

# Cursus Databanken

## **SELECT**-instructie

kolommen specificeren, aliassen, expressies,  
**DISTINCT, ORDER BY, LIMIT**

Docenten:

Damien Decorte

Wim Goedertier

Luc Vervoort

Tom Vande Wiele

Eric Juchtmans



**HO  
GENT**

# SHOW DATABASES toont een lijst van beschikbare databanken (schema's)

Voer volgende instructies één per één uit:

**SHOW DATABASES;**

**SHOW SCHEMAS;**

## Merk op:

- Elke SQL-instructie wordt afgesloten met een **puntkomma**.
- Bij MySQL zijn **DATABASES** en **SCHEMAS** **synoniemen**: er is geen verschil tussen. (Bij Microsoft SQL Server is er wel een verschil.)

```
bib SQL > SHOW DATABASES;
+-----+
| Database |
+-----+
| bib      |
| db_foundations |
| information_schema |
| mysql    |
| performance_schema |
| sakila   |
| sys      |
| world    |
+-----+
```

Als "bib" of "db\_foundations" ontbreken, importeer ze dan (zie 03-DBs-importeren)

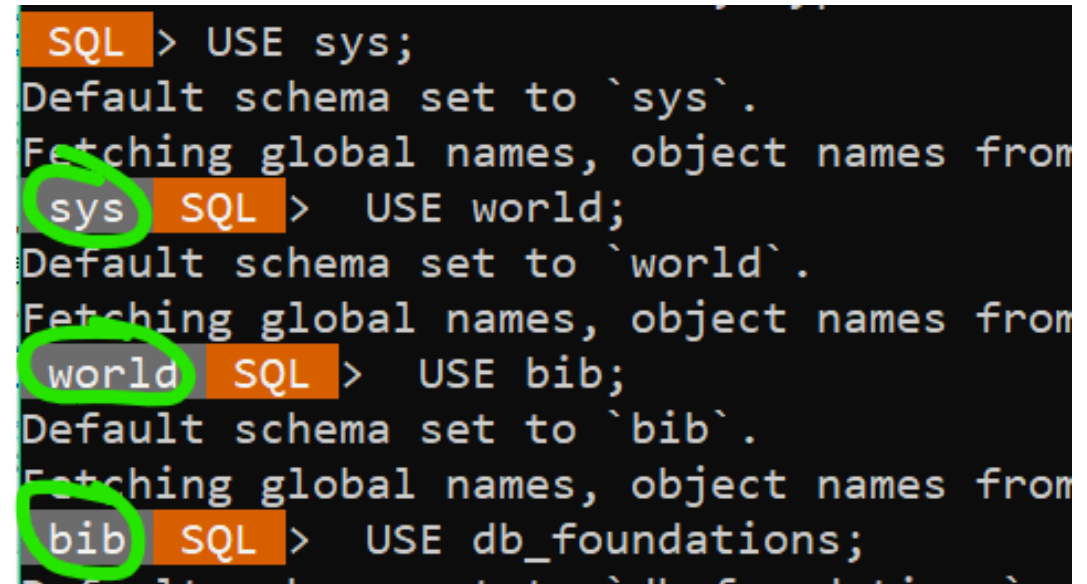
# USE activeert de databank (het schema) waarin je instructies wil uitvoeren

Voer volgende instructies één per één uit:

```
USE sys;
```

```
USE world;
```

```
USE bib;
```



```
SQL > USE sys;  
Default schema set to `sys`.  
Fetching global names, object names from  
sys SQL > USE world;  
Default schema set to `world`.  
Fetching global names, object names from  
world SQL > USE bib;  
Default schema set to `bib`.  
Fetching global names, object names from  
bib SQL > USE db_foundations;
```

## Merk op:

- De default-**prompt** van MySQL Shell toont welke databank er actief is.  
(Dat is niet bij elke prompt of bij elke CLI-tool het geval)

# HOOFDLETTERS vs. kleine letters

Voer volgende instructies één per één uit:

```
use bib;
```

```
USE BIB;
```

```
UsE bIb;
```

## Merk op:

- SQL is **niet case-sensitive**: hoofdletters of kleine letters maken geen verschil
- **MAAR** de algemene **afspraken** is, voor de leesbaarheid:
  - SQL-**keywords** (SHOW, DATABASES, USE, ...) → in hoofdletters
  - **namen** van databanken, tabellen, kolommen → in kleine letters
- Dit geldt vooral voor programmeer-code, bestanden, documentatie, ...

