

FileMaker® 15

SQL-Referenzhandbuch



© 2013-2016 FileMaker, Inc. Alle Rechte vorbehalten.

FileMaker, Inc.

5201 Patrick Henry Drive

Santa Clara, California 95054, USA

FileMaker und FileMaker Go sind Marken von FileMaker, Inc., eingetragen in den USA und anderen Ländern. Das Dateiodner-Logo und FileMaker WebDirect sind Marken von FileMaker, Inc. Alle anderen Marken sind Eigentum der jeweiligen Besitzer.

Die FileMaker-Dokumentation ist urheberrechtlich geschützt. Sie dürfen diese Dokumentation ohne schriftliche Genehmigung von FileMaker weder vervielfältigen noch verteilen. Diese Dokumentation darf ausschließlich mit einer gültigen, lizenzierten Kopie der FileMaker-Software verwendet werden.

Alle in den Beispielen erwähnten Personen, Firmen, E-Mail-Adressen und URLs sind rein fiktiv und jegliche Ähnlichkeit mit bestehenden Personen, Firmen, E-Mail-Adressen und URLs ist rein zufällig. Die Danksagungen und Urheberrechtshinweise finden Sie im entsprechenden Dokument, das mit der Software geliefert wurde. Die Erwähnung von Produkten und URLs Dritter dient nur zur Information und stellt keine Empfehlung dar. FileMaker, Inc. übernimmt keine Verantwortung für die Leistung dieser Produkte.

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

Edition: 01

Inhalt

Kapitel 1

Einführung

Über diese Referenz	5
Speicherort der FileMaker-Dokumentation	5
Über SQL	5
Verwenden einer FileMaker-Datenbank als Datenquelle	6
Verwenden der Funktion „SQLAusführen“	6

Kapitel 2

Unterstützte Standards

Unterstützung von Unicode-Zeichen	7
SQL-Anweisungen	7
SELECT-Anweisung	8
SQL-Klauseln	9
FROM-Klausel	9
WHERE-Klausel	10
GROUP BY-Klausel	11
HAVING-Klausel	11
UNION-Operator	12
ORDER BY-Klausel	12
OFFSET- und FETCH FIRST-Klauseln	13
FOR UPDATE-Klausel	14
DELETE-Anweisung	17
INSERT-Anweisung	17
UPDATE-Anweisung	19
CREATE TABLE-Anweisung	20
TRUNCATE TABLE-Anweisung	22
ALTER TABLE-Anweisung	22
CREATE INDEX-Anweisung	23
DROP INDEX-Anweisung	23
SQL-Ausdrücke	24
Feldnamen	24
Konstanten	24
Exponentialschreibweise	25
Numerische Operatoren	25
Zeichenoperatoren	25
Datumsoperatoren	26
Relationale Operatoren	27
Logische Operatoren	28
Priorität der Operatoren	29

SQL-Funktionen	30
Statistikfunktionen	30
Funktionen, die Zeichenfolgen zurückgeben	31
Funktionen, die Zahlen zurückgeben	32
Funktionen, die Datumswerte zurückgeben	34
Bedingte Funktion	34
FileMaker-Systemobjekte	35
FileMaker-Systemtabellen	35
FileMaker-Systemspalten	36
Reservierte SQL-Schlüsselwörter	37
<i>Index</i>	40

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 Tabellenauftritten 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 erhalten Sie unter <http://www.filemaker.com/documentation/de>.
- 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.

Speicherort der FileMaker-Dokumentation

- Wählen Sie in FileMaker Pro **Hilfe > Produktdokumentation**.
- Wählen Sie in FileMaker Server Admin Console **Hilfe > FileMaker Server Produktdokumentation**.
- Wenn Sie FileMaker-Dokumentation anzeigen oder herunterladen möchten oder sich über die Dokumentation informieren möchten, besuchen Sie <http://www.filemaker.com/documentation/de>.

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-Queries 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 Tabellenauftritten abrufen, die im Beziehungsdigramm 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 Beziehungsdigramme 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 zu Syntax und Verwendung der Funktion „SQLAusführen“ finden Sie in der [FileMaker Pro Hilfe](#).

