenname.spaltenname
CREATE INDEX ON tabellenname (spaltenname)
```

`CREATE INDEX` wird für eine einzelne Spalte unterstützt (Mehrspaltenindizes werden nicht unterstützt). Indizes sind bei Spalten nicht zulässig, die Containerfeldtypen, Statistikfeldern, Feldern mit globaler Speicherung oder nicht gespeicherten Formelfeldern in einer FileMaker-Datenbankdatei entsprechen.

Das Erstellen eines Index für eine Textspalte wählt die Speicheroption **Minimal** unter **Indizierung** für das entsprechende Feld in der FileMaker-Datenbankdatei automatisch aus. Das Erstellen eines Index für eine Nicht-Textspalte (oder eine als japanischer Text formatierte Spalte) wählt die Speicheroption **Alle** unter **Indizierung** für das entsprechende Feld in der FileMaker-Datenbankdatei automatisch aus.

Das Erstellen eines Index für eine beliebige Spalte wählt die Speicheroption **Indizes bei Bedarf automatisch erstellen** unter **Indizierung** für das entsprechende Feld in der FileMaker-Datenbankdatei automatisch aus.

FileMaker erstellt Indizes bei Bedarf automatisch. Die Verwendung von `CREATE INDEX` bewirkt, dass der Index direkt als nur „bei Bedarf“ erstellt wird.

Beispiel:

```
CREATE INDEX ON Verkaeufer.VerkauferID
```

DROP INDEX-Anweisung

Verwenden Sie die `DROP INDEX`-Anweisung, um einen Index aus einer Datenbankdatei zu entfernen. Das Format der `DROP INDEX`-Anweisung ist:

```
DROP INDEX ON tabellenname.spaltenname
DROP INDEX ON tabellenname (spaltenname)
```

Entfernen Sie einen Index, wenn Ihre Datenbankdatei zu groß ist oder Sie ein Feld nicht häufig in Abfragen verwenden.

Wenn Ihre Abfragen langsam ausgeführt werden und Sie mit einer sehr großen FileMaker-Datenbankdatei mit vielen indizierten Textfeldern arbeiten, sollten Sie in Erwägung ziehen, die Indizes einiger Felder zu entfernen. Erwägen Sie auch, die Indizes von Feldern zu entfernen, die Sie selten in `SELECT`-Anweisungen verwenden.

Das Entfernen eines Index für eine beliebige Spalte wählt die Speicheroption **Ohne** unter **Indizierung** für das entsprechende Feld in der FileMaker-Datenbankdatei automatisch aus und deaktiviert die Option **Indizes bei Bedarf automatisch erstellen**.

Das Attribut `PREVENT INDEX CREATION` wird nicht unterstützt.

Beispiel:

