



# Databases

8. Werken met meerdere tabellen  
(GROUP BY – JOIN – UNION)

# Inhoud

8.1. GROUP BY en statistische functies

8.2. JOIN

8.3. UNION

# Basisvorm SELECT statement

- SELECT voor raadplegen van één tabel

**SELECT** [ALL | **DISTINCT**] {\*}|uitdrukking [,uitdrukking ...]}

**FROM** tabelnaam

[**WHERE** voorwaarde(n)]

 **GROUP BY** kolomnaam [,kolomnaam ...]

[**HAVING** voorwaarde(n)]

[**ORDER BY** {kolomnaam|volgnr}{ASC|**DESC**}[,...]]

# **8.1. GROUP BY en statistische functies**

# Statistische functies

- Statistische functies (= aggregaatsfuncties)
  - SQL voorziet 5 standaardfuncties
    - **SUM**(uitdrukking): som
    - **AVG**(uitdrukking): gemiddelde
    - **MIN**(uitdrukking): minimum
    - **MAX**(uitdrukking): maximum
    - **COUNT**(\*|[DISTINCT] kolomnaam): aantal
  - Deze functies geven één antwoord per kolom (of groep) en mogen dus niet in een WHERE clause (= rijniveau) gebruikt worden

# Som en gemiddelde

## SUM

- Berekent de som van alle niet-NUL numerieke waarden in één kolom
- Voorbeeld: Geef de totale stockwaarde van alle producten

```
SELECT SUM(UnitsInStock * UnitPrice) AS inventoryvalue  
FROM products
```

Result Grid	
	inventoryvalue
▶	73953.3500

## AVG

- Berekent het gemiddelde van alle niet-NUL numerieke waarden
- Voorbeeld: Wat is de gemiddelde voorraad over alle producten?

```
SELECT AVG(unitsinstock) AS AverageStock  
FROM products
```

Result Grid	
	AverageStock
▶	40.5065

# Aantal rijen tellen

## COUNT(\*)

- Telt het aantal rijen van de selectie
- Voorbeeld: Tel het aantal producten

```
SELECT COUNT(*) AS Aantal  
FROM products
```

Result Grid	
	Filter Row
Aantal	77

## COUNT(kolomnaam)

- Telt het aantal niet-lege velden in een kolom
- Voorbeeld: Tel het aantal producten met een (niet-lege) categorie

```
SELECT COUNT(categoryid) AS cat_count  
FROM products
```

Result Grid	
	Filter Row
cat count	77

- Voorbeeld: Tel het aantal verschillende (niet-lege) categorieën bij de producten

```
SELECT COUNT(DISTINCT categoryid) AS cat_count  
FROM products
```

Result Grid	
	Filter Row
cat count	8

# Minimum en maximum

## MIN en MAX

- retourneert de kleinste en de grootste waarde in een kolom
- zowel op numerieke als alfanumerieke argumenten
- Voorbeeld: Wat is het goedkoopste en het duurste product?

```
SELECT MIN(unitprice) AS Minimum,  
       MAX(unitprice) AS Maximum  
FROM products
```

Minimum	Maximum
2.5000	263.5000

## Opmerkingen

1. Omdat een statistische functie maar **één antwoord** oplevert, moeten ofwel alle uitdrukkingen in de SELECT clausule een statistische functie bevatten, ofwel geen enkele! Dit verandert wanneer we group-by introduceren
2. Statistische functies houden geen rekening met **NULL-waarden**  
Uitzondering : COUNT(\*) >>> telt ook rijen die NULL bevatten

# Groeperen via GROUP BY

## GROUP BY clause

- Groeperen van rijen met gemeenschappelijke kenmerken
- Per groep wordt maar één rij teruggegeven!
- Voorbeeld: Welke categorieën zijn er binnen de producten?

```
SELECT CategoryID  
FROM Products  
GROUP BY CategoryID
```

CategoryID
1
2
3
4
5
6
7
8

- Elke groep is een afzonderlijke verzameling waarop statistische functies uitgevoerd kunnen worden

# Groeperen via GROUP BY

- Toon per categorie het aantal producten

```
SELECT CategoryID, COUNT(productID) AS Aantal  
FROM Products  
GROUP BY CategoryID
```

