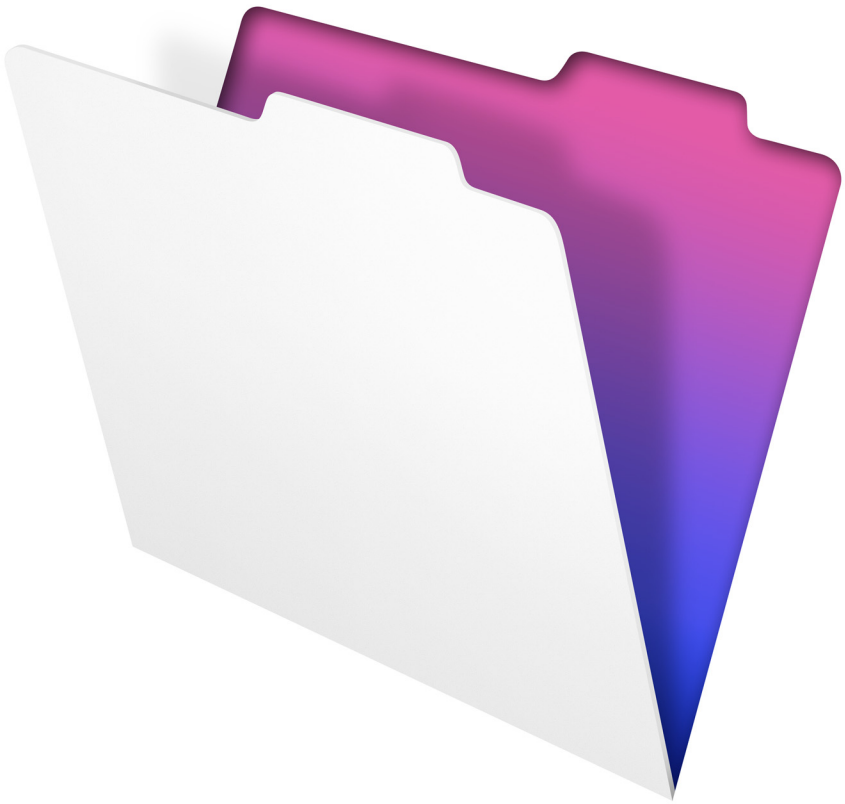
FileMaker® 13

SQL-referens



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

Mer information finns på webbplatsen http://www.filemaker.com/se.

Utgåva: 01

Innehåll

Kapitel 1
Introduktion 4

Om denna referens 4
Hitta PDF-dokumentation 4
Om SQL 4
Använda en FileMaker-databas som en datakälla 5
Använda ExecuteSQL-funktionen 5

Kapitel 2
Standarder som stöds 6

Stöd för Unicode-tecken 6
SQL-satser 6
SELECT-sats 7
SQL-satser 7
FROM-sats 8
WHERE-sats 9
GROUP BY-sats 9
HAVING-sats 9
UNION-operator 10
ORDER BY-sats 10
OFFSET- och FETCH FIRST-satser 11
FOR UPDATE-sats 12
DELETE-sats 15
INSERT-sats 15
UPDATE-sats 17
CREATE TABLE-sats 18
ALTER TABLE-sats 19
CREATE INDEX-sats 20
DROP INDEX-sats 20
SQL-satser 21
Fältnamn 21
Konstanter 21
Exponentiell/matematisk notation 22
Numeriska operatorer 22
Teckenoperatorer 23
Datumoperatorer 23
Relationsoperatorer 23
Logiska operatorer 25
Prioritetsordning för operatorer 25
SQL-funktioner 26
Statistikfunktioner 26
Funktioner som returnerar teckensträngar 27
Funktioner som returnerar siffror 28
Funktioner som returnerar datum 30
Villkorsfunktioner 30
Reserverade SQL-nyckelord 32

Index 35

Kapitel 1
Introduktion

Som databasutvecklare kan du använda FileMaker Pro för att skapa databaslösningar utan att kunna något om SQL. Men om du har viss kunskap om SQL kan du använda en FileMaker-databasfil som en ODBC- eller JDBC-datakälla och dela dina data med andra program via ODBC och JDBC. Du kan också använda ExecuteSQL-funktionen i FileMaker Pro för att hämta data från alla tabellförekomster i en FileMaker Pro-databas.

Denna referens beskriver vilka SQL-satser och standarder som stöds av FileMaker. FileMakers ODBC- och JDBC-klientdrivrutiner stöder alla SQL-uttryck som beskrivs i denna referens. ExecuteSQL-funktionen i FileMaker Pro stöder endast SELECT-satsen.

Om denna referens

- Information om hur du använder ODBC och JDBC med tidigare versioner av FileMaker Pro kan du hämta från http://www.filemaker.com/se/support/.
- Den här referensen förutsätter att du känner till grunderna i hur du använder FileMaker Pro-funktionerna, kodar ODBC- och JDBC-program och skapar SQL-frågor. Mer information om dessa ämnen finns i böcker från olika företag.
- I denna referens avser termen ”FileMaker Pro” både FileMaker Pro och FileMaker Pro Advanced, utom när specifika funktioner i FileMaker Pro Advanced beskrivs.

Hitta PDF-dokumentation

Hitta PDF-filer för FileMaker-dokumentation:

- Välj **Hjälp**-menyn > **Produktdokumentation** i FileMaker Pro.
- Välj **Hjälp**-menyn > **Produktdokumentation** i FileMaker Server.
- Ytterligare dokumentation hittar du på http://www.filemaker.com/se/support/. Uppdateringar till det här dokumentet kan också hämtas från den webbplatsen.

Om SQL

SQL, eller Structured Query Language, är ett programmeringsspråk som utformats för att ställa frågor till data från en relationsdatabas. Den primära sats som används för att ställa en fråga till en databas är SELECT-satsen.

Utöver språk för att ställa en fråga till en databas, innehåller SQL satser för att utföra datamanipulering som gör att du kan lägga till, uppdatera och ta bort data.

SQL innehåller även satser för att utföra datadefinitioner. Med dessa satser kan du skapa och ändra tabeller och index.

SQL-satser och -standarder som stöds av FileMaker beskrivs i kapitel 2, ”Standarder som stöds”.

Använda en FileMaker-databas som en datakälla

När du är värd för en FileMaker-databas som en ODBC- eller JDBC-datakälla kan FileMaker-data delas med ODBC- eller JDBC-kompatibla program. Programmen ansluter till FileMaker-datakällan via FileMaker-klientdrivrutiner, skapar och kör SQL-frågorna med ODBC eller JDBC, och bearbetar de data som hämtas från FileMaker-databaslösningen.

I *Handboken för FileMaker ODBC och JDBC* finns utförlig information om hur du kan använda FileMaker-programvara som en datakälla för ODBC- och JDBC-program.

FileMakers ODBC- och JDBC-klientdrivrutiner stöder alla SQL-satser som beskrivs i denna referens.

Använda ExecuteSQL-funktionen

Med ExecuteSQL-funktionen i FileMaker Pro kan du hämta data från tabellförekomster som namnges i relationsdiagrammet, men som är oberoende av några definierade relationer. Du kan hämta data från flera olika tabeller utan att skapa tabellkopplingar eller några relationer mellan tabellerna. I vissa fall kanske du kan minska relationsdiagrammens komplexitet genom att använda ExecuteSQL-funktionen.

Fälten som du ställer frågor till med ExecuteSQL-funktionen behöver inte vara i någon layout, så du kan använda ExecuteSQL-funktionen för att hämta data oberoende av layoutkontext. På grund av detta kontextoberoende kan manusens enkelhet förbättras när du använder ExecuteSQL-funktionen i manus. Du kan använda ExecuteSQL-funktionen överallt där du kan ange beräkningar, bland annat i diagram och rapporter.

ExecuteSQL-funktionen stöder endast SELECT-satsen, enligt beskrivningen i ”SELECT-sats” på sidan 7.

ExecuteSQL-funktionen godkänner även endast SQL-92 syntaxens ISO-datum- och tidsformat utan klammerparenteser ({}). ExecuteSQL-funktionen godkänner inte datum-, tids- och tidsstämpelkonstanter i ODBC/JDBC-format inom klammerparenteser

Information om syntax och användning av ExecuteSQL-funktionen finns i FileMaker Pro Hjälp.

Kapitel 2
Standarder som stöds

Denna referens beskriver vilka SQL-uttryck och ord som stöds av FileMaker. FileMakers ODBC- och JDBC-klientdrivrutiner stöder alla SQL-uttryck som beskrivs i detta kapitel. ExecuteSQL-funktionen i FileMaker Pro stöder endast SELECT-satsen.

Använd klientdrivrutinerna när du vill få tillgång till databaslösningar för FileMaker från ett program som är kompatibelt med ODBC eller JDBC. Databaslösningar för FileMaker kan ha antingen FileMaker Pro eller FileMaker Server som värd.

- ODBC-klientdrivrutinen stöder ODBC 3.5 nivå 1 och vissa funktioner i nivå 2.
- JDBC-klientdrivrutinen ger delvis stöd för specifikationen JDBC 3.0.
- ODBC- och JDBC-klientdrivrutinerna stöder båda grunderna i SQL-92 och vissa högre funktioner i SQL-92.

Stöd för Unicode-tecken

ODBC- och JDBC-klientdrivrutinerna stöder Unicode API. Om du däremot skapar ett eget program som använder klientdrivrutinerna, bör du använda ASCII för fältnamn, tabellnamn och filnamn (när något annat än ett Unicode-frågeverktyg eller -program används).

