Overzicht van de taal SQL

In dit hoofdstuk is een groot deel van de mogelijkheden van SQL voor selectiequery's besproken. Deze paragraaf bevat een korte, meer formele beschrijving van query's in SQL. Het is een samenvatting van het voorgaande.

De basisstructuur van een query is:

```
SELECT <kolommen en/of functies>

FROM <tabellen>

[ WHERE <voorwaarden> ]

[ ORDER BY <sorteer-kolommen> ]

[ GROUP BY <groepeer-kolommen> [ HAVING <groep-voorwaarden> ] ]
```

De eerste twee regels zijn verplicht. De andere hoeven niet voor te komen, dit wordt aangegeven door de blokhaken.

SELECT

Kolommen	Achter SELECT geef je op welke kolommen de resultaattabel van de query moet hebben. Zie: SQL: Kolommen uit een tabel tonen
Kolomnamen	Je duidt kolommen aan met hun naam. Kolomnamen worden gescheiden door komma's. Wanneer er verwarring mogelijk is doordat er kolommen met dezelfde naam voorkomen (koppelingen), geef je kolommen aan met *<tabelnaam>.<kolomnaam>* (punt ertussen). Zie: SQL: Combinaties van tabellen maken
Alle kolommen	Alle kolommen uit een tabel krijg je met * of *<tabelnaam>.** Zie: SQL: Kolommen uit een tabel tonen
Functies	Je kunt functies over een hele kolom laten uitrekenen: Aantal rijen: *COUNT(*)* Totaal: *SUM(<kolomnaam>)* Gemiddelde: *AVG(<kolomnaam>)* Maximum: *MAX(<kolomnaam>)* Minimum: *MIN(<kolomnaam>)* Meestal selecteer je of functies, of kolommen, niet door elkaar. Uitzondering is het groeperen, zie daar. Zie: SQL: Functies in een query
Andere namen voor	Je kunt een kolom een nieuwe naam geven door er *AS <nieuwenaam>*

kolommen	achter te zetten. Dit is vooral handig bij kolommen met de resultaten van berekende functies Zie: SQL: Functies in een query
DISTINCT	Door dit sleutelwoord direct achter SELECT te zetten, zullen er in het resultaat geen dubbele rijen verschijnen. Zie: SQL: Alleen verschillende rijen tonen

FROM

Brontabellen	Hier geef je de tabel of tabellen op waaruit de gebruikte gegevens gehaald moeten worden. Tussen de tabelnamen moeten komma's. Zie: SQL: Kolommen uit een tabel tonen
Pseudoniemen	Wanneer je direct achter een tabelnaam een afkorting van de tabelnaam zet, kun je die afkorting in de rest van de query gebruiken in plaats van de tabelnaam zelf. Zie: SQL: Koppelen met voorwaarden
Koppelingen	Wanneer je meer dan één tabel gebruikt, maakt het systeem een koppeling tussen de tabellen. Dit betekent dat alle mogelijke combinaties van rijen gemaakt zullen worden. Meestal is dit niet gewenst, bij WHERE kun je dan door middel van een koppelvoorwaarde aangeven welke combinaties gemaakt moeten worden. Zie: SQL: Koppelen met voorwaarden

WHERE

Selectievoorwaarden	Achter WHERE geef je met behulp van een voorwaarde aan welke rijen uit de brontabellen geselecteerd moeten worden. Zie: SQL: Voorwaarden in een query
Koppelvoorwaarden	Sommige voorwaarden dienen om aan te geven welke combinaties bij een koppeling gemaakt moeten worden. Meestal bestaat zo'n voorwaarde uit het gelijk stellen van kolommen die in beide tabellen voorkomen (bijvoorbeeld: *leerlingen.llnr = uitleningen.llnr*). Zie: SQL: Koppelen met voorwaarden
AND, OR, NOT	Afzonderlijke voorwaarden verbind je met AND of OR. In een reeks van AND en OR wordt de AND het eerst gedaan. Gebruik waar nodig haakjes. Met NOT kun je een voorwaarde omkeren. Zie: SQL: Samengestelde voorwaarden in een query en SQL: Voorwaarden omdraaien met NOT

