# Claris FileMaker

## Naslaggegevens voor SQL

Claris

Claris International Inc.
5201 Patrick Henry Drive
Santa Clara, California 95054, VS



Editie: 01

# Inhoudsopgave

# Hoofdstuk 1
# Inleiding

Als databaseontwikkelaar kunt u Claris™ FileMaker® Pro gebruiken om databaseoplossingen te maken zonder enige kennis van SQL. Maar als u echter kennis van SQL hebt, kunt u een FileMaker Pro-databasebestand gebruiken als een ODBC- of JDBC-gegevensbron om zo uw gegevens te delen met andere toepassingen via OBDC en JDBC. U kunt de functie ExecuteSQL van FileMaker Pro ook gebruiken om gegevens uit een tabel in een FileMaker Pro-database op te halen.

Deze naslaggegevens beschrijven de SQL-instructies en de standaarden die door FileMaker-software worden ondersteund. De ODBC- en JDBC-clientstuurprogramma's van FileMaker ondersteunen alle SQL-instructies die in deze naslaggegevens worden beschreven. De functie ExecuteSQL van FileMaker Pro ondersteunt alleen de instructie SELECT.

## Over deze naslaggegevens

- Voor informatie over het gebruik van ODBC en JDBC met lagere versies van FileMaker Pro bezoekt u het centrum voor productdocumentatie.
- Voor deze naslaggegevens wordt als uitgangspunt aangenomen dat u vertrouwd bent met de basiskennis van het gebruik van FileMaker Pro-functies, het coderen van ODBC- en JDBC-toepassingen en het maken van SQL-opvragen. Raadpleeg een handleiding van een andere leverancier voor meer informatie over deze onderwerpen.

## Over SQL

SQL, ofwel Structured Query Language, is een programmeertaal die is ontworpen voor het opvragen van gegevens uit een relationele database. De meeste gebruikte instructie voor het opvragen van gegevens uit een database is de instructie SELECT.

SQL is echter niet alleen een taal voor het opvragen van gegevens uit een database maar biedt ook instructies voor de bewerking van gegevens, zoals het toevoegen, bijwerken en verwijderen van gegevens.

Daarnaast biedt SQL ook instructies voor de definiëring van gegevens. Met deze instructies kunt u tabellen en indexen maken en wijzigen.

De SQL-instructies en standaarden die worden ondersteund door FileMaker-software worden beschreven in hoofdstuk 2, “Ondersteunde standaarden.”

## Een FileMaker Pro-database als een gegevensbron gebruiken

Wanneer u een FileMaker Pro-database host als een ODBC- of JDBC-gegevensbron, kunnen FileMaker-gegevens worden gedeeld met ODBC- en JDBC-compatibele toepassingen. De toepassingen maken verbinding met de FileMaker-gegevensbron met behulp van de FileMaker-clientstuurprogramma's, maken SQL-opvragen en voeren ze uit met behulp van ODBC of JDBC en verwerken de gegevens die uit de FileMaker Pro-databaseoplossing zijn opgehaald.

Raadpleeg FileMaker-handleiding voor ODBC en JDBC voor uitgebreide informatie over hoe u FileMaker-software kunt gebruiken als een gegevensbron voor ODBC- en JDBC-toepassingen.

De ODBC- en JDBC-clientstuurprogramma's van FileMaker ondersteunen alle SQL-instructies die in deze naslaggegevens worden beschreven.

# De functie ExecuteSQL gebruiken

Via de functie ExecuteSQL van FileMaker Pro kunt u gegevens ophalen uit tabellen die vermeld zijn in de relatiegrafiek maar onafhankelijk van gedefinieerde relaties zijn. U kunt gegevens uit meerdere tabellen ophalen zonder tabellen samen te voegen of relaties tussen tabellen te maken. In sommige gevallen kunt u de complexiteit van uw relatiegrafiek vereenvoudigen met behulp van de functie ExecuteSQL.

De velden die u opvraagt met de functie ExecuteSQL hoeven geen deel uit te maken van een lay-out. U kunt de functie ExecuteSQL dus gebruiken om gegevens op te halen die onafhankelijk van een lay-outcontext zijn. Vanwege deze contextonafhankelijkheid kan het gebruik van de functie ExecuteSQL in script de portabiliteit van de scripts verbeteren. U kunt de functie ExecuteSQL gebruiken waar u berekeningen opgeeft, inclusief voor het uitzetten van gegevens in een grafiek of in een rapport.

De functie ExecuteSQL ondersteunt alleen de SELECT-instructie die is beschreven in de sectie "De instructie SELECT" op pagina 8.

De functie ExecuteSQL accepteert ook alleen ISO-datum- en tijdopmaak van de syntaxis SQL-92 zonder accolades ({}). De functie ExecuteSQL accepteert geen ODBC/JDBC-opmaak voor data, tijden en tijdstempelconstanten tussen accolades.

Raadpleeg FileMaker Pro Help voor informatie over de syntaxis en het gebruik van de functie ExecuteSQL.

# Hoofdstuk 2
# Ondersteunde standaarden

Gebruik de FileMaker ODBC- en JDBC-clientstuurprogramma's om vanuit een ODBC- of JDBC-compatibele toepassing toegang te krijgen tot een FileMaker Pro-databaseoplossing. De FileMaker Pro-databaseoplossing kan door FileMaker Pro of FileMaker Server worden gehost.

- Het ODBC-clientstuurprogramma ondersteunt ODBC 3.0 Niveau 1:
- Het JDBC-clientstuurprogramma biedt gedeeltelijke ondersteuning voor de JDBC 3.0-specificatie.
- De ODBC- en JDBC-clientstuurprogramma's zijn beiden geschikt voor SQL-92 Entry Level en ondersteunen enkele functies van SQL-92 Intermediate Level.

## Ondersteuning voor Unicode-tekens

De ODBC- en JDBC-clientstuurprogramma's ondersteunen de Unicode-API. Als u echter een eigen toepassing maakt die de clientstuurprogramma's gebruikt, gebruikt u ASCII voor veld-, tabel- en bestandsnamen (bij gebruik van een niet-Unicode opvraaghulpprogramma of –toepassing).

Opmerking Gebruik `SQL_C_WCHAR` als u Unicode-gegevens wilt invoegen en ophalen.

## SQL-instructies

De ODBC- en JDBC-clientstuurprogramma's bieden ondersteuning voor de volgende SQL-instructies:

- SELECT (pagina 8)
- DELETE (pagina 17)
- INSERT (pagina 17)
- UPDATE (pagina 19)
- CREATE TABLE (pagina 20)
- TRUNCATE TABLE (pagina 21)
- ALTER TABLE (pagina 22)
- CREATE INDEX (pagina 22)
- DROP INDEX (pagina 23)

De clientstuurprogramma's ondersteunen ook de toewijzing van FileMaker-gegevenstypen aan die van ODBC-SQL en JDBC-SQL. Raadpleeg de FileMaker-handleiding voor ODBC en JDBC voor conversies van gegevenstypen. Voor meer informatie over het maken van SQL-opvragen kunt u een handboek van een andere leverancier raadplegen.

Opmerking De ODBC- en JDBC-clientstuurprogramma's bieden geen ondersteuning voor FileMaker Pro-portalen.

## De instructie SELECT

Gebruik de instructie `SELECT` om de kolommen op te geven die u wilt opvragen. Na de instructie `SELECT` volgen de kolomuitdrukkingen (vergelijkbaar met veldnamen) die u wilt opvragen (bijvoorbeeld `achternaam`). Uitdrukkingen kunnen rekenkundige bewerkingen of tekenreeksbewerkingen bevatten (bijvoorbeeld `SALARIS * 1.05`).

Voor de instructie `SELECT` kunnen verschillende elementen worden gebruikt:

```
SELECT [DISTINCT] {* | kolomuitdrukking [[AS] kolom_alias],...}
FROM tabel_naam [tabel_alias], ...
[ WHERE uitdr1 rel_operator uitdr2 ]
[ GROUP BY {kolomuitdrukking, ...} ]
[ HAVING uitdr1 rel_operator uitdr2 ]
[ UNION [ALL] (SELECT...) ]
[ ORDER BY {sorteeruitdrukking [DESC | ASC]}, ... ]
[ OFFSET n {ROWS | ROW} ]
[ FETCH FIRST [ n [ PERCENT ] ] { ROWS | ROW } {ONLY | WITH TIES } ]
[ FOR UPDATE [OF {kolomuitdrukking, ...}] ]
```

Waarden tussen accolades zijn optioneel.

`kolom_alias` kan worden gebruikt om de kolom een meer herkenbare naam te geven of om de langere kolomnaam in te korten.

### Voorbeeld

Wijs de alias `afdeling` toe aan de kolom `afd`:

```
SELECT afd AS afdeling FROM werkn
```

Veldnamen kunnen als prefix de tabelnaam of tabel-alias krijgen. Voorbeeld: `WERKN.ACHTERNAAM_WERKNEMER` of `W.ACHTERNAAM_WERKNEMER`, waarbij `E` de alias is voor de tabel `WERKN`.

De operator `DISTINCT` kan voorafgaan aan de eerste kolomuitdrukking. Met deze operator laat u dubbele rijen weg uit het resultaat van een opvraag.

### Voorbeeld

```
SELECT DISTINCT afd FROM werkn
```

## SQL-elementen

De ODBC- en JDBC-clientstuurprogramma's bieden ondersteuning voor de volgende SQL-elementen:

| Gebruik dit SQL-element | Om dit te doen |
|---|---|
| FROM (pagina 9) | Aangeven welke tabellen in de instructie `SELECT` worden gebruikt. |
| WHERE (pagina 11) | De voorwaarden opgeven waaraan records moeten voldoen om te worden opgehaald (zoals bij een FileMaker Pro-zoekopdracht). |
| GROUP BY (pagina 11) | De namen opgeven van een of meer velden waarop de resultaatwaarden moeten worden gegroepeerd. Dit element wordt gebruikt om een reeks totaalwaarden te verkrijgen door voor elke groep één rij als resultaat te geven (zoals bij een FileMaker Pro-subresumé). |
| HAVING (pagina 12) | Voorwaarden opgeven voor groepen records (bijvoorbeeld alleen de afdelingen weergeven waarvan het totaal van de salarissen meer dan € 20.000 bedraagt). |
| UNION (pagina 12) | De resultaten van twee of meer `SELECT`-instructies in één resultaat combineren. |
| ORDER BY (pagina 13) | Aangeven hoe de records worden gesorteerd. |
| OFFSET (pagina 13) | Aantal rijen vermelden die moeten worden overgeslagen voordat de rijen worden opgehaald. |
| FETCH FIRST (pagina 13) | Het aantal op te halen rijen opgeven. Het opgegeven aantal rijen is het maximale aantal rijen dat wordt gegeven hoewel een kleiner aantal rijen kan worden gegeven als de opvraag minder resultaten geeft dan het aantal opgegeven rijen. |
| FOR UPDATE (pagina 14) | Geplaatste (positioned) updates of verwijderingen uitvoeren via SQL-cursors. |

Opmerking Als u probeert gegevens op te halen uit een tabel zonder kolommen, geeft de `SELECT`-instructie niets als resultaat.

