DE HAAGSE HOGESCHOOL

Reader SQL

(DB-bevragen)

INF-B

Documentbeheer

Auteur	Datum	Versie
Merie Heijne	oktober 2003	1.0 voltijd
Paul Breukel	oktober 2003	2.0 voltijd
Merie Heijne	04-03-2004	3.0 voltijd
Annemarie Keijzer-Scholtes	11-11-2004	1.0 deeltijd
Annemarie Keijzer-Scholtes	november 2005	2.0 deeltijd
Arno Nederend	November 2005	2.1 voltijd
Rianne Béchet	Augustus 2007	2.2 voltijd
Hélène Weenink	Maart 2010	2.3 voltijd
Brigitte Derks	November 2013	2.4

INHOUDSOPGAVE

1. SQL: Query op 1 tabel

1.1 Inleiding

SQL (Structured Query Language) is een taal waarmee gegevens uit een (relationele) database opgehaald en bewerkt kunnen worden en waarmee een (relationele) database gemaakt kan worden. Het is dus zowel een *data manipulation language* (DML) als een *data definition language* (DDL). Met behulp van SQL kunnen opdrachten aan het DBMS gegeven worden om met een database te werken.
SQL is gebaseerd op het relationele model. In 1986 werd de eerste versie van SQL gestandaardiseerd door het ANSI (American National Standards Institute). Momenteel werken de meeste SQL producten met SQL-2 (1992). SQL-3 is op dit moment nog in ontwikkeling. SQL wordt meestal uitgesproken als S-Q-L, sommigen zeggen "sequel" dit heeft te maken met de naam van een voorganger van SQL, Sequel (Standard English Query Language).

Alle SQL-statements in dit en volgende hoofdstukken hebben betrekking op de voorbeeldtabellen S*tudent, Docent, Module en Resultaat.*

Docent (code, naam, geslacht, in_dienst, salaris)

code	naam	geslacht	in_dienst	salaris
kp	Kaptein	v	1/11/92	4600
bk	Bakker	m	1/9/96	3500
al	Alkemade	m	1/1/94	4500

Student (idcode, naam leeftijd, d_code)

idcode	naam	leeftijd	d_code
20015588	Jan	22	al
20013473	Els	21	kp
20016789	Harun	19	al
20010800	Thea	21	
20012300	Said	20	al

Module (code, naam, studiepunten, coördinator)

code	naam	studiepunten	coördinator
pp1	programmeren1	3	al
db1	databases1	2	bk
cs1	computersystemen1	2	bk
wd1	webdesign1	1	al

Resultaat (idccode, mcode, datum, cijfer)

idcode	mcode	datum	cijfer
20015588	pp1	15-03-01	8
20015588	db1	18-03-01	7
20016789	pp1	15-03-01	9
20013473	pp1	15-03-01	7
20013473	db1	18-03-01	8

1.2 Het Select-statement

1.2.1 From en Where

Met een select kun je gegevens opvragen uit een of meerdere tabellen. Vereenvoudigd ziet het select statement er als volgt uit:

SELECT <kolomna(a)m(en)>
FROM <tabelna(a)m(en)>
[WHERE <voorwaarde(n)>] ;

In hoofdletters staan de verplichte woorden, tussen < > namen die je zelf in moet vullen. De tekst tussen [ ] is niet verplicht. Sluit elk SQL-statement af met een punt-komma.

Wanneer je de namen van alle studenten wilt weten, kun je dat in SQL als volgt weergeven:
select naam
from student;

Het resultaat van deze query is weer een tabel en heeft de volgende inhoud:

naam
Jan
Els
Harun
Thea
Said

Van iedere rij in de tabel student wordt de inhoud van de kolom naam afgedrukt.
Je ziet dat de query in SQL in kleine letters is gegeven en dat de select en de from op een aparte regel staan. Dit hoeft niet, maar dit is volgens de lay-out regels die hier op school gelden. Het is de bedoeling dat je die ook volgt.

Je kunt ook meerdere kolommen afdrukken. Je wilt bijvoorbeeld de naam, idcode en leeftijd van iedere student weten:
select naam, idcode, leeftijd
from student;

Dit levert op:

naam	idcode	leeftijd
Jan	20015588	22
Els	20013473	21
Harun	20016789	19
Thea	20010800	21
Said	20012300	20

Je ziet dat de kolommen afgedrukt worden in de volgorde zoals ze in de select genoemd staan.

Achter de select kunnen ook berekeningen staan (uiteraard alleen bij numerieke waarden), deze kun je een eigen naam geven. Daarnaast heb je de mogelijkheid om kolommen een andere naam te geven. Beide kun je met behulp van = of **as** doen.
select docentcode = code , salaris * 12, salaris * 12 as jaarsalaris
from docent;

docentcode	salaris*12	jaarsalaris
kp	55200	55200
bk	42000	42000
al	54000	54000

Je kunt ook kolommen toevoegen:
select 'maandsalaris', salaris
from docent;

'maandsalaris'	salaris
maandsalaris	4600
maandsalaris	3500
maandsalaris	4500

Samenvattend: Na de select kun je aangeven welke kolommen getoond moeten worden, na de from geef je aan welke tabel(len) gebruikt gaat worden.