Obs! Använd `SQL_C_WCHAR` om du vill infoga och hämta Unicode-data.

SQL-satser

ODBC- och JDBC-klientdrivrutinerna har stöd för följande SQL-satser:

- SELECT (sidan 7)
- DELETE (sidan 15)
- INSERT (sidan 15)
- UPDATE (sidan 17)
- CREATE TABLE (sidan 18)
- ALTER TABLE (sidan 19)
- CREATE INDEX (sidan 20)
- DROP INDEX (sidan 20)

Klientdrivrutinerna stöder även FileMaker-datatypsmappning till datatyperna i ODBC SQL och JDBC SQL. Se *Handboken för FileMaker ODBC och JDBC* om datatypskonverteringar. Vidare information om hur du skapar SQL-frågor finns i allmänna böcker om SQL.

Obs! ODBC- och JDBC-klientdrivrutinerna stöder inte FileMaker-portaler.

SELECT-sats

Använd en `SELECT`-sats för att ange vilka kolumner du efterfrågar. Slutför `SELECT`-satsen med de kolumnuttryck (samma som fältnamnen) som du vill hämta (t.ex. `efternamn`). Uttrycken kan innehålla matematiska operationer eller strängmanipulationer (t.ex. `LÖN * 1.05`).

`SELECT`-satsen kan ha flera olika instruktioner:

```
SELECT [DISTINCT] {* | kolumnuttryck [[AS] kolumnalias],...}
FROM tabellnamn [tabellalias], ...
[ WHERE uttr1 rel_operator uttr2 ]
[ GROUP BY {kolumnuttryck, ...} ]
[ HAVING uttr1 rel_operator uttr2 ]
[ UNION [ALL] (SELECT...) ]
[ ORDER BY sorteringsuttryck [DESC | ASC], ... ]
[ OFFSET n {ROWS | ROW} ]
[ FETCH FIRST [ n [ PERCENT ] ] { ROWS | ROW } {ONLY | WITH TIES } ]
[ FOR UPDATE [OF {kolumnuttryck, ...}] ]
```

Poster inom hakparenteser är valfria.

Du kan använda `kolumnalias` för att ge kolumnen ett mer beskrivande namn eller för att förkorta ett långt kolumnnamn. Så här kan du t.ex. tilldela aliaset `avdelning` till kolumnen `avd`:

```
SELECT avd AS avdelning FROM anst
```

Fältnamn kan föregås av tabellnamnet eller tabellalias. Exempel: `ANST.EFTERNAMN` eller `A.EFTERNAMN`, där `A` är alias för tabellen `ANST`.

Operatorn `DISTINCT` kan föregå det första kolumnuttrycket. Denna operator eliminerar dubblerade rader från ett frågeresultat. Till exempel:

```
SELECT DISTINCT avd FROM anst
```

SQL-satser

ODBC- och JDBC-klientdrivrutinerna har stöd för följande SQL-satser:

Använd SQL-satsen	För att
FROM (sidan 8)	Ange vilka tabeller som används i `SELECT`-satsen.
WHERE (sidan 9)	Ange de villkor som posterna måste uppfylla för att hämtas (som en FileMaker Pro-sökpost).
GROUP BY (sidan 9)	Ange namnen på ett eller flera fält som de returnerade värdena ska grupperas efter. Denna instruktion används för att returnera en uppsättning statistikvärden genom att returnera en rad för varje grupp (som en delsumma i FileMaker Pro).
HAVING (sidan 9)	Ange villkor för grupper av poster (t.ex. endast visa de avdelningar som har löner som uppgår till mer än 200 000).
UNION (sidan 10)	Kombinera resultaten av två eller flera `SELECT`-satser till ett enda resultat.
ORDER BY (sidan 10)	Ange hur posterna är sorterade.
OFFSET (sidan 11)	Ange antalet rader som ska hoppas över innan rader börjar att hämtas.
FETCH FIRST (sidan 11)	Ange antalet rader som ska hämtas. Högst angivet antal rader returneras trots att färre rader kan returneras om frågan resulterar i mindre än det antal rader som anges.
FOR UPDATE (sidan 12)	Genomför positionsuppdateringar eller -raderingar via SQL-markörer.

Obs! Om du försöker hämta data från en tabell utan kolumner returnerar `SELECT`-satsen ingenting.

FROM-sats

FROM-satsen visar att tabellerna ska användas i SELECT-satsen. Formatet är:

```
FROM tabellnamn [tabellalias] [, tabellnamn [tabellalias]]
```

tabellnamn är namnet på en tabell i den aktuella databasen. Tabellnamnet måste börja med ett alfabetiskt tecken. Om tabellnamnet börjar med något annat än ett alfabetiskt tecken ska du omge det med dubbla citattecken (citatomsluten identifierare)

tabellalias kan användas för att ge tabellen ett mer beskrivande namn, förkorta ett längre tabellnamn eller infoga samma tabell i frågan fler än en gång (till exempel i självkopplingar).

Fältnamn börjar med ett alfabetiskt tecken. Om fältnamnet börjar med något annat än ett alfabetiskt tecken ska du omge det med dubbla citattecken (citatomsluten identifierare) Exempel: ExecuteSQL-satsen för fältet som heter _LASTNAME är:

```
SELECT "_LASTNAME" från anst
```

Fältnamn kan föregås av tabellnamnet eller tabellalias. Med specifikationen FROM anställd A kan du t.ex. hänvisa till fältet EFTERNAMN som A.EFTERNAMN. Tabellalias måste användas om SELECT-satsen kopplar en tabell till sig själv. Till exempel:

```
SELECT * FROM anställd A, anställd F WHERE A.chefs_id =
F.anställnings_id
```

Likhetstecknet (=) tar bara med matchande rader i resultatet.

Om du kopplar mer än en tabell och du vill radera alla rader som inte har motsvarande rader i båda källtabellerna, kan du använda INNER JOIN. Till exempel:

```
SELECT *
   FROM Säljare INNER JOIN Säljdata
   ON Säljare.Försäljar_ID = Säljdata.Försäljar_ID
```

Om du kopplar två tabeller men inte vill radera raderna i den första tabellen (den vänstra tabellen) kan du använda LEFT OUTER JOIN.

```
SELECT *
   FROM Säljare LEFT OUTER JOIN Säljdata
   ON Säljare.Försäljar_ID = Säljdata.Försäljar_ID
```

Alla rader från tabellen ”Säljare” visas i den kopplade tabellen.

Kommentar

- RIGHT OUTER JOIN stöds för närvarande inte.
- FULL OUTER JOIN stöds för närvarande inte.

WHERE-sats

WHERE-satsen anger vilka villkor som poster måste uppfylla för att kunna hämtas. WHERE-satsen innehåller villkor i formatet:

```
WHERE uttr1 rel_operator uttr2
```

uttr1 och uttr2 kan vara fältnamn, konstantvärden eller uttryck.

rel_operator är den relationsoperator som kopplar ihop de två uttrycken. Följande SELECT-sats hämtar t.ex. namnen på de anställda som tjänar minst 200 000 kronor.

```
SELECT efternamn,förnamn FROM anst WHERE lön >= 20000
```

WHERE-satsen kan också använda uttryck som:

```
WHERE uttr1 IS NULL
WHERE NOT uttr2
```

Obs! Om du använder fullständiga namn i SELECT-listan (projektionslistan) måste du även använda fullständiga namn i den tillhörande WHERE-satsen.

GROUP BY-sats

GROUP BY-satsen anger namnet på ett eller flera fält som de returnerade värdena ska grupperas efter. Du använder denna instruktion för att returnera en uppsättning statistikvärden. Instruktionen har följande format:

```
GROUP BY kolumner
```

kolumner måste matcha det kolumnuttryck som används i SELECT-satsen. Ett kolumnuttryck kan vara ett eller flera fältnamn i databastabellen åtskilda med kommatecken.

Exempel

Följande exempel summerar lönerna på varje avdelning.

```
SELECT avd_id, SUM(lön) FROM anst GROUP BY avd_id
```

Den här satsen returnerar en rad för varje separat avdelnings-ID. Varje rad innehåller avdelnings-ID:t och summan av lönerna för avdelningens anställda.

HAVING-sats

Med HAVING-satsen kan du ange villkor för grupper av poster (t.ex. endast visa de avdelningar som har löner som uppgår till mer än 200 000). Instruktionen har följande format:

```
HAVING uttr1 rel_operator uttr2
```

uttr1 och uttr2 kan vara fältnamn, konstantvärden eller uttryck. Dessa uttryck måste inte matcha ett kolumnuttryck i SELECT-satsen.

rel_operator är den relationsoperator som kopplar ihop de två uttrycken.

Exempel

Följande exempel returnerar endast de avdelningar vilkas lönesummor är större än 200 000:

```
SELECT avd_id, SUM (lön) FROM anst
   GROUP BY avd_id HAVING SUM (lön) > 200000
```

UNION-operator

Operatorn `UNION` kombinerar resultatet av två eller flera `SELECT`-satser till ett enda resultat. Resultatet är alla de returnerade posterna från `SELECT`-satserna. Som standard returneras inte dubblerade poster. Om du vill returnera dubbla poster använder du nyckelordet `ALL` (`UNION ALL`). Formatet är:

```
SELECT-sats UNION [ALL] SELECT-sats
```

När du använder `UNION`-operatorn måste urvalslistorna för varje `SELECT`-sats ha samma antal kolumnuttryck, med samma datatyper och anges i samma ordningsföljd. Till exempel:

```
SELECT efternamn, lön, anst_datum FROM anst UNION SELECT namn, lön,
födelsedatum FROM person
```

Detta exempel har samma antal kolumnuttryck och varje kolumnuttryck, i ordningsföljd, har samma datatyp.