## Het element FROM

Het element `FROM` geeft de tabellen aan die in de `SELECT`-instructie worden gebruikt. Hiervoor gebruikt u de volgende syntaxis:

```
FROM tabel_naam [tabel_alias] [, tabel_naam [tabel_alias]]
```

`tabel_naam` is de naam van een tabel in de huidige database. De tabelnaam moet beginnen met een letter. Als de tabelnaam begint met een ander teken dan een letter, plaatst u het tussen dubbele aanhalingstekens (ID tussen aanhalingstekens).

`tabel_alias` kan worden gebruikt om de tabel een meer herkenbare naam te geven of om een langere tabelnaam in te korten of om dezelfde tabel meerdere keren in de opvraag op te nemen (bijvoorbeeld in interne relaties).

Veldnamen beginnen met een letter. Als de veldnaam begint met een ander teken dan een letter, plaatst u het tussen dubbele aanhalingstekens (ID tussen aanhalingstekens).

### Voorbeeld

De instructie `ExecuteSQL` voor het veld `_ACHTERNAAM` is:

```
SELECT "_ACHTERNAAM" FROM werkn
```

Veldnamen kunnen als prefix de tabelnaam of -alias krijgen.

### Voorbeeld

Met de tabelspecificatie `FROM werknemer E` kunt u naar het veld `ACHTERNAAM` verwijzen als `E.ACHTERNAAM`. Tabelaliassen moeten worden gebruikt als de `SELECT`-instructie een tabel met zichzelf samenvoegt.

```
SELECT * FROM werknemer E, werknemer F WHERE E.manager_id = F.werknemer_id
```

Het gelijkteken (=) bevat alleen overeenkomende rijen in de resultaten.

Als u meer dan één tabel samenvoegt en u wilt alle rijen verwijderen die in beide brontabellen geen overeenkomende rijen hebben, kunt u `INNER JOIN` gebruiken.

### Voorbeeld

```
SELECT *
   FROM Verkopers INNER JOIN Verkoopgegevens
   ON Verkopers.Verkoper_ID = Verkoopgegevens.Verkoper_ID
```

Als u twee tabellen samenvoegt, maar u wilt geen rijen uit de eerste tabel (de “linkse” tabel) negeren, kunt u `LEFT OUTER JOIN` gebruiken.

### Voorbeeld

```
SELECT *
   FROM Verkopers LEFT OUTER JOIN Verkoopgegevens
   ON Verkopers.Verkoper_ID = Verkoopgegevens.Verkoper_ID
```

Elke rij uit de tabel “Verkopers” zal in de samengevoegde tabel verschijnen.

### Opmerkingen

- `RIGHT OUTER JOIN` wordt momenteel niet ondersteund.
- `FULL OUTER JOIN` wordt momenteel niet ondersteund.

## Het element WHERE

Het element `WHERE` geeft de voorwaarden op waaraan records moeten voldoen om te worden opgehaald. Het element `WHERE` bevat voorwaarden in deze vorm:

```
WHERE uitdr1 rel_operator uitdr2
```

`uitdr1` en `uitdr2` kunnen veldnamen, constante waarden of uitdrukkingen zijn.

`rel_operator` is de relationele operator die de twee expressies aan elkaar koppelt.

### Voorbeeld

Haal de namen van werknemers op die € 20.000 of meer verdienen.

```
SELECT achternaam,voornaam FROM werkn WHERE salaris >= 20000
```

Het element `WHERE` kan ook uitdrukkingen gebruiken zoals deze:

```
WHERE uitdr1 IS NULL
WHERE NOT uitdr2
```

Opmerking Als u in de `SELECT`-(projectie)lijst volledige namen gebruikt, moet u ook in de gerelateerde `WHERE`-instructie volledige namen gebruiken.

## Het element GROUP BY

Het element `GROUP BY` geeft de namen op van een of meer velden waarop de resultaatwaarden moeten worden gegroepeerd. Dit element wordt gebruikt om een reeks totaalwaarden te verkrijgen. Het is als volgt gestructureerd:

```
GROUP BY kolommen
```

Het bereik van het element `GROUP BY` is de tabeluitdrukking in het element `FROM`. Hierdoor moeten de kolomuitdrukkingen opgegeven door `columns` uit de tabellen komen die in het element `FROM` zijn opgegeven. Een kolomuitdrukking kan bestaan uit een of meer door komma's gescheiden veldnamen uit de databasetabel.

### Voorbeeld

Tel de salarissen in elke afdeling op.

```
SELECT afd_id, SUM (salaris) FROM werkn GROUP BY afd_id
```

Deze instructie geeft één rij voor elke verschillende afdelings-ID. Elke rij bevat de afdelings-ID en de som van de salarissen van de werknemers van de afdeling.

## Het element HAVING

Met het element HAVING kunt u voorwaarden opgeven voor groepen records (bijvoorbeeld alleen de afdelingen weergeven waarvan het totaal van de salarissen meer dan € 20.000 bedraagt). Het is als volgt gestructureerd:

```
HAVING uitdr1 rel_operator uitdr2
```

uitdr1 en uitdr2 kunnen veldnamen, constante waarden of uitdrukkingen zijn. Deze uitdrukkingen hoeven niet overeen te komen met een kolomexpressie in het SELECT-element.

rel_operator is de relationele operator die de twee expressies aan elkaar koppelt.

### Voorbeeld

Geef als resultaat de afdelingen waarvan de som van de salarissen groter is dan € 20.000.

```
SELECT afd_id, SUM (salaris) FROM werkn
   GROUP BY afd_id HAVING SUM (salaris) > 20000
```

## De operator UNION

De operator UNION combineert de resultaten van twee of meer SELECT-instructies in één resultaat. Het enige resultaat is alle resulterende records uit de SELECT-instructies. Dubbele records worden standaard niet in het resultaat opgenomen. Als u dubbele records in het resultaat wilt opnemen, gebruikt u het trefwoord ALL (UNION ALL). Hiervoor gebruikt u de volgende syntaxis:

```
SELECT instructie UNION [ALL] SELECT instructie
```

Bij het gebruik van de operator UNION moeten de selectielijsten voor elke SELECT-instructie hetzelfde aantal kolomuitdrukkingen hebben, met dezelfde gegevenstypen, en moeten ze in dezelfde volgorde worden opgegeven.

### Voorbeeld

```
SELECT achternaam, salaris, in_dienst FROM werkn UNION SELECT naam, salaris,
geboortedatum FROM persoon
```

Het volgende voorbeeld is niet geldig omdat de gegevenstypen van de kolomuitdrukkingen verschillen (SALARIS uit WERKN heeft een ander gegevenstype dan ACHTERNAAM uit OPSLAGEN). In dit voorbeeld heeft elke SELECT-instructie hetzelfde aantal kolomuitdrukkingen, maar bevinden de uitdrukkingen zich niet in dezelfde volgorde volgens gegevenstype.

### Voorbeeld

```
SELECT achternaam, salaris FROM werkn UNION SELECT salaris, achternaam FROM
opslagen
```

## Het element ORDER BY

Het element `ORDER BY` geeft aan hoe de records moeten worden gesorteerd. Als uw `SELECT`-instructie geen `ORDER BY`-element bevat, worden de records mogelijk in een willekeurige volgorde weergegeven.

Hiervoor gebruikt u de volgende syntaxis:

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

`sorteeruitdrukking` kan de veldnaam of het positienummer van de te gebruiken kolomuitdrukking zijn. Standaard wordt een oplopende (`ASC`) sorteervolgorde uitgevoerd.

### Voorbeelden

Sorteer eerst op `achternaam` en dan op `voornaam`.

```
SELECT werkn_id, achternaam, voornaam FROM werkn ORDER BY achternaam,
voornaam
```

In het tweede voorbeeld worden positienummers 2 en 3 gebruikt om dezelfde volgorde te verkrijgen als in het vorige voorbeeld waarbij `achternaam` en `voornaam` uitdrukkelijk waren opgegeven.

```
SELECT werkn_id, achternaam, voornaam FROM werkn ORDER BY 2,3
```

Opmerking FileMaker Server gebruikt een unicode binaire sorteervolgorde die verschilt van de taalsorteervolgorde in FileMaker Pro of de standaard taalonafhankelijke sorteervolgorde.

## De elementen OFFSET en FETCH FIRST

De elementen `OFFSET` en `FETCH FIRST` worden gebruikt om een specifiek bereik van rijen te verkrijgen dat begint vanaf een bepaald startpunt in een reeks resultaten. Door de mogelijkheid om het ophalen van rijen uit een grote reeks resultaten te beperken, kunt u 'bladeren per pagina' door de gegevens en verbetert u de efficiëntie.

Het element `OFFSET` geeft het aantal rijen aan die moeten worden overgeslagen voordat gegevens moeten worden geretourneerd. Als het element `OFFSET` niet wordt gebruikt in een `SELECT`-instructie, is de eerste rij 0. Het element `FETCH FIRST` geeft het aantal rijen op die moeten worden geretourneerd, hetzij als een geheel getal zonder teken groter dan of gelijk aan 1 hetzij als een percentage, vanaf het startpunt aangegeven in het element `OFFSET`. Als zowel `OFFSET` als `FETCH FIRST` worden gebruikt in een `SELECT`-instructie, moet eerst het element `OFFSET` worden gebruikt.