Stel je voor dat je nu alleen de naam en de leeftijd van de studenten wilt zien die als studiebegeleider 'al' hebben. Je wilt dus niet alle rijen uit de tabel zien, maar alleen die rijen waarbij de studiebegeleider 'al' is. Dit kun je aangeven na de where. De vraag "Geef de naam en leeftijd van de studenten die als studiebegeleider 'al' hebben" luidt in SQL:
select naam, leeftijd
from student
where d_code = 'al';

Dit geeft het volgende resultaat:

naam	leeftijd
Jan	22
Harun	19
Said	20

Na de where geef je aan aan welke voorwaarden de rij of tupel moet voldoen. De bovenstaande query wordt als volgt uitgevoerd; voor iedere rij in student wordt bekeken of deze rij voldoet aan de voorwaarde, dus of d_code gelijk is aan 'al', als dit waar is dan worden van deze rij de kolommen naam en leeftijd weergegeven.

De volgorde van uitvoering van de select is dus:
1. from haal de tabel op uit de database
2. where selecteer de rijen die gevraagd worden
3. select selecteer de kolommen die gevraagd worden en druk deze af.

Je ziet dat het ook bij deze query belangrijk is dat je op de hoogte bent van het datatype van de kolom d_code; is het alfanumeriek of numeriek. D-code is alfanumeriek en een waarde van d_code moet je tussen ' ' zetten. In de voorbeeld tabellen zijn kolommen salaris, leeftijd, studiepunten en cijfer numeriek, alle andere zijn alfanumeriek (in _dienst en datum zijn speciaal die zijn van het type date).

Je kunt na de where meer dan een voorwaarde aangeven, de verschillende voorwaarden kun je met elkaar combineren met een OR of AND, afhankelijk van de gegevens die je wilt hebben. Je wilt bijvoorbeeld alle gegevens van de studenten die als studiebegeleider al of kp hebben. In SQL :

select *
from student
where d_code = 'al' or d_code = 'kp';

idcode	naam	leeftijd	d_code
20015588	Jan	22	al
20013473	Els	21	kp
20016789	Harun	19	al
20012300	Said	20	al

Select * wil zeggen geef alle kolommen uit de tabel student. Je kunt natuurlijk ook alle kolommen noemen.
Een ander voorbeeld, je wilt alle namen van de studenten waarvan de studiebegeleider al is en die ouder zijn dan 20. In SQL:
select naam
from student
where d_code = 'al' and leeftijd > 20;

naam
Jan

Na de **where** staan dus een of meer voorwaarden; deze voorwaarden kunnen de volgende vormen aannemen:

1. Vergelijking. **where salaris = 3500**
   Operatoren die je kunt gebruiken zijn = > < >= <= en <>
2. Bereik **where leeftijd between 20 and 25**
3. Deelname **where leeftijd in (19,20,21)**
   Tussen de haakjes kunnen alleen constanten staan, deze constanten moeten van het zelfde datatype zijn.
4. Patroon **where naam like 'A%'**
   Het % teken betekent een of meer willekeurig karakters, het _ teken (de underscore) staat voor precies een karakter.
5. Null **where d_code is null**
   Let op **is** en niet = . Dit is omdat een NULL waarde nooit gelijk kan zijn aan een andere NULL waarde. Als dat zo zou zijn dan zou dat betekenen dat je iets weet van die NULL waarde. Het is overigens vreemd om te spreken van een NULL *waarde*, in de literatuur wordt dit echter meestal gedaan.

De voorwaarden of condities kunnen dus gecombineerd worden met behulp van **and** en **or.**
De condities worden van links naar rechts geevalueerd, met haakjes kan de volgorde veranderd worden.
where leeftijd = 19 or naam like 'S%' and d_code is null

Stel je wilt de naam weten van alle studenten die geen studiebegeleider hebben en van wie de naam begint met een T. In SQL;
select naam
from student
where d_code is null and naam like 'T%';

naam
Thea

Bovenstaande condities kunnen ook gecombineerd worden met **not**.

- **where not salaris = 3500**
- **where leeftijd not between 20 and 25**
- **where leeftijd not in (19,20)**
- **where naam not like 'A%'**
- **where d_code is not null;**

Je wilt de namen van alle modulen die niet met een P beginnen en die geen 3 studiepunten hebben. In SQL:

```
select naam
from module
where naam not like 'P%' and not studiepunten = 3;
```

naam
databases1
computersystemen1
webdesign1

Je kunt natuurlijk ook bij de tweede voorwaarde **studiepunten <> 3** gebruiken.