CategoryID	Aantal
1	12
2	12
3	13
4	10
5	7
6	6
7	5
8	12

- Toon per categorie het aantal producten, waarvan er meer dan 10 in stock zijn

```
SELECT CategoryID, COUNT(productID) AS Aantal  
FROM Products  
WHERE UnitsInStock > 10  
GROUP BY CategoryID
```

CategoryID	Aantal
1	12
2	9
3	10
4	8
5	7
6	3
7	4
8	10

# Groepering verfijnen via HAVING

## HAVING clausule (**GROEPEN SELECTEREN**)

- Toon het aantal producten voor elke categorie die meer dan 10 producten bevat

```
SELECT CategoryID, COUNT(productID) AS Aantal  
FROM Products  
GROUP BY CategoryID  
HAVING COUNT(productID) > 10
```

CategoryID	Aantal
1	12
2	12
3	13
8	12

- Toon het aantal producten voor elke categorie die meer dan 10 producten **in voorraad** bevat

```
SELECT CategoryID, COUNT(productID) AS Aantal  
FROM Products  
WHERE UnitsInStock > 0  
GROUP BY CategoryID  
HAVING COUNT(*) > 10
```

CategoryID	Aantal
1	12

# WHERE vs. HAVING

Verschil tussen WHERE en HAVING

- **WHERE = RIJEN SELECTEREN**
- **HAVING = GROEPEN SELECTEREN**
- Aggregatiefunctie(s) in SELECT?
  - Aggregatiefuncties niet gebruiken in WHERE, GROUP BY
  - Alle kolommen in SELECT die niet in een aggregatiefunctie voorkomen moeten in GROUP BY!

```
SELECT categoryID, MIN(unitprice) AS Minimum  
FROM products
```

Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated column 'northwind.products.CategoryID'; this is incompatible with sql\_mode=only\_full\_group\_by

# Oefeningen

NR	VNAAM	INIT	FNAAM	AFD	IN DIENST	CODE	NIV	GESL	GEBDAT	SALARIS
10	Christine	I	Haas	A00	650101	66	18	V	330814	52750
20	Michel	L	Theunis	B01	731001	61	18	M	480202	41250
30	Sally	A	Kramer	C01	750405	60	20	V	410511	38250
50	Johan	B	Geysen	E01	490817	58	16	M	250915	40175
60	Irving	F	Steur	D11	730914	55	16	M	450707	32250
70	Eva	D	Pulanski	D21	800930	56	16	V	530526	36170
90	Evelien	W	Hendriks	E11	700815	55	16	V	410515	29750
100	Theo	Q	Spencer	E21	800619	54	14	M	561218	26150
110	Vincent	G	Leman	A00	631205	58	19	M	291105	46500
120	Sean		Connors	A00	580516	58	14	M	421018	29250
130	Danielle	M	Scheire	C01	710728	55	16	V	250915	23800
140	Hilde	A	Nagels	C01	761215	56	18	V	460119	28420
150	Bruno		Adams	D11	720212	55	16	M	470517	25280
160	Els	R	Placke	D11	771011	54	17	V	550412	22250
170	Mats	J	Sierens	D11	780915	54	16	M	510105	24680
180	Marleen	S	Schouters	D11	730707	53	17	V	490221	21340
190	Jan	E	Wauters	D11	740726	53	16	M	520625	20450
200	David		De Bruyn	D11	660303	55	16	M	410529	27740
210	Willem	T	Jansens	D11	790411	25	17	M	530223	18270
220	Jennifer	K	Luyckx	D11	680829	55	18	V	480319	29840

tabel Werknemer

AFDNR	AFDNAAM	MANNR
A00	Computer	10
B01	Planning	20
C01	Informatie	30
D01	Ontwikkelingasc	50
E01	Support	60
D11	Administratie	70
D21	Software	80
E21	Tools	90

tabel Afdeling

**HO  
GENT**

# Oefeningen

- Tel het aantal werknemers uit de afdeling D11 en geef het maximum, minimum en gemiddeld salaris voor deze afdeling, alsook het aantal verschillende jobcodes uit deze afdeling. Geef ook de som van alle lonen betaald in afdeling D11
- Geef per afdeling, het afdnr en het aantal werknemers, gesorteerd volgens afdelingsnummer
- Idem, maar nu aflopend gesorteerd volgens aantal werknemers
- Idem maar nu wens je het aantal werknemers te kennen per afdeling en per jobcode
- Tel per afdeling het aantal mannen en vrouwen en sorteer volgens opklimmende afdeling en afdalend geslacht
- Geef een overzicht van de afdelingen die tenminste 2 werknemers hebben die meer dan 1000 verdienen