```
DROP INDEX ON Verkaeufer.Verkaeufernr
```

SQL-Ausdrücke

Verwenden Sie Ausdrücke in den Klauseln `WHERE`, `HAVING` und `ORDER BY` `SELECT`-Anweisungen, um detaillierte und raffinierte Datenbankabfragen zu erstellen. Gültige Ausdruckelemente sind:

- Feldnamen
- Konstanten
- Exponentialschreibweise
- Numerische Operatoren
- Zeichenoperatoren
- Datumsoperatoren
- Relationale Operatoren
- Logische Operatoren
- Funktionen

Feldnamen

Der gängigste Ausdruck ist ein einfacher Feldname wie `formel` oder `Vertriebsdaten.Rechnungsnr`.

Konstanten

Konstanten sind Werte, die sich nicht ändern. Zum Beispiel ist im Ausdruck `PREIS * 1,05` der Wert 1,05 eine Konstante. Oder Sie weisen der Konstante `Anzahl_der_Tage_im_Juni` einen Wert von 30 zu.

Sie müssen Zeichenkonstanten in einfachen Anführungszeichen (') angeben. Um ein einfaches Anführungszeichen in einer Zeichenkonstanten, die durch einfache Anführungszeichen eingeschlossen ist, aufzunehmen, verwenden Sie zwei einfache Anführungszeichen (z. B. `'ist''s'`).

Für ODBC- und JDBC-Anwendungen: FileMaker akzeptiert die Konstanten für ODBC/JDBC-Formatdatum, -Zeit und -Zeitstempel in geschweiften Klammern ({}).

Beispiele

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

FileMaker gestattet die Verwendung von Groß- und Kleinbuchstaben (`D`, `T`, `TS`) als Typangabe. Sie können eine beliebige Anzahl an Leerzeichen nach der Typangabe verwenden oder das Leerzeichen sogar weglassen.

FileMaker akzeptiert auch die ISO-Datums- und Zeitformate der SQL-92-Syntax ohne geschweifte Klammern.

Beispiele

- `DATE 'JJJJ-MM-TT'`
- `TIME 'HH:MM:SS'`
- `TIMESTAMP 'JJJJ-MM-TT HH:MM:SS'`

Die FileMaker Pro-Funktion „SQLAusführen“ akzeptiert nur ISO-Datums- und Zeitformate der SQL-92-Syntax ohne geschweifte Klammern.

Konstante	Akzeptable Syntax (Beispiele)
Text	`'Paris'`
Zahl	`1.05`

Datum	`DATE '2019-06-05'`
	`{ D '2019-06-05' }`
	`{06/05/2019}`
	`{06/05/19}`
	Hinweis Die Syntax mit zweistelliger Jahreszahl wird für das ODBC/JDBC-Format bzw. das SQL-92-Format nicht unterstützt.

Zeit	`TIME '14:35:10'`
	`{ T '14:35:10' }`
	`{14:35:10}`

Zeitstempel	`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}`
	Stellen Sie sicher, dass **Strenger Datentyp: Vierstellige Jahreszahl** nicht als Überprüfungsoption in der FileMaker-Datenbankdatei für ein Feld ausgewählt ist, das diese zweistellige Jahressyntax verwendet.
	Hinweis Die Syntax mit zweistelliger Jahreszahl wird für das ODBC/JDBC-Format bzw. das SQL-92-Format nicht unterstützt.

Wenn Sie Datums- und Zeitwerte eingeben, verwenden Sie das Format der Sprachumgebung der Datenbankdatei. Wenn die Datenbank z. B. in einem italienischen Sprachsystem erstellt wurde, verwenden Sie die italienischen Datums- und Zeitformate.

Exponentialschreibweise

Zahlen können auch in wissenschaftlicher Schreibweise angegeben werden.

Beispiel:

```
SELECT spalte1 / 3.4E+7 FROM tabelle1 WHERE formel < 3.4E-6 * spalte2
```

Numerische Operatoren

In Zahlenausdrücken können Sie folgende Operatoren aufnehmen: +, -, *, /, und ^ oder ** (Exponent).

Sie können numerischen Ausdrücken ein Plus (+) oder Minus (-) voranstellen.

Zeichenoperatoren

Sie können Zeichen verketten. In den folgenden Beispielen ist `nachname 'JONAS '` und vorname `'ROBERT '`:

Operator	Verkettung	Beispiel	Ergebnis
+	Führende Leerzeichen beibehalten	vorname + nachname	'ROBERT JONAS '
-	Führende Leerzeichen ans Ende bewegen	vorname - nachname	'ROBERTJONAS '

Datumsoperatoren

Sie können Datumswerte verändern. In den folgenden Beispielen ist `einst_datum = DATE '2019-01-30'`.

Operator	Wirkung auf Datum	Beispiel	Ergebnis
+	Einem Datum eine Anzahl von Tagen hinzufügen	`einst_datum` + 5	DATE '2019-02-04'
-	Die Anzahl der Tage zwischen zwei Datumswerten hinzufügen	`einst_datum - DATE '2019-01-01'`	`29`
	Einem Datum eine Anzahl von Tagen abziehen	`einstdatum - 10`	DATE '2019-01-20'

Weitere Beispiele