De elementen `OFFSET` en `FETCH FIRST` worden niet ondersteund in subopvragen.

### OFFSET-syntaxis

De `OFFSET`-syntaxis is:

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

`n` is een geheel getal zonder teken. Als `n` groter is dan het aantal rijen dat wordt geretourneerd in de reeks resultaten, wordt niets geretourneerd en verschijnt geen foutbericht.

`ROWS` is het hetzelfde als `ROW`.

### FETCH FIRST-syntaxis

De `FETCH FIRST`-syntaxis is:

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

`n` is het aantal rijen dat moet worden geretourneerd. De standaardwaarde is 1 als `n` wordt weggelaten.

`n` is een geheel getal zonder teken dat groter dan of gelijk aan 1 is tenzij het wordt gevolgd door `PERCENT`. Als `n` wordt gevolgd door `PERCENT`, kan de waarde een positieve fractionele waarde of een geheel getal zonder teken.

`ROWS` is het hetzelfde als `ROW`.

`WITH TIES` moet worden gebruikt met het element `ORDER BY`.

Via `WITH TIES` kunnen meer rijen worden geretourneerd dan opgegeven in de `FETCH`-waarde omdat peerrijen (rijen die niet verschillen op basis van het element `ORDER BY`) ook worden geretourneerd.

#### Voorbeelden

Geef als resultaat informatie van de zesentwintigste rij van de reeks resultaten die zijn gesorteerd op `achternaam` en dan op `voornaam`.

```
SELECT werkn_ID, achternaam, voornaam FROM werkn ORDER BY achternaam,
voornaam OFFSET 25 ROWS
```

Geef op dat u slechts tien rijen als resultaat wilt.

```
SELECT werkn_ID, achternaam, voornaam FROM werkn ORDER BY achternaam,
voornaam OFFSET 25 ROWS FETCH FIRST 10 ROWS ONLY
```

Geef als resultaat de tien rijen en hun peerrijen (rijen die niet verschillen op basis van het element `ORDER BY`):

```
SELECT werkn_ID, achternaam, voornaam FROM werkn ORDER BY achternaam,
voornaam OFFSET 25 ROWS FETCH FIRST 10 ROWS WITH TIES
```

## Het element FOR UPDATE

Het element `FOR UPDATE` vergrendelt records voor geplaatste (positioned) updates of verwijderingen via SQL-cursors. Hiervoor gebruikt u de volgende syntaxis:

```
FOR UPDATE [OF kolomuitdrukkingen]
```

`kolomuitdrukkingen` is een lijst met veldnamen in de databasetabel die u wilt gaan bijwerken, gescheiden door een komma. `kolomuitdrukkingen` is optioneel en wordt genegeerd.

#### Voorbeeld

Geef als resultaat alle records in de werknemersdatabase waarvan het veld `SALARIS` een hogere waarde dan € 20.000 bevat.

```
SELECT * FROM werkn WHERE salaris > 20000
   FOR UPDATE OF achternaam, voornaam, salaris
```

Bij het ophalen van records wordt elke record vergrendeld. Als de record wordt bijgewerkt of verwijderd, blijft de vergrendeling behouden tot u de wijziging vastlegt. Anders wordt de vergrendeling pas opgeheven wanneer u de volgende record opvraagt.

### Voorbeelden

| Zo gebruikt u een | Voorbeeld-SQL |
|---|---|
| tekstconstante | `SELECT 'CatDog' FROM Verkopers` |
| getalconstante | `SELECT 999 FROM Verkopers` |
| datumconstante | `SELECT DATE '2021-06-05' FROM Verkopers` |
| tijdconstante | `SELECT TIME '02:49:03' FROM Verkopers` |
| tijdstempelconstante | `SELECT TIMESTAMP '2021-06-05 02:49:03' FROM Verkopers` |
| tekstkolom | `SELECT Naam_bedrijf FROM Verkoopgegevens`<br>`SELECT DISTINCT Naam_bedrijf FROM Verkoopgegevens` |
| numerieke kolom | `SELECT Bedrag FROM Verkoopgegevens`<br>`SELECT DISTINCT Bedrag FROM Verkoopgegevens` |
| datumkolom | `SELECT Verkoopdatum FROM Verkoopgegevens`<br>`SELECT DISTINCT Verkoopdatum FROM Verkoopgegevens` |
| tijdkolom | `SELECT Verkooptijdstip FROM Verkoopgegevens`<br>`SELECT DISTINCT Verkooptijdstip FROM Verkoopgegevens` |
| tijdstempelkolom | `SELECT Tijdstempel_Verkocht FROM Verkoopgegevens`<br>`SELECT DISTINCT Tijdstempel_Verkocht FROM Verkoopgegevens` |
| BLOB[a]-kolom | `SELECT Bedrijfsbrochures FROM Verkoopgegevens`<br>`SELECT GETAS (Bedrijfslogo, 'JPEG') FROM Verkoopgegevens` |
| Jokerteken * | `SELECT * FROM Verkopers`<br>`SELECT DISTINCT * FROM Verkopers` |

a. Een BLOB (Binary Large Object) is een bestandscontainerveld in een FileMaker Pro-database.

### Opmerkingen bij de voorbeelden

Een `kolom` is een verwijzing naar een veld in het FileMaker Pro-databasebestand. (Het veld kan vele verschillende waarden bevatten.)

Het jokerteken asterisk (*) staat voor “alles”. In het voorbeeld `SELECT * FROM Verkopers`, is het resultaat alle kolommen in de tabel `Verkopers`. In het voorbeeld `SELECT DISTINCT * FROM Verkopers`, is het resultaat alle unieke rijen in de tabel `Verkopers` (geen duplicaten).

- FileMaker-software slaat geen gegevens op voor lege tekenreeksen. De volgende opvragen zullen bijgevolg geen records als resultaat opleveren:

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

- Als u `SELECT` met binaire gegevens gebruikt, moet u de functie `GetAs()` gebruiken om de stroom die als resultaat moet worden gegeven, op te geven. Lees het volgende gedeelte "De inhoud van een containerveld ophalen: de functie CAST() en GetAs()," voor meer informatie.

### De inhoud van een containerveld ophalen: de functie CAST() en GetAs()

U kunt bestandsverwijzingsinformatie, binaire gegevens of gegevens van een specifiek bestandstype uit een containerveld ophalen.

- Als u bestandsverwijzingsinformatie, zoals het pad naar een bestand, afbeelding of QuickTime-film uit een containerveld wilt ophalen, gebruikt u de functie `CAST()` met een `SELECT`-instructie.
- Als er bestandsgegevens of binaire JPEG-gegevens bestaan, haalt de `SELECT`-instructie met `GetAS(veldnaam, 'JPEG')` de gegevens in binaire vorm op; anders geeft de `SELECT`-instructie met veldnaam het resultaat `NULL`.

#### Voorbeeld

Gebruik de functie `CAST()` met een `SELECT`-instructie om bestandsverwijzingsinformatie op te halen.

```
SELECT CAST (Bedrijfsbrochures AS VARCHAR) FROM Verkoopgegevens
```

Als u in dit voorbeeld:

- in het containerveld een bestand zou invoegen met behulp van FileMaker Pro, maar alleen een verwijzing naar het bestand zou opslaan, haalt de `SELECT`-instructie de bestandsverwijzingsinformatie op als type `SQL_VARCHAR`.
- in het containerveld de inhoud van een bestand zou invoegen met behulp van FileMaker Pro, haalt de `SELECT`-instructie de naam van het bestand op.
- in het containerveld een bestand uit een andere toepassing zou importeren, geeft de `SELECT`-instructie '?' weer (het bestand verschijnt als **Untitled.dat** (Naamloos.dat) in FileMaker Pro).

U kunt de `SELECT`-instructie met de functie `GetAs()` gebruiken om de gegevens in binaire vorm op te halen op de volgende manieren:

- Wanneer u de functie `GetAs()` met de optie `DEFAULT` gebruikt, haalt u de masterstroom voor de container op zonder het stroomtype uitdrukkelijk te moeten definiëren.

  #### Voorbeeld

  ```
  SELECT GetAs (Bedrijfsbrochures, DEFAULT) FROM Verkoopgegevens
  ```

- Als u een afzonderlijke stroomtype uit een container wilt ophalen, gebruikt u de functie `GetAs()` met het bestandstype op basis van de manier waarop de gegevens in het containerveld in FileMaker Pro zijn ingevoegd.

  #### Voorbeeld

  Als de gegevens zijn ingevoegd met behulp van de opdracht **Invoegen** > **Bestand**, geeft u `'FILE'` op in de functie `GetAs()`.

  ```
  SELECT GetAs (Bedrijfsbrochures, 'FILE') FROM Verkoopgegevens
  ```

  #### Voorbeeld

  Als de gegevens zijn ingevoegd met behulp van de opdracht **Invoegen** > **Afbeeldingen**, met slepen en neerzetten of door ze vanaf het klembord te plakken, geeft u een van de bestandstypen op die in de volgende tabel zijn vermeld, zoals `'JPEG'`.

  ```
  SELECT GetAs (Bedrijfslogo, 'JPEG') FROM Bedrijf_Pictogrammen
  ```

| Bestandstype | Beschrijving |
|---|---|
| 'GIFf' | Graphics Interchange Format |
| 'JPEG' | Fotografische afbeeldingen |
| 'TIFF' | Rasterbestandsindeling voor digitale afbeeldingen |
| 'PDF' | Portable Document Format |
| 'PNGf' | Bitmapafbeeldingsindeling |

## De instructie DELETE

Gebruik de instructie `DELETE` om records uit een databasetabel te verwijderen. De instructie `DELETE` is als volgt gestructureerd:

```
DELETE FROM tabelnaam [ WHERE { voorwaarden } ]
```

Opmerking Het element `WHERE` bepaalt welke records moeten worden verwijderd. Als u het element `WHERE` niet toevoegt, worden alle records in de tabel verwijderd (maar blijft de tabel behouden).

### Voorbeeld

Verwijder een record uit de tabel `werkn`.

```
DELETE FROM werkn WHERE werkn_id = 'E10001'
```

Elke `DELETE`-instructie verwijdert elke record die aan de voorwaarden van het `WHERE`-element beantwoordt. In dit geval wordt elke record met de werknemer-ID `E10001` verwijderd. Aangezien werknemer-ID's uniek zijn in de tabel Werknemer, wordt er slechts één record verwijderd.