Följande exempel är inte giltigt eftersom datatyperna för kolumnuttrycken är olika (`LÖN` från `ANST` har en annan datatyp än `EFTERNAMN` från `LÖNEFÖRHÖJNING`). Detta exempel har samma antal kolumnuttryck i varje `SELECT`-sats, men satserna är inte i samma ordningsföljd som datatypen.

```
SELECT efternamn, lön FROM anst UNION SELECT lön, efternamn FROM
löneförhöjning
```

ORDER BY-sats

`ORDER BY`-satsen anger hur posterna ska sorteras. Formatet är:

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

`sorteringsuttryck` kan vara fältnamn, uttryck eller positionsnummer för det kolumnuttryck som ska användas. Som standard sker sorteringen i stigande ordning (`ASC`).

Om du t.ex. vill sortera efter `efternamn` och sedan efter `förnamn` använder du en av följande `SELECT`-satser:

```
SELECT anst_id, efternamn, förnamn FROM anst ORDER BY efternamn,
förnamn
```

eller

```
SELECT anst_id, efternamn, förnamn FROM anst ORDER BY 2,3
```

I det andra exemplet är `efternamn` det andra kolumnuttrycket som följer efter `SELECT`, så `ORDER BY 2` sorterar efter `efternamn`.

OFFSET- och FETCH FIRST-satser

OFFSET- och FETCH FIRST-satser används för att returnera ett angivet intervall av rader som börjar vid en särskild startpunkt i en resultatuppsättning. Möjligheten att begränsa de rader som hämtas från stora resultatuppsättningar gör att du kan ”bläddra sida” genom data och förbättrar effektiviteten.

OFFSET-satsen indikerar antalet rader som ska hoppas över innan data börjar att returneras. Om OFFSET-satsen inte används i en SELECT-sats är startraden 0. FETCH FIRST-satsen anger antalet rader som ska returneras, antingen som ett positivt heltal som är större än eller lika med 1 eller som en procentsats från den startpunkt som indikeras i OFFSET-satsen. Om både OFFSET och FETCH FIRST används i en SELECT-sats ska OFFSET-satsen komma först.

OFFSET- och FETCH FIRST-satser stöds inte i delfrågor.

OFFSET-format

OFFSET-formatet är:

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

n är ett positivt heltal. Om n är större än antalet rader som returneras i resultatuppsättningen returneras ingenting och inget felmeddelande visas.

ROWS är detsamma som ROW.

FETCH FIRST-format

FETCH FIRST-formatet är:

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

n är antalet rader som ska returneras. Standardvärdet är 1 om n utesluts, n är ett positivt heltal som är större än eller lika med 1 om det inte följs av PERCENT. Om n följs av PERCENT, kan värdet vara antingen ett positivt bråkdelsvärde eller ett positivt heltal.

ROWS är detsamma som ROW.

WITH TIES måste användas med ORDER BY-satsen.

WITH TIES gör att fler rader kan returneras än det värde som anges i FETCH eftersom peer-rader, de rader som inte är separata utifrån ORDER BY-satsen, också returneras.

Exempel

Exempel: För att returnera information från den tjugosjätte raden i resultatuppsättningen som sorterats efter efternamn och sedan efter förnamn, kan du använda följande SELECT-sats:

```
SELECT anst_id, efternamn, förnamn FROM anst ORDER BY efternamn,
förnamn OFFSET 25 ROWS
```

Så här anger du att du bara vill returnera tio rader:

```
SELECT anst_id, efternamn, förnamn FROM anst ORDER BY efternamn,
förnamn OFFSET 25 ROWS FETCH FIRST 10 ROWS ONLY
```

Så här returnerar du tio rader och deras peer-rader (rader som inte är separata utifrån `ORDER BY`-satsen):

```
SELECT anst_id, efternamn, förnamn FROM anst ORDER BY efternamn,
förnamn OFFSET 25 ROWS FETCH FIRST 10 ROWS WITH TIES
```

FOR UPDATE-sats

Instruktionen `FOR UPDATE` låser poster för positionsuppdateringar eller -raderingar via SQL-markörer. Formatet är:

```
FOR UPDATE [OF kolumnuttryck]
```

`kolumnuttryck` är en lista över de fältnamn i databastabellen som du vill uppdatera, avgränsade av kommatecken. `kolumnuttryck` är valfritt och ignoreras.

Exempel

Följande exempel returnerar alla poster i databasen över anställda som har ett värde i fältet `LÖN` som är högre än 20 000. När varje post hämtas är den låst. Om posten uppdateras eller raderas är posten låst tills du har gjort ändringen. I annat fall låses den upp när du hämtar nästa post.

```
SELECT * FROM anst WHERE lön > 20000
   FOR UPDATE OF efternamn, förnamn, lön
```

Fler exempel:

Med	SQL-kod
textkonstant	`SELECT 'KattHund' FROM Säljare`
numerisk konstant	`SELECT 999 FROM Säljare`
datumkonstant	`SELECT DATE '2012-06-05' FROM Säljare`
tidskonstant	`SELECT TIME '02:49:03' FROM Säljare`
tidsstämpelkonstant	`SELECT TIMESTAMP '2012-06-05 02:49:03' FROM Säljare`
textkolumn	`SELECT Företagsnamn FROM Säljdata` `SELECT DISTINCT Företagsnamn FROM Säljdata`
numerisk kolumn	`SELECT Belopp FROM Säljdata` `SELECT DISTINCT Belopp FROM Säljdata`
datumkolumn	`SELECT Försäljningsdatum FROM Säljdata` `SELECT DISTINCT Försäljningsdatum FROM Säljdata`
tidskolumn	`SELECT Försäljningstid FROM Säljdata` `SELECT DISTINCT Försäljningstid FROM Säljdata`
tidsstämpelkolumn	`SELECT Tidsstämpel_för_försäljning FROM Säljdata` `SELECT DISTINCT Tidsstämpel_för_försäljning FROM Säljdata`
BLOB[a]-kolumn	`SELECT Företagsbroschyrer FROM Säljdata` `SELECT GETAS(Företagslogotyp, 'JPEG') FROM Säljdata`
Jokertecken *	`SELECT * FROM Säljare` `SELECT DISTINCT * FROM Säljare`

a. En BLOB är ett containerfält i en FileMaker-databasfil.

Information om exemplen

En `kolumn` är en referens till ett fält i FileMaker-databasfilen. (Fältet kan innehålla många separata värden.)

Jokertecknet asterisk (*) är ett sätt att ange ”allt”. I exemplet `SELECT * FROM Säljare` är resultatet alla kolumner i tabellen `Säljare`. I exemplet `SELECT DISTINCT * FROM Säljare` är resultatet alla unika rader i tabellen `Säljare` (inga dubbletter).

- FileMaker lagrar inte data för tomma strängar, så följande frågor returnerar inga poster:

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

- Om du använder `SELECT` med binära data, måste du ange funktionen `GetAs()` för att ange vilken ström som ska returneras. Se nästa avsnitt, ”Hämta innehållet i ett containerfält: Funktionen CAST() och funktionen GetAs()”, för vidare information.

Hämta innehållet i ett containerfält: Funktionen CAST() och funktionen GetAs()

Du kan hämta binära data, filreferensinformation eller data med en specifik filtyp från ett containerfält.

Om det finns fildata eller binära data i JPEG hämtar `SELECT`-satsen med `GetAS(fältnamn, 'JPEG')` data i binär form. Annars returnerar `SELECT`-satsen med fältnamnet `NULL`.

Om du vill hämta filreferensinformation från ett containerfält, som sökvägen till en fil, bild eller QuickTime-film, använder du funktionen `CAST()` tillsammans med en `SELECT`-sats. Till exempel:

```
SELECT CAST(Företagsbroschyrer AS VARCHAR(NNN)) FROM Säljdata
```

Om du gjorde följande i det här exemplet:

- Infogade en fil i containerfältet med FileMaker Pro men bara lagrade en referens till filen. `SELECT`-satsen hämtar då filreferensinformationen som typen `SQL_VARCHAR`.
- Infogade innehållet i en fil i containerfältet med FileMaker Pro. Då hämtar `SELECT`-satsen namnet på filen.
- Importerade en fil i containerfältet från något annat program. Då visar `SELECT`-uttrycket ”?” (filen visas som **Namnlös.dat** i FileMaker Pro).