Kapitel 2

Unterstützte Standards

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

Verwenden Sie die 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. 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	Für
FROM (Seite 9)	anzuzeigen, welche Tabellen in der <code>SELECT</code> -Anweisung verwendet werden.
WHERE (Seite 10)	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 11)	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 <code>SELECT</code> -Anweisungen in einem einzigen Ergebnis zu kombinieren.
ORDER BY (Seite 12)	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 Tabellename muss mit einem Zeichen aus dem Alphabet beginnen. Wenn der Tabellename 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 Query 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). Die `ExecuteSQL`-Anweisung für das Feld namens `_NACHNAME` lautet zum Beispiel:

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

Der Tabellename oder der Tabellenalias kann Feldnamen vorangestellt werden. 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. Beispiel:

```
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 Verkaeufuer INNER JOIN Vertriebsdaten
ON Verkaeufuer.Verkaeufuernr = Vertriebsdaten.Verkaeufuernr
```

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

```
SELECT *
FROM Verkaeufuer LEFT OUTER JOIN Vertriebsdaten
ON Verkaeufuer.Verkaeufuernr = Vertriebsdaten.Verkaeufuernr
```

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

Notizen

- `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. Die folgende `SELECT`-Anweisung ruft zum Beispiel die Namen der Angestellten ab, die ein Gehalt von mehr als 20.000 Euro haben.

```
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 `columns` 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:

Das folgende Beispiel summiert die Gehälter jeder Abteilung.

```
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:

Das folgende Beispiel gibt 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
```

Dieses Beispiel hat die gleiche Anzahl an Spaltenausdrücken und jeder Spaltenausdruck in der Reihenfolge hat den gleichen Datentyp.

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.

```
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.

Um zum Beispiel zuerst nach `nachname` und dann nach `vorname` zu sortieren, können Sie eine der folgenden `SELECT`-Anweisungen verwenden:

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

ODER

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

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.

Hinweis FileMaker SQL verwendet die Unicode-Binärsortierfolge, die sich von der FileMaker Pro-Sortierfolge unterscheidet, die für die Sprachsortierung bzw. für die sprachneutrale Standardsortierfolge verwendet wird.

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 zurückgegeben 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 Nicht-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

Um zum Beispiel Informationen aus der sechszwanzigsten Zeile der Ergebnismenge, sortiert nach `nachname` und dann nach `vorname` zurückzugeben, verwenden Sie folgende `SELECT`-Anweisung:

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

Um anzugeben, 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
```

Um die zehn Zeilen und ihre Peer-Zeilen (Zeilen, die basierend auf der ORDER BY-Klausel nicht eindeutig sind) zurückzugeben:

```
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:

Das folgende Beispiel gibt alle Datensätze in der Angestellten-Datenbank zurück, die einen GEHALT-Feldwert von mehr als 20.000 Euro besitzen. 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.

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

Weitere Beispiele:

Verwenden von	Beispiel-SQL
Textkonstante	<code>SELECT 'KatzeHund' FROM Verkaeufer</code>
Zahlenkonstante	<code>SELECT 999 FROM Verkaeufer</code>
Datumskonstante	<code>SELECT DATE '2016-06-05' FROM Verkaeufer</code>
Zeitkonstante	<code>SELECT TIME '02:49:03' FROM Verkaeufer</code>
Zeitstempelkonstante	<code>SELECT TIMESTAMP '2016-06-05 02:49:03' FROM Verkaeufer</code>
Textspalte	<code>SELECT Firmenname FROM Vertriebsdaten</code> <code>SELECT DISTINCT Firmenname FROM Vertriebsdaten</code>
Zahlenspalte	<code>SELECT Betrag FROM Vertriebsdaten</code> <code>SELECT DISTINCT Betrag FROM Vertriebsdaten</code>
Datumsspalte	<code>SELECT Verkaufsdatum FROM Vertriebsdaten</code> <code>SELECT DISTINCT Verkaufsdatum FROM Vertriebsdaten</code>
Zeitspalte	<code>SELECT Verkaufszeit FROM Vertriebsdaten</code> <code>SELECT DISTINCT Verkaufszeit FROM Vertriebsdaten</code>
Zeitstempelspalte	<code>SELECT Verkaufszeitstempel FROM Vertriebsdaten</code> <code>SELECT DISTINCT Verkaufszeitstempel FROM Vertriebsdaten</code>
BLOB ^a -Spalte	<code>SELECT Firmenbroschueren FROM Vertriebsdaten</code> <code>SELECT GETAS(Firmenlogo, 'JPEG') FROM Vertriebsdaten</code>
Jokerzeichen *	<code>SELECT * FROM Verkaeufer</code> <code>SELECT DISTINCT * FROM Verkaeufer</code>

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

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.