Waarden	Zet enkele aanhalingstekens '...' om tekstconstanten of datums. Getallen kun je zonder aanhalingstekens invoeren. NB: Notatie van datums verschilt per systeem. Zie: SQL: Voorwaarden in een query
Operatoren	Gebruik =, <, >, >=, <= of <> om te vergelijken. De < en > duiden bij tekst op alfabetische volgorde: < betekent dan 'eerder in het alfabet'. Zie: SQL: Voorwaarden in een query
LIKE	Gebruik LIKE in plaats van = als je niet exacte gelijkheid bedoelt. In de tekstconstante kun je dan _ (Access: ?) gebruiken voor één onbekende letter, % (Access: *) voor willekeurig veel letters. Voorbeeld: *NAAM LIKE '_ob%'* selecteert 'Bob' 'Rob' en 'Robert', maar niet 'Jacob'. Zie: SQL: Vergelijkingen met LIKE
IS NULL	Gebruik *IS NULL* om te testen of er in een veld iets is ingevuld. Zie: SQL: Zoeken naar niet ingevulde velden
Subquery, IN	Binnen een query kun je een subquery maken voor een tussenresultaat. De voorwaarde is dan: *<kolom> IN (<subquery>)*. Hiermee kun je nagaan of de gegevens in <kolom> voorkomen in de tabel die door de subquery wordt opgeleverd. De subquery moet een tabel met één kolom opleveren. Als je zeker weet dat de subquery maar één rij oplevert, bijvoorbeeld doordat er een functie als MAX gebruikt wordt, kun je in plaats van IN ook = gebruiken. Zie: SQL: Subquery's met IN
Subquery, NOT IN	Om uit te zoeken of er in een andere tabel juist *geen* samenhangende gegevens zijn, kun je een subquery met NOT IN gebruiken. De voorwaarde is dan: *<kolom> NOT IN (<subquery>)*. Zie: SQL: Zoeken naar wat er niet is
EXISTS *(alleen verdieping)*	De voorwaarde: *EXISTS (<subquery>)* test of de subquery een tabel met één of meer rijen oplevert. *NOT EXISTS (<subquery>)* test juist of de subquery niets oplevert. Zie: SQL: Gecorreleerde subquery's en EXISTS

ORDER BY

Sorteervolgorde	Achter ORDER BY kunnen een of meer kolommen worden opgegeven, door middel van de kolomnaam of het rangnummer van de bij SELECT opgegeven kolommen. Er wordt gesorteerd op de eerstgenoemde kolom, bij gelijkheid op de tweede enzovoort.

	Zie: SQL: Sorteren van de uitvoer
ASC, DESC	Na elke kolom kun je ASC (oplopend) of DESC (aflopend) opgeven. Standaard is oplopend. Zie: SQL: Sorteren van de uitvoer

GROUP BY .... HAVING

Groeperen	Met GROUP BY kun je de rijen van een tabel indelen in groepen die dezelfde gegevens in een kolom (of combinatie van kolommen) hebben. Die kolom(men) geef je achter GROUP BY op. Zie: SQL: Groeperen in een query
Functies	Bij SELECT zet je dezelfde kolommen als achter GROUP BY, plus functies voor het aantal, totalen en dergelijke. Deze functies worden dan voor elke groep apart uitgerekend. Zie: SQL: Groeperen in een query
HAVING *(alleen verdieping)*	Bij HAVING kun je voorwaarden opgegeven die voor de groepen moeten gelden. Er is een belangrijk verschil met de voorwaarden die je bij WHERE opgeeft. Met de voorwaarden bij WHERE worden rijen geselecteerd, voordat er gegroepeerd wordt. De voorwaarden bij HAVING gelden voor de groepen en de functies die hierin worden gebruikt. Groepen met meer dan tien rijen selecteer je bijvoorbeeld met *HAVING COUNT(*) > 10*. Zie: SQL: Groeperen met voorwaarden