Om du vill hämta data från ett containerfält, kan du använda `GetAs()`-funktionen. Du kan använda `DEFAULT`-tillvalet eller ange filtypen. `DEFAULT`-tillvalet hämtar containerns originaldirektuppspelning utan att explicit behöva definiera uppspelningstyp:

```
SELECT GetAs(Företagsbroschyrer, DEFAULT) FROM  Säljdata
```

Om du vill hämta en enskild uppspelningstyp från en container använder du funktionen `GetAs()` tillsammans med filens typ utifrån hur data sattes in i containerfältet i FileMaker Pro. Till exempel:

- Om data sattes in med hjälp av kommandot **Sätt in** > **Fil** anger du `'FIL'` i funktionen `GetAs()`. Till exempel:

  ```
  SELECT GetAs(Företagsbroschyrer, 'FIL') FROM Säljdata
  ```

- Om data sattes in med hjälp av kommandot **Sätt in** > **Ljud** (Standardljud – MAC OS X raw-format) anger du `'snd'` i funktionen `GetAs()`. Till exempel:

  ```
  SELECT GetAs(Företags_möte, 'snd') FROM Företagets_Nyhetsbrev
  ```

- Om data sattes in med hjälp av kommandot **Sätt in** > **Bild**, dra och släpp eller om det klistrades in från urklipp, anger du någon av filtyperna som visas i listan i följande tabell. Till exempel:

```
SELECT GetAs(Företags_logotyp, 'JPEG') FROM Företag
```

Filformat	Beskrivning	Filformat	Beskrivning
'GIFf'	Graphics Interchange Format	'PNTG'	MacPaint
'JPEG'	Fotografiska bilder	'.SGI'	Allmänt bitmappsformat
'JP2 '	JPEG 2000	'TIFF'	Rasterfilformat för digitala bilder
'PDF '	Portable Document Format	'TPIC'	Targa
'PNGf'	Bitmappsbildformat	'8BPS'	PhotoShop (PSD)

DELETE-sats

Använd `DELETE`-satsen när du vill ta bort poster från en databastabell. `DELETE`-satsen har följande format:

```
DELETE FROM tabellnamn [ WHERE { villkor } ]
```

Obs! `WHERE`-satsen avgör vilka poster som ska raderas. Om du inte inkluderar nyckelordet `WHERE` raderas alla poster i tabellen (men själva tabellen lämnas intakt).

Exempel

Ett exempel på en `DELETE`-sats för tabellen Anställda:

```
DELETE FROM anst WHERE anst_id = 'E10001'
```

`DELETE`-satsen tar bort alla poster som uppfyller villkoren i instruktionen `WHERE`. I det här fallet raderas alla poster som har anställnings-ID `E10001`. Eftersom varje anställnings-ID är unikt i tabellen Anställda, raderas endast en post.

INSERT-sats

Använd `INSERT`-satsen om du vill skapa poster i en databastabell. Du kan ange något av följande:

- En lista över värden som ska infogas som en ny post
- En `SELECT`-sats som kopierar data från en annan tabell som ska infogas som en uppsättning nya poster

`INSERT`-satsen har följande format:

```
INSERT INTO tabellnamn [(kolumnnamn, ...)] VALUES (uttr, ...)
```

`kolumnnamn` är en valfri lista över kolumnnamn som ger tillgång till namnet och ordningsföljden för de kolumner vilkas värde anges i instruktionen `VALUES`. Om du utelämnar `kolumnnamn` måste värdeuttrycken (`uttr`) ge värden för alla kolumner som är definierade i tabellen. De måste också komma i samma ordningsföljd som kolumnerna definierades i tabellen. `kolumnnamn` kan också ange en fältrepetition, t.ex. `lastDates[4]`.

uttr är den lista över uttryck som ger värdena för den nya postens kolumner. Normalt är uttrycken konstanta värden för kolumnerna (men de kan också vara en delfråga). Värden för teckensträngar måste omslutas av enkla citationstecken ('). Om du vill ta med ett enkelt citationstecken i ett värde för en teckensträng som omsluts av enkla citationstecken, använder du två enkla citationstecken tillsammans (t.ex. 'Don''t').

Delfrågor måste omges av parenteser.

I följande exempel infogas en lista över uttryck:

```
INSERT INTO anst (efternamn, förnamn, anst_id, lön, anst_datum)
   VALUES ('Andersson', 'Anders', 'E22345', 27500, DATE '2013-06-05')
```

Varje INSERT-sats lägger till en post i databastabellen. I det här fallet har en post lagts till i databastabellen över anställda, ANST. Värden har angetts för fem kolumner. De återstående kolumnerna i tabellen tilldelas ett tomt värde, dvs. Null.

Obs! I containerfält kan du bara använda INSERT med text, såvida du inte förbereder ett parameteruttryck och strömmar data från programmet. Om du vill använda binära data kan du helt enkelt tilldela filnamnet genom att omge det med enkla citationstecken eller använda funktionen PutAs(). När du anger filnamnet härleds filtypen från filtillägget:

```
INSERT INTO tabellnamn (containernamn) VALUES(? AS
'filnamn.filtillägg')
```

Filtyper som inte stöds kommer att sättas in som typen FIL.

Ange typen när du använder funktionen PutAs(): PutAs(kol, 'typ'), där typvärdet är en filtyp som stöds, enligt beskrivningen i ”Hämta innehållet i ett containerfält: Funktionen CAST() och funktionen GetAs()” på sidan 14.

SELECT-satsen är en fråga som returnerar värden för varje kolumnnamn-värde som anges i listan över kolumnnamn. Att använda en SELECT-sats i stället för en lista över värdeuttryck medför att du kan välja en uppsättning rader från en tabell och infoga den i en annan tabell med en enda INSERT-sats.

Här är ett exempel på en INSERT-sats som använder sig av en SELECT-sats:

```
INSERT INTO anst1 (förnamn, efternamn, anst_id, avd, lön)
   SELECT förnamn, efternamn, anst_id, avd, lön FROM anst
   WHERE avd = 'D050'
```

I den här typen av INSERT-satser måste antalet kolumner som ska infogas matcha antalet kolumner i SELECT-satsen. Listan över kolumner som ska infogas måste motsvara kolumnerna i SELECT-satsen på samma sätt som den måste motsvara en lista över värdeuttryck i den andra typen av INSERT-satser. Exempel: Den första infogade kolumnen motsvarar den första valda kolumnen; den andra infogade kolumnen motsvarar den andra valda, osv.

Storleken och datatypen för dessa motsvarande kolumner måste överensstämma. Varje kolumn i listan SELECT bör ha en datatyp som ODBC- eller JDBC-drivrutinen accepterar vid en vanlig INSERT/UPDATE av den motsvarande kolumnen i listan INSERT. Värdena trunkeras när storleken på värdet i kolumnen i listan SELECT är större än storleken på den motsvarande kolumnen i listan INSERT.

SELECT-satsen beräknas innan några värden infogas.

UPDATE-sats

Använd `UPDATE`-satsen om du vill ändra poster i en databastabell. `UPDATE`-satsen har följande format:

```
UPDATE tabellnamn SET kolumnnamn = uttr, ... [ WHERE {villkor } ]
```

`kolumnnamn` är namnet på en kolumn vars värde ska ändras. Det går att ändra flera kolumner i ett och samma uttryck.

`uttr` är kolumnens nya värde.

Normalt är uttrycken konstanta värden för kolumnerna (men de kan också vara en delfråga). Värden för teckensträngar måste omslutas av enkla citationstecken ('). Om du vill ta med ett enkelt citationstecken i ett värde för en teckensträng som omsluts av enkla citationstecken, använder du två enkla citationstecken tillsammans (t.ex. `'Don''t'`).

Delfrågor måste omges av parenteser.

Instruktionen `WHERE` är en valfri, giltig instruktion. Den bestämmer vilka poster som uppdateras.

Exempel

Ett exempel på en `UPDATE`-sats för tabellen Anställda:

```
UPDATE anst SET lön=32000, undantag=1 WHERE anst_id = 'E10001'
```

Uttrycket `UPDATE` ändrar alla poster som uppfyller villkoren i instruktionen `WHERE`. I det här fallet ändras lönen och undantagsstatus för alla anställda som har anställnings-ID `E10001`. Eftersom varje anställnings-ID är unikt i tabellen Anställda, uppdateras endast en post.

Här följer ett exempel där en delfråga används:

```
UPDATE anst SET lön = (SELECT medel(lön) FROM anst) WHERE anst_id =
'E10001'
```

I det här fallet ändras lönen till medellönen i företaget för de anställda som har anställnings-ID `E10001`.

Obs! I containerfält kan du bara använda `UPDATE` med text, såvida du inte förbereder ett parameteruttryck och strömmar data från programmet. Om du vill använda binära data kan du helt enkelt tilldela filnamnet genom att omge det med enkla citationstecken eller använda funktionen `PutAs()`. När du anger filnamnet härleds filtypen från filtillägget:

```
UPDATE tabellnamn SET (containernamn) = ? AS 'filnamn.filtillägg'
```

Filtyper som inte stöds kommer att sättas in som typen FIL.

Ange typen när du använder funktionen `PutAs()`: `PutAs(kol, 'typ')`, där typvärdet är en filtyp som stöds, enligt beskrivningen i ”Hämta innehållet i ett containerfält: Funktionen CAST() och funktionen GetAs()” på sidan 14.

CREATE TABLE-sats

Använd en `CREATE TABLE`-sats om du vill skapa en tabell i en databasfil. `CREATE TABLE`-satsen har följande format:

```
CREATE TABLE tabellnamn ( tabellelementlista [,
tabellelementlista...] )
```

I instruktionen anger du namnet och datatypen för varje kolumn.

