howest
hogeschool

Syllabus

Databases

Mileto Di Marco
Kristien Roels
Bart Soete

Inhoudsopgave

H4 - Het raadplegen van data m.b.v. SQL

1 Inleiding

In dit hoofdstuk wordt het raadplegen van data uit een relationele databank m.b.v. SQL behandeld.
Het SQL-statement dat hiervoor gebruikt wordt, is het SELECT-statement. Dit is een zeer krachtig statement waarmee zowel eenvoudige als complexe vragen over de data in de databank kunnen opgelost worden.

De voorbeelddatabank die gebruikt wordt in dit hoofdstuk is de bibliotheek-databank.
Vergeet dus niet om van de bibliotheek-databank de actieve databank te maken vooraleer SELECT-statements uit te voeren:
USE bibliotheek

Enkele voorbeelden van vragen die kunnen opgelost worden met een SELECT-statement:

- Geef van alle boeken uit de databank de titel, de auteur en de uitgever.
- Geef per categorie het aantal boeken.
- Geef de boeken die nog nooit uitgeleend zijn.
- Geef de boeken die het meest uitgeleend zijn.

Voor een volledige bespreking van het SELECT-statement, met voorbeelden, zie
https://docs.microsoft.com/en-us/sql/t-sql/queries/select-transact-sql

2 Syntax SELECT-statement

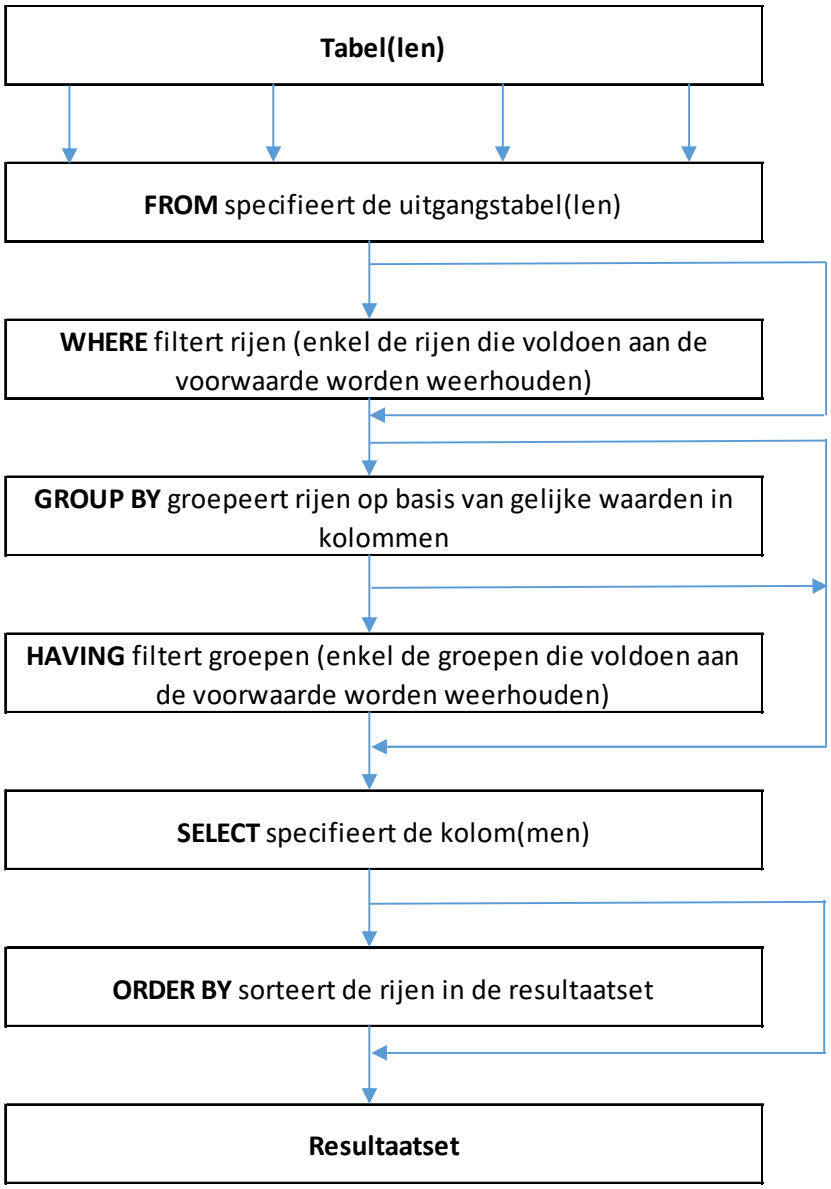
	Volgorde van uitvoeren en werking
SELECT-clausule	5. specifieert de kolom(men)
FROM-clausule	1. specifieert de tabel(len)
[WHERE-clausule]	2. filtert rijen
[GROUP BY-clausule]	3. groepeert rijen
[HAVING-clausule]	4. filtert groepen
[ORDER BY-clausule]	6. sorteert de rijen in de resultaatset

Het SELECT-statement bestaat uit een aantal clausules (componenten). Niet alle clausules zijn verplicht op te nemen: de optionele clausules staan tussen vierkante haken.

De volgorde van de clausules moet zoals in de bovenstaande syntaxbeschrijving zijn. Zo moet de WHERE-clausule ná de FROM-clausule en vóór de ORDER BY-clausule komen.