## De instructie INSERT

Gebruik de instructie `INSERT` om records in een databasetabel te maken. U kunt een van de volgende waarden opgeven:

- Een lijst met waarden die als een nieuwe record moeten worden ingevoegd
- Een `SELECT`-instructie die gegevens uit een andere tabel kopieert die als een nieuwe reeks records moeten worden ingevoegd.

De instructie `INSERT` is als volgt gestructureerd:

```
INSERT INTO tabelnaam [(kolomnaam, ...)] VALUES (uitdr, ...)
```

`kolomnaam` is een optionele lijst met kolomnamen die de naam en volgorde van de kolommen aangeeft waarvan de waarde in het `VALUES`-element zijn opgegeven. Als u `kolomnaam` weglaat, moeten de waarde-uitdrukkingen (`uitdr`) waarden bieden voor alle kolommen die in de tabel zijn gedefinieerd en moeten ze dezelfde volgorde hebben als de kolommen die voor de tabel zijn gedefinieerd. `kolomnaam` kan ook een veldherhaling opgeven, zoals `laatsteDatums[4]`.

`uitdr` is de lijst met uitdrukkingen die de waarden voor de kolommen van de nieuwe record geeft. Doorgaans zijn de uitdrukkingen constante waarden voor de kolommen (maar ze kunnen ook een subopvraag zijn). Tekenreekswaarden moet u tussen enkele aanhalingstekens (') plaatsen. Als u een dergelijk aanhalingsteken wilt invoegen in een tekenreekswaarde die zelf al tussen enkele aanhalingstekens is geplaatst, gebruikt u twee opeenvolgende enkele aanhalingstekens (bijvoorbeeld: `'Programma''s'`).

Subopvragen moeten tussen haakjes worden geplaatst.

### Voorbeeld

Voeg een lijst met uitdrukkingen in.

```
INSERT INTO werkn (achternaam, voornaam, werkn_id, salaris, in_dienst)
   VALUES ('Smit', 'Jan', 'E22345', 27500, DATE '2019-06-05')
```

Elke INSERT-instructie voegt één record aan de databasetabel toe. In dit geval is een record toegevoegd aan de werknemersdatabasetabel werkn. Waarden worden opgegeven voor vijf kolommen. Aan de resterende kolommen in de tabel wordt een lege waarde (Null) toegewezen.

Opmerking In containervelden kunt u alleen tekst INVOEGEN, tenzij u een instructie met parameters voorbereidt en de gegevens vanuit uw toepassing stroomsgewijs overbrengt. Als u binaire gegevens wilt gebruiken, kunt u de bestandsnaam gewoon toewijzen door deze tussen enkele aanhalingstekens te plaatsen of de functie PutAs() te gebruiken. Wanneer de bestandsnaam wordt opgegeven, wordt het bestandstype afgeleid van de bestandsextensie:

```
INSERT INTO tabelnaam (containernaam) VALUES(? AS 'bestandsnaam.bestandsextensie')
```

Niet-ondersteunde bestandstypen worden ingevoegd als het type FILE.

Bij het gebruik van de functie PutAs() geeft u het type op: PutAs (kol, 'type'), waarbij de typewaarde een ondersteund bestandstype is dat is beschreven in "De inhoud van een containerveld ophalen: de functie CAST() en GetAs()" op pagina 16.

De SELECT-instructie is een opvraag die als resultaat waarden geeft voor elke kolomnaam-waarde die in de lijst met kolomnamen is opgegeven. Als u een SELECT-instructie gebruikt in plaats van een lijst met waarde-uitdrukkingen kunt u een reeks rijen uit een tabel selecteren en deze in een andere tabel invoegen met behulp van één INSERT-instructie.

### Voorbeeld

Voeg in met een SELECT-instructie.

```
INSERT INTO werkn1 (voornaam, achternaam, werkn_id, afd, salaris)
   SELECT voornaam, achternaam, werkn_id, afd, salaris FROM werkn
   WHERE afd = 'D050'
```

In dit type INSERT-instructie moet het aantal in te voegen kolommen overeenkomen met het aantal kolommen in de SELECT-instructie. De lijst met in te voegen kolommen moet overeenkomen met de kolommen in de SELECT-instructie, net zoals deze ook zou moeten overeenkomen met een lijst met waarde-uitdrukkingen in het andere type INSERT-instructie. De eerste ingevoegde kolom komt bijvoorbeeld overeen met de eerste geselecteerde kolom; de tweede ingevoegde kolom met de tweede geselecteerde kolom, enzovoort.

De grootte en het gegevenstype van deze overeenkomende kolommen moet compatibel zijn. Elke kolom in de SELECT-lijst moet een gegevenstype hebben dat het ODBC- of JDBC-clientstuurprogramma accepteert bij een normale INSERT/UPDATE van de overeenkomstige kolom in de INSERT-lijst. Waarden worden afgekapt wanneer de waarde in de kolom van de SELECT-lijst groter is dan die van de overeenkomende kolom van de INSERT-lijst.

De SELECT-instructie wordt geëvalueerd voordat waarden worden ingevoegd.

## De instructie UPDATE

Gebruik de instructie `UPDATE` om records in een databasetabel te wijzigen. De instructie `UPDATE` is als volgt gestructureerd:

```
UPDATE tabelnaam SET kolomnaam = uitdr, ... [ WHERE { voorwaarden } ]
```

`kolomnaam` is de naam van een kolom waarvan de waarden moeten worden gewijzigd. Met één instructie kunnen meerdere kolommen worden gewijzigd.

`uitdr` is de nieuwe waarde voor de kolom.

Doorgaans zijn de uitdrukkingen constante waarden voor de kolommen (maar ze kunnen ook een subopvraag zijn). Tekenreekswaarden moet u tussen enkele aanhalingstekens (') plaatsen. Als u een dergelijk aanhalingsteken wilt invoegen in een tekenreekswaarde die zelf al tussen enkele aanhalingstekens is geplaatst, gebruikt u twee opeenvolgende enkele aanhalingstekens (bijvoorbeeld: `'Programma''s'`).

Subopvragen moeten tussen haakjes worden geplaatst.

Het `WHERE`-element kan elk geldig element zijn. Dit bepaalt welke records worden bijgewerkt.

### Voorbeeld

`UPDATE`-instructie voor de tabel `werkn`.

```
UPDATE werkn SET salaris=32000, vrijstelling=1 WHERE werkn_id = 'E10001'
```

De `UPDATE`-instructie wijzigt elke record die aan de voorwaarden van het `WHERE`-element beantwoordt. In dit geval worden het salaris en de status Vrijstelling gewijzigd voor alle werknemers met de werknemer-id `E10001`. Aangezien werknemer-id's uniek zijn in de tabel Werknemer, wordt er slechts één record bijgewerkt.

### Voorbeeld

`UPDATE`-instructie voor de tabel `werkn` met een subopvraag.

```
UPDATE werkn SET salaris = (SELECT gem(salaris) FROM werkn) WHERE werkn_id
= 'E10001'
```

In dit geval wordt het salaris gewijzigd in het gemiddelde salaris van het bedrijf voor de werknemer met werknemer-ID `E10001`.

Opmerking In containervelden kunt u alleen tekst `BIJWERKEN`, tenzij u een instructie met parameters voorbereidt en de gegevens vanuit uw toepassing stroomsgewijs overbrengt. Als u binaire gegevens wilt gebruiken, kunt u de bestandsnaam gewoon toewijzen door deze tussen enkele aanhalingstekens te plaatsen of de functie `PutAs()` te gebruiken. Wanneer de bestandsnaam wordt opgegeven, wordt het bestandstype afgeleid van de bestandsextensie:

```
UPDATE tabelnaam SET (containernaam) = ? AS 'bestandsnaam.bestandsextensie'
```

Niet-ondersteunde bestandstypen worden ingevoegd als het type FILE.

Bij het gebruik van de functie `PutAs()` geeft u het type op: `PutAs (kol, 'type')`, waarbij de typewaarde een ondersteund bestandstype is dat is beschreven in "De inhoud van een containerveld ophalen: de functie CAST() en GetAs()" op pagina 16.

## De instructie CREATE TABLE

Gebruik de instructie `CREATE TABLE` om een tabel in een databasebestand te maken. De instructie `CREATE TABLE` is als volgt gestructureerd:

```
CREATE TABLE tabelnaam ( lijst_tabelelement [, lijst_tabelelement...] )
```

In de instructie geeft u de naam en het gegevenstype van elke kolom op.

- `tabelnaam` is de naam van de tabel. `tabelnaam` mag maximaal 100 tekens lang zijn. Er mag niet al een tabel met dezelfde naam zijn gedefinieerd. De tabelnaam moet beginnen met een letter. Als de tabelnaam begint met een ander teken dan een letter, plaatst u het tussen dubbele aanhalingstekens (ID tussen aanhalingstekens).
- De syntaxis voor `lijst_tabelelement` is:

  ```
  veldnaam veldtype [[herhalingen]]
  [DEFAULT expr] [UNIQUE | NOT NULL | PRIMARY KEY | GLOBAL]
  [EXTERNAL tekenreeks_relatief_pad [SECURE | OPEN tekenreeks_berekeningspad]]
  ```

  - `naam_veld` is de naam van het veld. Veldnamen moeten uniek zijn. Veldnamen beginnen met een letter. Als de veldnaam begint met een ander teken dan een letter, plaatst u het tussen dubbele aanhalingstekens (ID tussen aanhalingstekens).

    **Voorbeeld**

    De instructie `CREATE TABLE` voor het veld `_ACHTERNAAM` is:

    ```
    CREATE TABLE "_WERKNEMER" (ID INT PRIMARY KEY, "_VOORNAAM" VARCHAR(20),
    "_ACHTERNAAM" VARCHAR(20))
    ```

  - Voor de instructie `CREATE TABLE herhalingen`, geeft u een veldherhaling op door na het veldtype een getal van 1 tot en met 32.000 tussen haakjes te plaatsen.