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. Beispiel:

```
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).

Um Daten aus einem Containerfeld abzurufen, verwenden Sie die die Funktion `GetAs()`. Sie können auch die Option `DEFAULT` nutzen oder den Dateityp angeben. Die Option `DEFAULT` ruft den Masterstream für den Container ab, ohne dass der Streamtyp explizit zu definieren ist:

```
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. Beispiel:

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

- Wenn die Daten mit dem Befehl **Einfügen > Ton** (Standardton – MAC OS X-Rohformat) eingefügt wurden, geben Sie `'snd'` in der Funktion `GetAs()` an. Beispiel:

```
SELECT GetAs(Firmenmeeting, 'snd') FROM Firma_Newsletter
```


- Wenn die Daten mit dem Befehl **Einfügen > Bild** eingefügt wurden, ziehen und legen Sie ab, fügen Sie aus der Zwischenablage ein, geben Sie einen der Dateitypen ein, die in der folgenden Tabelle genannt werden. Beispiel:

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

Dateityp	Beschreibung	Dateityp	Beschreibung
'GIFf'	Graphics Interchange Format	'PNTG'	MacPaint
'JPEG'	Fotografische Bilder	' .SGI'	Generisches Bitmap-Format
'JP2 '	JPEG 2000	'TIFF'	Raster-Dateiformat für digitale Bilder
'PDF '	Portable Document Format	'TPIC'	Targa
'PNGf'	Bitmap-Bildformat	'8BPS'	PhotoShop (PSD)

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:

Ein Beispiel einer `DELETE`-Anweisung in der Angestellten-Tabelle lautet:

```
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.

Das folgende Beispiel fügt eine Liste von Ausdrücken ein:

```
INSERT INTO ang (nachname, vorname, angnr, gehalt, einst_datum)
VALUES ('Schmidt, 'Johann', 'E22345', 27500, DATE '05.06.2016')
```

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 nur Text `INSERT`, 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.

Hier ein Beispiel einer `INSERT`-Anweisung, die eine `SELECT`-Anweisung verwendet:

```
INSERT INTO angl (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.

Beispiele

Ein Beispiel einer `UPDATE`-Anweisung in der Angestellten-Tabelle lautet:

```
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.

Hier sehen Sie ein Beispiel 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 nur Text UPDATE, 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 Tabellename muss mit einem Zeichen aus dem Alphabet beginnen. Wenn der Tabellename 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]]
```

- `feldname` 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). Die `CREATE TABLE`-Anweisung für das Feld namens `_NACHNAME` lautet zum Beispiel:

```
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	<code>CREATE TABLE T1 (C1 VARCHAR, C2 VARCHAR (50), C3 VARCHAR (1001), C4 VARCHAR (500276))</code>
Textspalte, NOT NULL	<code>CREATE TABLE T1NN (C1 VARCHAR NOT NULL, C2 VARCHAR (50) NOT NULL, C3 VARCHAR (1001) NOT NULL, C4 VARCHAR (500276) NOT NULL)</code>
Zahlenspalte	<code>CREATE TABLE T2 (C1 DECIMAL, C2 DECIMAL (10,0), C3 DECIMAL (7539,2), C4 DECIMAL (497925,301))</code>
Datumsspalte	<code>CREATE TABLE T3 (C1 DATE, C2 DATE, C3 DATE, C4 DATE)</code>
Zeitspalte	<code>CREATE TABLE T4 (C1 TIME, C2 TIME, C3 TIME, C4 TIME)</code>
Zeitstempelspalte	<code>CREATE TABLE T5 (C1 TIMESTAMP, C2 TIMESTAMP, C3 TIMESTAMP, C4 TIMESTAMP)</code>
Spalte für Containerfeld	<code>CREATE TABLE T6 (C1 BLOB, C2 BLOB, C3 BLOB, C4 BLOB)</code>
Spalte für extern gespeichertes Containerfeld	<code>CREATE TABLE T7 (C1 BLOB EXTERNAL 'Dateien/MeineDatenbank/' SECURE)</code> <code>CREATE TABLE T8 (C1 BLOB EXTERNAL 'Dateien/MeineDatenbank/' OPEN 'Objects')</code>

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 („Datei 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	<code>ALTER TABLE Verkaeufner ADD C1 VARCHAR</code>
Spalten entfernen	<code>ALTER TABLE Verkaeufner DROP C1</code>
Legt den Standardwert für eine Spalte fest	<code>ALTER TABLE Verkaeufner ALTER Firma SET DEFAULT 'FileMaker'</code>
Entfernt den Standardwert für eine Spalte	<code>ALTER TABLE Verkaeufner ALTER Firma DROP DEFAULT</code>

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.VerkaeferID
```