Het resultaat van een SELECT-statement is een tabel. Men spreekt over de resultaatset.

Hieronder staat een schematische voorstelling van de verwerking van een SELECT-statement:



3 De SELECT-clausule en de FROM-clausule

De eenvoudigste vorm van het SELECT-statement bestaat uit een SELECT-clausule en een FROM-clausule. Deze vorm van het SELECT-statement heeft de volgende syntax:

SELECT [DISTINCT] kolommenlijst
FROM tabel

De resultaatset van dit eenvoudigste SELECT-statement bevat alle rijen uit de tabel die vermeld wordt in de FROM-clausule.

De kolommenlijst bevat een opsomming van de kolommen (gescheiden door een komma) die in de resultaatset moeten voorkomen.
*Het is ook mogelijk om een asterisk * te schrijven: dan worden alle kolommen uit de tabel weergegeven.*

DISTINCT elimineert de duplicaatrijen in de resultaatset.

Geef de titel en het jaar (van uitgave) van alle boeken.
select titel, jaar from boeken

Geef alle gegevens van alle boeken.
select * from boeken

Geef een lijst van de plaatsen waar klanten wonen.
select distinct woonplaats from klanten

4 De ORDER BY-clausule

De ORDER BY-clausule sorteert de rijen in de resultaatset. De eenvoudigste vorm van deze clausule heeft de volgende syntax:

ORDER BY kolomnaam [ASC | DESC] [, …]

De kolomnaam geeft aan op welke kolom er gesorteerd wordt. Er kan gesorteerd worden op meerdere kolommen [, …]; deze kolommen moeten niet noodzakelijk in de SELECT-clausule staan.

ASC betekent stijgend sorteren en is de default; DESC betekent dalend sorteren.

Geef een alfabetische lijst van alle auteurs.
select * from auteurs order by familienaam, voornaam

Geef een lijst van alle boeken, gesorteerd per jaar en binnen een jaar alfabetisch op titel.
select * from boeken order by jaar, titel

Geef een lijst van alle uitleningen, gesorteerd op dalende datum en binnen een datum stijgend op boekid.
select * from uitleningen order by datum desc, boekid asc

Het aantal rijen dat weergegeven wordt, kan beperkt worden door gebruik te maken van OFFSET en FETCH. De syntax van de ORDER BY-clausule is dan als volgt:

ORDER BY kolomnaam [ASC | DESC] [, …]
OFFSET aantal ROWS
[FETCH NEXT aantal ROWS ONLY]

OFFSET geeft aan hoeveel rijen moeten overgeslagen worden, vooraleer rijen worden weergegeven.

FETCH geeft aan hoeveel rijen moeten weergegeven worden nadat OFFSET-aantal rijen zijn overgeslagen.

Geef de eerste 5 boeken volgens boekid.
select * from boeken order by boekid offset 0 rows fetch next 5 rows only

Sla de eerste 5 boeken over (volgens boekid) en geef al de andere boeken weer.
select * from boeken order by boekid offset 5 rows

Om enkel de eerste n rijen weer te geven in de resultaatset kan de SELECT-clausule uitgebreid worden met TOP (n). TOP (n) werkt steeds in samenhang met ORDER BY. De syntax is als volgt:

SELECT TOP (n) [WITH TIES] kolommenlijst
FROM-clausule
ORDER BY-clausule

TOP (n) geeft aan dat de eerste n rijen (volgens de sorteervolgorde) moeten weergegeven worden.

De laatste plaats kan met WITH TIES ingenomen worden door verschillende rijen als die rijen dezelfde waarde hebben in de kolommen gebruikt in de ORDER BY-clausule.

Geef de eerste 5 boeken volgens boekid.
select top (5) * from boeken order by boekid

Geef de 2 meest recent uitgegeven boeken.
select top (2) with ties * from boeken order by jaar desc

5 De WHERE-clausule

De WHERE-clausule filtert rijen: enkel de rijen die aan de voorwaarde voldoen worden weergegeven. De syntax van deze clausule:

WHERE voorwaarde

De voorwaarde kan vele vormen aannemen:

expressie { = | < > | ! = | > | > = | ! > | < | < = | ! < } expressie
expressie IS [NOT] NULL
expressie [NOT] BETWEEN expressie AND expressie
stringexpressie [NOT] LIKE patroon
expressie [NOT] IN (expressie [,...])

Een expressie kan een kolomnaam zijn, een constante of een “echte” expressie. In een “echte” expressie wordt een berekening uitgevoerd of wordt een scalaire functie gebruikt (zie verder).

Het is ook mogelijk om samengestelde voorwaarden te gebruiken. Merk hierbij op dat de NOT-operator de hoogste prioriteit heeft, daarna komt de AND-operator en tot slot de OR-operator. Uiteraard kunnen haakjes gebruikt worden om aan te geven in welke volgorde de logische operatoren NOT, AND en OR moeten uitgevoerd worden.

[NOT] voorwaarde1 { AND | OR } [NOT] voorwaarde2

Geef een alfabetische lijst van de klanten uit Brugge.
select * from klanten where woonplaats = 'Brugge' order by familienaam, voornaam
Merk op dat het vergelijken van tekst standaard hoofdletterongevoelig (case insensitive) is.