    **Voorbeeld**

    ```
    WERKNEMER_ID INT[4]
    ACHTERNAAM VARCHAR(20)[4]
    ```

  - `veldtype` kan een van de volgende typen zijn: `NUMERIC`, `DECIMAL`, `INT`, `DATE`, `TIME`, `TIMESTAMP`, `VARCHAR`, `CHARACTER VARYING`, `BLOB`, `VARBINARY`, `LONGVARBINARY`, of `BINARY VARYING`. Voor `NUMERIC` en `DECIMAL` kunt u de precisie en schaal opgeven. Bijvoorbeeld: `DECIMAL(10,0)`. Voor `TIME` en `TIMESTAMP` kunt u de precisie opgeven. Bijvoorbeeld: `TIMESTAMP(6)`. Voor `VARCHAR` en `CHARACTER VARYING` kunt u de lengte van de tekenreeks opgeven.

    **Voorbeeld**

    ```
    VARCHAR(255)
    ```

  - Met het trefwoord `DEFAULT` kunt u een standaardwaarde voor een kolom instellen. Voor `uitdr` kunt u een constante waarde of uitdrukking gebruiken. Toegestane uitdrukkingen zijn `USER`, `USERNAME`, `CURRENT_USER`, `CURRENT_DATE`, `CURDATE`, `CURRENT_TIME`, `CURTIME`, `CURRENT_TIMESTAMP`, `CURTIMESTAMP` en `NULL`.

- Als u een kolom definieert als `UNIQUE`, wordt automatisch de bevestigingsoptie **Uniek** geselecteerd voor het overeenkomende veld in het FileMaker Pro-databasebestand.
- Als u een kolom definieert als `NOT NULL`, wordt automatisch de bevestigingsoptie **Niet leeg** geselecteerd voor het overeenkomende veld in het FileMaker Pro-databasebestand. In FileMaker Pro wordt het veld op het tabblad **Velden** van het dialoogvenster Database beheren gemarkeerd als een **Vereiste waarde**.
- Als u een kolom als een containerveld wilt definiëren, gebruikt u `BLOB`, `VARBINARY` of `BINARY VARYING` voor het `type_veld`.
- Als u een kolom wilt definiëren als een containerveld waarin externe gegevens worden opgeslagen, gebruikt u het trefwoord `EXTERNAL`. De `tekenreeks_relatief_pad` definieert de map waarin de gegevens extern zijn opgeslagen, in verhouding tot de locatie van de FileMaker Pro-database. Dit pad moet worden opgegeven als de basismap in het FileMaker Pro-dialoogvenster Containers beheren. U dient `SECURE` op te geven voor veilige opslag of `OPEN` voor open opslag. Als u open opslag gebruikt, is `tekenreeks_berekeningspad` de map in de map `tekenreeks_relatief_pad` waarin containerobjecten moeten worden opgeslagen. Het pad moet slashes (/) in de mapnaam gebruiken.

### Voorbeelden

| Zo gebruikt u een | Voorbeeld-SQL |
|---|---|
| tekstkolom | `CREATE TABLE T1 (C1 VARCHAR, C2 VARCHAR (50), C3 VARCHAR (1001), C4 VARCHAR (500276))` |
| tekstkolom, `NOT NULL` | `CREATE TABLE T1NN (C1 VARCHAR NOT NULL, C2 VARCHAR (50) NOT NULL, C3 VARCHAR (1001) NOT NULL, C4 VARCHAR (500276) NOT NULL)` |
| numerieke kolom | `CREATE TABLE T2 (C1 DECIMAL, C2 DECIMAL (10,0), C3 DECIMAL (7539,2), C4 DECIMAL (497925,301))` |
| datumkolom | `CREATE TABLE T3 (C1 DATE, C2 DATE, C3 DATE, C4 DATE)` |
| tijdkolom | `CREATE TABLE T4 (C1 TIME, C2 TIME, C3 TIME, C4 TIME)` |
| tijdstempelkolom | `CREATE TABLE T5 (C1 TIMESTAMP, C2 TIMESTAMP, C3 TIMESTAMP, C4 TIMESTAMP)` |
| kolom voor containerveld | `CREATE TABLE T6 (C1 BLOB, C2 BLOB, C3 BLOB, C4 BLOB)` |
| kolom voor containerveld voor externe opslag | `CREATE TABLE T7 (C1 BLOB EXTERNAL 'Bestanden/MijnDatabase/' SECURE)`<br>`CREATE TABLE T8 (C1 BLOB EXTERNAL 'Bestanden/MijnDatabase/' OPEN 'Objecten')` |

## De instructie TRUNCATE TABLE

Gebruik de instructie `TRUNCATE TABLE` om snel alle records in de opgegeven tabel te verwijderen, waardoor de tabel geen gegevens meer bevat.

```
TRUNCATE TABLE tabelnaam
```

U kunt geen `WHERE`-element bij de instructie `TRUNCATE TABLE` opgeven. De instructie `TRUNCATE TABLE` verwijdert alle records.

Alleen de records in de tabel opgegeven door `tabelnaam` worden verwijderd. Dit is niet van invloed op records uit gerelateerde tabellen.

De instructie `TRUNCATE TABLE` moet alle records in de tabel kunnen vergrendelen om de recordgegevens te verwijderen. Als een record in de tabel is vergrendeld door een andere gebruiker, geeft FileMaker-software foutcode 301 (“Record wordt al door een andere gebruiker gebruikt”).

## De instructie ALTER TABLE

Gebruik de instructie `ALTER TABLE` om de structuur van een bestaande tabel in een databasebestand te wijzigen. In elke instructie kunt u slechts één kolom wijzigen. De instructie `ALTER TABLE` is als volgt gestructureerd:

```
ALTER TABLE tabelnaam ADD [COLUMN] kolomdefinitie
ALTER TABLE tabelnaam DROP [COLUMN] niet-gekwalificeerde_kolomnaam
ALTER TABLE tabelnaam ALTER [COLUMN] kolomdefinitie SET DEFAULT uitdr
ALTER TABLE tabelnaam ALTER [COLUMN] kolomdefinitie DROP DEFAULT
```

Voordat u de instructie `ALTER TABLE` gebruikt, moet u wel de tabelstructuur kennen en weten hoe u deze wilt wijzigen.

### Voorbeelden

| Om dit te doen | Voorbeeld-SQL |
|---|---|
| kolommen toevoegen | `ALTER TABLE Verkopers ADD C1 VARCHAR` |
| kolommen verwijderen | `ALTER TABLE Verkopers DROP C1` |
| de standaardwaarde voor een kolom instellen | `ALTER TABLE Verkopers ALTER Bedrijf SET DEFAULT 'Claris'` |
| de standaardwaarde voor een kolom verwijderen | `ALTER TABLE Verkopers ALTER Bedrijf DROP DEFAULT` |

Opmerking `SET DEFAULT` en `DROP DEFAULT` hebben geen invloed op bestaande rijen in de tabel, maar wijzigen de standaardwaarde voor rijen die vervolgens aan de tabel worden toegevoegd.

## De instructie CREATE INDEX

Gebruik de instructie `CREATE INDEX` om zoekacties in uw databasebestand te versnellen. De instructie `CREATE INDEX` is als volgt gestructureerd:

```
CREATE INDEX ON tabelnaam.kolomnaam
CREATE INDEX ON tabelnaam (kolomnaam)
```

`CREATE INDEX` wordt ondersteund voor één kolom (indices van meerdere kolommen worden niet ondersteund). Indices zijn niet toegestaan op kolommen die overeenstemmen met containerveldtypen, resumévelden, velden met de optie globale opslag, of niet-opgeslagen berekeningvelden in een FileMaker Pro-databasebestand.

Als u voor een tekstkolom een index maakt, wordt in **Indexeren** automatisch de opslagoptie **Minimaal** geselecteerd voor het overeenkomstige veld in het FileMaker Pro-databasebestand. Als u voor een niet-tekstkolom (of een kolom die als Japanse tekst is opgemaakt) een index maakt, wordt automatisch de opslagoptie **Alles** in **Indexeren** geselecteerd voor het overeenkomstige veld in het FileMaker Pro-databasebestand.

Als u voor een willekeurige kolom een index maakt, wordt in **Indexeren** automatisch de opslagoptie **Automatisch indices maken indien nodig** geselecteerd voor het overeenkomstige veld in het FileMaker Pro-databasebestand.

FileMaker-software maakt automatisch indices indien nodig. Door `CREATE INDEX` te gebruiken, wordt de index onmiddellijk opnieuw opgebouwd in plaats van op verzoek.

**Voorbeeld**

```
CREATE INDEX ON Verkopers.Verkoper_id
```

### De instructie DROP INDEX

Gebruik de instructie `DROP INDEX` om een index uit een databasebestand te verwijderen. De instructie `DROP INDEX` is als volgt gestructureerd:

```
DROP INDEX ON tabelnaam.kolomnaam
DROP INDEX ON tabelnaam (kolomnaam)
```

Verwijder een index wanneer uw databasebestand te groot is of als u in opvragen niet vaak een veld gebruikt.

Als uw opvragen slechte resultaten geven en u werkt met een extreem groot FileMaker Pro-databasebestand met een groot aantal geïndexeerde tekstvelden, kunt u overwegen om de index van bepaalde velden weg te laten. U kunt ook overwegen om de index weg te laten van velden die u zelden gebruikt in `SELECT`-instructies.

Als u voor een willekeurige kolom een index weglaat, wordt in **Indexeren** automatisch de opslagoptie **Geen** geselecteerd en wordt de opslagoptie **Automatisch indices maken indien nodig** uitgeschakeld voor het overeenkomstige veld in het FileMaker Pro-databasebestand.

Het kenmerk `PREVENT INDEX CREATION` wordt niet ondersteund.

**Voorbeeld**

```
DROP INDEX ON Verkopers.Verkoper_id
```

## SQL-uitdrukkingen

Gebruik uitdrukkingen in `WHERE`-, `HAVING`- en `ORDER BY`-elementen van `SELECT`-instructies om gedetailleerde en geavanceerde databaseopvragen te maken. Geldige uitdrukkingselementen zijn:

- Veldnamen
- Constanten
- Exponentiële/wetenschappelijke opmaak
- Numerieke operatoren
- Operatoren voor lettertekens
- Datumoperatoren
- Relationele operatoren
- Logische operatoren
- Functies

### Veldnamen

De meest algemene uitdrukking is een eenvoudige veldnaam, zoals `berek` of `Verkoopgegevens.Factuur_ID`.