- `tabellnamn` är namnet på tabellen. `tabellnamn` har en begränsning på 100 tecken. Det får inte redan finnas en tabell med samma namn. Tabellnamnet måste börja med ett alfabetiskt tecken. Om tabellnamnet börjar med något annat än ett alfabetiskt tecken ska du omge det med dubbla citationstecken (citatomsluten identifierare)
- Formatet för `tabellelementlista` är:

  ```
  fältnamn fälttyp [DEFAULT uttr]
  [UNIQUE | NOT NULL | PRIMARY KEY | GLOBAL]
  [EXTERNAL relativ_sökväg [SECURE | OPEN beräknad_sökväg]]
  ```

  - `fältnamn` är namnet på fältet. Ett annat fält i samma tabell får inte ha samma namn. Du anger en fältrepetition med hjälp av ett tal inom hakparenteser. Till exempel: `lastDates[4]`. Fältnamn börjar med ett alfabetiskt tecken. Om fältnamnet börjar med något annat än ett alfabetiskt tecken ska du omge det med dubbla citationstecken (citatomsluten identifierare) Exempel: `CREATE TABLE`-satsen för fältet som heter `_LASTNAME` är:

    ```
    CREATE TABLE "_EMPLOYEE" (ID INT PRIMARY KEY, "_FIRSTNAME"
    VARCHAR(20), "_LASTNAME" VARCHAR(20))
    ```

  - `fälttyp` kan vara ett av följande: `NUMERIC`, `DECIMAL`, `INT`, `DATE`, `TIME`, `TIMESTAMP`, `VARCHAR`, `CHARACTER VARYING`, `BLOB`, `VARBINARY`, `LONGVARBINARY` eller `BINARY VARYING`. Du kan ange noggrannheten och skalan för `NUMERIC` och `DECIMAL`. Till exempel: `DECIMAL(10,0)`. Du kan ange noggrannheten för `TIME` och `TIMESTAMP`. Till exempel: `TIMESTAMP(6)`. Du kan ange stränglängden för `VARCHAR` och `CHARACTER VARYING`. Till exempel: `VARCHAR(255)`.
  - Med nyckelordet `DEFAULT` kan du ange ett standardvärde för en kolumn. Till `uttryck` kan du använda ett konstant värde eller ett uttryck. Tillåtna uttryck är `USER`, `USERNAME`, `CURRENT_USER`, `CURRENT_DATE`, `CURDATE`, `CURRENT_TIME`, `CURTIME`, `CURRENT_TIMESTAMP`, `CURTIMESTAMP` och `NULL`.
  - Om du anger att en kolumn ska vara `UNIQUE` aktiveras automatiskt kontrolltillvalet **Unikt** för motsvarande fält i FileMaker-databasfilen.
  - Om du anger att en kolumn ska vara `NOT NULL` aktiveras automatiskt kontrolltillvalet **Ej tomt** för motsvarande fält i FileMaker-databasfilen. Fältet flaggas med texten **Ej tomt** på fliken **Fält** i dialogrutan Hantera databas i FileMaker Pro.

- När du vill definiera en kolumn som ett containerfält använder du `BLOB`, `VARBINARY` eller `BINARY VARYING` som `fälttyp`.
- När du vill definiera en kolumn som ett containerfält som lagrar data externt använder du nyckelordet `EXTERNAL`. Med `relativ_sökväg` definieras mappen där data lagras externt i förhållande till FileMaker-databasen. Sökvägen måste anges som baskatalogen i dialogrutan Hantera containrar i FileMaker Pro. Du måste ange antingen `SECURE` för säker lagring eller `OPEN` för öppen lagring. Om du använder öppen lagring är `beräknad_sökväg` mappen i `relativ_sökväg` där containerobjekten sparas. I sökvägen måste det finnas snedstreck (/) i mappens namn.

Exempel

Med	SQL-kod
textkolumn	`CREATE TABLE T1 (C1 VARCHAR, C2 VARCHAR (50), C3 VARCHAR (1001), C4 VARCHAR (500276))`
textkolumn, `NOT NULL`	`CREATE TABLE T1NN (C1 VARCHAR NOT NULL, C2 VARCHAR (50) NOT NULL, C3 VARCHAR (1001) NOT NULL, C4 VARCHAR (500276) NOT NULL)`
numerisk kolumn	`CREATE TABLE T2 (C1 DECIMAL, C2 DECIMAL (10,0), C3 DECIMAL (7539,2), C4 DECIMAL (497925,301))`
datumkolumn	`CREATE TABLE T3 (C1 DATE, C2 DATE, C3 DATE, C4 DATE)`
tidskolumn	`CREATE TABLE T4 (C1 TIME, C2 TIME, C3 TIME, C4 TIME)`
tidstämpelkolumn	`CREATE TABLE T5 (C1 TIMESTAMP, C2 TIMESTAMP, C3 TIMESTAMP, C4 TIMESTAMP)`
kolumn för containerfält	`CREATE TABLE T6 (C1 BLOB, C2 BLOB, C3 BLOB, C4 BLOB)`
kolumn för containerfält för extern lagring	`CREATE TABLE T7 (C1 BLOB EXTERNAL 'Filer/MinDatabas/' SECURE)` `CREATE TABLE T8 (C1 BLOB EXTERNAL 'Filer/MinDatabas/' OPEN 'Objekt')`

ALTER TABLE-sats

Använd en `ALTER TABLE`-sats när du vill ändra strukturen i en befintlig tabell i en databasfil. Du kan bara ändra en kolumn i varje instruktion. `ALTER TABLE`-satsen har följande format:

```
ALTER TABLE tabellnamn ADD [COLUMN] kolumndefinition

ALTER TABLE tabellnamn DROP [COLUMN] kolumnnamn

ALTER TABLE tabellnamn ALTER [COLUMN] kolumndefinition SET DEFAULT uttr

ALTER TABLE tabellnamn ALTER [COLUMN] kolumndefinition DROP DEFAULT
```

Du måste känna till strukturen i tabellen och hur du vill ändra den innan du använder `ALTER TABLE`-satsen.

Exempel

För att	SQL-kod
lägga till kolumner	`ALTER TABLE Säljare ADD C1 VARCHAR`
ta bort kolumner	`ALTER TABLE Säljare DROP C1`
ange standardvärdet för en kolumn	`ALTER TABLE Säljare ALTER Företag SET DEFAULT 'FileMaker'`
ta bort standardvärdet för en kolumn	`ALTER TABLE Säljare ALTER Företag DROP DEFAULT`

Obs! `SET DEFAULT` och `DROP DEFAULT` påverkar inte befintliga rader i en tabell, men ändrar standardvärdet för rader som sedan läggs till i tabellen.

CREATE INDEX-sats

Använd en `CREATE INDEX`-sats när du vill söka snabbare i en databasfil. `CREATE INDEX`-satsen har följande format:

```
CREATE INDEX ON tabellnamn.kolumnnamn
CREATE INDEX ON tabellnamn (kolumnnamn)
```

`CREATE INDEX` kan användas för en enskild kolumn (indexering i flera kolumner stöds inte). Indexeringar kan inte göras på kolumner som motsvarar containerfälttyper, statistikfält, fält som använder tillvalet för global lagring eller beräkningsfält vilkas värden inte lagras i en FileMaker-databasfil.

Om du skapar ett index för en textkolumn aktiveras automatiskt indexeringstillvalet **Minimal** under **Indexering** för motsvarande fält i FileMaker-databasfilen. Om du skapar ett index för en icke-textkolumn (eller en kolumn utformad för japansk text) aktiveras automatiskt indexeringstillvalet **Allt** under **Indexering** för motsvarande fält i FileMaker-databasfilen.

Om du skapar ett index för en kolumn (vilken som helst) aktiveras automatiskt indexeringstillvalet **Skapa index automatiskt vid behov** under **Indexering** för motsvarande fält i FileMaker-databasfilen.

FileMaker skapar automatiskt de index som behövs. Med `CREATE INDEX` skapas indexet omedelbart i stället för på begäran.

Exempel

```
CREATE INDEX ON Säljare.Försäljar_ID
```

DROP INDEX-sats

Använd en `DROP INDEX`-sats när du vill ta bort ett index från en databasfil. `DROP INDEX`-satsen har följande format:

```
DROP INDEX ON tabellnamn.kolumnnamn
DROP INDEX ON tabellnamn (kolumnnamn)
```

Ta bort ett index när databasfilen är för stor eller om du sällan använder ett fält i frågorna.

Om dina frågor ger dåligt resultat och du arbetar med en mycket stor FileMaker-databasfil med många indexerade textfält, bör du överväga att ta bort index från några av fälten. Du kan också ta bort index från fält som du sällan använder i `SELECT`-satser.

Om du tar bort ett index från en kolumn (vilken som helst) kommer indexeringstillvalet **Ingen** att aktiveras och rutan **Skapa index automatiskt vid behov** att avmarkeras under **Indexering** för motsvarande fält i FileMaker-databasfilen.

Attributet `PREVENT INDEX CREATION` stöds inte.

Exempel

```
DROP INDEX ON Säljare.Försäljar_ID
```

SQL-satser

Använd uttryck i `WHERE`-, `HAVING`- och `ORDER BY`-instruktioner i `SELECT`-uttryck när du vill skapa detaljerade och avancerade databasfrågor. De giltiga uttryckselementen är följande:

- Fältnamn
- Konstanter
- Exponentiell/matematisk notation
- Numeriska operatorer
- Teckenoperatorer
- Datumoperatorer
- Relationsoperatorer
- Logiska operatorer
- Funktioner

Fältnamn

Det vanligaste uttrycket är ett enkelt fältnamn, t.ex. `beräkn` eller `Säljdata.Faktura_ID`.

Konstanter

Konstanter är värden som inte ändras. I uttrycket `PRIS * 1,05` är värdet 1,05 en konstant. Du kan också använda värdet 30 i konstanten `Antal_dagar_i_juni`.