Geef de boeken uit categorie 2 die niet uitgegeven zijn in 1993. Sorteer op dalend uitgiftejaar.
select * from boeken where categorieid = 2 and jaar != 1993 order by jaar desc

Geef de boeken die uitgegeven zijn tussen 1970 en 1999 (beide grenzen inbegrepen). Sorteer op jaar.
select * from boeken where jaar >= 1970 and jaar <= 1999 order by jaar
select * from boeken where jaar between 1970 and 1999 order by jaar

Geef de boeken de uitgegeven zijn in 1963, 1983, 1993 of 1999. Sorteer op jaar.

```
select *
from boeken
where jaar = 1963 or jaar = 1983 or jaar = 1993 or jaar = 1999
order by jaar
```

```
select *
from boeken
where jaar in (1963, 1983, 1993, 1999)
order by jaar
```

Geef een alfabetische lijst van de klanten waarvan de familienaam begint met de letter V.

```
select *
from klanten
where familienaam like 'V%'
order by familienaam
```

Het patroon dat gebruikt wordt bij de LIKE-operator is een string waarin wildcards kunnen gebruikt worden.
Betekenis van deze wildcards:

%	0, 1 of meer tekens
_	1 teken

Geef een alfabetische lijst van de klanten met een letter V in de familienaam.

```
select *
from klanten
where familienaam like '%V%'
order by familienaam
```

Geef een alfabetische lijst op titel van de boeken waarvoor het jaar niet ingevuld is.

```
select *
from boeken
where jaar is null
order by titel
```

Geef een alfabetische lijst op titel van de boeken waarvoor het jaar ingevuld is.

```
select *
from boeken
where jaar is not null
order by titel
```

Geef een chronologisch overzicht van de uitleengegevens na 03/05/2013.

```
select *
from uitleningen
where datum > '2013-05-03'
order by datum
```

Geef de boeken die uitgegeven zijn in een even jaar. Sorteer op jaar.

```
select *
from boeken
where jaar % 2 = 0
order by jaar
```

De rekenkundige operatoren + - * / en % kunnen gebruikt worden in een expressie.

6 Scalaire functies

Scalaire functies werken op 1 waarde en leveren 1 waarde af. In deze paragraaf worden enkele interessante scalaire functies besproken. Een scalaire functie kan gebruikt worden waar een expressie verwacht wordt.

Voor een gedetailleerde beschrijving van alle scalaire functies, ingedeeld in categorieën, zie SQL Server Help https://docs.microsoft.com/en-us/sql/t-sql/functions/functions. Het is ook mogelijk om een functienaam in de query-editor in te geven, de cursor op de functienaam te plaatsen en daarna F1 te drukken om hulp te krijgen over deze functie.

Merk op dat elk RDBMS zijn eigen scalaire functies heeft. Zo zijn de scalaire functies van MS SQL Server anders dan de scalaire functies van MySQL. Merk tot slot ook op dat het mogelijk is om zelf functies te definiëren: men spreekt over user-defined functions (UDF).

6.1 Datum- en tijdfuncties

Day(date) levert de dag uit de datum als een geheel getal.
Month(date) levert de maand uit de datum als een geheel getal.
Year(date) levert het jaar uit de datum als een geheel getal.

Geef de boeken (boekid) die uitgeleend werden in het jaar 2013.

```
select boekid, datum
from uitleningen
where year(datum) = 2013
order by boekid
```

GetDate() levert de current serverdatum en -tijd (tot milliseconden-niveau).

Geef de boeken (boekid) die dit jaar uitgeleend werden.

```
select boekid, datum
from uitleningen
where year(datum) = year(getdate())
order by boekid
```

DatePart(datepart, date) levert een integer die het gevraagde deel van de datum-tijd voorstelt.

Geef de boeken (boekid) die dit jaar uitgeleend werden.

```
select boekid, datum
from uitleningen
where datepart(year, datum) = datepart(year, getdate())
order by boekid
```

DateAdd(datepart, number, date) telt een interval bij de opgegeven datum-tijd.

DateDiff(datepart, startdate, enddate) levert het verschil tussen 2 datum-tijden. Er wordt enkel gekeken naar het gevraagde gedeelte, niet naar de volledige datum-tijd.

Geef bij elke rij uit de tabel uitleningen, het aantal jaar geleden dat het boek werd uitgeleend.
select *, datediff(year, datum, getdate()) as aantalJaarGeleden from uitleningen
aantalJaarGeleden is een kolomnaam voor de berekende kolom; men spreekt over een alias.

6.2 Stringfuncties

Replace(bronstring, zoekstring, vervangstring) vervangt in de bronstring alle voorkomens van de zoekstring door de vervangstring.

Geef alle auteurs, gesorteerd op familienaam. Spaties mogen niet meespelen in het bepalen van de sorteervolgorde.
select * from auteurs order by replace(familienaam, ' ', '')

Charindex(zoekstring, string) levert de positie af waar de zoekstring de eerste keer gevonden wordt in de string.

Substring(string, startpositie, lengte) levert een deelstring af.

Right(string, lengte) levert het rechtse gedeelte van de string.
Left(string, lengte) levert het linkse gedeelte van de string.

Geef van de boektitels die uit meerdere woorden bestaan, het eerste woord.
select *, left(titel, charindex(' ', titel)) as eersteWoord from boeken where charindex(' ', titel) != 0

Upper(string) levert de string in hoofdletters.
Lower(string) levert de string in kleine letters.
Ltrim(string) levert de string zonder voorloopspaties.
Rtrim(string) levert de string zonder naloopspaties.

