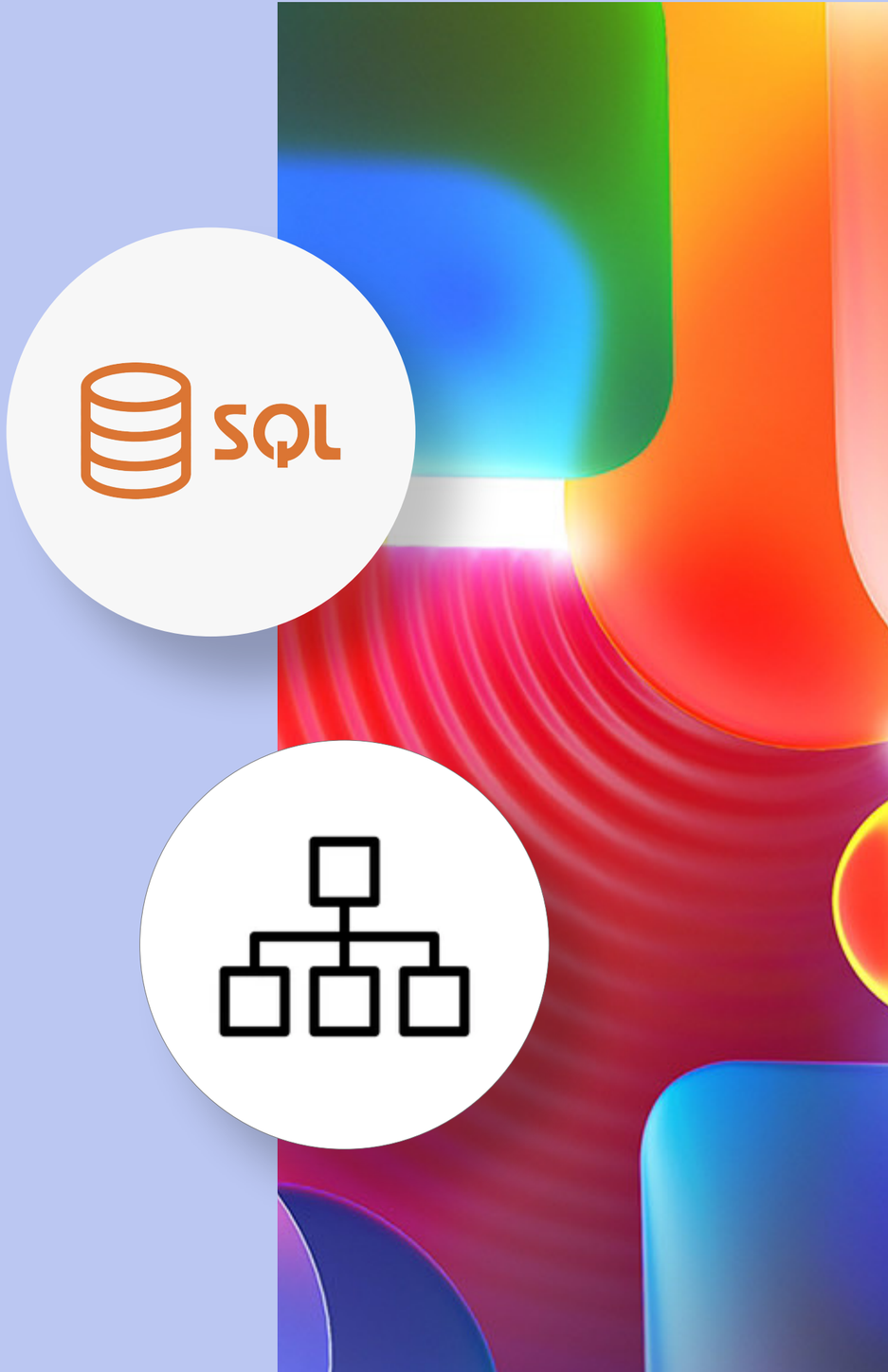
Modelling & Databases

Les 11:

SQL > DQL advanced



Onderwerpen les:

- Vergelijkingsoperatoren
- Aliassen voor kolomnamen
- Rekenkundige operatoren
- Logische operatoren
- Geaggregeerde functies
- JOINS
- Opdracht

Leerdoelen

- Aan het einde van de dag kun je verschillende operatoren gebruiken in SQL
- Aan het einde van de dag weet je wat joins zijn en kun je deze gebruiken in queries

SQL DQL

Vergelijkingsoperatoren

Comparison/Relational Operator (vergelijkingsoperator) is een wiskundig symbool om twee waarden te vergelijken. Het resultaat kan TRUE of FALSE en UNKNOWN zijn.