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.Verkaeuferrnr
```

SQL-Ausdrücke

Verwenden Sie 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 ({}), zum Beispiel:

- `{D '2016-06-05'}`
- `{T '14:35:10'}`
- `{TS '2016-06-05 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:

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

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

Konstante	Akzeptable Syntax (Beispiele)
Text	'Paris'
Zahl	1.05
Datum	DATE '2016-06-05' { D '2016-06-05' } {06/05/2016} {06/05/16} 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 '2016-06-05 14:35:10' { TS '2016-06-05 14:35:10' } {06/05/2016 14:35:10} {06/05/16 14:35:10} Stellen Sie sicher, dass Streng, 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 verketteten.

Beispiele

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.

Beispiele

In den folgenden Beispielen ist `einst_datum` DATE '30.01.2016'.

Operator	Wirkung auf Datum	Beispiel	Ergebnis
+	Einem Datum eine Anzahl von Tagen hinzufügen	<code>einst_datum + 5</code>	DATE '2016-02-04'
-	Die Anzahl der Tage zwischen zwei Datumswerten hinzufügen	<code>einst_datum - DATE '01.01.2016'</code>	29
-	Einem Datum eine Anzahl von Tagen abziehen	<code>einstdatum - 10</code>	DATE '2016-01-20'

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)

Beispiele

```
SELECT Vertriebsdaten.Rechnungsnr FROM Vertriebsdaten
      WHERE Vertriebsdaten.Verkaeuferrnr = '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)
```

Beispiele

```
SELECT * FROM Vertriebsdaten WHERE Vertriebsdaten.Firmenname
NOT LIKE '%Universität' AND Vertriebsdaten.Betrag > 3000
```

```
SELECT * FROM VertriebsdatenWHERE (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	Hinweis
7	AND
8	OR

Das folgende Beispiel verdeutlicht die Bedeutung der Priorität:

```
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. Beispiel:

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

ruft Angestellte in Abteilung A101 ab, die entweder mehr als 40.000 Euro verdienen oder nach dem Mittwoch, 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: <code>SUM(GEHALT)</code> gibt die Summe aller Gehaltsfeldwerte zurück.
AVG	Der Mittelwert der Werte in einem Zahlenfeldausdruck. Beispiel: <code>AVG(GEHALT)</code> gibt den Mittelwert aller Gehaltsfeldwerte zurück.
COUNT	Die Anzahl der Werte in einem Feldausdruck. Beispiel: <code>COUNT(NAME)</code> gibt die Anzahl aller Namenswerte zurück. Bei Verwendung von <code>COUNT</code> mit einem Feldnamen gibt <code>COUNT</code> die Anzahl der Feldwerte ungleich null zurück. Ein spezielles Beispiel ist <code>COUNT(*)</code> , das die Anzahl der Datensätze in einer Menge zurückgibt, einschließlich der Datensätze mit null Werten.
MAX	Der Maximalwert in einem Feldausdruck. Beispiel: <code>MAX(GEHALT)</code> gibt den maximalen Gehaltsfeldwert zurück.
MIN	Der Minimalwert in einem Feldausdruck. Beispiel: <code>MIN(GEHALT)</code> gibt den minimalen Gehaltsfeldwert zurück.

Beispiele

```
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:

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

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

```
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('Konrad',2,3) ergibt 'onr' SUBSTR('Konrad',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 '2008-12-25') ergibt '2008-12-25'
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() wird abgelehnt. Verwenden Sie stattdessen den SQL-Standard CURRENT_TIME.

Beispiele

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

```
SELECT RTRIM(' ' + Verkaeufers.Verkaeufersnr) AS agg FROM Verkaeufers
```

```
SELECT TRIM(SPACE(1) + Verkaeufers.Verkaeufersnr) AS agg FROM Verkaeufers
```

