Structured Query Language (SQL)

Huub de Beer

Eindhoven, 4 juni 2011

Database: in essentie

- 0 of meer tabellen
- elke tabel nul of meer kolommen (of velden)
- elke tabel nul of meer unieke rijen
- elke query werkt op tabellen
- elke query levert een nieuwe tabel op
- een lege tabel is ook een tabel

Je hebt altijd te maken met tabellen.

Die kunnen leeg zijn

Ik leer SQL en neem mee ...

- SQL is door mensen bedacht en zit logisch in elkaar
- Een oplossing zien $=$ een oplossing begrijpen $\neq$ begrijpen hoe je zelf tot die oplossing komt
- Leren door oefenen
- Boek en de presentaties: veel voorbeelden en (nieuwe) SQL code.
- Probeer de voorbeelden uit (zeker als je ze niet begrijpt)
- Databases zijn op eckartnet te vinden
- Tijdens de presentatie gebruik ik de school database (van gisteren)

Informatie opzoeken: voer een query uit

SQL Query

SELECT veld_A, veld_B, ...
FROM tabelnaam
WHERE conditie
ORDER BY veld_B;

Op zijn NLs

Maak een nieuwe tabel met daarin de kolommen veld_A, veld_B, enzovoorts. Haal de waarden uit de tabel met naam tabelnaam. Kopieer alleen die rijen die aan de conditie voldoen. Sorteer de resultaattabel op veld_B.

Informatie opzoeken: voer een query uit

SQL Query

SELECT veld_A, veld_B, ...
FROM tabelnaam
WHERE conditie
ORDER BY veld_B;

Op zijn NLs

Maak een nieuwe tabel met daarin de kolommen veld_A, veld_B, enzovoorts. Haal de waarden uit de tabel met naam tabelnaam. Kopieer alleen die rijen die aan de conditie voldoen. Sorteer de resultaattabel op veld_B.

Eenvoudige queries: namen van leerlingen

Selecteer de namen van alle leerlingen

```
SELECT achternaam
FROM leerling;
```

Achternamen komen vaker voor: haal de dubbelen weg

Selecteer de namen van alle leerlingen; elke naam komt maar een keer voor; sorteer op alfabet (a-z)

```
SELECT DISTINCT achternaam
FROM leerling
ORDER BY achternaam;
```

DISTINCT voor een veldnaam in de **SELECT** clausule verwijdert dubbelen in de resultaattabel.

Eenvoudige queries: namen van leerlingen

Selecteer de namen van alle leerlingen

```
SELECT achternaam
FROM leerling;
```

Achternamen komen vaker voor: haal de dubbelen weg

Selecteer de namen van alle leerlingen; elke naam komt maar een keer voor; sorteer op alfabet (a-z)

```
SELECT DISTINCT achternaam
FROM leerling
ORDER BY achternaam;
```

DISTINCT voor een veldnaam in de **SELECT** clausule verwijdert dubbelen in de resultaattabel.

Eenvoudige queries: namen en leeftijden van leerlingen

Selecteer de namen en leeftijden van alle leerlingen van 16 jaar en ouder, gesorteerd van oud naar jong

SELECT achternaam, 2010 − **YEAR**(geboorte_datum) **AS** leeftijd
FROM leerling
WHERE 2010 − **YEAR**(geboorte_datum) >= 16
ORDER BY leeftijd **DESC**;

Je kunt eenvoudige rekensommen met velden maken. Hernoem een veldnaam met **AS** (werkt ook met tabellen). **YEAR** (datum) levert het jaar van een datum op. **DESC** en **ASC** in de **ORDER BY** clausule sorteert respectievelijk aflopend of oplopend.

Eenvoudige queries: samengestelde condities

Selecteer de voornaam, achternaam en woonplaats van alle leerlingen die jonger zijn dan 16 en wiens achternaam met een V begint

```
SELECT voornaam, achternaam, woonplaats
FROM leerling
WHERE achternaam LIKE 'V%' AND
2010 − YEAR(geboorte_datum) < 16;
```

LIKE voor patroonherkenning: % voor 0 of meer tekens; _ voor precies een teken. De conditie is een Booleaanse expressie.

Eenvoudige queries: samengestelde condities

Selecteer alle gegevens van alle leerlingen die jonger zijn dan 16 en wiens achternaam met een ‘V’ begint of de woonplaats met een ‘e’ eindigt

```
SELECT *
FROM leerling
WHERE (
achternaam LIKE 'V%' OR woonplaats LIKE '%e'
) AND 2010 − YEAR(geboorte_datum) < 16;
```

* in de **SELECT** clausule betekent “alle kolommen”. Let op de haakjes!

Aggregatiefuncies: samennemen van waarden in een kolom

Naast uitspraken en berekeningen met waarden in een rij kun je ook uitspraken doen over een hele kolom. Je gebruikt dan zogenaamde aggregatiefuncties. Deze komen alleen in de **SELECT** en **HAVING** clausule voor!