## Constanten

Constanten zijn waarden die niet wijzigen. In de uitdrukking `PRIJS * 1,05` is de waarde 1,05 een constante. U kunt ook de waarde 30 toewijzen aan de constante `Aantal_dagen_in_juni`.

Tekenreeksconstanten moet u tussen enkele aanhalingstekens (') plaatsen. Als u een dergelijk aanhalingsteken wilt invoegen in een tekenreeksconstante die zelf al tussen enkele aanhalingstekens is geplaatst, gebruikt u twee opeenvolgende enkele aanhalingstekens (bijvoorbeeld: `'Programma''s'`).

Voor ODBC- en JDBC-toepassingen accepteert FileMaker-software datum-, tijd- en tijdstempelconstanten met de ODBC/JDBC-opmaak tussen accolades ({}).

### Voorbeelden

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

FileMaker-software accepteert typeaanduidingen (`D`, `T`, `TS`) in hoofdletters of kleine letters. Na de typeaanduiding kunt u een willekeurig aantal spaties plaatsen of de spatie zelfs overslaan.

FileMaker-software accepteert ook ISO-datum- en tijdopmaak van de syntaxis SQL-92 zonder accolades.

### Voorbeelden

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

De functie ExecuteSQL van FileMaker Pro accepteert alleen ISO-datum- en tijdopmaak van de syntaxis SQL-92 zonder accolades.

| Constante | Toegestane syntaxis (voorbeelden) |
|---|---|
| Tekst | `'Parijs'` |
| Getal | `1.05` |
| Datum | `DATE '2019-06-05'`<br>`{ D '2019-06-05' }`<br>`{06/05/2019}`<br>`{06/05/19}`<br>Opmerking De syntaxis van twee jaarcijfers wordt niet ondersteund voor de ODBC/JDBC- of SQL-92-opmaak. |

| Constante | Toegestane syntaxis (voorbeelden) |
|---|---|
| Tijd | `TIME '14:35:10'`<br>`{ T'14:35:10' }`<br>`{14:35:10}` |
| Tijdstempel | `TIMESTAMP '2019-06-05 14:35:10'`<br>`{ TS '2019-06-05 14:35:10' }`<br>`{06/05/2019 14:35:10}`<br>`{06/05/19 14:35:10}`<br>Zorg ervoor dat **Restrictie gegevenstype: Jaartal in 4 cijfers** niet is geselecteerd als een bevestigingsoptie in het FileMaker Pro-databasebestand voor een veld dat deze syntaxis van 2 jaarcijfers gebruikt.<br>Opmerking De syntaxis van twee jaarcijfers wordt niet ondersteund voor de ODBC/JDBC- of SQL-92-opmaak. |

Bij het invoeren van datum- en tijdwaarden gebruikt u de taalinstellingen van het databasebestand. Als de database bijvoorbeeld in een Italiaans taalsysteem is gemaakt, gebruikt u de Italiaanse datum- en tijdopmaak.

## Exponentiële/wetenschappelijke opmaak

Getallen kunnen worden uitgedrukt met behulp van wetenschappelijke opmaak.

### Voorbeeld

```
SELECT kolom1 / 3.4E+7 FROM tabel1 WHERE berek < 3.4E-6 * kolom2
```

## Numerieke operatoren

U kunt de volgende operatoren gebruiken in getaluitdrukkingen: +, -, *, / en ^ of ** (exponentberekening).

U kunt numerieke uitdrukkingen laten voorafgaan door een monadisch plusteken (+) of minteken (-).

## Operatoren voor lettertekens

U kunt lettertekens samenvoegen. In het volgende is de `achternaam 'DIJKSTRA '` en de voornaam `'ROB '`.

| Operator | Samenvoeging | Voorbeeld | Resultaat |
|---|---|---|---|
| + | Met behoud van volgspaties | voornaam + achternaam | 'ROB DIJKSTRA ' |
| - | Volgspaties verplaatsen naar het einde | voornaam - achternaam | 'ROBDIJKSTRA ' |

## Datumoperatoren

U kunt datums wijzigen. In het volgende is `in_dienst` `DATE '2019-01-30'`.

| Operator | Effect op datum | Voorbeeld | Resultaat |
|---|---|---|---|
| + | Voeg een aantal dagen aan een datum toe | `in_dienst` + 5 | DATE '2019-02-04' |
| - | Zoek het aantal dagen tussen twee datums | `in_dienst - DATE '2019-01-01'` | `29` |
| | Trek een aantal dagen van een datum af | `in_dienst - 10` | DATE '2019-01-20' |

### Extra voorbeelden

```
SELECT Verkoopdatum, Verkoopdatum + 30 AS totaal FROM Verkoopgegevens
SELECT Verkoopdatum, Verkoopdatum - 30 AS totaal FROM Verkoopgegevens
```

## Relationele operatoren

| Operator | Betekenis |
|---|---|
| = | Gelijk aan |
| <> | Niet gelijk aan |
| > | Groter dan |
| >= | Groter dan of gelijk aan |
| < | Kleiner dan |
| <= | Kleiner dan of gelijk aan |
| LIKE | Komt overeen met een patroon |
| NOT LIKE | Komt niet overeen met een patroon |
| IS NULL | Gelijk aan Null |
| IS NOT NULL | Niet gelijk aan Null |
| BETWEEN | Reeks waarden tussen een boven- en ondergrens |
| IN | Een onderdeel van een reeks opgegeven waarden of een onderdeel van een subopvraag |
| NOT IN | Geen onderdeel van een reeks opgegeven waarden of geen onderdeel van een subopvraag |
| EXISTS | ‘True’ (Waar) als een subopvraag minimaal één record als resultaat geeft |
| ANY | Vergelijkt een waarde met elke resultaatwaarde van een subopvraag (operator moet vooraf worden gegaan door =, <>, >, >=, < of <=). =Any komt overeen met In |
| ALL | Vergelijkt een waarde met elke resultaatwaarde van een subopvraag (operator moet vooraf worden gegaan door =, <>, >, >=, < of <=) |

### Voorbeeld

```
SELECT Verkoopgegevens.Factuur_ID FROM Verkoopgegevens
   WHERE Verkoopgegevens.Verkopercode = 'SP-1'
SELECT Verkoopgegevens.Bedrag FROM Verkoopgegevens WHERE
Verkoopgegevens.Factuur_ID <> 125
SELECT Verkoopgegevens.Bedrag FROM Verkoopgegevens WHERE
Verkoopgegevens.Bedrag > 3000
SELECT Verkoopgegevens.Verkooptijdstip FROM Verkoopgegevens
   WHERE Verkoopgegevens.Verkooptijdstip < '12:00:00'
SELECT Verkoopgegevens.Bedrijfsnaam FROM Verkoopgegevens
   WHERE Verkoopgegevens.Bedrijfsnaam LIKE '%Universiteit'
SELECT Verkoopgegevens.Bedrijfsnaam FROM Verkoopgegevens
   WHERE Verkoopgegevens.Bedrijfsnaam NOT LIKE '%Universiteit'
SELECT Verkoopgegevens.Bedrag FROM Verkoopgegevens WHERE
Verkoopgegevens.Bedrag IS NULL
SELECT Verkoopgegevens.Bedrag FROM Verkoopgegevens WHERE
Verkoopgegevens.Bedrag IS NOT NULL
SELECT Verkoopgegevens.Factuur_ID FROM Verkoopgegevens
   WHERE Verkoopgegevens.Factuurnummer BETWEEN 1 AND 10
SELECT COUNT (Verkoopgegevens.Factuur_ID) AS totaal
   FROM Verkoopgegevens WHERE Verkoopgegevens.Factuur_ID IN (50,250,100)
SELECT COUNT (Verkoopgegevens.Factuur_ID) AS totaal
   FROM Verkoopgegevens WHERE Verkoopgegevens.Factuur_ID NOT IN (50,250,100)
SELECT COUNT (Verkoopgegevens.Factuur_ID) AS totaal FROM Verkoopgegevens
   WHERE Verkoopgegevens.FACTUUR_ID NOT IN (SELECT Verkoopgegevens.Factuur_ID
   FROM Verkoopgegevens WHERE Verkoopgegevens.Verkoper_ID = 'SP-4')
SELECT *
   FROM Verkoopgegevens WHERE EXISTS (SELECT Verkoopgegevens.Bedrag
   FROM Verkoopgegevens WHERE Verkoopgegevens.Verkoper_ID IS NOT NULL)
SELECT *
   FROM Verkoopgegevens WHERE Verkoopgegevens.Bedrag = ANY (SELECT
Verkoopgegevens.Bedrag
   FROM Verkoopgegevens WHERE Verkoopgegevens.Verkoper_ID = 'SP-1')
SELECT *
   FROM Verkoopgegevens WHERE Verkoopgegevens.Bedrag = ALL (SELECT
Verkoopgegevens.Bedrag
   FROM Verkoopgegevens WHERE Verkoopgegevens.Verkoper_ID IS NULL)
```

## Logische operatoren

U kunt twee of meer voorwaarden combineren. De voorwaarden moeten aan elkaar gekoppeld zijn met `AND` of `OR`, bijvoorbeeld:

```
salaris = 40000 AND vrijstelling = 1
```

De logische NOT-operator wordt gebruikt om een betekenis om te keren, bijvoorbeeld:

```
NOT (salaris = 40000 AND vrijstelling = 1)
```

### Voorbeeld

```
SELECT * FROM Verkoopgegevens WHERE Verkoopgegevens.Bedrijfsnaam
   NOT LIKE '%Universiteit' AND Verkoopgegevens.Bedrag > 3000
SELECT * FROM Verkoopgegevens WHERE (Verkoopgegevens.Bedrijfsnaam
   LIKE '%Universiteit' OR Verkoopgegevens.Bedrag > 3000)
   AND Verkoopgegevens.Verkopercode = 'SP-1'
```

## Prioriteit van operatoren

Naarmate uitdrukkingen complexer worden, wordt ook de volgorde waarin de uitdrukkingen worden geëvalueerd belangrijker. Deze tabel geeft de volgorde aan waarin de operatoren worden geëvalueerd. De operatoren op de eerste regel worden eerst geëvalueerd, enzovoort. Operatoren op dezelfde regel worden van links naar rechts in de uitdrukking geëvalueerd.

| Prioriteit | Operator |
|---|---|
| 1 | Monadisch minteken `'-'`, Monadisch plusteken `'+'` |
| 2 | ^, ** |
| 3 | *, / |
| 4 | +, - |
| 5 | =, <>, <, <=, >, >=, Like, Not Like, Is Null, Is Not Null, Between, In, Exists, Any, All |
| 6 | Not |
| 7 | AND |
| 8 | OR |

### Voorbeelden

```
WHERE salaris > 40000 OR in_dienst > (DATE '2008-01-30'} AND afd =
'D101'
```

Aangezien AND eerst wordt geëvalueerd, haalt u met deze opvraag de werknemers uit afdeling D101 op die in dienst zijn getreden na woensdag 30 januari 2008, maar ook elke werknemer die meer dan € 40.000 verdient, ongeacht de afdeling of de datum van indiensttreding.