```
SELECT Verkaufsdatum, Verkaufsdatum + 30 AS agg FROM Vertriebsdaten
SELECT Verkaufsdatum, Verkaufsdatum - 30 AS agg FROM Vertriebsdaten
```

Relationale Operatoren

Operator	Bedeutung
=	Ist gleich
<>	Ist ungleich
>	Größer als
>=	Größer oder gleich
<	Kleiner als
<=	Kleiner oder gleich
LIKE	Entspricht einem Muster
NOT LIKE	Entspricht nicht einem Muster
IS NULL	Ist gleich null
IS NOT NULL	Ist nicht gleich null
BETWEEN	Bereich von Werten zwischen einer unteren und oberen Grenze
IN	Teil einer Menge von angegebenen Werten oder Teil einer Unterabfrage
NOT IN	Nicht Teil einer Menge von angegebenen Werten oder Teil einer Unterabfrage
EXISTS	'Wahr', wenn eine Unterabfrage wenigstens einen Datensatz zurückgibt
ANY	Vergleicht einen Wert mit jedem Wert, der von einer Unterabfrage zurückgegeben wird (dem Operator muss ein =, <>, >, >=, <, oder <= vorangestellt sein); =ANY entspricht IN
ALL	Vergleicht einen Wert mit jedem Wert, der von einer Unterabfrage zurückgegeben wird (dem Operator muss ein =, <>, >, >=, <, oder <= vorangestellt sein)

Beispiel:

```
SELECT Vertriebsdaten.Rechnungsnr FROM Vertriebsdaten
   WHERE Vertriebsdaten.Verkaeufernr = 'SP-1'
SELECT Vertriebsdaten.Betrag FROM Vertriebsdaten
   WHERE Vertriebsdaten.Rechnungsnr <> 125
SELECT Vertriebsdaten.Betrag FROM Vertriebsdaten
   WHERE Vertriebsdaten.Betrag > 3000
SELECT Vertriebsdaten.Verkaufszeit FROM Vertriebsdaten
   WHERE Vertriebsdaten.Verkaufszeit < '12:00:00'
SELECT Vertriebsdaten.Firmenname FROM Vertriebsdaten
   WHERE Vertriebsdaten.Firmenname LIKE '%Universität'
SELECT Vertriebsdaten.Firmenname FROM Vertriebsdaten
   WHERE Vertriebsdaten.Firmenname NOT LIKE '%Universität'
SELECT Vertriebsdaten.Betrag FROM Vertriebsdaten
   WHERE Vertriebsdaten.Betrag IS NULL
SELECT Vertriebsdaten.Betrag FROM Vertriebsdaten
   WHERE Vertriebsdaten.Betrag IS NOT NULL
SELECT Vertriebsdaten.Rechnungsnr FROM Vertriebsdaten
   WHERE Vertriebsdaten.Rechnungsnr BETWEEN 1 AND 10
SELECT COUNT (Vertriebsdaten.Rechnungsnr) AS agg
   FROM Vertriebsdaten WHERE Vertriebsdaten.Rechnungsnr IN (50,250,100)
SELECT COUNT (Vertriebsdaten.Rechnungsnr) AS agg
   FROM Vertriebsdaten WHERE Vertriebsdaten.Rechnungsnr NOT IN (50,250,100)
SELECT COUNT (Vertriebsdaten.Rechnungsnr) AS agg FROM Vertriebsdaten
   WHERE Vertriebsdaten.Rechnungsnr NOT IN (SELECT
Vertriebsdaten.Rechnungsnr
   FROM Vertriebsdaten WHERE Vertriebsdaten.Verkaeufernr = 'SP-4')
SELECT *
   FROM Vertriebsdaten WHERE EXISTS (SELECT Vertriebsdaten.Betrag
   FROM Vertriebsdaten WHERE Vertriebsdaten.Verkaeufernr IS NOT NULL)
SELECT *
   FROM Vertriebsdaten WHERE Vertriebsdaten.Betrag = ANY (SELECT
Vertriebsdaten.Betrag
   FROM Vertriebsdaten WHERE Vertriebsdaten.Verkaeufernr = 'SP-1')
SELECT *
   FROM Vertriebsdaten WHERE Vertriebsdaten.Betrag = ALL (SELECT
Vertriebsdaten.Betrag
   FROM Vertriebsdaten WHERE Vertriebsdaten.Verkaeufernr IS NULL)
```

Logische Operatoren

Sie können zwei oder mehrere Bedingungen kombinieren. Die Bedingungen müssen mit `AND` oder `OR` in Beziehung stehen:

```
gehalt = 40000 AND steuerfrei = 1
```

Der logische Operator NOT wird verwendet, um die Bedeutung umzukehren:

```
NOT (gehalt = 40000 AND steuerfrei = 1)
```

Beispiel:

```
SELECT * FROM Vertriebsdaten WHERE Vertriebsdaten.Firmenname
   NOT LIKE '%Universität' AND Vertriebsdaten.Betrag > 3000
SELECT * FROM Vertriebsdaten WHERE (Vertriebsdaten.Firmenname
   LIKE '%Universität' OR Vertriebsdaten.Betrag > 3000)
   AND Vertriebsdaten.Verkaeufernr = 'SP-1'
```

Priorität der Operatoren

Wenn die Ausdrücke komplexer werden, wird die Reihenfolge wichtig, in der die Ausdrücke ausgewertet werden. Diese Tabelle zeigt die Reihenfolge, in der die Operatoren ausgewertet werden. Die Operatoren in der ersten Zeile werden zuerst ausgewertet usw. Operatoren in der gleichen Zeile werden im Ausdruck von links nach rechts ausgewertet.

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

Beispiele

```
WHERE gehalt > 40000 OR einst_datum (DATE '2008-01-30') AND abt =
'A101'
```

Weil AND zuerst ausgewertet wird, ruft diese Abfrage Angestellte in Abteilung A101 ab, die nach dem Mittwoch, 30. Januar 2008 eingestellt wurden, sowie jeden Angestellten mit mehr als 40.000 Euro Gehalt, unabhängig von Abteilung oder Einstellungsdatum.

Um die Klausel in einer anderen Reihenfolge auszuwerten, verwenden Sie Klammern um die zuerst auszuwertenden Bedingungen.

```
WHERE (gehalt > 40000 OR einst_datum > DATE '2008-01-30') AND abt =
'A101'
```

Dieses Beispiel ruft Angestellte in Abteilung A101 ab, die entweder mehr als 40.000 Euro verdienen oder nach dem 30. Januar 2008 eingestellt wurden.

SQL-Funktionen

FileMaker SQL unterstützt viele Funktionen, die Sie in Ausdrücken verwenden können. Einige der Funktionen geben Buchstabenzeichenfolgen, einige Zahlen, einige Datumswerte und einige Werte zurück, die von Bedingungen abhängen, die die Funktionsargumente erfüllen.

Statistikfunktionen

Statistikfunktionen geben einen Wert aus einer Menge von Datensätzen zurück. Sie können eine Statistikfunktion als Teil einer SELECT-Anweisung mit einem Feldnamen (zum Beispiel AVG(GEHALT)) oder in Kombination mit einem Spaltenausdruck (zum Beispiel AVG(GEHALT * 1,07)) verwenden.

Sie können dem Spaltenausdruck den DISTINCT-Operator voranstellen, um doppelte Werte zu eliminieren.

Beispiel:

```
COUNT (DISTINCT nachname)
```

In diesem Beispiel werden nur eindeutige Nachnamenswerte gezählt.

Statistikfunktion	Ergebnis
SUM	Die Summe der Werte in einem Zahlenfeldausdruck. Beispiel: SUM(GEHALT) gibt die Summe aller Gehaltsfeldwerte zurück.
AVG	Der Mittelwert der Werte in einem Zahlenfeldausdruck. Beispiel: AVG(GEHALT) gibt den Mittelwert aller Gehaltsfeldwerte zurück.
COUNT	Die Anzahl der Werte in einem Feldausdruck. Beispiel: COUNT(NAME) gibt die Anzahl aller Namenswerte zurück. Bei Verwendung von COUNT mit einem Feldnamen gibt COUNT die Anzahl der Feldwerte ungleich null zurück. Ein spezielles Beispiel ist COUNT(*), das die Anzahl der Datensätze in einer Menge zurückgibt, einschließlich der Datensätze mit null Werten.
MAX	Der Maximalwert in einem Feldausdruck. Beispiel: MAX(GEHALT) gibt den maximalen Gehaltsfeldwert zurück.
MIN	Der Minimalwert in einem Feldausdruck. Beispiel: MIN(GEHALT) gibt den minimalen Gehaltsfeldwert zurück.

Beispiel:

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

Sie können eine Statistikfunktion nicht als Argument für andere Funktionen verwenden. Sonst gibt FileMaker den Fehlercode 8309 („Ausdrücke mit Statistikfunktionen werden nicht unterstützt.“) zurück. Die folgende Anweisung ist zum Beispiel ungültig, da die Statistikfunktion SUM nicht als Argument für die Funktion ROUND verwendet werden kann:

Beispiel:

```
SELECT ROUND(SUM(Gehalt), 0) FROM Loehne
```

Statistikfunktionen können jedoch Funktionen verwenden, die Zahlen als Argumente liefern. Die folgende Anweisung ist gültig:

Beispiel:

```
SELECT SUM(ROUND(Gehalt), 0) FROM Loehne
```

Funktionen, die Zeichenfolgen zurückgeben

Funktionen, die Zeichenfolgen zurückgeben	Beschreibung	Beispiel
CHR	Konvertiert einen ASCII-Code in eine Zeichenfolge mit einem Zeichen	`CHR(67)` ergibt **C**
CURRENT_USER	Gibt die zur Verbindungszeit angegebene Anmeldekennung zurück	
DAYNAME	Gibt den Namen des Tages zurück, der einem angegebenen Datum entspricht	
RTRIM	Entfernt nachfolgende Leerzeichen aus einer Zeichenfolge	`RTRIM('  ABC ')` ergibt `'  ABC'`
TRIM	Entfernt führende und nachfolgende Leerzeichen aus einer Zeichenfolge	`TRIM(' ABC ')` ergibt `'ABC'`
LTRIM	Entfernt führende Leerzeichen aus einer Zeichenfolge	`LTRIM(' ABC')` ergibt `'ABC'`
UPPER	Ändert jeden Buchstaben einer Zeichenfolge zu Großbuchstaben	`UPPER('Allen')` ergibt `'ALLEN'`
LOWER	Ändert jeden Buchstaben einer Zeichenfolge zu Kleinbuchstaben	`LOWER('Allen')` ergibt `'allen'`
LEFT	Gibt die Zeichen angefangen links zurück	`LEFT('Mattson',3)` ergibt `'Mat'`
MONTHNAME	Ergibt den Namen des Kalendermonats	
RIGHT	Gibt die Zeichen angefangen rechts zurück	`RIGHT('Mattson',4)` ergibt `'tson'`
SUBSTR SUBSTRING	Gibt eine Unterzeichenfolge einer Zeichenfolge mit Parametern der Zeichenfolge, dem ersten zu extrahierenden Zeichen und der Anzahl der zu extrahierenden Zeichen zurück (optional)	`SUBSTR('Conrad',2,3)` ergibt `'onr'` `SUBSTR('Conrad',2)` ergibt `'onrad'`
SPACE	Erzeugt eine Zeichenfolge mit Leerzeichen	`SPACE(5)` ergibt `'  '`

STRVAL	Konvertiert einen Wert beliebigen Typs in eine Buchstabenzeichenfolge	`STRVAL('Woltman')` ergibt `'Woltman'`
		`STRVAL(5 * 3)` ergibt `'15'`
		`STRVAL(4 = 5)` ergibt `'False'`
		`STRVAL(DATE '2019-12-25')` ergibt `'2019-12-25'`

TIME TIMEVAL	Ergibt die Tageszeit als Zeichenfolge	Um 21:49 ergibt `TIME()` **21:49:00**
USERNAME USER	Gibt die zur Verbindungszeit angegebene Anmeldekennung zurück	

Hinweis Die Funktion `TIME()` ist veraltet. Verwenden Sie stattdessen den SQL-Standard `CURRENT_TIME`.

Beispiel:

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

SELECT RTRIM(' ' + Verkaeufer.Verkaeufernr) AS agg FROM Verkaeufer

SELECT TRIM(SPACE(1) + Verkaeufer.Verkaeufernr) AS agg FROM Verkaeufer

SELECT LTRIM(' ' + Verkaeufer.Verkaeufernr) AS agg FROM Verkaeufer

SELECT UPPER(Verkaeufer.Verkaeufernr) AS agg FROM Verkaeufer

SELECT LOWER(Verkaeufer.Verkaeufernr) AS agg FROM Verkaeufer

SELECT LEFT(Verkaeufer.Verkaeufernr, 5) AS agg FROM Verkaeufer

SELECT RIGHT(Verkaeufer.Verkaeufernr, 7) AS agg FROM Verkaeufer

SELECT SUBSTR(Verkaeufer.Verkaeufernr, 2, 2) +
SUBSTR(Verkaeufer.Verkaeufernr, 4, 2) AS agg FROM Verkaeufer

SELECT SUBSTR(Verkaeufer.Verkaeufernr, 2) +
SUBSTR(Verkaeufer.Verkaeufernr, 4) AS agg FROM Verkaeufer

SELECT SPACE(2) + Verkaeufer.Verkaeufernr AS Verkaeufernr FROM Verkaeufer

SELECT STRVAL('60506') AS agg FROM Vertriebsdaten WHERE
Vertriebsdaten.Rechnung = 1
```