- **COUNT**(achternaam): tel het aantal achternamen
- **COUNT**(**DISTINCT** achternaam): tel het aantal achternamen, neem dubbelen niet mee. DISTINCT werk op alle aggregatiefuncties.
- **MAX**(geboorte_datum): geef de jongste datum (= grootste waarde). Veld moet sorteerbaar zijn
- **MIN**(geboorte_datum): geef de oudste datum (= kleinste waarde). Veld moet sorteerbaar zijn
- **AVG**(ll_nr): geef de gemiddelde datum (veld moet rekenbaar zijn, niet altijd zinnig ...)
- **SUM**(ll_nr): tel alle prijzen op (veld moet rekenbaar zijn)

Vooral handig bij **GROUP BY**

Volledige SQL query

SQL Query

```
SELECT veld_A, veld_B, ...
FROM tabelnaam
WHERE conditie_W
GROUP BY veld_G
HAVING conditie_H
ORDER BY veld_B;
```

Op zijn NLs

Maak een nieuwe tabel met daarin de kolommen veld_A, veld_B, enzovoorts. Haal de waarden uit de tabel met naam tabelnaam. Kopieer alleen die rijen die aan de conditie_W voldoen. Groepeer de resultaten per veld_G en neem alleen die groepen op die voldoen aan conditie_H. Sorteer de resultaattabel op veld_B.

Volledige SQL query

SQL Query

SELECT veld_A, veld_B, ...
FROM tabelnaam
WHERE conditie_W
GROUP BY veld_G
HAVING conditie_H
ORDER BY veld_B;

Op zijn NLs

Maak een nieuwe tabel met daarin de kolommen veld_A, veld_B, enzovoorts. Haal de waarden uit de tabel met naam tabelnaam. Kopieer alleen die rijen die aan de conditie_W voldoen. Groepeer de resultaten per veld_G en neem alleen die groepen op die voldoen aan conditie_H. Sorteer de resultaattabel op veld_B.

Eenvoudige queries: geef het aantal leerlingen per klas

Het aantal leerlingen per klas

```
SELECT klas_nr, COUNT(ll_nr) AS aantal
FROM leerling
GROUP BY klas_nr;
```

Elke leerling zit in een klas (vreemde sleutel klas_nr). Tel het aantal leerlingen en geef dat aantal en het klas_nr. Groepeer per klas_nr. Het effect: tel het aantal leerlingen per groep, dus per klas(nummer)

De resultaattabel bevat rijen die over groepen gaan, niet meer over losse rijen .... (probeer dit zelf uit, zorg dat je het verschil begrijpt)

Eenvoudige queries: welke leerling(en) is het oudst?

Het aantal leerlingen per klas

```
SELECT *, MIN(geboorte_datum)
FROM leerling;
```

Let op, er kunnen meerdere leerlingen zijn met dezelfde kleinste geboortedatum. Er zit er maar een in de resultaattabel (onbekend welke).

BTW onderstaande oplossing werkt niet, waarom niet?

Het aantal leerlingen per klas

```
SELECT *
FROM leerling
WHERE geboorte_datum = MIN(geboorte_datum);
```

Oplossing?

Eenvoudige queries: welke leerling(en) is het oudst?

Het aantal leerlingen per klas

```
SELECT *, MIN(geboorte_datum)
FROM leerling;
```

Let op, er kunnen meerdere leerlingen zijn met dezelfde kleinste geboortedatum. Er zit er maar een in de resultaattabel (onbekend welke).

BTW onderstaande oplossing werkt niet, waarom niet?

Het aantal leerlingen per klas

```
SELECT *
FROM leerling
WHERE geboorte_datum = MIN(geboorte_datum);
```

Oplossing?

Eenvoudige queries: geef het aantal leerlingen per klas, maar alleen van die klassen die minder dan 5 leerlingen hebben

Het aantal leerlingen per klas

```
SELECT klas_nr, COUNT(ll_nr) AS aantal
FROM leerling
GROUP BY klas_nr
HAVING COUNT(ll_nr) < 5
ORDER BY aantal;
```

De **HAVING** clausule werkt alleen op groepen (de "**WHERE**" van **GROUP BY**, als het ware). Je mag er aggregatiefuncties gebruiken. De groepen worden gesorteerd van klein naar groot (de kleinste klas staat bovenaan).

Hoe leer ik SQL?

In principe zijn de meeste queries/opgaven niet overdreven moeilijk.

- Heel veel informatie: lastig te verwerken/onthouden/reproduceren uit het hoofd
- Oefen: probeer voorbeelden uit en maak opgaven
- Voorbeelddatabases beschikbaar: download, XAMPP, phpmyadmin, importeer en je kunt aan de gang
- Het is handig om XAMPP thuis eens te installeren
- Zorg dat je het effect van een query begrijpt
- Zorg dat je zelf een query kunt schrijven: Geen trial-and-error

Maar: de opgaven worden steeds moeilijker. Zonder goede basis ga je onzin opschrijven en dat levert niets op (ook geen punten “voor het proberen” ...)