Geef de titels van de boeken in hoofdletters, zonder eventuele voorloop- en naloopspaties. Sorteer de titels alfabetisch; houd daarbij geen rekening met spaties in de titel.
select upper(ltrim(rtrim(titel))) as titelHoofdletters from boeken order by replace(upper(ltrim(rtrim(titel))), ' ', '')

6.3 Soundex functies

Soundex is een systeem om gelijkklinkende woorden of namen te vinden. Het is in het begin van de jaren 1900 ontwikkeld. Als een gesproken naam moet opgezocht worden, dan vindt Soundex alle namen die klinken zoals de ingegeven naam. Vb Nielsen klinkt als Nelson en klinkt als Neilson. Soundex berekent een soundex-code. Indien 2 woorden dezelfde code hebben, dan klinken ze hetzelfde. Nielsen, Nelson en Neilson krijgen soundex-code N425. Deze code wordt als volgt berekend:

De eerste letter wordt behouden. De volgende 3 tekens krijgen een code:

1 voor B, F, P, V
2 voor C, G, J, K, Q, S, X, Z
3 voor D, T
4 voor L
5 voor M, N
6 voor R

De tekens A, E, I, O, U, H, W, Y worden weggelaten.

Soundex(string) berekent de soundex-code van de string

Geef een alfabetische lijst van alle uitgevers met de corresponderende soundexcode.
select *, soundex(uitgever) as soundexcode from uitgevers order by uitgever

Difference(string, string) levert het soundex-verschil tussen 2 strings. Het resultaat is een getal tussen 1 en 4. 4 betekent dezelfde soundex-code.

Geef een lijst van de uitgevers waarvan de naam klinkt als QUE.
select * from uitgevers where difference(uitgever, 'que') = 4

6.4 Conversiefuncties

Impliciete typeconversie gebeurt automatisch (zonder gebruik te maken van functies): bijvoorbeeld van het type nchar naar het type nvarchar.

Expliciete typeconversie vereist het gebruik van de functies cast() of convert().

Voor een overzicht van alle mogelijke impliciete en expliciete typeconversies: zie SQL Server Help https://docs.microsoft.com/en-us/sql/t-sql/functions/cast-and-convert-transact-sql.

Cast(expressie as datatype) converteert de expressie naar het gewenste datatype.

Convert(datatype, expressie [,style]) converteert de expressie naar het gewenste datatype. De stijl wordt gebruikt bij de uitvoer tijdens de conversie van een datum-tijd naar een karaktergebaseerd type. De stijl wordt gebruikt bij de invoer tijdens de conversie van tekst naar een datum-tijd.

Geef de huidige datum als karakterdata dd-mm-yyyy.
select convert(char(10), getdate(), 105) as vandaag

Geef van alle rijen uit de uitleningen-tabel: het uitleennummer, de datum en de geformatteerde datum.
select uitleenid, datum, convert(varchar(20), datum, 105) as datumAnders from uitleningen

7 Aggregaatfuncties (verzamelfuncties)

Aggregaatfuncties werken op een verzameling waarden en leveren 1 waarde af. In deze paragraaf worden een aantal interessante aggregaatfuncties besproken.

Count(*)	Telt het aantal rijen	
Count(expressie)	Telt het aantal niet NULL-waarden in de expressie	
Count(distinct expressie)	Telt het aantal verschillende niet NULL-waarden in de expressie	
Min(expressie)	Levert de kleinste niet NULL-waarde uit de expressie	Numeriek, string, datetime datatype
Max(expressie)	Levert de grootste niet NULL-waarde uit de expressie	Numeriek, string, datetime datatype
Sum([distinct] expressie)	Levert de som van de [verschillende] niet NULL-waarden uit de expressie	Numeriek datatype
Avg([distinct] expressie)	Levert het gemiddelde van de [verschillende] niet NULL-waarden uit de expressie	Numeriek datatype

De expressie is bijna altijd een kolomnaam.

De aggregaatfuncties kunnen NIET in de WHERE-clausule voorkomen!

Geef het aantal boeken.
select count(*) as aantal from boeken
--Let op! De volgende query is FOUT: select titel, count(*) as aantal from boeken --Een aggregaatfunctie kan niet gecombineerd worden met een reeks van waarden.

Geef het aantal boeken, het eerste jaar van uitgifte, het meest recente jaar van uitgifte en het verschil tussen beide.
select count(*) as aantal, min(jaar) as eersteJaar, max(jaar) as recentsteJaar, max(jaar) - min(jaar) as verschil from boeken

Geef het aantal verschillende jaren waarin boeken zijn uitgegeven.
select count(distinct jaar) as aantalJaren from boeken

Geef het aantal klanten uit Brugge.
select count(*) as aantal from klanten where woonplaats = 'Brugge'

Geef het aantal verschillende letters waarmee de namen van klanten beginnnen.
select count(distinct substring(familienaam, 1, 1)) as aantalVerschillendeEersteLetters from klanten

Geef het gemiddelde jaar van uitgifte van de boeken uit categorie 3.
select avg(jaar) as gemiddelde from boeken where categorieid = 3
select sum(jaar) / count(jaar) as gemiddelde from boeken where categorieid = 3
Onderstaande query is FOUT omdat de kolom jaar NULL-waarden kan bevatten: select sum(jaar) / count(*) as gemiddelde from boeken where categorieid = 3

8 De inner join van tabellen

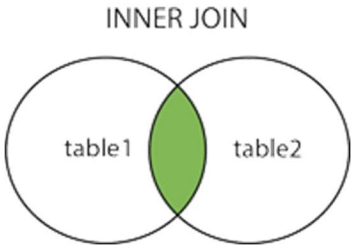
Gegevens uit verschillende tabellen combineren, gebeurt met join-operaties.