Funktionen, die Zahlen zurückgeben

Funktionen, die Zahlen zurückgeben	Beschreibung	Beispiel
ABS	Ergibt den absoluten Wert des numerischen Ausdrucks	
ATAN	Ergibt den Arcustangens des Arguments als Winkel in Bogenmaß	
ATAN2	Ergibt den Arcustangens der x- und y-Koordinaten als Winkel in Bogenmaß	
CEIL CEILING	Ergibt den kleinsten Ganzzahlwert, der größer oder gleich dem Argument ist	
DEG DEGREES	Ergibt die Anzahl an Grad des Arguments, das ein Winkel ist, in Bogenmaß	
DAY	Gibt den Tagesteil eines Datums zurück	`DAY (DATE '2019-01-30')` ergibt **30**
DAYOFWEEK	Gibt den Tag der Woche (1-7) eines Datumsausdrucks zurück	`DAYOFWEEK(DATE '2004-05-01')` ergibt **7**
MOD	Teilt zwei Zahlen und gibt den Rest der Division zurück	`MOD(10,3)` ergibt **1**
EXP	Ergibt einen Wert, der die Basis des natürlichen Logarithmus (e) hoch des Arguments ist	
FLOOR	Ergibt den größten Ganzzahlwert, der kleiner oder gleich dem Argument ist	
HOUR	Gibt den Stundenteil eines Werts zurück	
INT	Gibt den ganzzahligen Teil einer Zahl zurück	`INT(6,4321)` ergibt **6**
LENGTH	Gibt die Länge einer Zeichenfolge zurück	`LENGTH('ABC')` ergibt **3**
MONTH	Gibt den Monatsteil eines Datums zurück	`MONTH (DATE '2019-01-30')` ergibt **1**
LN	Ergibt den natürlichen Logarithmus des Arguments	
LOG	Ergibt den natürlichen Logarithmus des Arguments	
MAX	Gibt die größere von zwei Zahlen zurück	`MAX(66,89)` ergibt **89**
MIN	Gibt die kleinere von zwei Zahlen zurück	`MIN(66,89)` ergibt **66**
MINUTE	Gibt den Minutenteil eines Werts zurück	
NUMVAL	Konvertiert eine Buchstabenzeichenfolge in eine Zahl. Die Funktion schlägt fehl, wenn die Buchstabenzeichenfolge keine gültige Zahl ist	`NUMVAL('123')` ergibt **123**
PI	Gibt den konstanten Wert der mathematischen Konstante pi zurück	
RADIANS	Gibt das Bogenmaß für ein Argument zurück, das in Grad ausgedrückt ist	
ROUND	Rundet eine Zahl	`ROUND(123.456,0)` ergibt **123** `ROUND(123.456,2)` ergibt **123,46** `ROUND(123.456,-2)` ergibt **100**