# Lees de foutmeldingen goed !!

Voer volgende instructies één per één uit (mèt de typ-fouten):

UUUSE bib;

USE bbbib;

Lees de foutmeldingen **volledig**

Probeer ze te begrijpen

Meestal gaat het om een "human error" 😊



# Voorkom fouten: gebruik de TAB-toets (= auto completion)

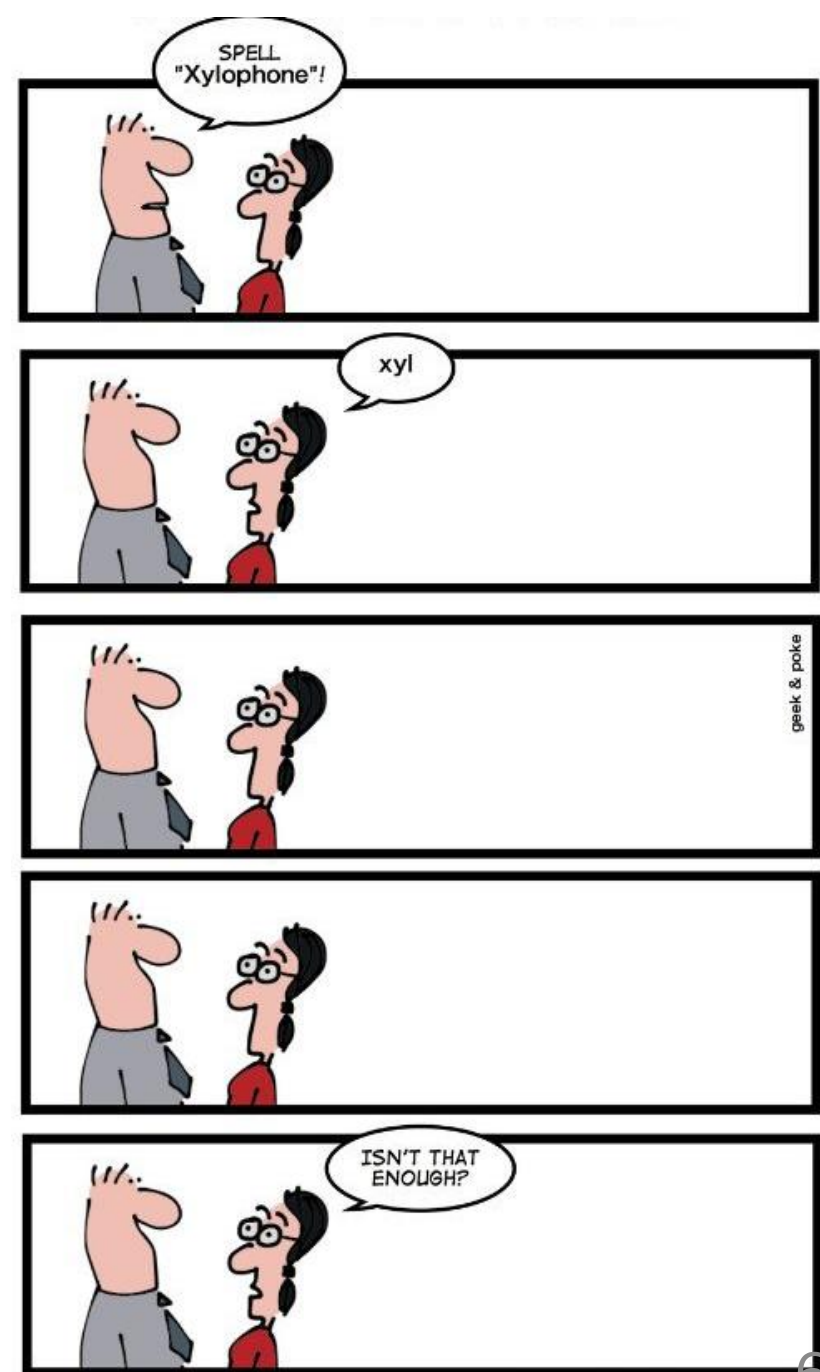
Voer de instructie **"USE bib;"** in  
door volgende toetsen één voor één in te tikken:

**u s TAB SPATIE b TAB ;**

Als er meerdere mogelijkheden zijn, moet je 2x op  
de TAB-toets drukken, om alle mogelijkheden te zien.

Probeer dit uit:

**u TAB TAB s TAB SPATIE s TAB TAB ...**



# Wees lui en recycleer: gebruik "het pijltje omhoog" (= history)

Druk verschillende keren op de toets met "het pijltje omhoog" (↑)  
Je ziet alle instructies die je reeds intikte.