Bij een inner join of kortweg join van 2 tabellen bevat de resultaatset 'gecombineerde' rijen. Zo'n 'gecombineerde' rij is een rij van tabel1 naast een rij van tabel2 áls aan de opgegeven join-voorwaarde voldaan is.

Meestal gebeurt de join tussen 2 tabellen waarbij de foreign key van de ene tabel moet overeenkomen met de primary key van de andere tabel.

De syntax van het SELECT-statement ziet er dan als volgt uit:

```
SELECT-clausule
FROM tabel1
   JOIN tabel2 ON join-voorwaarde1
   [ JOIN tabel3 ON join-voorwaarde2 ] …
[ WHERE-clausule ]
[ GROUP BY-clausule ]
[ HAVING-clausule ]
[ ORDER BY-clausule]
```



Geef alle boeken met hun uitgever. Sorteer op uitgever.
select * from boeken join uitgevers on boeken.uitgeverid = uitgevers.uitgeverid order by uitgevers.uitgever

De kolom uitgeverid komt in beide tabellen (boeken en uitgevers) voor. In de join-voorwaarde moet deze kolomnaam dan voorafgegaan worden door de tabelnaam om het onderscheid tussen de 2 kolommen te kunnen maken. Men spreekt over het kwalificeren van een kolom.

```
select boeken.*, uitgevers.uitgever
from boeken
join uitgevers on boeken.uitgeverid = uitgevers.uitgeverid
order by uitgevers.uitgever
```

```
select b.*, u.uitgever
from boeken as b
join uitgevers as u on b.uitgeverid = u.uitgeverid
order by u.uitgever
```

In het bovenstaande voorbeeld wordt met een alias gewerkt voor de tabelnamen. Dit maakt het schrijven van de query wat gemakkelijker.

Geef de titel en het jaar van de boeken die uitgegeven zijn bij De Bezige Bij, ná 1970. Sorteer op titel.

```
select b.titel, b.jaar
from boeken as b
join uitgevers as u on b.uitgeverid = u.uitgeverid
where u.uitgever = 'De Bezige Bij'
and b.jaar > 1970
order by b.titel
```

Geef een gesorteerde lijst van de titels van de boeken uit de categorie Wetenschappelijk. Geef ook de auteur weer.

```
select b.titel, a.familienaam, a.voornaam
from boeken as b
join auteurs as a on b.auteurid = a.auteurid
join categorieen as c on b.categorieid = c.categorieid
where c.categorie = 'wetenschappelijk'
order by b.titel
```

Geef van alle boeken de titel, het jaar van uitgifte, de uitgever, auteur en categorie. Sorteer op titel.

```
select b.titel, b.jaar, u.uitgever, a.familienaam, a.voornaam, c.categorie
from boeken as b
join uitgevers as u on b.uitgeverid = u.uitgeverid
join auteurs as a on b.auteurid = a.auteurid
join categorieen as c on b.categorieid = c.categorieid
order by b.titel
```

Geef het aantal boeken dat De Bezige Bij heeft uitgegeven.

```
select count(*) as aantal
from boeken as b
join uitgevers as u on b.uitgeverid = u.uitgeverid
where u.uitgever = 'De Bezige Bij'
```

Geef een alfabetische lijst van de boeken uit categorie 2 of 3 die uitgeleend werden in 2013. Geef de titel en de uitleendatum.

```
select b.titel, u.datum
from boeken b
join uitleningen u on b.boekid = u.boekid
where b.categorieid in (2, 3) and year(u.datum) = 2013
order by b.titel
```

```
select b.titel, u.datum
from boeken b
join uitleningen u on b.boekid = u.boekid
where (b.categorieid = 2 or b.categorieid = 3) and year(u.datum) = 2013
order by b.titel
```

Geef van de klanten die een boek uitleenden: de klantgegevens, de uitleendatum en de titel van het boek. Sorteer op klantid.

```
select k.*, u.datum, b.titel
from klanten k
join uitleningen u on k.klantid = u.klantid
join boeken b on u.boekid = b.boekid
order by k.klantid
```

Geef de gegevens van de klanten die een boek van de auteur Breemeersch uitleenden.

```
select k.*
from klanten k
join uitleningen u on k.klantid = u.klantid
join boeken b on u.boekid = b.boekid
join auteurs a on b.auteurid = a.auteurid
where a.familienaam = 'Breemeersch'
order by k.klantid
```

Ook een join van een tabel met zichzelf is mogelijk. Dan móet er gewerkt worden met een alias voor de tabelnamen om het onderscheid tussen de 2 tabellen te kunnen maken.