Funktionen, die Zahlen zurückgeben	Beschreibung	Beispiel
SECOND	Gibt den Sekundenteil eines Werts zurück	
SIGN	Ein Indikator des Vorzeichens des Arguments: -1 für negativ, 0 für 0 und 1 für positiv	
SIN	Gibt den Sinus des Arguments zurück	
SQRT	Gibt die Quadratwurzel des Arguments zurück	
TAN	Gibt den Tangens des Arguments zurück	
YEAR	Gibt den Jahresteil eines Datums zurück	YEAR (DATE '2019-01-30') ergibt **2019**

Funktionen, die Datumswerte zurückgeben

Funktionen, die Datumswerte zurückgeben	Beschreibung	Beispiel
CURDATE CURRENT_DATE	Gibt das heutige Datum zurück	
CURTIME CURRENT_TIME	Gibt die aktuelle Uhrzeit zurück	
CURTIMESTAMP CURRENT_TIMESTAMP	Gibt den aktuellen Zeitstempelwert zurück	
TIMESTAMPVAL	Konvertiert eine Buchstabenzeichenfolge in einen Zeitstempel	TIMESTAMPVAL('2019-01-30 14:00:00') ergibt den Zeitstempelwert
DATE TODAY	Gibt das heutige Datum zurück	Wenn heute der 21.11.19 ist, ergibt DATE() **2019-11-21**
DATEVAL	Konvertiert eine Buchstabenzeichenfolge in ein Datum	DATEVAL('2019-01-30') ergibt **30.01.2019**

Hinweis Die Funktion DATE() ist veraltet. Verwenden Sie stattdessen den SQL-Standard CURRENT_DATE.

Bedingte Funktion

Bedingte Funktion	Beschreibung	Beispie
CASE WHEN	**Einfaches CASE-Format** Vergleicht den Wert von *eingabe_ausdr* mit den Werten der Argumente von *wert_ausdr*, um das Ergebnis zu bestimmen. CASE *eingabe_ausdr* {WHEN *wert_ausdr* THEN *ergebnis*...} [ELSE *ergebnis*] END	SELECT Rechnungsnr, CASE Firmenname WHEN 'Exportiert GB' THEN 'Exportiert GB gefunden' WHEN 'Hausmöbellieferanten' THEN 'Hausmöbellieferanten gefunden' ELSE 'Exportiert weder GB noch Hausmöbellieferanten' END, Verkaeufernr FROM Vertriebsdaten
	Gesuchtes CASE-Format Gibt ein Ergebnis basierend darauf zurück, ob die in einem WHEN-Ausdruck angegebene Bedingung wahr ist. CASE {WHEN *boolescher_ausdr* THEN *ergebnis...*} [ELSE *ergebnis*] END	SELECT Rechnungsnr, Betrag, CASE WHEN Betrag > 3000 THEN 'Über 3000' WHEN Betrag < 1000 THEN 'Unter 1000' ELSE 'Zwischen 1000 und 3000' END, Verkaeufernr FROM Vertriebsdaten
COALESCE	Gibt den ersten Wert zurück, der nicht NULL ist	SELECT Verkaeufernr, COALESCE(Vertriebsleiter, Verkaeufer) FROM Verkäufer
NULLIF	Vergleicht zwei Werte und gibt NULL zurück, wenn die zwei Werte gleich sind; ansonsten gibt sie den ersten Wert zurück	SELECT Rechnungsnr, NULLIF(Betrag, -1), Verkaeufernr FROM Vertriebsdaten

FileMaker-Systemobjekte

FileMaker-Datenbankdateien umfassen die folgenden Systemobjekte, auf die Sie mittels SQL-Abfragen zugreifen können.