Als u de evaluatie in een andere volgorde wilt laten uitvoeren, plaatst u de voorwaarden die eerst moeten worden geëvalueerd tussen haakjes.

```
WHERE (salaris > 40000 OR in_dienst > DATE '2008-01-30'} AND afd =
'D101'
```

Met dit voorbeeld haalt u werknemers uit de afdeling D101 op die meer dan € 40.000 verdienen of die na woensdag 30 januari 2008 in dienst zijn getreden.

# SQL-functies

Claris biedt een implementatie van de SQL-standaard voor het FileMaker Platform en ondersteunt veel functies die u in uitdrukkingen kunt gebruiken. Sommige functies geven als resultaat tekenreeksen, andere geven als resultaat getallen of data, en nog andere geven als resultaat waarden die afhangen van voorwaarden die worden voldaan door de functie-argumenten.

## Totaalfuncties

Totaalfuncties geven als resultaat één waarde uit een reeks records. U kunt een totaalfunctie gebruiken als onderdeel van een `SELECT`-instructie, of met een veldnaam (bijvoorbeeld `AVG(SALARIS)`), of in combinatie met een kolomuitdrukking (bijvoorbeeld `AVG(SALARIS * 1.07)`).

U kunt de kolomuitdrukking laten voorafgaan door de operator `DISTINCT` om dubbele waarden weg te laten.

### Voorbeeld

```
COUNT (DISTINCT achternaam)
```

In dit voorbeeld worden alleen unieke waarden voor achternaam geteld.

| Totaalfunctie | Geeft dit als resultaat |
|---|---|
| `SUM` | Het totaal van de waarden in een numerieke velduitdrukking. `SUM (SALARIS)` geeft bijvoorbeeld als resultaat de som van alle waarden van het veld Salaris. |
| `AVG` | Het gemiddelde van de waarden in een numerieke velduitdrukking. `AVG (SALARIS)` geeft bijvoorbeeld als resultaat het gemiddelde van alle waarden van het veld Salaris. |
| `COUNT` | Het aantal waarden in een willekeurige velduitdrukking. `COUNT (NAAM)` geeft bijvoorbeeld naamwaarden als resultaat. Bij het gebruik van `COUNT` met een veldnaam geeft `COUNT` als resultaat het aantal veldwaarden dat niet null is. Een speciaal voorbeeld is `COUNT(*)`. Dat geeft als resultaat het aantal records in de reeks, met inbegrip van records met 'null'-waarden. |
| `MAX` | De maximale waarde in een willekeurige velduitdrukking. `MAX(SALARIS)` geeft bijvoorbeeld als resultaat de maximale waarde in het veld Salaris. |
| `MIN` | De minimale waarde in een willekeurige velduitdrukking. `MIN(SALARIS)` geeft bijvoorbeeld als resultaat de minimale waarde in het veld Salaris. |

### Voorbeeld

```
SELECT SUM (Verkoopgegevens.Bedrag) AS totaal FROM Verkoopgegevens
SELECT AVG (Verkoopgegevens.Bedrag) AS totaal FROM Verkoopgegevens
SELECT COUNT (Verkoopgegevens.Bedrag) AS totaal FROM Verkoopgegevens
SELECT MAX (Verkoopgegevens.Bedrag) AS totaal FROM Verkoopgegevens
   WHERE Verkoopgegevens.Bedrag < 3000
SELECT MIN (Verkoopgegevens.Bedrag) AS totaal FROM Verkoopgegevens
   WHERE Verkoopgegevens.Bedrag > 3000
```

U kunt geen totaalfunctie gebruiken als een argument bij andere functies. Als u dat toch doet, geeft FileMaker-software foutcode 8309 (“Expressions involving aggregations are not supported”). De volgende instructie is bijvoorbeeld ongeldig omdat de totaalfunctie `SUM` niet kan worden gebruikt als een argument bij de functie `ROUND`:

### Voorbeeld

```
SELECT ROUND(SUM(Salaris), 0) FROM Loonlijst
```

Totaalfuncties kunnen wel functies gebruiken die getallen als argumenten geven. De volgende instructie is geldig.

### Voorbeeld

```
SELECT SUM(ROUND(Salaris, 0)) FROM Loonlijst
```

## Functies die tekenreeksen als resultaat geven

| Functies die tekenreeksen als resultaat geven | Beschrijving | Voorbeeld |
|---|---|---|
| CHR | Converteert een ASCII-code naar een tekenreeks van één teken | CHR (67) geeft als resultaat **C**. |
| CURRENT_USER | Geeft als resultaat de aanmeldings-id die op het tijdstip van aanmelding is opgegeven | |
| DAYNAME | Geeft als resultaat de naam van de dag die met een opgegeven datum overeenkomt | |
| RTRIM | Verwijdert volgspaties uit een tekenreeks | RTRIM ('  ABC ') geeft als resultaat '  ABC' |
| TRIM | Verwijdert voorloop- en volgspaties uit een tekenreeks | TRIM (' ABC ') geeft als resultaat 'ABC' |
| LTRIM | Verwijdert voorloopspaties uit een tekenreeks | LTRIM (' ABC') geeft als resultaat 'ABC' |
| UPPER | Wijzigt elke letter van een tekenreeks in een hoofdletter | UPPER ('Dijkstra') geeft als resultaat 'DIJKSTRA' |
| LOWER | Wijzigt elke letter van een tekenreeks in een kleine letter | LOWER ('Dijkstra') geeft als resultaat 'dijkstra' |
| LEFT | Geeft als resultaat de uiterst linkse tekens van een tekenreeks | LEFT ('Marion',3) geeft als resultaat 'Mar' |
| MONTHNAME | Geeft als resultaat de naam van de kalendermaand | |
| RIGHT | Geeft als resultaat de uiterst rechtse tekens van een tekenreeks | RIGHT ('Marion',4) geeft als resultaat rion |
| SUBSTR<br>SUBSTRING | Geeft als resultaat een subtekenreeks van een tekenreeks, met parameters van de tekenreeks, het eerste te extraheren teken en het aantal te extraheren tekens (optioneel) | SUBSTR('Jeroen',2,3) geeft als resultaat 'ero'<br>SUBSTR('Jeroen',2) geeft als resultaat 'eroen' |
| SPACE | Genereert een tekenreeks die uit spaties bestaat | SPACE(5) geeft als resultaat '  ' |
| STRVAL | Converteert een waarde van elk willekeurig type naar een tekenreeks | STRVAL('Woltman') geeft als resultaat 'Woltman'<br>STRVAL(5 * 3) geeft als resultaat '15'<br>STRVAL(4 = 5) geeft als resultaat 'False'<br>STRVAL(DATE '2019-12-25') geeft als resultaat '2019-12-25' |
| TIME<br>TIMEVAL | Geeft als resultaat het tijdstip van de dag als een tekenreeks | Om 21:49 geeft TIME () dit als resultaat: **21:49:00** |
| USERNAME<br>USER | Geeft als resultaat de aanmeldings-id die op het tijdstip van aanmelding is opgegeven | |

Opmerking De functie TIME() wordt niet meer gebruikt. Gebruik in plaats daarvan de SQL-standaard CURRENT_TIME.

### Voorbeeld

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

SELECT RTRIM(' ' + Verkopers.Verkoper_ID) AS totaal FROM Verkopers

SELECT TRIM(SPACE(1) + Verkopers.Verkoper_ID) AS totaal FROM Verkopers

SELECT LTRIM(' ' + Verkopers.Verkoper_ID) AS totaal FROM Verkopers

SELECT UPPER(Verkopers.Verkoper) AS totaal FROM Verkopers

SELECT LOWER(Verkopers.Verkoper) AS totaal FROM Verkopers

SELECT LEFT(Verkopers.Verkoper, 5) AS totaal FROM Verkopers

SELECT RIGHT(Verkopers.Verkoper, 7) AS totaal FROM Verkopers

SELECT SUBSTR(Verkopers.Verkoper_ID, 2, 2) + SUBSTR(Verkopers.Verkoper_ID,
4, 2) AS totaal FROM Verkopers

SELECT SUBSTR(Verkopers.Verkoper_ID, 2) + SUBSTR(Verkopers.Verkoper_ID, 4)
AS totaal FROM Verkopers