Geef de klanten met een naamgenoot (zelfde familienaam).

```
select k1.*
from klanten k1
join klanten k2 on k1.familienaam = k2.familienaam and k1.klantid != k2.klantid
order by k1.familienaam
```

```
select k1.*
from klanten k1
join klanten k2 on k1.familienaam = k2.familienaam
where k1.klantid != k2.klantid
order by k1.familienaam
```

9 De outer join van tabellen

Er zijn 3 soorten outer joins. Outer joins breiden de inner join uit met rijen uit de linker en/of rechter tabel.

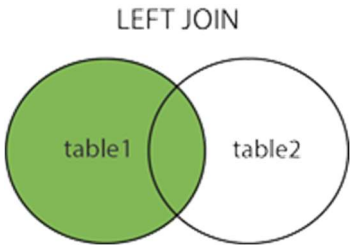
Een left (outer) join bevat de rijen van de inner join én de rijen uit de linker tabel die niet voorkomen in de inner join. Deze rijen worden gecombineerd met NULL-waarden aan de rechterkant.

Een right (outer) join bevat de rijen van de inner join én de rijen uit de rechter tabel die niet voorkomen in de inner join. Deze rijen worden gecombineerd met NULL-waarden aan de linkerkant.

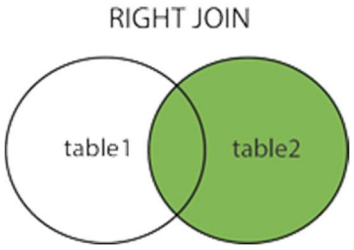
Een full (outer) join bevat de rijen van de inner join én de rijen uit de linker tabel die niet voorkomen in de inner join (deze rijen worden gecombineerd met NULL-waarden aan de rechterkant) én de rijen uit de rechter tabel die niet voorkomen in de inner join (deze rijen worden gecombineerd met NULL-waarden aan de linkerkant). Een full (outer) join wordt niet zo vaak gebruikt en wordt in deze cursus niet verder behandeld.

De syntax van het SELECT-statement ziet er dan als volgt uit:

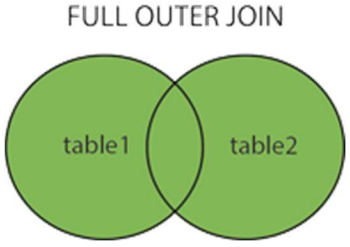
```
SELECT-clausule
FROM tabel1
   LEFT JOIN tabel2 ON join-voorwaarde1
   [ LEFT JOIN tabel3 ON join-voorwaarde2 ] …
[ WHERE-clausule]
[ GROUP BY-clausule ]
[ HAVING-clausule ]
[ ORDER BY-clausule]
```




```
SELECT-clausule
FROM tabel1
   RIGHT JOIN tabel2 ON join-voorwaarde1
   [ RIGHT JOIN tabel3 ON join-voorwaarde2 ] …
[ WHERE-clausule]
[ GROUP BY-clausule ]
[ HAVING-clausule ]
[ ORDER BY-clausule]
```




```
SELECT-clausule
FROM tabel1
   FULL JOIN tabel2 ON join-voorwaarde1
   [ FULL JOIN tabel3 ON join-voorwaarde2 ] …
[ WHERE-clausule]
[ GROUP BY-clausule ]
[ HAVING-clausule ]
[ ORDER BY-clausule]
```



Geef van alle klanten de klantgegevens en de uitleengegevens. Als een klant niets heeft uitgeleend, wordt hij toch opgenomen in de lijst met NULL bij de uitleengegevens. Sorteer op klantid.
select * from klanten k left join uitleningen u on k.klantid = u.klantid

order by k.klantid
select k.*, u.* from uitleningen u right join klanten k on k.klantid = u.klantid order by k.klantid

Geef een overzicht van alle boekid's en titels, met daarbij, als ze uitgeleend zijn, de uitleendatum, anders niets. Sorteer op boekid.
select b.boekid, b.titel, u.datum from boeken b left join uitleningen u on b.boekid = u.boekid order by b.boekid

Geef alle uitgevers met hun boeken en de categoriegegevens. Ook als een uitgever geen boeken heeft uitgegeven, moet hij opgenomen worden in de lijst. Sorteer op uitgeverid.
select * from uitgevers u left join boeken b on u.uitgeverid = b.uitgeverid left join categorieen c on b.categorieid = c.categorieid order by u.uitgeverid

10 Data groeperen: de GROUP BY- en HAVING-clausule

De GROUP BY-clausule maakt groepen van rijen, meestal om een aggregaatfunctie toe te passen op elke groep. Het SELECT-statement levert 1 resultaatrij per groep. Deze resultaatrij bevat meestal de kolom(men) waarop gegroepeerd werd en aggregaatfuncties die op de groep werken.