1.2.2 Distinct

Het is mogelijk om dezelfde waarde slechts een keer weer te geven. Stel je wilt weten welke docenten studiebegeleider zijn. In SQL:

select d_code
from student;

d_code
al
kp
al
al

Voor iedere rij in de tabel student wordt nu de d_code afgedrukt, en dat betekent dat al drie keer wordt getoond. Met behulp van distinct kun je duplicaten elimineren. Alleen verschillende waarden worden getoond.

select distinct d_code
from student;

d_code
al
kp

Je ziet dat ook de niet bekende studiebegeleider van Thea wordt afgedrukt. Een NULL waarde is dus anders dan een niet-NULL waarde.

Wat nu als er meer dan een NULL waarde in de tabel staat; student Kees is toegevoegd en heeft geen studiebegeleider:

idcode	naam	leeftijd	d_code
20015588	Jan	22	al
20013473	Els	21	kp
20016789	Harun	19	al
20010800	Thea	21	
20012345	Kees	19	
20012300	Said	20	al

select distinct d_code
from student;

d_code
al
kp

Je zou verwachten dat twee maal een lege kolom gegeven gaat worden; een NULL waarde is niet gelijk aan een andere NULL waarde. Bij de distinct wordt er echter van de betekenis uitgegaan; de waarde onbekend komt meerdere keren voor en wordt dus één keer getoond als duplicaten weggelaten moeten worden.

1.2.3 Order by

Je kunt het eindresultaat ook gesorteerd weer laten geven, hiervoor gebruik je de order by. Bijvoorbeeld: 'Geef alle gegevens van de docenten en sorteer dit op naam' . In SQL:
select *
from docent
order by naam;

code	naam	geslacht	in dienst	salaris
al	Alkemade	m	1/1/94	4500
bk	Bakker	m	1/9/96	3500
kp	Kaptein	v	1/11/92	4600

Het resultaat wordt oplopend gesorteerd (asc, ascending) Als je wilt dat er aflopend gesorteerd wordt, moet je desc (descending) opnemen dus
order by naam desc;

Houd goed in de gaten dat de order by alleen iets zegt over de wijze waarop het resultaat wordt afgedrukt; het gaat uitsluitend om de lay-out ! De order by staat als laatste statement in de select en wordt ook als laatste uitgevoerd. Dus bij "Geef alle gegevens van de mannelijke docenten en sorteer dit op naam":
select *
from docent
where geslacht = 'm'
order by naam;

Uitvoering:
1. from; haal de tabel docent uit de database
2. where; selecteer alle rijen waarvan het geslacht 'm' is
3. select; selecteer alle kolommen
4. order by; sorteer dit op naam en druk af

1.2.4 Kolomfuncties.

De vorige SQL statements hadden betrekking op gegevens die ergens in rij/tupel van de tabel zouden kunnen voorkomen. Het is ook mogelijk om query's samen te stellen over gegevens die niet als zodanig in de tabel staat, maar met behulp van de gegevens uit de tabel afgeleid kunnen worden. Hiervoor maak je gebruik van de kolomfuncties.

count(*)	telt het aantal rijen
count(leeftijd)	telt het aantal rijen waarbij leeftijd niet NULL is)
count (distinct leeftijd)	telt het aantal rijen waarbij duplicaten en NULL waarden niet meegeteld worden.
sum(salaris)	telt alle niet NULL salarissen bij elkaar op.
avg(salaris)	bepaalt het gemiddelde van alle niet NULL salarissen.
min(leeftijd)	bepaalt de kleinste waarde van leeftijd (NULL uitgezonderd).
max(leeftijd)	bepaalt de grootste waarde van leeftijd (NULL uitgezonderd).

De sum en avg functie kunnen (uiteraard) alleen bij numerieke waarden gebruikt worden. Een kolomfunctie levert als resultaat altijd precies een rij; één waarde.
Bijvoorbeeld: 'Geef het aantal studenten'. In SQL:

select count(*)
from student;

count(*)
6

Opmerking: Als kolomnaam wordt hier count(*) gegeven, dit is niet bij alle implementaties van SQL het geval.

Een ander voobeeld: 'Geef het aantal studenten, het aantal studenten met een studiebegeleider, het aantal studenten met een verschillende studiebegeleider, de laagste en de hoogste leeftijd'
In SQL:
select count(*), count(d_code), count (distinct d_code), min(leeftijd), max(leeftijd)
from student;

count(*)	count(d code)	count(distinct d code)	min(leeftijd)	max(leeftijd)
5	4	2	19	22

Let op dat de kolomfuncties dus geen NULL meenemen op count(*) na.
Dat betekent dat op de vraag 'Geef het gemiddelde salaris van de docenten' de query
select sum(salaris) / count(*)
from docent;

een ander antwoord zou kunnen geven dan
select avg(salaris)
from docent;

Zoals gezegd geven kolomfuncties precies één waarde op; in bovenstaande query wordt van alle rijen uit de tabel docent, het gemiddelde salaris berekend. Dit betekent dat je geen kolomnaam kunt noemen in de select bij een kolomfunctie. Dus niet:
select naam, min(leeftijd) // foutmelding !!!
from student;

Deze query kan niet uitgevoerd worden, want er staan 5 rijen in de tabel, van de 5 rijen wordt 5 keer een naam geselecteerd maar min(leeftijd) levert precies één waarde op, namelijk voor de hele groep studenten wordt bepaald wie de jongste is.