SELECT SPACE(2) + Verkopers.Verkoper_ID AS Verkoper_ID FROM Verkopers

SELECT STRVAL('60506') AS totaal FROM Verkoopgegevens WHERE
Verkoopgegevens.Factuur = 1
```

## Functies die getallen als resultaat geven

| Functies die getallen als resultaat geven | Beschrijving | Voorbeeld |
|---|---|---|
| ABS | Geeft als resultaat de absolute waarde van de numerieke uitdrukking | |
| ATAN | Geeft als resultaat de arctangens van het argument als een hoek, uitgedrukt in radialen | |
| ATAN2 | Geeft als resultaat de arctangens van de x- en y-coördinaten als een hoek, uitgedrukt in radialen | |
| CEIL<br>CEILING | Geeft als resultaat de kleinste geheel-getalwaarde die groter is dan of gelijk is aan het argument | |
| DEG<br>DEGREES | Geeft als resultaat het aantal graden van het argument, als een hoek, uitgedrukt in radialen | |
| DAY | Geeft als resultaat het daggedeelte van een datum | `DAY(DATE '2019-01-30')` geeft als resultaat **30** |
| DAYOFWEEK | Geeft als resultaat de dag van de week (1-7) van een datumuitdrukking | `DAYOFWEEK(DATE '2004-05-01')` geeft als resultaat **7** |
| MOD | Deelt twee getallen en geeft als resultaat de rest van de deling | `MOD(10,3)` geeft als resultaat **1** |
| EXP | Geeft als resultaat een waarde die de basis is van de natuurlijke logaritme (e) tot de macht van een door het argument opgegeven getal | |
| FLOOR | Geeft als resultaat de grootste geheel-getalwaarde die kleiner is dan of gelijk is aan het argument | |
| HOUR | Geeft als resultaat het urengedeelte van een tijdwaarde | |
| INT | Geeft als resultaat het 'geheel getal'-gedeelte van een getal | `INT(6.4321)` geeft als resultaat **6** |
| LENGTH | Geeft als resultaat de lengte van een tekenreeks | `LENGTH('ABC')` geeft als resultaat **3** |
| MONTH | Geeft als resultaat het maandgedeelte van een datum | `MONTH(DATE '2019-01-30')` geeft als resultaat **1** |
| LN | Geeft als resultaat de natuurlijke logaritme van het argument | |
| LOG | Geeft als resultaat de algemene logaritme van het argument | |
| MAX | Geeft als resultaat het grootste van twee getallen | `MAX(66,89)` geeft als resultaat **89** |
| MIN | Geeft als resultaat het kleinste van twee getallen | `MIN(66,89)` geeft als resultaat **66** |
| MINUTE | Geeft als resultaat het minutengedeelte van een tijdwaarde | |
| NUMVAL | Converteert een tekenreeks naar een getal. De functie werkt niet als de tekenreeks geen geldig getal is. | `NUMVAL ('123')` geeft als resultaat **123** |
| PI | Geeft als resultaat de constante waarde van de wiskundige constante pi | |
| RADIANS | Geeft als resultaat het aantal radialen voor een argument dat in graden is uitgedrukt | |

| Functies die getallen als resultaat geven | Beschrijving | Voorbeeld |
|---|---|---|
| ROUND | Rondt een getal af | ROUND(123.456,0) geeft als resultaat **123**<br>ROUND(123.456,2) geeft als resultaat **123,46**<br>ROUND(123.456,-2) geeft als resultaat **100** |
| SECOND | Geeft als resultaat het secondengedeelte van een tijdwaarde | |
| SIGN | Een indicator van het teken van het argument: -1 voor negatief, 0 voor 0 en 1 voor positief | |
| SIN | Geeft als resultaat de sinus van een argument | |
| SQRT | Geeft als resultaat de vierkantswortel van een argument | |
| TAN | Geeft als resultaat de tangens van een argument | |
| YEAR | Geeft als resultaat het jaargedeelte van een datum | YEAR(DATE '2019-01-30') geeft als resultaat **2019** |

## Functies die datums als resultaat geven

| Functies die datums als resultaat geven | Beschrijving | Voorbeeld |
|---|---|---|
| CURDATE<br>CURRENT_DATE | Geeft als resultaat de huidige datum | |
| CURTIME<br>CURRENT_TIME | Geeft als resultaat de huidige tijd | |
| CURTIMESTAMP<br>CURRENT_TIMESTAMP | Geeft als resultaat de huidige tijdstempelwaarde | |
| TIMESTAMPVAL | Converteert een tekenreeks naar een tijdstempel | TIMESTAMPVAL('2019-01-30 14:00:00') geeft de tijdstempelwaarde |
| DATE<br>TODAY | Geeft als resultaat de huidige datum | Als het vandaag 21-11-2019 is, geeft DATE() als resultaat **{2019-11-21}** |
| DATEVAL | Converteert een tekenreeks naar een datum | DATEVAL ('2019-01-30') geeft als resultaat **2019-01-30** |

Opmerking De functie DATE() wordt niet meer gebruikt. Gebruik in plaats daarvan de SQL-standaard CURRENT_DATE.

## Voorwaardelijke functies

| Voorwaardelijke functies | Beschrijving | Voorbeeld |
|---|---|---|
| CASE WHEN | **Eenvoudige CASE-syntaxis**<br><br>Vergelijkt de waarde van *invoer_uitdr* met de waarden van de argumenten *waarde_uitdr* om het resultaat te bepalen.<br><br>CASE *invoer_uitdr*<br>{WHEN *waarde_uitdr* THEN *resultaat*...}<br>[ELSE *resultaat*]<br>END | SELECT<br>Factuur_ID,<br>CASE Naam_bedrijf<br>WHEN 'Export VK' THEN 'Export VK gevonden'<br>WHEN 'Leveranciers huismeubilair' THEN 'Leveranciers huismeubilair gevonden'<br>ELSE 'Geen export VK noch Leveranciers huismeubilair'<br>END,<br>Verkoper_ID<br>FROM<br>Verkoopgegevens |
| | **Gezochte CASE-syntaxis**<br><br>Geeft een resultaat naargelang de opgegeven voorwaarde door een WHEN-uitdrukkingen waar is.<br><br>CASE<br>{WHEN *Booleaanse_uitdr* THEN *resultaat...*}<br>[ELSE *resultaat*]<br>END | SELECT<br>Factuur_ID,<br>Bedrag,<br>CASE<br>WHEN Bedrag > 3000 THEN 'Hoger dan 3000'<br>WHEN Bedrag < 1000 THEN 'Lager dan 3000'<br>ELSE 'Tussen 1000 en 3000'<br>END,<br>Verkoper_ID<br>FROM<br>Verkoopgegevens |
| COALESCE | Geeft de eerste waarde die niet NULL is. | SELECT<br>Verkoper_ID,<br>COALESCE(Verkoopmanager, Verkoper)<br>FROM<br>Verkopers |
| NULLIF | Vergelijkt twee waarden en geeft NULL als resultaat als de twee waarden gelijk zijn; anders wordt de eerste waarde als resultaat gegeven. | SELECT<br>Factuur_ID,<br>NULLIF(Bedrag, -1),<br>Verkoper_ID<br>FROM<br>Verkoopgegevens |

# FileMaker-systeemobjecten

De FileMaker Pro-databasebestanden bevatten de volgende systeemobjecten waar u met SQL-opvragen toegang toe hebt.

## FileMaker-systeemtabellen

Elke FileMaker Pro-databasebestand bevat twee systeemtabellen: FileMaker_Tables en FileMaker_Fields. Bij ODBC-toepassingen worden deze tabellen toegevoegd aan de informatie die door de catalogusfunctie SQLTables wordt gegeven. Bij JDBC-toepassingen worden deze tabellen toegevoegd aan de informatie die door de DatabaseMetaData-methode getTables wordt gegeven. De tabellen kunnen ook in ExecuteSQL-functies worden gebruikt.

### FileMaker_Tables

De tabel FileMaker_Tables bevat informatie over de databasetabellen die in het FileMaker Pro-bestand zijn gedefinieerd.

De tabel FileMaker_Tables bevat een rij voor elke tabelvermelding in de relatiegrafiek met de volgende kolommen:

- TableName - De naam van de tabelvermelding.
- TableId - De unieke ID voor de tabelvermelding.
- BaseTableName - De naam van de basistabel vanwaar de tabelvermelding is gemaakt.
- BaseFileName - De FileMaker Pro-bestandsnaam voor het databasebestand dat de basistabel bevat.
- ModCount - Het totale aantal wijzigingen aan de definitie van deze tabel.

**Voorbeeld**

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

### De tabel FileMaker_Fields

De tabel FileMaker_Fields bevat informatie over de velden die in het FileMaker Pro-bestand zijn gedefinieerd.

De tabel FileMaker_Fields bevat de volgende kolommen:

- TableName - De naam van de tabel dat het veld bevat.
- FieldName - De naam van het veld.
- FieldType - Het SQL-gegevenstype van het veld.
- FieldId - De unieke ID voor het veld.
- FieldClass - Een van drie waarden: Resumé, voor resumévelden; Berekening, voor berekende resultaten; of Normaal.
- FieldReps - Het aantal herhalingen van het veld.
- ModCount - Het totale aantal wijzigingen aan de definitie van deze tabel.

**Voorbeeld**

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

## FileMaker-systeemkolommen

FileMaker-software voegt systeemkolommen (velden) toe aan alle rijen (records) in alle tabellen die in het FileMaker Pro-bestand zijn gedefinieerd. Bij ODBC-toepassingen worden deze kolommen toegevoegd aan de informatie die door de catalogusfunctie SQLSpecialColumns wordt gegeven. Bij JDBC-toepassingen worden deze kolommen toegevoegd aan de informatie die door de DatabaseMetaData-methode getVersionColumns wordt gegeven. De kolommen kunnen ook in ExecuteSQL-functies worden gebruikt.

### De kolom ROWID

De systeemkolom `ROWID` bevat de unieke ID van de record. Dit is dezelfde waarde die de functie Get (RecordID) van FileMaker Pro als resultaat geeft.

### De kolom ROWMODID

De systeemkolom `ROWMODID` bevat het totale aantal wijzigingen aan de huidige record. Dit is dezelfde waarde die de functie Get (TellingRecordwijzigingen) van FileMaker Pro als resultaat geeft.

#### Voorbeeld

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

# Gereserveerde SQL-trefwoorden

De volgende sectie bevat gereserveerde trefwoorden die niet mogen worden gebruikt als naam voor kolommen, tabellen, aliassen of andere zelf gedefinieerde objecten. Als syntaxisfouten optreden, worden die mogelijk veroorzaakt door het gebruik van een van deze gereserveerde woorden. Indien u een van deze trefwoorden wilt gebruiken, dient u dit tussen aanhalingstekens te plaatsen om te voorkomen dat het als een trefwoord wordt verwerkt.

### Voorbeeld

Gebruik het trefwoord `DEC` als naam voor een gegevenselement.

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

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

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

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

# Index

## J



## K



## L



## M



## N



## O



## P



## R



## S

## T



## U



## V



## W



## Y