# Oplossingen

- Select count(\*), max(salaris), min(salaris), avg(salaris), sum(salaris), count(distinct code) from werknemer where afd = 'D11'
- Select afd, count(\*) from werknemer group by afd order by afd
- Select afd, count(\*) from werknemer group by afd order by count(\*) desc
- Select afd, code, count(\*) from werknemer group by afd, code  
(hier groeperen op code, anders krijg je een fout!)
- Select afd, gesl, count(\*) from werknemer group by afd, gesl  
order by afd, gesl desc
- Select afd from werknemer where salaris > 1000 group by afd  
having count(\*) >= 2

## **8.2. JOIN**

# (INNER) JOIN

- Selecteren van kolommen uit meerdere tabellen
  - **JOIN** keyword specificeert de samen te voegen tabellen
  - **ON** keyword specificeert hoe samen te voegen
- Produceert één resultset waarin de rijen uit die tabellen gekoppeld worden (op basis van de ON-criteria)
- Basisvorm (**ANSI JOIN (SQL-92)**) vs. **Old style join**)

**SELECT** uitdrukking  
**FROM** tabel **JOIN** tabel **ON** voorwaarde  
[**JOIN** tabel **ON** voorwaarde]\*

**SELECT** uitdrukking  
**FROM** tabel, tabel [, tabel]\*  
**WHERE** voorwaarde

# INNER JOIN

- Koppelen van rijen uit één tabel met rijen uit een andere tabel op basis van gemeenschappelijke waarden in de overeenkomstige kolommen
  - De relatie tussen de velden in de verschillende tabellen kan je uitdrukken met vergelijkingsoperatoren: =, <, >, <>, >=, <=
  - Meestal wordt = gebruikt. Dit heet ook **EQUI-JOIN**

# INNER JOIN

- Geef alle bestellingen die niet bestemd zijn voor België, Frankrijk of Duitsland, met alle producten in die bestellingen en hun hoeveelheden
  - ANSI JOIN (SQL-92)

```
SELECT orders.OrderID, ShipCountry, ProductID, Quantity
FROM orders JOIN order_details
    ON orders.OrderID = order_details.OrderID
WHERE ShipCountry not in ('Belgium','France','Germany')
```

- Of “old style”

```
SELECT orders.OrderID, ShipCountry, ProductID, Quantity
FROM orders, order_details
WHERE orders.OrderID = order_details.OrderID
    AND ShipCountry not in ('Belgium','France','Germany')
```

Result Grid	Filter Rows:	Exports:	Wrap Cell Content:	Fetch rows:
OrderID	ShipCountry	ProductID	Quantity	
10250	Brazil	41	10	
10250	Brazil	51	35	
10250	Brazil	65	15	
10253	Brazil	31	20	
10253	Brazil	39	42	
10253	Brazil	49	40	
10254	Switzerland	24	15	
10254	Switzerland	55	21	

# Aliassen

- Gebruik van tabel-aliassen (via ‘as’ of **spatie**)
  - ANSI JOIN (SQL-92)

```
SELECT o.OrderID, ShipCountry, ProductID, Quantity
FROM orders o JOIN order_details od ON o.OrderID = od.OrderID
WHERE ShipCountry not in ('Belgium','France','Germany')
```

- “old style join”

```
SELECT o.OrderID, ShipCountry, ProductID, Quantity
FROM orders o, order_details od
WHERE o.OrderID = od.OrderID AND ShipCountry not in ('Belgium','France','Germany')
```

## Opmerkingen

1. Als een kolomnaam in meerdere tabellen (gebruikt in de query) voorkomt, dan **moet** die steeds worden voorafgegaan door de **tabelnaam (of alias)**
2. Inner joins geven enkel die rijen terug die voldoen aan de ON conditie. Dit betekent dat als een rij in de eerste tabel niet matcht met een rij uit de tweede tabel (bijv. een order zonder producten), de rij niet zal geretourneerd worden en omgekeerd

# INNER JOIN van meerdere tabellen

- Voorbeeld: Voeg bij het vorige resultaat nu ook de namen van de producten toe
  - ANSI JOIN (SQL-92)

```
SELECT o.OrderID, ShipCountry, p.ProductName, Quantity
FROM orders o
JOIN order_details od ON o.OrderID = od.OrderID
JOIN products p ON p.ProductID = od.ProductID
WHERE ShipCountry not in ('Belgium','France','Germany')
```