1.2.5 Group by

Het is wel mogelijk om een functie te gebruiken voor meer dan een groep. In bovenstaande voorbeelden werd steeds gekeken naar alle studenten. Het is echter mogelijk om binnen studenten weer groepjes te maken.
select d_code, count(*)
from student
group by d_code;

Het statement wordt als volgt uitgevoerd;

1. from; haal de tabel student uit de database
2. group by d_code; zet de rijen uit student met dezelfde d_code bij elkaar dus:

alle studenten met d_code al : 20015588 Jan 22
20016789 Harun 19
20012300 Said 20

alle studenten met d_code kp: 20013473 Els 21

alle studenten met d_code NULL 20010800 Thea 21

3. select; selecteer d_code en tel per d_code het aantal rijen en druk dit af

d_code	count(*)
	1
al	3
kp	1

Stel dat je niet wilt dat NULL mee wordt genomen. Hoe moet de query dan aangepast worden ?

Een ander voorbeeld: 'Geef per coordinator het aantal modules waarbij alleen die modules geteld worden die meer dan een studiepunt hebben'. In SQL:

```
select coordinator, count(*)
from module
where studiepunten > 1
group by coordinator;
```

coordinator	count(*)
al	1
bk	2

Uitvoering:
1. from; haal de tabel module uit de database
2. where; selecteer die rijen waarbij studiepunten > 1 is
3. group by; zet de rijen met dezelfde coordinator bij elkaar
4. select; selecteer coordinator en tel het aantal rijen per coordinator en druk beide af.

Het zal duidelijk zijn dat de kolomnaam die je noemt in de select ook de kolomnaam is waarop groepjes gevormd worden.
Je kunt op meer dan een kolom groeperen:

```
select coordinator, code, count(*)
from module
where studiepunten > 1
group by coordinator, code;
```

In dit geval heeft dat niet zoveel zin omdat code steeds uniek is. De query geeft de volgende uitvoer:

coordinator	code	count(*)
bk	cs1	1
bk	db1	1
al	pp1	1

2 SQL: Query op meerdere tabellen

2.1 Queries op meerdere tabellen: de join

Het komt natuurlijk regelmatig voor dat je gegevens uit meer dan een tabel wilt hebben. Je wilt bijvoorbeeld de d_code en het salaris van de docent weten die studiebegeleider is van student 20013473. Je hebt hiervoor de tabel student nodig om te zien welke d_code bij deze student hoort en vervolgens kun je met die d_code het salaris van de docent achterhalen. Wellicht zou je denken dat de query om bovenstaande op te lossen er als volgt uit ziet. Maar deze query geeft een verkeerd resultaat !

select d_code, code, salaris // code uit docent wordt ook gevraagd ter verduidelijking//
from student, docent
where idcode = '20013473';

d_code	code	salaris
kp	kp	4600
kp	bk	3500
kp	al	4500

Hoe komt dit nu? Na de from staan twee tabellen genoemd. De from wordt als eerste uitgevoerd en beide tabellen worden uit de database gehaald. De rijen van docent en student worden met elkaar gecombineerd, er wordt een *cartesisch product* van deze tabellen gemaakt. Dus alle rijen van student worden gecombineerd met alle rijen van docent. In totaal 3 * 5 = 15 rijen !

Ter illustratie:
select *
from student, docent;

Uitvoering:
De query wordt als volgt uitgevoerd:
1. from; haal alle rijen van student en docent uit de database en maak een cartesisch product
2. select; selecteer alle kolommen

idcode	naam	leeftijd	d_code	code	naam	geslacht	in_dienst	salaris
20015588	Jan	22	al	kp	Kaptein	v	1-11-92	4600
20015588	Jan	22	al	bk	Bakker	m	1-9-96	3500
20015588	Jan	22	al	al	Alkemade	m	1-1-94	4500
20013473	Els	21	kp	kp	Kaptein	v	1-11-92	4600
20013473	Els	21	kp	bk	Bakker	m	1-9-96	3500
20013473	Els	21	kp	al	Alkemade	m	1-1-94	4500
20016789	Harun	19	al	kp	Kaptein	v	1-11-92	4600
20016789	Harun	19	al	bk	Bakker	m	1-9-96	3500
20016789	Harun	19	al	al	Alkemade	m	1-1-94	4500
20010800	Thea	21		kp	Kaptein	v	1-11-92	4600
20010800	Thea	21		bk	Bakker	m	1-9-96	3500
20010800	Thea	21		al	Alkemade	m	1-1-94	4500
20012300	Said	20	al	kp	Kaptein	v	1-11-92	4600
20012300	Said	20	al	bk	Bakker	m	1-9-96	3500
20012300	Said	20	al	al	Alkemade	m	1-1-94	4500