In SQL gebruiken we voornamelijk bij WHERE:
WHERE <expression> [comparison operator] <expression>;

Voorbeeld

producten met een prijs groter dan 10

```
SELECT  * FROM artikel
WHERE adviesprijs > 10;
```

SQL DQL

Vergelijkingsoperatoren

>	groter dan
<	kleiner dan
>=	groter of gelijk aan
<=	kleiner of gelijk aan
!=	niet gelijk aan
=	gelijk aan

- **https://www.w3schools.com/sql/sql_operators.asp**
- **https://www.tutlane.com/tutorial/sql-server/sql-comparison-operators**

SQL DQL

Aliassen voor kolommen

SQL aliassen worden gebruikt om een tijdelijke naam te geven aan tabellen of kolommen. Hierdoor worden kolomnamen meer leesbaar, soms wordt het gebruikt om naar een andere taal te vertalen of om een naam te geven aan een functie.

Voorbeeld

vertaling naar engels tabel en kolom. Geven benaming aan een nieuwe kolom onstaan door gebruik van een funcite

```
SELECT  klantnr AS customernr, count(bestnr) AS totalorders
FROM bestelling AS orders
GROUP BY customernr;
```

https://www.w3schools.com/sql/sql_alias.asp

SQL DQL

Rekenkundige operatoren

In een query kunnen naast kolomwaarden ook berekende waarden opgevraagd worden. Bijvoorbeeld aantal * prijs AS totaalbedrag of prijs - 20 AS kortingsbedrag etc. Hiervoor gebruiken we Arithmetic Operators (rekenkundige operatoren)

+	Optellen
-	Aftrekken
*	Vermenigvuldigen
/	Delen
%	Modulo (Resterend)

Voorbeeld

\# Welke artikelen hebben te weinig voorraad?

```
SELECT fabnr, artnr, vrd - minvrd AS tekort
FROM artikel
WHERE vrd - minvrd < 0;
```

- **https://www.w3schools.com/sql/sql_operators.asp**
- **https://www.programiz.com/sql/operators**

SQL DQL

Logische operatoren

Logical operators (logische operatoren) worden in SQL gebruikt om te testen of meerdere statements als totaal TRUE, FALSE of UNKNOWN zijn. Bijvoorbeeld:
WHERE salaris > 2500 AND salaris < 4000; of
WHERE continent = ‘Europa’ OR continent = ‘Amerika’;

AND	TRUE als alle voorwaarden gescheiden door AND TRUE zijn.
BETWEEN	TRUE als het argument binnen het vergelijkingsbereik valt
LIKE	TRUE als het argument overeenkomt met een patroon
NOT	Geeft een record weer als de voorwaarde(n) NIET WAAR is (zijn).
OR	TRUE als een van de voorwaarden gescheiden door OR TRUE is

Zie meer op w3 schools

https://www.w3schools.com/sql/sql_operators.asp

SQL DQL

Geaggregeerde functies

Aggregate functions (geaggregeerde functies) worden in SQL vaak gebruikt met het group by statement. Je kunt bijvoorbeeld de gemiddelde prijs berekenen van de aankopen per klant. Of de maximale aankoopsom per klant etc.

COUNT	De functie COUNT() retourneert het aantal rijen dat overeenkomt met een opgegeven criterium.
AVG	De functie AVG() retourneert de gemiddelde waarde van een numerieke kolom.
SUM	De functie SUM() retourneert de totale som van een numerieke kolom.
MIN	De functie MIN() retourneert de kleinste waarde van de geselecteerde kolom.
MAX	De functie MAX() retourneert de grootste waarde van de geselecteerde kolom.