Värden för teckenkonstanter måste omslutas av enkla citationstecken ('). Om du vill ta med ett enkelt citationstecken i en teckenkonstant som omsluts av enkla citationstecken, använder du två enkla citationstecken tillsammans (t.ex. `'Don''t'`).

För ODBC- och JDBC-program godkänner FileMaker ODBC/JDBC-formatet för datum-, tid- och tidsstämpelkonstanter inom klammerparenteser ({}), till exempel:

- `{D '2012-06-05'}`
- `{T '14:35:10'}`
- `{TS '2012-06-05 14:35:10'}`

Typspecificeraren (`D`, `T`, `TS`) kan vara versal eller gemen. Du kan använda ett valfritt antal mellanslag efter typspecifikationen. Du kan till och med utelämna mellanslaget.

FileMaker godkänner även SQL-92 syntaxens ISO-datum- och tidformat utan klammerparenteser:

- `DATE 'YYYY-MM-DD'`
- `TIME 'HH:MM:SS'`
- `TIMESTAMP 'YYYY-MM-DD HH:MM:SS'`

ExecuteSQL-funktionen i FileMaker Pro godkänner endast SQL-92 syntaxens ISO-datum- och tidsformat utan klammerparenteser.

Konstant	Giltig syntax (exempel)
Text	`'Paris'`
Numeriskt	`1,05`

Datum	`DATE '2012-06-05'`
	`{ D '2012-06-05' }`
	`{2012-06-05}`
	`{2012-06-05}`
	Obs! Syntax med tvåsiffrigt årtal stöds inte för ODBC/JDBC-format eller SQL-92-format.

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

Tidsstämpel	`TIMESTAMP '2012-06-05 14:35:10'`
	`{ TS '2012-06-05 14:35:10'}`
	`{2012-06-05 14:35:10}`
	`{2012-06-05 14:35:10}`
	Se till att **Av typen**: **Fyrsiffrigt årtal** inte är markerat som valideringstillval i FileMaker-databasfilen för ett fält som använder syntax med tvåsiffrigt årtal.
	Obs! Syntax med tvåsiffrigt årtal stöds inte för ODBC/JDBC-format eller SQL-92-format.

När du anger datum- och tidsvärden måste deras format stämma med de nationella inställningarna för databasfilen. Om databasen till exempel skapades i ett system med italienska språkinställningar måste du använda italienska datum- och tidsformat.

Exponentiell/matematisk notation

Tal kan uttryckas med matematisk notation.

Exempel

```
SELECT kolumn1, 3.4E+7 FROM tabell1 WHERE beräkn < 3.4E-6 * kolumn2
```

Numeriska operatorer

Du kan ta med följande operatorer i numeriska uttryck: +, -, *, /, ^ och ** (exponent).

Du kan låta numeriska uttryck föregås av ett unärt plus (+) eller minus (-).

Teckenoperatorer

Du kan sammanlänka tecken.

Exempel

I följande exempel är `efternamn 'JANSSON '` och förnamn `'ROBERT ':`

Operator	Sammanlänkning	Exempel	Resultat
+	Behåll avslutande tomma tecken	förnamn + efternamn	”ROBERT JANSSON ”
-	Flytta avslutande tomma tecken till slutet	förnamn - efternamn	”ROBERTJANSSON ”

Datumoperatorer

Du kan ändra datum.

Exempel

I exemplen nedan är `anst_datum DATUM '2013-01-30'.`

Operator	Effekt på datum	Exempel	Resultat
+	Lägg till ett antal dagar i ett datum	`anst_datum` +5	DATE '2013-02-04'
-	Hitta antalet dagar mellan två datum	`anst_datum - DATUM '2013-01-01'`	`29`
	Subtrahera ett antal dagar från ett datum	`anst_datum - 10`	DATE '2013-01-20'

Fler exempel:

```
SELECT Försäljnings_Datum, Försäljnings_Datum + 30 AS agg FROM Sälj_Data
SELECT Försäljningsdatum, Försäljningsdatum - 30 AS agg FROM Säljdata
```

Relationsoperatorer

Operator	Betydelse
`=`	Lika med
`<>`	Inte lika med
`>`	Större än
`>=`	Större än eller lika med
`<`	Mindre än
`<=`	Mindre än eller lika med
`LIKE`	Matcha ett mönster
`NOT LIKE`	Matcha inte ett mönster
`IS NULL`	Lika med Null
`IS NOT NULL`	Inte lika med Null
`BETWEEN`	Intervall av värden mellan en undre och en övre gräns
`IN`	En medlem av en uppsättning angivna värden eller en medlem av en delfråga
`NOT IN`	Inte en medlem av en uppsättning angivna värden eller en medlem av en delfråga

Operator	Betydelse
EXISTS	”Sant” om en delfråga returneras som minst en post
ANY	Jämför ett värde med varje värde som returneras av en delfråga (operatorn måste föregås av =, <>, >, >=, < eller <=); =ANY motsvarar IN
ALL	Jämför ett värde med varje värde som returneras av en delfråga (operatorn måste föregås av =, <>, >, >=, < eller <=)

Exempel

```
SELECT Säljdata.Fakturanummer FROM Säljdata
   WHERE Säljdata.Försäljar_ID = 'FS-1'

SELECT Säljdata.Summa FROM Säljdata WHERE Säljdata.Faktura_ID <> 125

SELECT Säljdata.Summa FROM Säljdata WHERE Säljdata.Summa > 3000

SELECT Säljdata.Försäljningstid FROM Säljdata
   WHERE Säljdata.Försäljningstid < '12:00:00'

SELECT Säljdata.Företagsnamn FROM Säljdata
   WHERE Säljdata.Företagsnamn LIKE '%Universitet'

SELECT Säljdata.Företagsnamn FROM Säljdata
   WHERE Säljdata.Företagsnamn NOT LIKE '%Universitet'

SELECT Säljdata.Summa FROM Säljdata WHERE Säljdata.Summa IS NULL

SELECT Säljdata.Summa FROM Säljdata WHERE Säljdata.Summa IS NOT NULL

SELECT Säljdata.Fakturanummer FROM Säljdata
   WHERE Säljdata.Fakturanummer BETWEEN 1 AND 10

SELECT COUNT(Säljdata.Faktura_ID) AS agg
   FROM Säljdata WHERE Säljdata.INVOICE_ID IN (50,250,100)

SELECT COUNT(Säljdata.Faktura_ID) AS agg
   FROM Säljdata WHERE Säljdata.INVOICE_ID NOT IN (50,250,100)

SELECT COUNT (Säljdata.Faktura_ID) AS agg FROM Säljdata
   WHERE Säljdata.INVOICE_ID NOT IN (SELECT Säljdata.Faktura_ID
   FROM Säljdata WHERE Säljdata.Försäljar_ID = 'FS-4')

SELECT *
   FROM Säljdata WHERE EXISTS (SELECT Säljdata.Summa
   FROM Säljdata WHERE Säljdata.Försäljar_ID IS NOT NULL)

SELECT *
   FROM Säljdata WHERE Säljdata.Summa = ANY (SELECT Säljdata.Summa
   FROM Säljdata WHERE Säljdata.Försäljar_ID = 'FS-1')

SELECT *
   FROM Säljdata WHERE Säljdata.Summa = ALL (SELECT Säljdata.Summa
   FROM Säljdata WHERE Säljdata.Försäljar_ID IS NULL)
```

Logiska operatorer

Du kan kombinera två eller flera villkor. Relationer måste skapas mellan villkoren med hjälp av `AND` eller `OR`, t.ex.:

```
lön = 40000 AND undantag = 1
```

Du använder den logiska NOT-operatorn för att göra innebörden till den omvända, t.ex.:

```
NOT (lön = 40000 AND undantag = 1)
```

Exempel

```
SELECT * FROM Säljdata WHERE Säljdata.Företagsnamn
   NOT LIKE '%Universitet' AND Säljdata.Summa > 3000

SELECT * FROM Säljdata WHERE (Säljdata.Företagsnamn
   LIKE '%Universitet' OR Säljdata.Summa > 3000)
   AND Säljdata.Försäljar_ID = 'FS-1'
```

Prioritetsordning för operatorer

Varefter uttryck blir mer komplicerade, blir det viktigare i vilken ordning uttrycket utvärderas. Tabellen visar i vilken ordning operatorerna utvärderas. Operatorerna på första raden utvärderas först och så vidare. Operatorer på samma rad utvärderas från vänster till höger i uttrycket.

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

I följande exempel visas vikten av prioritetsordning:

```
WHERE lön > 40000 OR anst_datum > (DATE '2008-01-30') AND avd = 'D101'
```

Eftersom AND utvärderas först hämtar denna fråga alla anställda på avdelning D101 som anställts efter 30 januari 2008 samt alla anställda som tjänar mer än 40000, oavsett avdelning och anställningsdatum.

Om du vill tvinga instruktionen att utvärderas i en annan ordning, omsluter du de villkor du vill utvärdera först inom parentes. Till exempel:

```
WHERE (lön > 40000 OR anst_datum > DATE '2008-01-30') AND avd = 'D101'
```

hämtar anställda på avdelning D101 som antingen tjänar mer än 40000 eller som anställdes efter 30 januari 2008.

SQL-funktioner

FileMaker SQL har stöd för många funktioner som du kan använda i uttryck. Vissa av funktionerna returnerar teckensträngar, vissa returnerar tal, vissa returnerar datum och vissa returnerar värden som beror på villkor som uppfylls av funktionsargumenten.