Uit deze tabel worden nu vervolgens de gegevens gehaald. Dit verklaart de 3 rijen die je krijgt bij de query:

```
select  d_code , code, salaris
from student, docent
where idcode = '20013473';
```

want de 4de,5de en 6de rij voldoen aan de voorwaarde die je noemt bij de where.
De rij die je wilt hebben is uiteraard rij 4. Hoe kun je dat voor elkaar krijgen ? Door gebruik te maken van het attribuut dat beide tabellen koppelt namelijk d_code en code. Je gaat dus gebruik maken van de vreemde sleutel in student, d_code, die verwijst naar de primaire sleutel in docent.
De query wordt als volgt:

```
select  d_code, code, salaris
from student, docent
where idcode = '20013473' and d_code = code;
```

d_code	code	salaris
kp	kp	4600

Het combineren van meerdere tabellen op deze manier noem je een inner join. Je haalt gegevens op uit tabellen op grond van een waarde die in beide tabellen voorkomt.

De join kun je ook op een andere manier aangeven in SQL:

```
select d_code, code, salaris
from student join docent on d_code = code
where idcode = '20013473';
```

Deze manier zal verder gebruikt worden.
Wanneer je tabellen combineert, kan het zijn dat een naam van een kolom meer dan een keer voorkomt. In dit voorbeeld is dit naam. Je kunt deze kolommen van elkaar onderscheiden door voor de kolom de naam van de tabel te noemen, docent.naam en student.naam
De vraag: geef de naam van de student en de naam van de studiebegeleider ,wordt in SQL:

```
select student.naam, docent.naam
from student join docent on d_code = code;
```

naam	naam
Jan	Alkemade
Els	Kaptein
Harun	Alkemade
Said	Alkemade

Aan de uitvoer van de query kun je goed zien dat het om een inner join gaat; de student zonder studiebegeleider is niet opgenomen en ook de docent zonder studenten doet niet mee.

Een voorbeeld met drie tabellen:
'Geef de namen van de studenten die een resultaat hebben voor de module programmeren1'

Om deze vraag te kunnen beantwoorden, heb je gegevens uit de tabel student, module en resultaat nodig. Student moet je koppelen aan resultaat via idcode en module moet je koppelen via code en mcode. In SQL:

```
select student.naam
from student join resultaat on student.idcode = resultaat.idcode
             join module on module.code = resultaat.mcode
where module.naam = 'programmeren1';
```

naam
Jan
Harun
Els

Een voorbeeld met vier tabellen:
'Geef de namen van de docenten die een module coordineren waar studenten een 9 voor gehaald hebben en geef de namen van de studenten' wordt in SQL:

```
select docent.naam, student.naam
from docent join module on docent.code  = coordinator
            join  resultaat on module.code = mcode
            join   student on resultaat.idcode = student.idcode
where cijfer = 9;
```

naam	naam
Alkemade	Harun

Wanneer je het omslachtig vindt om steeds de naam van de tabel te noemen, kun je ook gebruik maken van een alias!

```
select s.naam, d.naam
from student s join docent d  on d_code = code;
        // i.p.v. student kun je s , i.p.v. docent kun je d gebruiken
```

Je kunt ook een tabel met zichzelf 'joinen'. Stel dat je een overzicht wilt hebben van alle studenten met dezelfde leeftijd als Els (en je weet de leeftijd van Els niet).
Je kunt niet volstaan met alleen de tabel student want je kent de leeftijd van Els niet. Je moet eigenlijk alle rijen van student af en die rijen vergelijken met de leeftijd van Els. Je hebt dus de tabel student 2 keer nodig. Bij een query zoals deze *moet* je een alias gebruiken; je gebruikt namelijk 2 maal dezelfde tabel. Hieronder zie je het cartesisch product van student * student

idcode	naam	leeftijd	d code	idcode	naam	leeftijd	d code
20015588	Jan	22	al	20015588	Jan	22	al
20013473	Els	21	kp	20015588	Jan	22	al
20016789	Harun	19	al	20015588	Jan	22	al
20010800	Thea	21		20015588	Jan	22	al
20012300	Said	20	al	20015588	Jan	22	al
20015588	Jan	22	al	20013473	Els	21	kp
20013473	Els	21	kp	20013473	Els	21	kp
20016789	Harun	19	al	20013473	Els	21	kp
20010800	Thea	21		20013473	Els	21	kp
20012300	Said	20	al	20013473	Els	21	kp
20015588	Jan	22	al	20016789	Harun	19	al
20013473	Els	21	kp	20016789	Harun	19	al
20016789	Harun	19	al	20016789	Harun	19	al
20010800	Thea	21		20016789	Harun	19	al
20012300	Said	20	al	20016789	Harun	19	al
20015588	Jan	22	al	20010800	Thea	21	
20013473	Els	21	kp	20010800	Thea	21	
20016789	Harun	19	al	20010800	Thea	21	
20010800	Thea	21		20010800	Thea	21	
20012300	Said	20	al	20010800	Thea	21	
20015588	Jan	22	al	20012300	Said	20	al
20013473	Els	21	kp	20012300	Said	20	al
20016789	Harun	19	al	20012300	Said	20	al
20010800	Thea	21		20012300	Said	20	al
20012300	Said	20	al	20012300	Said	20	al