```
SELECT LTRIM(' ' + Verkaeufers.Verkaeufersnr) AS agg FROM Verkaeufers
```

```
SELECT UPPER(Verkaeufers.Verkaeufersnr) AS agg FROM Verkaeufers
```

```
SELECT LOWER(Verkaeufers.Verkaeufersnr) AS agg FROM Verkaeufers
```

```
SELECT LEFT(Verkaeufers.Verkaeufersnr, 5) AS agg FROM Verkaeufers
```

```
SELECT RIGHT(Verkaeufers.Verkaeufersnr, 7) AS agg FROM Verkaeufers
```

```
SELECT SUBSTR(Verkaeufers.Verkaeufersnr, 2, 2) +  
SUBSTR(Verkaeufers.Verkaeufersnr, 4, 2) AS agg FROM Verkaeufers
```

```
SELECT SUBSTR(Verkaeufers.Verkaeufersnr, 2) +  
SUBSTR(Verkaeufers.Verkaeufersnr, 4) AS agg FROM Verkaeufers
```

```
SELECT SPACE(2) + Verkaeufers.Verkaeufersnr AS Verkaeufersnr FROM  
Verkaeufers
```

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

Funktionen, die Zahlen zurückgeben

Funktionen, die Zahlen zurückgeben

Funktion	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 Tageteil eines Datums zurück	DAY (DATE '2016-01-30') ergibt 30
DAYOFWEEK	Gibt den Tag der Woche (1-7) eines Datumsausdrucks zurück	DAYOFWEEK (DATE '2004-05-01') ergibt 7

Funktionen, die Zahlen zurückgeben	Beschreibung	Beispiel
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 '2016-01-30') 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
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 '2016-01-30') ergibt 2016

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('2016-01-30 14:00:00') ergibt den Zeitstempelwert
DATE TODAY	Gibt das heutige Datum zurück	Wenn heute der 21.11.2016 ist, ergibt DATE() 2016-11-21
DATEVAL	Konvertiert eine Buchstabenzeichenfolge in ein Datum	DATEVAL('2016-01-30') ergibt 30.01.2016

Hinweis Die Funktion DATE() wird abgelehnt. Verwenden Sie stattdessen den SQL-Standard CURRENT_DATE.

Bedingte Funktion

Bedingte Funktion	Beschreibung	Beispiel
CASE WHEN	<p>Einfaches CASE-Format</p> <p>Vergleicht dem Wert von <i>eingabe_ausdr</i> mit den Werten der Argumente von <i>wert_ausdr</i>, um das Ergebnis zu bestimmen.</p> <pre>CASE <i>eingabe_ausdr</i> {WHEN <i>wert_ausdr</i> THEN <i>ergebnis...</i>} [ELSE <i>ergebnis</i>] END</pre>	<pre>SELECT Rechnungsnr, CASE Firmenname WHEN 'Exportiert GB' THEN 'Exportiert GB gefunden' WHEN 'Hausmöbellieferanten' THEN 'Hausmöbellieferanten gefunden' ELSE 'Exportiert weder GB noch Hausmöbellieferanten' END, Verkaufsnr FROM Vertriebsdaten</pre>
	<p>Gesuchtes CASE-Format</p> <p>Gibt ein Ergebnis basierend darauf zurück, ob die in einem WHEN-Ausdruck angegebene Bedingung wahr ist.</p> <pre>CASE {WHEN <i>boolescher_ausdr</i> THEN <i>ergebnis...</i>} [ELSE <i>ergebnis</i>] END</pre>	<pre>SELECT Rechnungsnr, Betrag, CASE WHEN Betrag > 3000 THEN 'Über 3000' WHEN Betrag < 1000 THEN 'Unter 1000' ELSE 'Zwischen 1000 und 3000' END, Verkaufsnr FROM Vertriebsdaten</pre>

Bedingte Funktion	Beschreibung	Beispiel
COALESCE	Gibt den ersten Wert zurück, der nicht NULL ist	<pre>SELECT Verkaufsnr, COALESCE(Vertriebsleiter, Verkaufsnr) FROM Verkäufer</pre>