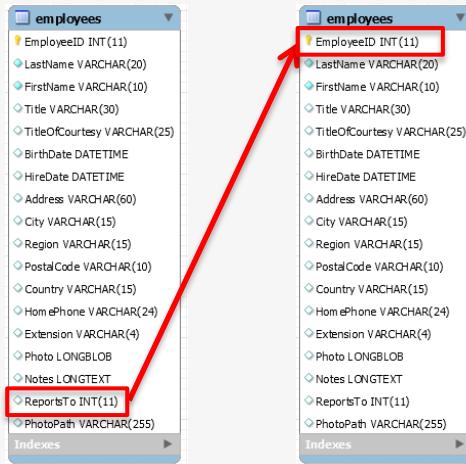
- “old style join”

```
SELECT o.OrderID, ShipCountry, p.ProductName, Quantity
FROM orders o, order_details od, products p
WHERE o.OrderID = od.OrderID
AND p.ProductID = od.ProductID
AND ShipCountry not in ('Belgium','France','Germany')
```

OrderID	ShipCountry	ProductName	Quantity
10251	Brazil	Jack's New England Clam Chowder	10
10252	Brazil	Manjimup Dried Apples	35
10253	Brazil	Louisiana Fiery Hot Pepper Sauce	15
10254	Brazil	Gorgonzola Telino	20
10255	Brazil	Chartreuse verte	42
10256	Brazil	Maxilaku	40
10257	Switzerland	Guaraná Fantástica	15
10258	Switzerland	Pâté chinois	21

# INNER JOIN van een tabel met zichzelf

- Voorbeeld: Toon het ID en de naam van alle werknemers en van hun manager



Employee.ReportsTo = vreemde sleutel  
die verwijst naar Manager.EmployeeID

Employee	lastName	Manager	lastName
1	Davolio	2	Fuller
3	Leverling	2	Fuller
4	Peacock	2	Fuller
5	Buchanan	2	Fuller
6	Suyama	5	Buchanan
7	King	5	Buchanan
8	Callahan	2	Fuller
9	Dodsworth	5	Buchanan

```
SELECT e.EmployeeID AS Employee, e.LastName,  
       e2.EmployeeID AS Manager, e2.LastName  
FROM Employees e JOIN Employees e2 ON e.ReportsTo = e2.EmployeeID
```

# OUTER JOIN

- Retourneert alle records van één tabel, zelfs als er geen gerelateerd record bestaat in de andere tabel
- Er zijn 3 types outer join
  - LEFT OUTER JOIN
    - Retourneert alle rijen van de eerst (= voor keyword JOIN) genoemde tabel in de FROM clause (SQL-92)
  - RIGHT OUTER JOIN
    - Retourneert alle rijen van de tweede (= na keyword JOIN) genoemde tabel in de FROM clause (SQL-92)
  - FULL OUTER JOIN (bestaat **niet** in MySQL)
    - Retourneert alle rijen uit beide tabellen, ook als die geen corresponderende rij hebben in andere tabel (SQL-92)

# LEFT OUTER JOIN

- Voorbeeld:

Geef de bedrijfsnamen van alle klanten en de ID van de bestellingen die ze plaatsten. Ook klanten die nog nooit een bestelling geplaatst hebben worden in de lijst opgenomen

```
SELECT c.CompanyName, o.OrderID  
FROM Customers c LEFT OUTER JOIN Orders o  
ON o.customerid = c.customerid
```

OUTER mag je weglaten

CompanyName	OrderID
Familia Arquibaldo	10581
Familia Arquibaldo	10650
Familia Arquibaldo	10725
FISSA Fabrica Inter....	NULL
Folies gourmandes	10408

# RIGHT OUTER JOIN

- Voorbeeld: Geef een overzicht van alle orders met de naam van het bedrijf erbij. Ook orders die niet aan een klant gelinkt zijn worden in de lijst opgenomen

```
SELECT c.CompanyName, o.OrderID  
FROM Customers c RIGHT OUTER JOIN Orders o  
ON o.customerid = c.customerid;
```

Er zullen geen null-waarden zijn voor CompanyName (alle orders zijn dus gelinkt aan een bedrijf)