De SQLquery om alle gegevens van de studenten met dezelfde leeftijd als Els af te drukken is dan:

```
select  A.idcode, A.naam, A.leeftijd
from student A join student B on A.leeftijd = B.leeftijd
where  B.naam = 'Els';
```

idcode	naam	leeftijd
20013473	Els	21
20010800	Thea	21

2.2 Queries op meerdere tabellen: de subselect

Subquery's of subselects zijn query's die opgenomen worden binnen een andere query.
Voorbeeld: Je wilt de namen van de docenten die coordinator van een module zijn. Dat kun je met behulp van een join oplossen, maar ook met behulp van een subselect.
Bekijk de volgende query:

```
select naam
from docent
where code in ( select coordinator from module );
```

De verwerking hiervan gaat als volgt. Eerst wordt de subselect
(select coordinator from module) uitgevoerd.

Dit levert een verzameling van waarden op. In dit geval ('al', 'bk')
Daarna wordt de eerste select uitgevoerd dus:

```
select naam
from docent
where code in ('al', 'bk');
```

Voor iedere rij in docent wordt vervolgens gekeken of code een waarde heeft die in deze deelverzameling voorkomt. Indien dit het geval is, wordt naam afgedrukt.

naam
Alkemade
Bakker

Met een subselect wordt dus eerst de subselect uitgevoerd en vervolgens de select.
Belangrijk is dat wat voor de **in** staat in dit geval **code** van hetzelfde datatype is als de waarden in de deelverzameling.
In de subselect kan altijd maar één expressie (kolomnaam, kolomfunctie) staan. In de subselect kan geen distinct of order by gebruikt worden.
Een subselect kan zelf ook weer een subselect bevatten. Bijvoorbeeld 'Geef de namen van de studenten die voor de module programmeren1 een resultaat gehaald hebben'. In SQL:

```
select naam
from student
where idcode in (select idcode
                 from resultaat
                 where mcode in ( select mcode
                                  from module
                                  where naam = 'programmeren1'));
```

naam
Jan
Harun
Els

Eerst wordt **select mcode from module where naam = 'programmeren1'** uitgevoerd. Dit levert precies één waarde op, namelijk 'pp1'. Vervolgens wordt **select idcode from resultaat where mcode in ('pp1')** uitgevoerd, dit levert een aantal waarden op namelijk 20015588, 20013473 en 20016789. Tenslotte wordt **select naam from student where idcode in ('20015588', '20013473', '20016789')**

De IN operator moet altijd gebruikt worden als het om meerdere waarden gaat. Weet je zeker dat je subquery slecht één waarde oplevert dan kun je ook = gebruiken. Toegepast in het vorige voorbeeld:

```
select naam
from student
where idcode in (select idcode
                 from resultaat
                 where mcode  = ( select mcode
                                  from module
                                  where naam = 'programmeren1'));
```

Er zijn situaties waarbij je alleen een join kunt gebruiken, namelijk als je gegevens uit meer dan een tabel wilt afdrukken. Bij een subselect kun je alleen de gegevens van de eerste select, in dit geval student, weergeven. De volgende query is bijvoorbeeld niet als een subselect op te lossen:

```
select student.naam, docent.naam
from student join docent on d_code = code;
```

De vraag 'Geef alle studenten met dezelfde leeftijd als 'Els' is met een subselect heel eenvoudig op te lossen:

```
select naam, leeftijd
from student
where leeftijd in (select leeftijd
                   from student
                   where naam = 'Els');
```

Soms kun je een antwoord alleen met een subselect vinden. Stel je wilt de namen van alle docenten die geen coordinator zijn afdrukken. Dit kan niet met een join, die is namelijk gebaseerd op overeenkomst en niet op verschil. De query:

```
select docent.naam
from docent join module on docent.code <> coordinator;
```

geeft het volgende, onjuiste resultaat:

naam
Kaptein
Bakker
Kaptein
Alkemade
Kaptein
Alkemade
Kaptein
Bakker

Met een subselect wordt de SQL-query:

```
select naam
from docent
where code not in (select coordinator
                   from module);
```

naam
Kaptein

2.3 Queries met subqueries

In een query kan een andere query opgenomen worden, dit wordt een subquery of een geneste query genoemd. Het resultaat van de geneste query wordt gebruikt voor selectie in de buitenste query.