NULLIF	Vergleicht zwei Werte und gibt NULL zurück, wenn die zwei Werte gleich sind; ansonsten gibt sie den ersten Wert zurück	<pre>SELECT Rechnungsnr, NULLIF(Betrag, -1), Verkaufsnr FROM Vertriebsdaten</pre>

FileMaker-Systemobjekte

FileMaker-Datenbankdateien umfassen die folgenden Systemobjekte, auf die Sie mittels SQL-Queries 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 Beziehungsdigramm eine Zeile mit folgenden Spalten:

- TableName - der Name des Tabellenauftritts.
- 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 gespeichert wurde.

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.

Die folgende Anweisung `CREATE TABLE` zeigt die Verwendung des Schlüsselworts `DEC` als Datenelementname.

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

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

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

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

Index

A

ABS, Funktion 32
ALL, Operator 27
ALTER TABLE (SQL-Anweisung) 22
ANY, Operator 27
ATAN, Funktion 32
ATAN2, Funktion 32
Ausdrücke in SQL 24
Ausschnitte 7

B

BaseFileName 35
BaseTableName 35
BETWEEN, Operator 27
Binärdaten, Verwendung in SELECT 15
BLOB-Datentyp, Verwendung in SELECT 15

C

CASE WHEN-Funktion 34
CAST-Funktion 16
CEIL, Funktion 32
CEILING, Funktion 32
CHR, Funktion 31
COALESCE-Funktion 35
Containerfeld
 extern gespeichert 21
 mit CREATE TABLE-Anweisung 21
 mit GetAs-Anweisung 16
 mit INSERT-Anweisung 18
 mit PutAs-Funktion 18
 mit SELECT-Anweisung 16
 mit UPDATE-Anweisung 20
CREATE INDEX (SQL-Anweisung) 23
CREATE TABLE (SQL-Anweisung) 20
CURDATE, Funktion 34
CURRENT USER, Funktion 31
CURRENT_DATE, Funktion 34
CURRENT_TIME, Funktion 34
CURRENT_TIMESTAMP, Funktion 34
CURRENT_USER, Funktion 31
Cursor in ODBC 14
CURTIME, Funktion 34
CURTIMESTAMP, Funktion 34

D

DATE, Funktion 34
Dateien, Verwendung in Containerfeldern 16
DATEVAL, Funktion 34
Datumsformate 24
Datumsoperatoren in SQL-Ausdrücken 26
DAY, Funktion 32

DAYNAME, Funktion 31
DAYOFWEEK, Funktion 32
DEFAULT (SQL-Klausel) 21
DEG, Funktion 32
DEGREES, Funktion 32
DELETE (SQL-Anweisung) 17
DISTINCT, Operator 8
Dokumentation 5
DROP INDEX (SQL-Anweisung) 23

E

EXISTS, Operator 27
EXP, Funktion 33
Exponentialschreibweise in SQL-Ausdrücken 25
EXTERNAL (SQL-Klausel) 21

F

Feldnamen in SQL-Ausdrücken 24
Feldwiederholungen 18, 20
FETCH FIRST (SQL-Klausel) 13
FieldClass 36
FieldId 36
FieldName 36
FieldReps 36
FieldType 36
FileMaker Server-Dokumentation 5
FileMaker_Fields 35
FileMaker_Tables 35
FLOOR, Funktion 33
FOR UPDATE (SQL-Klausel) 14
FROM (SQL-Klausel) 9
FULL OUTER JOIN 10
Funktion "LENGTH" 33
Funktion „SQLAusführen“ 6, 7
Funktionen in SQL-Ausdrücken 30

G

GetAs-Funktion 16
GROUP BY (SQL-Klausel) 11

H

HAVING (SQL-Klausel) 11
HOUR, Funktion 33

I

IN, Operator 27
INNER JOIN 10
INSERT (SQL-Anweisung) 17
INT, Funktion 33
IS NOT NULL, Operator 27
IS NULL, Operator 27

- J**
- JDBC-Client-Treiber
 - Ausschnitte 7
 - Unicode-Unterstützung 7
 - Join 10
- K**