De syntax van de GROUP BY-clausule is eenvoudig:

GROUP BY kolomnaam [, …]

Geef voor elk jaar van uitgifte het aantal boeken.
select jaar, count(*) as aantal from boeken group by jaar
De onderstaande query is FOUT omdat in de SELECT-clausule enkel groepsinformatie kan staan en geen detailinformatie van rijen uit de groep: select *, count(*) as aantal from boeken group by jaar

Geef per woonplaats het aantal klanten.
select woonplaats, count(*) as aantal from klanten group by woonplaats

Geef per auteur de auteurid, het aantal boeken dat hij geschreven heeft, het eerste jaar van uitgifte en het laatste jaar van uitgifte.
select auteurid, count(*) as aantal, min(jaar) as eerste, max(jaar) as laatste from boeken group by auteurid

Idem vorige vraag, maar zorg dat ook de auteurs die geen boek schreven in de lijst voorkomen.
select a.auteurid, count(boekid) as aantal, min(jaar) as eerste, max(jaar) as laatste from boeken b right join auteurs a on b.auteurid = a.auteurid group by a.auteurid

Geef per woonplaats het aantal uitleningen.
select woonplaats, count(*) as aantal from klanten k join uitleningen u on k.klantid = u.klantid group by woonplaats

Geef per maand het aantal uitgeleende boeken.
select month(datum) as maand, count(*) as aantal from uitleningen group by month(datum)

Geef van elke klant uit Brugge de klantid, de naam en het aantal uitgeleende boeken.
select k.klantid, k.familienaam, k.voornaam, count(uitleenid) as aantal from klanten k left join uitleningen u on k.klantid = u.klantid where k.woonplaats = 'Brugge' group by k.klantid, k.familienaam, k.voornaam

Groepen kunnen gefilterd worden met de HAVING-clausule. Enkel de groepen die aan de voorwaarde voldoen worden weerhouden. De voorwaarde kan bevatten: de kolom(men) waarop gegroepeerd werd en aggregaatfuncties.
Merk op dat de HAVING-clausule niet kan gebruikt worden zonder een voorafgaande GROUP BY-clausule.

De syntax van de HAVING-clausule:

HAVING voorwaarde

Geef voor elk jaar van uitgifte het aantal boeken. Enkel de jaren waarin meer dan 1 boek werd uitgegeven worden weerhouden.
select jaar, count(*) as aantal from boeken group by jaar having count(*) > 1

Geef per auteur de auteurid en het aantal boeken dat hij geschreven heeft. Enkel de auteurid's van de auteurs die meer dan 1 boek schreven worden weerhouden.
select auteurid, count(*) as aantal from boeken group by auteurid having count(*) > 1

Geef de namen van de categorieën met meer dan 6 boeken.
select categorie, count(*) as aantal from boeken b join categorieen c on b.categorieid = c.categorieid group by categorie having count(*) > 6

11 Subquery's

Een subquery is een query binnen een query. Anders gezegd: een SELECT-statement binnen een SELECT-statement.

Een subquery moet steeds tussen haakjes staan en kan op de volgende plaatsen gebruikt worden binnen een SELECT-statement:

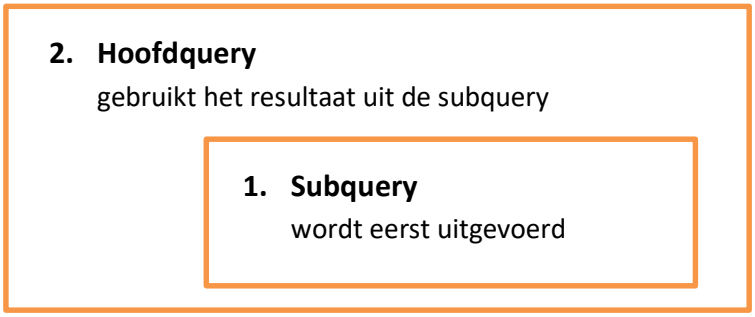
WHERE expressie vgloperator (subquery)	De subquery levert 1 waarde op
WHERE expressie [NOT] IN (subquery)	De subquery levert 1 waarde op of een lijst van waarden
WHERE [NOT] EXISTS (subquery)	Er wordt nagegaan of de subquery [geen] resultaatrijen oplevert
SELECT (subquery)	De subquery levert 1 waarde op
FROM (subquery)	De subquery levert een resultaatset op

Vele query's met een join kunnen (eenvoudiger) geschreven worden met een subquery.

Er zijn 2 soorten subquery's: niet-gecorreleerde subquery's en gecorreleerde subquery's. Beide soorten worden hieronder behandeld.

11.1 Niet-gecorreleerde subquery's

Een niet-gecorreleerde subquery werkt onafhankelijk van de hoofdquery (er is vanuit de subquery géén verwijzing naar de hoofdquery). De subquery wordt 1 maal uitgevoerd en het resultaat wordt gebruikt in de hoofdquery.



Geef de gegevens van de boeken die in hetzelfde jaar zijn uitgegeven als het boek 'Programmeren met LISP'. Sorteer op titel.
select * from boeken where jaar = (select jaar from boeken where titel = 'Programmeren met LISP') order by titel

Geef de gegevens van de boeken met het kleinste jaar van uitgifte. Sorteer op titel.
select * from boeken where jaar = (select min(jaar) from boeken) order by titel
De onderstaande query is FOUT: een aggregaatfunctie kan niet in de WHERE-clausule. select * from boeken where jaar = min(jaar) order by titel

Geef een alfabetische lijst op titel van de boeken die reeds uitgeleend werden.
select * from boeken where boekid in (select boekid from uitleningen) order by titel
select b.* from boeken b join uitleningen u on b.boekid = u.boekid order by titel

Geef een alfabetische lijst op titel van de boeken die nog niet uitgeleend werden.

```
select *
from boeken
where boekid not in (select boekid
                     from uitleningen)
order by titel
```

```
select b.*
from boeken b
left join uitleningen u on b.boekid = u.boekid
where u.boekid is null
order by titel
```