Geef de namen van de studenten met als studiebegeleider ‘Kaptein’:

```
select naam
from student
where d_code = (select code
                from docent
                where naam = ‘Kaptein’);
```

naam
Els

De geneste query heeft als resultaat precies één waarde namelijk ‘kp’ en daarmee wordt iedere rij uit de buitenste query mee vergeleken en indien waar afgedrukt dus:

```
select naam
from student
where d_code = (‘kp’)
```

Wanneer de geneste query meer dan een resultaat oplevert, kun je de = (of <>, <, <=, >, >=) niet gebruiken, dan gebruik je de IN (of de NOT IN):

```
select naam
from student
where d_code in (select code
                 from docent
                 where geslacht = ‘m’);
```

naam
Jan
Harun
Said

De geneste query levert op (‘bk’, ‘al’), daarmee worden vervolgens alle rijen van student mee vergeleken. In beide voorbeelden werden verschillende tabellen in de buitenste en geneste query gebruikt en werd er via primaire en vreemde sleutel gegevens opgehaald. Beide queries zijn een variant op de join. Er is echter een verschil, bij de subselect kun je geen gegevens van de geneste query tonen, in beide queries kun je alleen gegevens van de student laten zien, niet van de docent. In sommige situaties kun je alleen een subselect gebruiken:

Geef de namen van de studenten die nog geen resultaat hebben:

```
select naam
from student
where idcode not in (select idcode
                     from resultaat);
```

naam
Said

Je kunt hier geen join gebruiken omdat in de join conditie gezocht wordt naar een overeenkomst tussen primaire en vreemde sleutel. (Waarom kun je dit niet oplossen met <> ?)

Een ander voorbeeld waarbij je alleen de subselect kunt gebruiken:

Geef de namen van de studenten die ouder zijn dan de gemiddelde leeftijd van alle studenten:

```
select naam
from student
where leeftijd > (select AVG(leeftijd)
                  from student);
```

naam
Jan
Els
Thea

of
Geef de namen van de studenten die wel een resultaat hebben voor pp1 maar niet voor db1

```
select naam
from student
where idcode in (select idcode
                 from resultaat
                     where mcode = ‘pp1’)
      and
      idcode not in (select idcode
                     from resultaat
                     where mcode = ‘db1’);
```

naam
Els
Harun
Thea

Bij de NOT in doet zich een probleem voor indien er NULL als waarden in de geneste tabel staan:
Geef de namen van de docenten die geen studiebegeleider zijn

```
select *
from docent
where code not in ( select d_code
                    from student);
```

Deze query geeft het volgende resultaat:

code	naam	geslacht	in_dienst	salaris
				0

Hoe kan dit nu ? Dit komt omdat in SQL het resultaat van een vergelijking true, false of unknown kan zijn. Bij een vergelijking met NULL is het resultaat unknown. Dus

docent.code not in (‘al’, ‘kp’, NULL) geeft

al not in (‘al’, ‘kp’, NULL) ----> false
kp not in (‘al’, ‘kp’, NULL) ----> false
bk not in (‘al’, ‘kp’, NULL) -----> unknown.

De tupel met docent bk wordt dus niet gegeven, want de vergelijking levert niet true op. We willen dit uiteraard wel zien. Dat betekent dat de query als volgt uitgebreid moet worden:

```
select *
from docent
where code not in ( select d_code
                    from student
                    where d_code is not null);
```

Dit geeft het gewenste resultaat:

code	naam	geslacht	in_dienst	salaris
Bk	Bakker	m	1-9-1996	3500

3 SQL: Outer joins

Van alle modulen wordt via de vreemde sleutel coördinator de naam die bij deze coördinator hoort in docent opgehaald. Wat gebeurt er nu als er geen vreemde sleutel staat aangegeven bij een module, de coördinator van databases1 is bijvoorbeeld nog niet bekend ?

code	naam	studiepunten	coordinator
pp1	programmeren1	3	al
db1	databases1	2	
cs1	computersystemen1	2	bk
wd1	webdesign1	1	al

Het resultaat van

select module.naam, docent.naam
from module join docent on docent.code = coordinator;

is dan:

module.naam	docent.naam
programmeren1	Alkemade
computersystemen1	Bakker
webdesign1	Alkemade

Met andere woorden alleen die modulen die een coordinator hebben worden weergegeven. Deze join wordt ook wel de innerjoin genoemd. Alleen die rijen die daadwerkelijk een koppeling hebben naar de andere tabel worden weergegeven.
Wanneer je wilt dat ook de modulen die geen coordinator hebben weergegeven worden, moet je een outerjoin gebruiken. De outerjoin kan een left outerjoin of een right outerjoin zijn.

Vergelijk:

select module.naam, docent.naam
from module left join docent on docent.code = coordinator;

module.naam	docent.naam
programmeren1	Alkemade
databases1	
computersystemen1	Bakker
webdesign1	Alkemade

select module.naam, docent.naam
from module right join docent on docent.code = coordinator;

module.naam	docent.naam
	Kaptein
computersystemen1	Bakker
webdesign1	Alkemade
programmeren1	Alkemade

Left outerjoin betekent: neem ALLE rijen van module, de tabel links van het woord join. Dus alle rijen van module worden getoond met wel of niet een docentnaam
Right outerjoin betekent: neem ALLE rijen van docent, de tabel rechts van het woord join. Dus alle rijen van docent worden getoond met eventueel een modulenaam (en als een docent coördinator is van meerdere modulen, dan komt hij natuurlijk ook meerdere keren voor).