CompanyName	OrderID
Comércio Mineiro	11042
Consolidated Holdings	10435
Consolidated Holdings	10462
Consolidated Holdings	10848
Drachenblut Delikat...	10363

# Voorbeeld

T1

C1	C2
1	A
1	B
2	B

```
SELECT *
FROM T1 JOIN T2 ON T1.c1=T2.k1
```

T2

k1	k2
1	X
1	Y
3	Y

```
SELECT *
FROM T1 LEFT JOIN T2 ON T1.c1=T2.k1
```



```
SELECT *
FROM T1 RIGHT JOIN T2 ON T1.c1=T2.k1
```



```
SELECT *
FROM T1 FULL OUTER JOIN T2 ON T1.c1=T2.k1
```

T1.C1	T1.C2	T2.K1	T2.K2
1	A	1	X
1	A	1	Y
1	B	1	X
1	B	1	Y

T1.C1	T1.C2	T2.K1	T2.K2
1	A	1	X
1	A	1	Y
1	B	1	X
1	B	1	Y
2	B	NULL	NULL

T1.C1	T1.C2	T2.K1	T2.K2
1	A	1	X
1	B	1	X
1	A	1	Y
1	B	1	Y
NULL	NULL	3	Y

T1.C1	T1.C2	T2.K1	T2.K2
1	A	1	X
1	A	1	Y
1	B	1	X
1	B	1	Y
2	B	NULL	NULL
NULL	NULL	3	Y

# CROSS JOIN

Bij een cross join is het aantal rijen in de resultaattabel gelijk aan het aantal rijen in de eerste tabel maal het aantal rijen in de tweede tabel

- ANSI JOIN (SQL-92)

```
SELECT CONCAT(e1.FirstName,' ',e1.LastName) AS Groepslid1,  
       CONCAT(e2.FirstName,' ',e2.LastName) AS Groepslid2  
FROM Employees e1 CROSS JOIN Employees e2;
```

- “old style join”

```
SELECT CONCAT(e1.FirstName,' ',e1.LastName) AS Groepslid1,  
       CONCAT(e2.FirstName,' ',e2.LastName) AS Groepslid2  
FROM Employees e1, Employees e2;
```

Groepslid1	Groepslid2
John Dodsworth	Nancy Davolio
Maura Callahan	Nancy Davolio
Albert King	Nancy Davolio
Michael Suyama	Nancy Davolio
Steven Buchanan	Nancy Davolio
Mark L. Barnes	Nancy Davolio

## **8.3. UNION**

# UNION

```
SELECT ... FROM ... WHERE ...
UNION
SELECT ... FROM ... WHERE ...
ORDER BY ...
```

Via een UNION combineer je het resultaat van 2 of meerdere query's in één resultaattabel

- De resultaten van de 2 SELECT opdrachten moeten evenveel kolommen bevatten, in dezelfde volgorde, van hetzelfde type, en met dezelfde eigenschappen (bv NULL-waarden toegelaten)
- De kolomnamen/titels van de UNION zijn deze van de eerste SELECT
- Het resultaat bevat echter steeds alleen unieke rijen
- Aan het einde van de UNION kan je een ORDER BY toevoegen. In deze clausule mag geen kolomnaam of uitdrukking voorkomen indien kolomnamen van beide query's verschillend zijn. Gebruik in dat geval kolomnummers
- Ook INTERSECT en EXCEPT bestaan

# UNION

- Voorbeeld: geef een overzicht van alle werknemers (naam en voornaam, stad en postcode) en alle klanten (naam, stad en postcode)

```
SELECT firstname + ' ' + lastname AS name, city, postalcode
```

```
FROM Employees
```

**UNION**

```
SELECT companynname, city, postalcode  
FROM Customers
```

Omdat de kolomnamen van de UNION komen uit de eerste select, dien je de alias 'name' in de tweede select niet meer te herhalen

name	city	postalcode
Alfreds Futterkiste	Berlin	12209
Ana Trujillo Empa...	México D...	05021
Antonio Moreno ...	México D...	05023
Around the Horn	London	WA1 1DP
Berglunds snabb...	Luleå	S-958 22
Blauer See Delika...	Mannheim	68306
Blondesddsl père...	Strasbourg	67000
Bólido Comidas p...	Madrid	28023
Bon app'	Marseille	13008
Bottom-Dollar M...	Tsawassen	T2F 8M4