- **https://www.w3schools.com/sql/sql_count_avg_sum.asp**
- **https://www.w3schools.com/sql/sql_min_max.asp**

SQL DQL

Geaggregeerde functies

Voorbeelden

Zoek het duurste artikel

```
SELECT max(adviesprijs) AS duurtste_artikel
FROM artikel;
```

Geef in een tabel weer hoeveel fabrieken er zijn per stad

```
SELECT plaats, count(plaats) AS totaal_fab_plaats
FROM fabriek
GROUP BY plaats;
```

SQL DQL

Geaggregeerde functies

Voorbeelden

```
#  Gemiddelde afviesprijs artikelen
SELECT avg(adviesprijs) AS gemiddelde_adviesprijs
FROM artikel;
```

SQL DQL

JOINS

Een join wordt gebruikt om records te combineren uit meerdere tabellen, gebasseerd op de relatie tussen de kolommen.
Voor onze fabriek willen we misschien de bestelde artikelen ophalen per klant. In een overzicht wil je zien: klantnr, klantnaam, bestnr, artn, artikelnaam, aantal en bestelprijs. Om dit in 1 overzicht te krijgen gebruik je joins en gebruik je kolommen uit de benodigde tabellen.

(INNER) JOIN	- Retourneert records met overeenkomende waarden in beide tabellen
LEFT JOIN	- Retourneert alle records uit de linkertabel en de overeenkomende records uit de rechtertabel
RIGHT JOIN	- Retourneert alle records uit de rechtertabel en de overeenkomende records uit de linkertabel
FULL OUTER JOIN	- Returns all records when there is a match in either left or right table
SELF JOIN	- Een self-join is een gewone join, maar de tabel is met zichzelf verbonden.

SQL DQL

JOINS

(INNER) JOIN

De INNER JOIN selecteert records met overeenkomende waarden in beide tabellen. INNER JOIN = JOIN

SELECT .. FROM .. INNER JOIN .. ON

Voorbeeld

Haal de klantinformatie op bij de bestellingen.

```
SELECT  bestelling.bestnr, bestelling.datum, klant.naam, klant.adres, klant.plaats
FROM bestelling
INNER JOIN klant ON bestelling.klantnr = klant.klantnr;
```

SQL DQL

JOINS

(INNER) JOIN

Voorbeeld

Zelfde kortere manier

```
SELECT  b.bestnr, b.datum, k.naam, k.adres, k.plaats
FROM bestelling b
INNER JOIN klant k ON b.klantnr = k.klantnr;
```

SQL DQL

JOINS

LEFT JOIN

De LEFT JOIN geeft alle records uit de linkertabel (tabel1) en de overeenkomende records uit de rechtertabel (tabel2). Het resultaat is 0 records van de rechterkant, als er geen match is.

Voorbeeld

Haal de klantinformatie op bij de bestellingen.

```
SELECT  b.bestnr, b.datum, k.naam, k.adres, k.plaats
FROM klant k
LEFT JOIN bestelling b ON b.klantnr = k.klantnr;
```

Geeft nu ook klanten weer zonder bestelling

SQL DQL

JOINS

RIGHT JOIN

De RIGHT JOIN retourneert alle records uit de rechtertabel (tabel2) en de overeenkomende records uit de linkertabel (tabel1). Het resultaat is 0 records van de linkerkant, als er geen match is.

Voorbeeld

Haal de klantinformatie op bij de bestellingen.

```
SELECT  b.bestnr, b.datum, k.naam, k.adres, k.plaats
FROM bestelling b
LEFT JOIN klant k ON b.klantnr = k.klantnr;
```

Geeft nu ook bestelling weer zonder klanten (wat in deze DB niet kan)

SQL DQL

JOINS

FULL OUTER JOIN

De FULL OUTER JOIN retourneert alle records wanneer er een overeenkomst is in linker (tabel1) of rechter (tabel2) tabelrecords. FULL OUTER JOIN = OUTER JOIN

Voorbeeld

Haal de klantinformatie op bij de bestellingen.

```
SELECT  b.bestnr, b.datum, k.naam, k.adres, k.plaats
FROM bestelling b
LEFT JOIN klant k ON b.klantnr = k.klantnr;
```

klant k

bestelling b

Geeft nu in beide situaties alles van klant ook zonder bestelling en alles van bestelling ook zonder klant

SQL DQL

Opdracht

In deze casus bouw je een ERD, database, voeg je data toe en zoek je data uit de database.