- Konstanten in SQL-Ausdrücken 24
- L**
- Leere Werte in Spalten 18
 - Leere Zeichenfolge, Verwendung in SELECT 15
 - Leerzeichen 26
 - LEFT OUTER JOIN 10
 - LEFT, Funktion 31
 - LIKE, Operator 27
 - LN, Funktion 33
 - LOG, Funktion 33
 - Logische Operatoren in SQL-Ausdrücken 28
 - LOWER, Funktion 31
 - LTRIM, Funktion 31
- M**
- MAX, Funktion 33
 - MIN, Funktion 33
 - MINUTE, Funktion 33
 - MOD, Funktion 33
 - ModCount 35, 36
 - MONTH, Funktion 33
 - MONTHNAME, Funktion 31
- N**
- NICHT-Operator 28
 - NOT IN, Operator 27
 - NOT LIKE, Operator 27
 - NOT NULL (SQL-Klausel) 21
 - NULLIF-Funktion 35
 - Nullwert 18
 - Numerische Operatoren in SQL-Ausdrücken 25
 - NUMVAL, Funktion 33
- O**
- ODBC-Client-Treiber
 - Ausschnitte 7
 - Unicode-Unterstützung 7
 - ODBC-Standards, Einhaltung 7
 - ODER-Operator 28
 - OFFSET (SQL-Klausel) 13
 - Online-Dokumentation 5
 - Operatorpriorität bei SQL-Ausdrücken 29
 - ORDER BY (SQL-Klausel) 12
 - OUTER JOIN 10
- P**
- PDFs 5
 - Peer-Zeilen 13, 14
 - PI, Funktion 33
 - Positioned Updates und Deletes 14
 - PREVENT INDEX CREATION 23
 - PutAs-Funktion 18, 20
- R**
- RADIANS, Funktion 33
 - Relationale Operatoren in SQL-Ausdrücken 27
 - Reservierte SQL-Schlüsselwörter 37
 - RIGHT OUTER JOIN 10
 - RIGHT, Funktion 31
 - ROUND, Funktion 33
 - ROWID, Systemspalte 36
 - ROWMODID, Systemspalte 36
 - RTRIM, Funktion 31
- S**
- Schlüsselwörter, reservierte SQL- 37
 - SECOND, Funktion 33
 - SELECT (SQL-Anweisung) 8
 - Binärdaten 15
 - BLOB-Datentyp 15
 - leere Zeichenfolge 15
 - SIGN, Funktion 33
 - SIN, Funktion 33
 - Sortierfolge 12
 - SPACE, Funktion 31
 - Spaltenaliasse 8
 - SQL_C_WCHAR, Datentyp 7
 - SQL-92 7
 - SQL-Anweisungen
 - ALTER TABLE 22
 - CREATE INDEX 23
 - CREATE TABLE 20
 - DELETE 17
 - DROP INDEX 23
 - INSERT 17
 - Reservierte Schlüsselwörter 37
 - SELECT 8
 - TRUNCATE TABLE 22
 - Unterstützung durch Client-Treiber 7
 - UPDATE 19
 - SQL-Ausdrücke 24
 - Datumsoperatoren 26
 - Exponential- bzw. wissenschaftliche Schreibweise 25
 - Feldnamen 24
 - Funktionen 30
 - Konstanten 24
 - Logische Operatoren 28
 - Numerische Operatoren 25
 - Operatorpriorität 29
 - Relationale Operatoren 27
 - Zeichenoperatoren 25

SQL-Standards, Einhaltung 7
SQL-Statistikfunktionen 30
SQRT, Funktion 33
Standards, Einhaltung 7
Statistikfunktionen in SQL 30
STRVAL, Funktion 31
SUBSTR, Funktion 31
SUBSTRING, Funktion 31
Syntaxfehler 37
Systemtabellen 35

T

Tabellenaliasse 8, 9
TableId 35
TableName 35, 36
TAN, Funktion 33
TIME, Funktion 31
TIMESTAMPVAL, Funktion 34
TIMEVAL, Funktion 31
TODAY, Funktion 34
TRIM, Funktion 31
TRUNCATE TABLE (SQL-Anweisung) 22

U

UND-Operator 28
Unicode-Unterstützung 7
UNION (SQL-Operator) 12
UNIQUE (SQL-Klausel) 21
Unterabfragen 18
UPDATE (SQL-Anweisung) 19
UPPER, Funktion 31
USERNAME, Funktion 31

V

VALUES (SQL-Klausel) 18

W

WHERE (SQL-Klausel) 10
WITH TIES (SQL-Klausel) 13

Y

YEAR, Funktion 33

Z

Zeichenfolge, Funktionen 31
Zeichenoperatoren in SQL-Ausdrücken 25
Zeitformate 24
Zeitstempelformate 24