Statistikfunktioner

Statistikfunktioner returnerar ett enskilt värde från en uppsättning poster. Du kan använda en statistikfunktion som en del av en `SELECT`-sats, med ett fältnamn (t.ex. `AVG(lön)` eller i kombination med ett kolumnuttryck (t.ex. `AVG(lön * 1.07)`).

Du kan låta kolumnuttrycket föregås av `DISTINCT`-operatorn för att eliminera dubblerade värden. Till exempel:

```
COUNT (DISTINCT efternamn)
```

I det här exemplet räknas endast unika efternamn.

Statistikfunktionen...	**...returnerar följande**
`SUM`	Totalsumman för värdena i det numeriska fältuttrycket. Till exempel returnerar `SUM(Lön)` summan av alla värden i fältet Lön.
`AVG`	Medelvärdet för värdena i ett numeriskt fältuttryck. Till exempel returnerar `AVG(Lön)` medelvärdet av alla värden i fältet Lön.
`COUNT`	Antal värden i ett fältuttryck. Till exempel returnerar `COUNT(Namn)` antalet värden i fältet Namn. När du använder `COUNT` med ett fältnamn returnerar `COUNT` antalet fältvärden som inte är tomma. Ett specialexempel är `COUNT(*)`, som returnerar antal poster i uppsättningen, även tomma poster.
`MAX`	Maxvärdet för ett fältuttryck. Till exempel returnerar `MAX(Lön)` det maximala värdet i fältet Lön.
`MIN`	Minimivärdet för ett fältuttryck. Till exempel returnerar `MIN(Lön)` det minsta värdet i fältet Lön.

Exempel

```
SELECT SUM (Säljdata.Summa) AS agg FROM Säljdata

SELECT AVG (Säljdata.Summa) AS agg FROM Säljdata

SELECT COUNT (Säljdata.Summa) AS agg FROM Säljdata

SELECT MAX (Säljdata.Summa) AS agg FROM Säljdata
   WHERE Säljdata.Summa < 3000

SELECT MIN (Säljdata.Summa) AS agg FROM Säljdata
   WHERE Säljdata.Summa > 3000
```

Funktioner som returnerar teckensträngar

Funktioner som returnerar teckensträngar	Beskrivning	Exempel
CHR	Konverterar en ASCII-kod till en sträng bestående av ett tecken	CHR(67) returnerar **C**
CURRENT_USER	Returnerar det inloggnings-ID som angavs vid anslutningstiden	
DAYNAME	Returnerar namnet på den dag som motsvarar ett visst datum	
RTRIM	Tar bort avslutande blanksteg från en sträng	RTRIM(' ABC ') returnerar ' ABC'
TRIM	Tar bort inledande och avslutande blanksteg från en sträng	TRIM(' ABC ') returnerar 'ABC'
LTRIM	Tar bort inledande blanksteg från en sträng	LTRIM(' ABC') returnerar 'ABC'
UPPER	Ändrar varje bokstav i en sträng till versal	UPPER('Allen') returnerar 'ALLEN'
LOWER	Ändrar varje bokstav i en sträng till gemen	LOWER('Allen') returnerar 'allen'
LEFT	Returnerar tecknen längst till vänster i en sträng	LEFT('Mattson',3) returnerar 'Mat'
MONTHNAME	Returnerar namnet på kalendermånaden	
RIGHT	Returnerar tecknen längst till höger i en sträng	RIGHT('Mattson',4) returnerar 'tson'
SUBSTR SUBSTRING	Returnerar en delsträng i en sträng, med parametrar för strängen, det första tecknet att extrahera och antal tecken som ska extraheras (valfritt)	SUBSTR('Konrad',2,3) returnerar 'onr' SUBSTR('Konrad',2) returnerar 'onrad'
MELLANSLAG	Genererar en sträng med blanksteg	SPACE(5) returnerar ' '
STRVAL	Konverterar ett värde av vilken typ som helst till en teckensträng	STRVAL('Hallman') returnerar 'Hallman' STRVAL(5 * 3) returnerar '15' STRVAL(4 = 5) returnerar 'Falskt' STRVAL(DATE '2008-12-25') returnerar '2008-12-25'
TIME TIMEVAL	Returnerar klockslaget som en sträng	Klockan 21:49, TIME() returnerar **21:49:00**
USERNAME USER	Returnerar det inloggnings-ID som angavs vid anslutningstiden	

Obs! TIME()-funktionen används inte längre. Använd i stället SQL-standarden CURRENT_TIME.

Exempel

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

SELECT RTRIM(' ' + Säljare.Försäljar_ID) AS agg FROM Säljare

SELECT TRIM(SPACE(1) + Säljare.Försäljar_ID) AS agg FROM Säljare

SELECT LTRIM(' ' + Säljare.Försäljar_ID) AS agg FROM Säljare

SELECT UPPER(Säljare.Försäljare) AS agg FROM Säljare

SELECT LOWER(Säljare.Försäljare) AS agg FROM Säljare

SELECT LEFT (Säljare.Försäljare, 5) AS agg FROM Säljare

SELECT RIGHT (Säljare.Försäljare, 7) AS agg FROM Säljare

SELECT SUBSTR(Säljare.Försäljar_ID, 2, 2) + SUBSTR(Säljare.Försäljar_ID,
4, 2) AS agg FROM Säljare

SELECT SUBSTR(Säljare.Försäljar_ID, 2) + SUBSTR(Säljare.Försäljar_ID, 4)
AS agg FROM Säljare