Voor een boekenwinkel ga je een database bouwen voor het online shoppen.

- De boekenwinkel heeft 4 filialen waarvan adres, plaats en telefoonnummer in de database moet komen te staan.
- Uiteraard komen boeken in de database. Een boek heeft een ISBN, jaar van uitgave, titel, auteur, uitgever en een prijs. Van de auteur willen we zijn naam, geboortejaar en URL(naar zijn wikipedia). Van de uitgever willen we een naam, adres, plaats, telefoonnummer en URL (website van de uitgever).
- Een klant kan alleen boeken toevoegen aan een winkelmandje zodra de klant een online account heeft. Een account heeft een email, naam, telefoonnummer, adres en plaats.
- De ingelogde klant kan boeken toevoegen aan een online winkelwagentje, uiteraard kan de klant 1 of meerdere van hetzelfde boek toevoegen aan de winkelwagen.

SQL DQL

Opdracht

- de database houdt bij hoeveel van elk boek aanwezig is in elk filiaal.
- Voor het gemak doen we niks met afrekenen en versturen in deze opdracht.

1.1 ERD

Maak van de database een conceptueel, logisch en fysiek model. Denk goed na over wat je nodig hebt en de relaties tussen de entiteiten, tabellen.

1.2 DDL

Maak de database en tabellen aan de hand van een DDL script.

SQL DQL

Opdracht

1.3 Voeg data toe

Maak van de database een conceptueel, logisch en fysiek model. Denk goed na over wat je nodig hebt en de relaties tussen de entiteiten, tabellen.

Filiaal

Rokin 9, Amsterdam, 0205231481
Oudegracht 112-b, Utrecht, 0302335200
Coolsingel 129, Den Haag, 0104132070
Kerkstraat 27, 's-hertogenbosch, 0733020100

Boeken

Stop je 10 favoriete boeken in de database en de daarbij horende auteurs en uitgevers.

SQL DQL

Opdracht

Filiaal

Nu je boeken in de database hebt kun je ook in de database zetten hoeveel van elk boek aanwezig is in het filiaal. Elk boek kan x aantal keer in elk filiaal aanwezig zijn. Bepaal zelf hoeveel een boek aanwezig is in een filiaal. Elk boek minimaal 50 in totaal verspreid over de filialen.

Klanten

a_devries@adelaarsnest.nl, Arend de Vries, 0612345678, Adelaarsnest 12, Haarlem
bboer@boeren.nl, Bert Boer, 0623456789, Mesthoop 1, Dorp
c_kordaat@gmail.com, Cornelis Kordaat, 0634567890, Kerkstraat 11, Utrecht

SQL DQL

Opdracht

Gebeurtenissen

- Arend logt in en stop 1 exemplaar van boek ‘a’ en 1 exemplaar van boek ‘b ’in zijn winkelmandje
- Bert logt in en stop 4 exemplaren van boek ‘c’ in zijn winkelmandje.
- Cornelis logt in en stop 3 exemplaren van boek ‘a’ in zijn winkelmandje.

- Arend logt opnieuw in en stop in zijn nieuwe winkelmandje 1 exemplaar van boek ‘c’.

SQL DQL

Opdracht

1.4 data opzoeken

Nu er data in de database zit kunnen we queries uitvoeren.

1.4.1

Haal van alle boeken ISBN, naam op en daarbij hoeveel deze in totaal in een winkelmandje zitten.

1.4.2

Haal van klant Arend op zijn email en naam en de namen van alle boeken in zijn winkelmandjes inclusief het aantal van die boeken.

1.4.3

Haal van alle filialen het adres en de namen en hoeveelheid van de aanwezige boeken.

SQL DQL

Opdracht

1.4.4

Haal de naam van alle auteurs op en de namen van de uitgevers waar zij boeken voor hebben geschreven.

1.4.5

Haal alle boeken op die beginnen met een a of b.

1.4.5

Haal van alle uitgevers de naam op en de gemiddelde prijs per boek van deze uitgevers.

1.4.6

Hoeveel boeken duurder dan 10 euro zitten er in alle winkelwagens?