Voor een overzicht:

`\history`

(druk evt. eerst `Ctrl+C` )

## Tip:

Bewaar je SQL-instructies als studiemateriaal voor later.

Doe op het einde van de les `\history`  
en copy-and-paste de instructies in een tekstbestand.



# SHOW TABLES toont welke tabellen er in de actieve databank zitten

Voer volgende instructies uit:

```
USE bib;
```

```
SHOW TABLES;
```

```
USE db_foundations;
```

```
SHOW TABLES;
```

```
u s TAB SPATIE b TAB ;
```

```
s h o TAB SPATIE t a TAB s ;
```

```
2x ↑ 4x BCKSPACE d TAB ;
```

```
2x ↑ ENTER
```



Tables_in_bib
auteur
boeken
categorie
klant
uitgever
uitleen

Als je de exacte naam van een tabel niet meer weet zal je **SHOW TABLES** moeten doen...



# SELECT-instructie (of SELECT-*statement*)

Onderstaand SELECT-statement toont de inhoud van **alle** kolommen en **alle** rijen (de volledige inhoud, dus) uit de tabel "boeken"

```
USE bib;
```

```
SELECT * FROM boeken;
```

4x ↑

s e l TAB \* f r TAB b o TAB

De lijst die je krijgt noemen we de resultatenlijst.

id	auteur_id	publ_jaar	titel	cat_id	uitg_id
81	1	1953	Kaas	2	4
82	2	1962	Jan De Lichte	2	5
83	2	1964	Geuzenboek	2	5
84	3	1983	Het verdriet van België ½	2	6
84	3	1986	De geruchten	1	6

# SELECT-instructie (of SELECT-*statement*)

```
| 118 |      11 |      1999 | Tuinieren voor gevo  
| 119 |      11 |      1998 | Japanse tuinen  
| 120 |      11 |      2000 | Afrikaanse tuinen  
+-----+-----+-----+-----+  
20 rows in set (0.0010 sec)  
b10 SQL >
```

## Merk op:

Onderaan vind je steeds:

- het aantal rijen in de resultatenlijst
- hoeveel seconden de server nodig had om de SQL-instructie te verwerken.

# Spaties en "newlines"

Probeer het volgende:

<b>SELECT</b>	*	s e l	TAB	<b>2x</b>	SPACE	*	<b>ENTER</b>
<b>FROM</b>	<b>boeken;</b>	f r	TAB	SPACE	b o	TAB	; ENTER

Daarna:

↑ gevolgd door **ENTER**

## Merk op:

- SQL is *niet* gevoelig voor **extra spaties** of **newlines**
- MySQL Shell verwijdt de "newlines" zelfs uit de history

# Layout-afspraken

Sommige mensen zien liever:

```
SELECT * FROM boeken;
```

(compacter)

Andere mensen zien liever:

```
SELECT *
```

```
FROM boeken;
```

(overzichtelijker)

Hierover zijn **geen algemene** afspraken.

De afspraken kunnen verschillen van bedrijf tot bedrijf, van cursus tot cursus, van docent tot docent, van situatie tot situatie, ...

# Vooruitblik:

## Algemene structuur SELECT-instructie

De verschillende *clausules* van de SELECT-instructie zullen 1 voor 1 geïntroduceerd worden:

```
SELECT ...  
FROM ...  
WHERE ...  
GROUP BY ...  
    HAVING ...  
ORDER BY ...  
    LIMIT ...
```

Als er veel clausules zijn, is het soms beter om elke clausule op een aparte regel te zetten (afhankelijk van de situatie).

# De SELECT-clausule: selecteer kolommen

## Voorbeeld:

Toon de inhoud van **BEPAAALDE kolommen**  
(en alle rijen) uit de tabel "boeken"

```
SELECT id, titel  
FROM boeken;
```

id	titel
81	Kaas
82	Jan De Lichte
83	Geuzenboek
84	Het verdriet van België
94	De geruchten
96	De koele minnaar

## OPDRACHT:

Maak een resultatenlijst met  
(van links naar rechts) **titel, publicatiejaar** en **categorie-ID**  
van alle boeken in de tabel "boeken"

# De SELECT-clausule: vervelende kolomnamen

Stel dat je een tabel “flights” hebt met o.a. de kolommen “from” en “to”. Als je dan de kolom “from” wilt selecteren, dan krijg je een syntax-error.

```
SELECT from FROM flights;
```

Dit kan je oplossen door “from” te prefixen met de tabelnaam “flights” *of* door backticks (`) rond de “from” te plaatsen

```
SELECT flights.from FROM flights;  
SELECT `from` FROM flights;
```

Als je zelf een database ontwerpt, kies dan (indien mogelijk) **nooit** een “**keyword**” (SELECT, FROM, WHERE, ORDER, BY, ...) als **kolomnaam**.