SELECT SPACE(2) + Säljare.Försäljar_ID AS Försäljar_ID FROM Säljare

SELECT STRVAL('60506') AS agg FROM Säljdata WHERE Säljdata.Faktura = 1
```

Funktioner som returnerar siffror

Funktioner som returnerar siffror	Beskrivning	Exempel
ABS	Returnerar det absoluta värdet för ett numeriskt uttryck	
ATAN	Returnerar argumentets arcustangens som en vinkel uttryckt i radianer	
ATAN2	Returnerar arcustangensen för x- och y-koordinater som en vinkel uttryckt i radianer	
CEIL CEILING	Returnerar det minsta heltalsvärdet som är större än eller lika med argumentet	
DEG DEGREES	Returnerar antalet grader för argumentet, vilket är vinkeln uttryckt i radianer	
DAY	Returnerar dagen i ett datum	`DAY(DATE '2012-01-30')` returnerar **30**
DAYOFWEEK	Returnerar veckodagen (1-7) i ett datumuttryck	`DAYOFWEEK(DATE '2004-05-01')` returnerar **7**
MOD	Dividerar två tal och returnerar återstoden av divisionen	`MOD(10,3)` returnerar **1**

Funktioner som returnerar siffror	Beskrivning	Exempel
EXP	Returnerar ett värde som är basen för den naturliga logaritmen (e) upphöjt till ett värde som anges av argumentet	
FLOOR	Returnerar det största heltalsvärdet som är mindre än eller lika med argumentet	
HOUR	Returnerar timkomponenten i ett värde	
INT	Returnerar heltalsdelen av ett tal	`INT(6.4321)` returnerar **6**
LENGTH	Returnerar längden på en sträng	`LENGTH('ABC')` returnerar **3**
MONTH	Returnerar månaden i ett datum	`MONTH(DATE '2012-01-30')` returnerar **1**
LN	Returnerar argumentets naturliga logaritm	
LOG	Returnerar argumentets vanliga logaritm	
MAX	Returnerar det högsta av två tal	`MAX(66,89)` returnerar **89**
MIN	Returnerar det lägsta av två tal	`MIN(66,89)` returnerar **66**
MINUTE	Returnerar minutkomponenten i ett värde	
NUMVAL	Konverterar en teckensträng till ett tal. Funktionen misslyckas om teckensträngen inte är ett giltigt tal.	`NUMVAL('123')` returnerar **123**
PI	Returnerar det konstanta värdet av den matematiska konstanten pi	
RADIANS	Returnerar antalet radianer för ett argument som uttrycks i grader	
ROUND	Avrundar ett tal	`ROUND(123 456,0)` returnerar **123** `ROUND(123 456,2)` returnerar **123,46** `ROUND(123 456,-2)` returnerar **100**
SECOND	Returnerar sekundkomponenten i ett värde	
SIGN	En indikator på tecknet i argumentet: -1 för negativt, 0 för 0 och 1 för positivt	
SIN	Returnerar sinus för argumentet	
SQRT	Returnerar argumentets kvadratrot	
TAN	Returnerar tangensen för argumentet	
YEAR	Returnerar året i ett datum	`YEAR(DATE '2013-01-30')` returnerar **2013**

Funktioner som returnerar datum

Funktioner som returnerar datum	Beskrivning	Exempel
`CURDATE` `CURRENT_DATE`	Returnerar dagens datum	
`CURTIME` `CURRENT_TIME`	Returnerar aktuell tid	
`CURTIMESTAMP` `CURRENT_TIMESTAMP`	Returnerar aktuellt tidsstämpelvärde	
`TIMESTAMPVAL`	Konverterar en teckensträng till en tidsstämpel	`TIMESTAMPVAL('2013-01-30 14:00:00')` returnerar dess tidsstämpelvärde.
`DATE` `TODAY`	Returnerar dagens datum	Om dagens datum är 11/21/2013, returnerar `DATE()` **2013-11-21**
`DATEVAL`	Konverterar en teckensträng till ett datum	`DATEVAL('2013-01-30')` returnerar **2013-01-30**

Obs! `DATE()`-funktionen används inte längre. Använd i stället SQL-standarden CURRENT_DATE.

Villkorsfunktioner

Villkorsfunktioner	Beskrivning	Exempel

`CASE WHEN`	**Enkelt CASE-format**
	
	Jämför värdet för *indata_exp* med värdena för argumenten *värde_exp* för att fastställa resultatet.
	
	`CASE` *indata_exp*
	`{WHEN` *värde_exp* `THEN` *resultat*`...}`
	`[ELSE` *resultat*`]`
	END
		`SELECT`
		`  Faktura_ID,`
		`  CASE Företagsnamn`
		`  WHEN 'UK-exporter' THEN 'Hittade UK-exporter'`
		`  WHEN 'Leverantörer av heminredning' THEN 'Hittade leverantörer av heminredning'`
		`  ELSE 'Varken UK-exporter eller Leverantörer av heminredning'`
		`  END,`
		`  Försäljar_ID`
		`FROM`
		`  Sälj_Data`

	Sökt CASE-format
	
	Returnerar ett resultat utifrån om villkoret som anges av en WHEN-sats är sant.
	
	`CASE`
	`{WHEN` *logiskt_uttr* `THEN` *resultat*`...}`
	`[ELSE` *resultat*`]`
	END
		`SELECT`
		`  Faktura_ID,`
		`  Summa,`
		`  CASE`
		`  WHEN Summa > 3000 THEN 'Över 3000'`
		`  WHEN Summa > 1000 THEN 'Under 3000'`
		`  ELSE 'Mellan 1000 och 3000'`
		`  END,`
		`  Försäljar_ID`
		`FROM`
		`  Sälj_Data`

Villkorsfunktioner	Beskrivning	Exempel
COALESCE	Returnerar det första värdet som inte är NULL	SELECT Försäljar_ID, COALESCE(Försäljnings_Chef, Försäljare) FROM Säljare
NULLIF	Jämför två värden och returnerar NULL om de två värdena är lika. Annars returneras det första värdet.	SELECT Faktura_ID, NULLIF(Summa, -1), Försäljar_ID FROM Sälj_Data

Reserverade SQL-nyckelord

I avsnittet visas de reserverade nyckelord som inte får användas som namn på kolumner, tabeller, alias eller andra användardefinierade objekt. Om du får syntax-fel kan det bero på att du har använt ett av dessa reserverade nyckelord. Om du vill använda något av dessa nyckelord måste citationstecken användas för att förhindra termen från att behandlas som ett nyckelord.

Följande `CREATE TABLE`-sats visar t.ex. hur du använder `DEC`-nyckelordet som namn på ett dataelement.

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

ABSOLUTE	CHAR	CURTIME
ACTION	CHARACTER	CURTIMESTAMP
ADD	CHARACTER_LENGTH	DATE
ALL	CHAR_LENGTH	DATEVAL
ALLOCATE	CHECK	DAY
ALTER	CHR	DAYNAME
AND	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
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
NSERT
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
UNION
UNIQUE
UNKNOWN
UPDATE
UPPER
USAGE
USER
USERNAME
USING
VALUE
VALUES
VARBINARY
VARCHAR
VARYING
VIEW
WHEN
WHENEVER
WHERE
WITH
WORK
WRITE
YEAR
ZONE

Index

A

ABS-funktionen 28
ALL-operatorn 24
ALTER TABLE (SQL-sats) 19
AND-operatorn 25
ANY-operatorn 24
ATAN2-funktionen 28
ATAN-funktionen 28

B

BETWEEN-operatorn 23
binära data, använda i SELECT 13
BLOB-datatyp, använda i SELECT 13

C

CASE WHEN-funktionen 30
CEIL-funktionen 28
CEILING-funktionen 28
CHR-funktionen 27
COALESCE-funktionen 31
containerfält
  lagrade externt 19
  med CREATE TABLE-sats 19
  med GetAs-funktion 14
  med INSERT-sats 16
  med PutAs-funktion 16
  med SELECT-sats 14
  med UPDATE-sats 17
CREATE INDEX (SQL-sats) 20
CREATE TABLE (SQL-sats) 18
CURDATE-funktionen 30
CURRENT USER-funktionen 27
CURRENT_DATE-funktionen 30
CURRENT_TIME-funktionen 30
CURRENT_TIMESTAMP-funktionen 30
CURRENT_USER-funktionen 27
CURTIME-funktionen 30
CURTIMESTAMP-funktionen 30

D

DATE-funktionen 30
DATEVAL-funktionen 30
datumformat 21
datumoperatorer i SQL-satser 23
DAY-funktionen 28
DAYNAME-funktionen 27
DAYOFWEEK-funktionen 28
DEFAULT (SQL-sats) 18
DEG-funktionen 28
DEGREES-funktionen 28
DELETE (SQL-sats) 15
delfrågor 16
DISTINCT-operatorn 7
DROP INDEX (SQL-sats) 20

E

ExecuteSQL-funktion 5, 6
EXISTS-operatorn 24
EXP-funktionen 29
exponentiell notation i SQL-satser 22
EXTERNAL (SQL-sats) 19

F

fältnamn i SQL-satser 21
fältrepetitioner 18
FETCH FIRST (SQL-sats) 11
filer, använda i containerfält 14
FLOOR-funktionen 29
FOR UPDATE (SQL-sats) 12
FROM (SQL-sats) 8
FULL OUTER JOIN 8
funktionen CAST 14
funktionen GetAs 14
funktioner i SQL-satser 26

G

GROUP BY (SQL-sats) 9

H

HAVING (SQL-sats) 9
HOUR-funktionen 29

I

INNER JOIN 8
IN-operatorn 23
INSERT (SQL-sats) 15
INT-funktionen 29
IS NOT NULL-operator 23
IS NULL-operatorn 23

K

kolumnalias 7
konstanter i SQL-satser 21
koppla 8

L

LEFT OUTER JOIN 8
LEFT-funktionen 27
LENGTH-funktionen 29
LIKE operatorer 23

LN-funktionen 29
LOG-funktionen 29
logiska operatorer i SQL-satser 25
LOWER-funktionen 27
LTRIM-funktionen 27

M

markörer via ODBC 12
matematisk notation i SQL-satser 22
MAX-funktionen 29
MIN-funktionen 29
MINUTE-funktionen 29
MOD-funktionen 28
MONTH-funktionen 29
MONTHNAME-funktionen 27

N

nollvärde 16
NOT IN-operatorn 23
NOT LIKE-operatorn 23
NOT NULL (SQL-sats) 18
NOT-operatorn 25
NULLIF-funktionen 31
numeriska operatorer i SQL-satser 22
NUMVAL-funktionen 29
nyckelord, reserverade SQL 32

O

ODBC-standardkompatibilitet 6
OFFSET (SQL-sats) 11
ORDER BY (SQL-sats) 10
OR-operatorn 25
OUTER JOIN 8

P

peer-rader 11, 12
PI-funktionen 29
portaler 6
Portaler för JDBC-klientdrivrutiner 6
Portaler för ODBC-klientdrivrutiner 6
positionsuppdateringar och -raderingar 12
PREVENT INDEX CREATION 21
prioritetsordning för operatorer i SQL-satser 25
PutAs-funktionen 16, 17

R

RADIANS-funktionen 29
relationsoperatorer i SQL-satser 23
reserverade SQL-nyckelord 32
RIGHT OUTER JOIN 8
RIGHT-funktionen 27
ROUND-funktionen 29
RTRIM-funktionen 27

S

SECOND-funktionen 29
SELECT (SQL-sats) 7
  binära data 13
  BLOB-datatyp 13
  tom sträng 13
SIGN-funktionen 29
SIN-funktionen 29
SPACE-funktionen 27
SQL_C_WCHAR-datatyp 6
SQL-92 6
SQL-satser 21
  ALTER TABLE 19
  CREATE INDEX 20
  CREATE TABLE 18
  datumoperatorer 23
  DELETE 15
  DROP INDEX 20
  exponentiell och matematisk notation 22
  fältnamn 21
  funktioner 26
  INSERT 15
  konstanter 21
  logiska operatorer 25
  numeriska operatorer 22
  prioritetsordning för operatorer 25
  relationsoperatorer 23
  reserverade nyckelord 32
  som stöds av klientdrivrutiner 6
  teckenoperatorer 23
  UPDATE 17
SQL-satserna
  SELECT 7
SQL-standardkompatibilitet 6
SQL-statistikfunktioner 26
SQRT-funktionen 29
standardkompatibilitet 6
statistikfunktioner i SQL 26
strängfunktioner 27
STRVAL-funktionen 27
SUBSTR-funktionen 27
SUBSTRING-funktionen 27
syntaxfel 32

T

tabellalias 7, 8
TAN-funktionen 29
teckenoperatorer i SQL-satser 23
tidformat 21
tidsstämpelformat 21
TIME-funktionen 27
TIMESTAMPVAL-funktionen 30
TIMEVAL-funktion 27
TODAY-funktionen 30
tom sträng, använda i SELECT 13
tomma tecken 23
tomma värden i kolumner 16
TRIM-funktionen 27

U

Unicode-stöd 6
Unicode-stöd för JDBC-klientdrivrutin 6
Unicode-stöd för ODBC-klientdrivrutin 6
UNION (SQL-operator) 10
UNIQUE (SQL-sats) 18
UPDATE (SQL-sats) 17
UPPER-funktionen 27
USERNAME-funktionen 27
uttryck i SQL 21

V

VALUES (SQL-sats) 15

W

WHERE (SQL-sats) 9
WITH TIES (SQL-sats) 11

Y

YEAR-funktionen 29