FileMaker-Systemtabellen

Jede FileMaker-Datenbankdatei enthält zwei Systemtabellen: FileMaker_Tables und FileMaker_Fields. Für ODBC-Anwendungen sind diese Tabellen in den Informationen enthalten, die die Katalogfunktion SQLTables zurückgibt. Für JDBC-Anwendungen sind diese Tabellen in den Informationen enthalten, die die DatabaseMetaData-Methode getTables zurückgibt. Die Tabellen können auch in SQLAusführen-Funktionen verwendet werden.

FileMaker_Tables

Die Tabelle „FileMaker_Tables“ enthält Informationen über die in der FileMaker-Datei definierten Datenbanktabellen.

Die Tabelle „FileMaker_Tables“ enthält für jedes Tabellenauftreten im Beziehungsdiagramm eine Zeile mit folgenden Spalten:

- TableName - der Name des Tabellenauftretens.
- TableId - die eindeutige ID für das Tabellenauftreten.
- BaseTableName – der Name der Basistabelle, aus der das Tabellenauftreten erstellt wurde.
- BaseFileName – der FileMaker-Dateiname für die Datenbankdatei, die die Basistabelle enthält.
- ModCount – die Anzahl, wie oft Änderungen an der Definition dieser Tabelle geschrieben wurden.

Beispiel:

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

Tabelle „FileMaker_Fields“

Die Tabelle „FileMaker_Fields“ enthält Informationen über die in der FileMaker-Datei definierten Felder.

Die Tabelle „FileMaker_Fields“ enthält die folgenden Spalten:

- TableName – der Name der Tabelle, die das Feld enthält.
- FieldName – der Name des Felds.
- FieldType – der SQL-Datentyp des Felds.
- FieldId – die eindeutige ID für das Feld.
- FieldClass - einer von drei Werten: „Summary“ für Auswertungsfelder, „Calculate“ für berechnete Ergebnisse oder „Normal“.
- FieldReps – die Anzahl der Wiederholungen des Felds.
- ModCount – die Anzahl, wie oft Änderungen an der Definition dieser Tabelle geschrieben wurden.

Beispiel:

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

FileMaker-Systemspalten

FileMaker fügt allen Zeilen (Datensätzen) in allen Tabellen, die in der FileMaker-Datei definiert sind, Systemspalten (Felder) hinzu. Für ODBC-Anwendungen sind diese Spalten in den Informationen enthalten, die die Katalogfunktion SQLSpecialColumns zurückgibt. Für JDBC-Anwendungen sind diese Spalten in den Informationen enthalten, die die DatabaseMetaData-Methode getVersionColumns zurückgibt. Diese Spalten können auch in SQLAusführen-Funktionen verwendet werden.

ROWID-Spalte

Die Systemspalte `ROWID` enthält die eindeutige ID-Zahl des Datensatzes. Das ist der gleiche Wert, den die FileMaker Pro-Funktion Hole(DatensatzIDNr) zurückgibt.

ROWMODID-Spalte

Die Systemspalte `ROWMODID` enthält die Anzahl der Änderungen am aktuellen Datensatz, die geschrieben wurden. Das ist der gleiche Wert, den die FileMaker Pro-Funktion Hole(DatensatzÄnderungenAnzahl) zurückgibt.

Beispiel:

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

Reservierte SQL-Schlüsselwörter

In diesem Abschnitt werden die reservierten Schlüsselwörter aufgeführt, die nicht als Namen für Spalten, Tabellen, Aliasse oder andere benutzerdefinierte Objekte verwendet werden dürfen. Syntaxfehler können auf die Verwendung dieser reservierten Schlüsselwörter zurückzuführen sein. Wenn Sie eines dieser Schlüsselwörter verwenden möchten, müssen Sie Anführungszeichen einsetzen, um zu vermeiden, dass dieses Wort als Schlüsselwort behandelt wird.

Beispiel:

Verwenden Sie das Schlüsselwort `DEC` als Datenelementname.

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

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

Index

J

K

L

M

N

O

P

R

S

T

U

V

W

Y

Z