# De SELECT-clausule: rekenen met kolommen

Herinner je uit de wiskunde-lessen:

- eerst de haakjes uitwerken
- dan maal (\*) en gedeeld door (/)
- dan pas plus (+) en min (-)

Voorbeelden:

```
USE db_foundations;
```

```
SELECT last_name, salary, 12*salary+1000  
FROM employees;
```

```
SELECT last_name, salary, 12*(salary+1000)  
FROM employees;
```

## OPDRACHT:

Genereer volgende  
resultatenlijst

last_name	salary	12*salary+1000	12*(salary+1000)
King	24000.00	289000.00	300000.00
Kochhar	17000.00	205000.00	216000.00



# De SELECT-clausule: tekst samenvoegen (**concat()**)

## Voorbeeld:

```
SELECT concat(first_name, ' ', last_name, ' is ', job_id)  
FROM employees;
```

## Merk op:

De kolom-kop (header)  
ziet er wel vreemd uit.

```
+-----+  
| concat(first_name, ' ', last_name, ' is ', job_id) |  
+-----+  
| Steven King is AD_PRES |  
| Neena Kochhar is AD_VP |  
| Lex De Haan is AD_VP |  
| Alexander Hunold is IT_PROG |  
| Bruce Ernst is IT_PROG |
```

# De SELECT-clausule: kolom-aliasen (**AS**)

**Voorbeeld**: een aangepast **kolomlabel** (header) bekom je als volgt:

```
SELECT concat(first_name,' ',last_name,' is ',job_id) AS "Who is who ?"  
FROM employees;
```

De “**AS**” is voor de leesbaarheid, je *mag* ze ook weglaten.

## **OPDRACHT**:

Maak een resultatenlijst met  
achternaam (**label “Naam”**)  
en jaarloon (12 x salaris + 1000, **label “Jaarloon”**)  
van elke medewerker.

Naam	Jaarloon
King	289000.00
Kochhar	205000.00
De Haan	205000.00
Hunold	109000.00
Ernst	73000.00

# De SELECT-clausule: **DISTINCT**

**Voorbeeld**: met of zonder "dubbels" in de resultatenlijst ?

```
SELECT job_id, department_id  
FROM employees;
```

→ 20 rows returned

```
SELECT DISTINCT job_id, department_id  
FROM employees;
```

→ **13** rows returned

In de praktijk zijn dubbels in de output vaak overbodig of storend.

Stel je dus steeds de vraag of het gebruik van DISTINCT niet aangewezen is.

## **OPDRACHT**:

Maak een lijst van alle manager-id's die voorkomen in de tabel "employees"  
(9 rows returned)

# De **ORDER BY**-clausule

De volgorde van de rijen in een tabel of in een resultatenlijst is *ONBESTEMD*, ze wordt bepaald door "het toeval".

```
SELECT * FROM employees;
```

ORDER BY: geeft een rangschikking aan de rijen in de resultatenlijst (hier volgens job\_id)

```
SELECT * FROM employees ORDER BY job_id;
```

Let op:

De volgorde van de employees met **dezelfde** job\_id is nog steeds *ONBESTEMD*

# De ORDER BY-clausule

ORDER BY: rangschikking op basis van *kolomnaam*:

```
SELECT * FROM employees ORDER BY job_id;
```

ORDER BY: rangschikking op basis van *kolom"nummer"*:

```
SELECT last_name, first_name, job_id  
FROM employees ORDER BY 3;
```

(in dit voorbeeld: 3<sup>de</sup> kolom, dus eigenlijk ook job\_id)

# De ORDER BY-clausule

ORDER BY: rangschikking op basis van een “expressie”

```
SELECT last_name, 12*salary+1000  
FROM employees  
ORDER BY 12*salary+1000 ;
```

ORDER BY: rangschikking op basis van een kolomalias

```
SELECT last_name, 12*salary+1000 AS 'Jaarloon'  
FROM employees  
ORDER BY Jaarloon;
```

# De ORDER BY-clausule

**Opgelet:** aanhalingstekens/quotes ( ' of " ) zijn niet toegelaten bij ORDER BY  
(MySQL geeft geen foutmelding, maar negeert de ORDER BY !!☹)

```
SELECT last_name, 12*salary+1000 AS Jaarloon  
FROM employees  
ORDER BY "Jaarloon";
```