In de praktijk wordt de left join het meest gebruikt. Deze ga je in het vervolg ook gebruiken.

Wanneer gebruik je nu de outerjoin ? In ieder geval als in de vraagstelling het woord ALLE genoemd wordt, dat geeft duidelijk aan dat alle rijen van de genoemde tabel getoond moeten worden. Bijvoorbeeld, geef van ALLE studenten de naam en de naam van de studieloopbaanbegeleider. Alle rijen van student moeten getoond worden:

```
select student.naam, docent.naam
from student left join docent on d_code = code;
```

student.naam	docent.naam
Thea	
Jan	Alkemade
Harun	Alkemade
Said	Alkemade
Els	Kaptein

Bij het 'joinen' van meerdere tabellen kun je de left join en de innerjoin door elkaar gebruiken:
Bijvoorbeeld: Geef van de resultaten het cijfer en de naam van de student en geef van al deze studenten de naam van de studieloopbaanbegeleider

Hier moet een inner join gebruikt worden tussen resultaat en student en moet een outer join gebruikt worden met docent:

```
select student.naam, cijfer, docent.naam
from student  join resultaat on student.idcode = resultaat.idcode
              left outer join docent on d_code = docent.code;
```

student.naam	cijfer	docent.naam
Jan	8	Alkemade
Jan	7	Alkemade
Els	7	Kaptein
Harun	9	Alkemade
Thea	7	

Als we nu ook nog willen weten wat de naam is van de module waar het resultaat voor behaald is, dat krijgen we de volgende query:

```
select student.naam, cijfer, docent.naam, module.naam
from student  join resultaat on student.idcode = resultaat.idcode
              left outer join docent on d_code = docent.code
              join module on resultaat.mcode = module.code;
```

met de volgende uitvoer:

student.naam	cijfer	docent.naam	module.naam
Jan	8	Alkemade	programmeren1
Jan	7	Alkemade	databases1
Harun	9	Alkemade	programmeren1
Els	7	Kaptein	programmeren1
Thea	7		programmeren1

3.1 De having

```
select module.naam, count(cijfer) as aantal
from resultaat join module on  resultaat.mcode = module.code
group by module.naam
```

naam	aantal
databases1	1
programmeren1	4

Als je het aantal cijfers van iedere module wilt hebben, moet je een outer join gebruiken dus:

```
select module.naam, count(cijfer) as aantal
from  module left join resultaat on resultaat.mcode = module.code
group by module.naam
```

naam	aantal
computersystemen1	0
databases1	1
programmeren1	4
webdesign1	0

De bovenstaande queries worden als volgt uitgevoerd:

1. from: de tabellen worden geselecteerd, module en resultaat, en de rijen daaruit die voldoen aan de joinconditie worden bepaald.
2. group by: de rijen worden gegroepeerd op modulenaam
3. select: de modulenaam en het aantal cijfers per modulenaam wordt getoond.

Wanneer er eisen gesteld worden aan de rijen na de group by, er worden dus eisen gesteld aan de groep die gevormd is, bijvoorbeeld alleen de modulen met meer dan 3 cijfers moeten getoond worden, dan moet de having gebruikt worden:

```
select module.naam, count(cijfer) as aantal
from resultaat join module on resultaat.mcode = module.code
group by module.naam
having count(cijfer) > 3;
```

naam	aantal
programmeren1	4

De uitvoering van deze query::
1. from: de tabellen worden geselecteerd, module en resultaat, en de rijen daaruit die voldoen aan de joinconditie worden bepaald.
2. group by: de rijen worden gegroepeerd op modulenaam.
3. having: de groepen die voldoen aan meer dan drie cijfers worden bepaald
4. select: de modulenaam en het aantal cijfers per modulenaam wordt getoond

De having komt alleen voor in combinatie met de group by. Bij de voorwaarde in de having kunnen alleen functies (zoals de count) of constanten staan.
In de where kunnen ook voorwaarden genoemd worden, verschil met de having is dat het bij de having altijd om voorwaarden aan een groep en bij de where om voorwaarden aan een enkele rij gaat.

Vraag:
Waarom is het niet zinvol om bij de laatste query een outer join te gebruiken ?

Een tweede voorbeeld: Geef de gemiddelde leeftijd van de studenten per studiebegeleider, waarbij de gemiddelde leeftijd groter dan 20 moet zijn

```
select docent.naam, AVG(leeftijd) as gem_leeftijd
from docent join student on d_code = code
group by docent.naam
having AVG(leeftijd) >= 20;
```

naam	gem_leeftijd
Kaptein	21