→ geeft een fout resultaat

Een kolomalias met een spatie is dus **niet bruikbaar** in de ORDER BY-clausule

```
SELECT last_name, 12*salary+1000 AS 'Het Jaarloon'  
FROM employees  
ORDER BY 'Het Jaarloon';
```

→ geeft een fout resultaat

# De ORDER BY-clausule

## OPDRACHT:

Toon alle medewerkers, gerangschikt op salaris,  
maar toon ***enkel*** employee\_id, first\_name, last\_name

- Je kan ook **sorteren op een kolom die niet getoond wordt** in de SELECT-clausule



# De ORDER BY-clausule

## **OPDRACHT:**

Maak een resultatenlijst met alle info van alle medewerkers, gerangschikt op manager\_id

- Hebben alle medewerkers een manager\_id?  
Wat gebeurt er met de NULL-values?  
(in MySQL: NULLs first / in Oracle DB: NULLs last / SQL Server: ???)

# De ORDER BY-clausule ordenen in meerdere rangordes

ORDER BY: je kan **op meerdere kolommen sorteren**

VOORBEELD:

```
SELECT last_name, department_id, salary  
FROM employees  
ORDER BY department_id, salary;
```

→ Eerst op *department\_id*, binnen hetzelfde department op *salary*

# De ORDER BY-clausule

**OPDRACHT**: een resultatenlijst met alle medewerkers,

-- gerangschikt volgens salaris (laagste salaris bovenaan)

EMPLOYEE ID	FIRST NAME	LAST NAME	EMAIL	PHONE NUMBER	HIRE DATE	JOB ID	SALARY	COMMISSION PCT
144	Peter	Vargas	PVARGAS	650.121.2004	1998-07-09	ST_CLERK	2500.00	NU
143	Randall	Matos	RMATOS	650.121.2874	1998-03-15	ST_CLERK	2600.00	NU
142	Curtis	Davies	CDAVIES	650.121.2994	1997-01-29	ST_CLERK	3100.00	NU
141	Trenna	Dei	TDAIS	650.121.8000	1995-10-17	ST_CLERK	3500.00	NU

-- **VOORBEELD** : een resultatenlijst met alle medewerkers,

-- gerangschikt volgens anciënniteit (hoogste anciënniteit bovenaan)

EMPLOYEE ID	FIRST NAME	LAST NAME	EMAIL	PHONE NUMBER	HIRE DATE	JOB ID
100	Steven	King	SKING	515.123.4567	1987-06-17	AD_PFI
200	Jennifer	Whalen	JWHALEN	515.123.4444	1987-09-17	AD_AS
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1989-09-21	AD_VF
103	Alexander	Hunold	AHUNOLD	590.423.4567	1990-01-03	IT_PRI

-- Hoe rangschik je het hoogste salaris of de laagste anciënniteit bovenaan? Dat zie je straks.

# De ORDER BY-clausule: **ASC** en **DESC** oplopend ordenen, aflopend ordenen

-- VOORBEELDEN:

```
SELECT last_name, department_id, salary  
FROM employees  
ORDER BY department_id, salary DESC;    → DESCENDING = aflopend
```

```
SELECT last_name, department_id, salary  
FROM employees  
ORDER BY department_id DESC, salary DESC; → DESCENDING = aflopend
```

```
SELECT last_name, department_id, salary  
FROM employees  
ORDER BY department_id ASC, salary DESC; → ASCENDING = oplopend  
                                              = default behavior
```

# De ORDER BY-clausule: **LIMIT** en **OFFSET**

SELECT employee\_id, first\_name FROM employees

ORDER BY employee\_id **LIMIT** 3 ;

→ toon enkel de eerste 3 lijnen  
(dus: de 1<sup>ste</sup>, de 2<sup>de</sup> en de 3<sup>de</sup> lijn)

SELECT employee\_id, first\_name FROM employees

ORDER BY employee\_id **LIMIT** 3 **OFFSET** 6 ;

→ sla de eerste 6 lijnen over  
en toon de volgende 3 lijnen  
(dus: lijn 7, 8 en 9)

Andere notatie:

SELECT employee\_id, first\_name FROM employees

ORDER BY employee\_id **LIMIT** 6 , 3 ;

→ idem als LIMIT 3 OFFSET 6

## **OPDRACHT:**

Toon de 4<sup>de</sup> medewerker indien gerangschikt op employee\_id (Antw: Alexander)