Geef een alfabetische lijst van de klanten die nog geen enkel boek uitleenden.

```
select *
from klanten
where klantid not in (select klantid
                      from uitleningen)
order by familienaam, voornaam
```

Geef een alfabetische lijst van de auteurs die een boek hebben uitgegeven in 1995 in de categorie non-fictie.

```
select *
from auteurs
where auteurid in (select auteurid
                   from boeken
                   where jaar = 1995
                   and categorieid = (select categorieid
                                      from categorieen
                                      where categorie = 'non-fictie'))
order by familienaam, voornaam
```

Geef een overzicht van de boeken uit de categorie wetenschappelijk met de uitgevergegevens en de auteurgegevens. Sorteer op boekid.

```
select b.boekid, b.titel, b.jaar,
       u.uitgever,
       a.familienaam, a.voornaam
from boeken b
join uitgevers u on b.uitgeverid = u.uitgeverid
join auteurs a on b.auteurid = a.auteurid
where b.categorieid = (select categorieid
                       from categorieen
                       where categorie = 'wetenschappelijk')
order by b.boekid
```

```
select b.boekid, b.titel, b.jaar,
       u.uitgever,
       a.familienaam, a.voornaam
from boeken b
join uitgevers u on b.uitgeverid = u.uitgeverid
join auteurs a on b.auteurid = a.auteurid
```

```
join categorieen c on b.categorieid = C.categorieid
where categorie = 'wetenschappelijk'
order by b.boekid
```

11.2 Gecorreleerde subquery's

Een gecorreleerde subquery is afhankelijk van de hoofdquery (er is vanuit de subquery een verwijzing naar de hoofdquery). Voor elke rij in de hoofdquery wordt de subquery uitgevoerd.

1. **Hoofdquery**
   Voor elke rij in de hoofdquery, wordt de subquery uitgevoerd
   2. **Subquery**
      er is een verwijzing naar de hoofdquery

Geef de klanten met een naamgenoot (zelfde familienaam) in een andere plaats.

Oplossing met een gecorreleerde subquery:

```
select *
from klanten k1
where k1.familienaam IN (select familienaam
                         from klanten
                         where familienaam = k1.familienaam
                         and woonplaats != k1.woonplaats)
order by k1.familienaam
```

Oplossing met een join:

```
select k1.*
from klanten k1
join klanten k2 on k1.familienaam = k2.familienaam
where k1.woonplaats != k2.woonplaats
order by k1.familienaam
```

Geef de meest recent uitgegeven boeken uit elke categorie.

```
select *
from boeken b1
where jaar = (select max(jaar)
              from boeken b2
              where b2.categorieid = b1.categorieid)
order by categorieid
```

Geef alle gegevens van de boeken met het aantal keer dat ze uitgeleend werden. Sorteer dalend op dit aantal.

```
select *,
```

```
    (select count(*)
            from uitleningen
            where boekid = boeken.boekid) as aantal
from boeken
order by aantal desc
```

Geef de boeken die het meest uitgeleend zijn.

```
select top 1 with ties *,
    (select count(*)
            from uitleningen
            where boekid = boeken.boekid) as aantal
from boeken
order by aantal desc
```

Geef een alfabetische lijst van de klanten die nog geen enkel boek uitleenden.

Oplossing met een gecorreleerde subquery:

```
select *
from klanten
where not exists
        (select *
         from uitleningen
         where klantid = klanten.klantid)
order by familienaam, voornaam
```

Met EXISTS wordt nagegaan of het resultaat van de gecorreleerde subquery niet leeg is.
Met NOT EXISTS wordt nagegaan of het resultaat van de gecorreleerde subquery leeg is.

Oplossing met een niet-gecorreleerde subquery (zie ook hierboven):

```
select *
from klanten
where klantid not in (select klantid
                      from uitleningen)
order by familienaam
```

Geef de auteurs van wie er een boek in de databank zit.

Met een gecorreleerde subquery:

```
select *
from auteurs
where exists (select *
              from boeken
              where auteurid = auteurs.auteurid)
order by auteurid
```

Met een niet-gecorreleerde subquery:

```
select *
from auteurs
where auteurid IN (select auteurid
                   from boeken)
order by auteurid
```

Met een join:

```
select distinct auteurs.*
```

```
from auteurs
join boeken on auteurs.auteurid = boeken.auteurid
order by auteurid
```

Geef de auteurs van wie er geen boek in de databank zit.

Met een gecorreleerde subquery:

```
select *
from auteurs
where not exists (select *
                  from boeken
                  where auteurid = auteurs.auteurid)
order by auteurid
```

Met een niet-gecorreleerde subquery:

```
select *
from auteurs
where auteurid NOT IN (select auteurid
                       from boeken)
order by auteurid
```

12 SELECT INTO

Het SELECT-statement met de INTO-clausule maakt een nieuwe tabel met de resultaatset van het SELECT-statement.

De nieuw gemaakte tabel heeft kolommen met dezelfde naam en type als de lijst in het SELECT-statement. De rijen zijn de resultaatrijen van het SELECT-statement. Let op: de constraints worden niet overgenomen, evenmin als de indexen.

Als de WHERE-clausule van het SELECT-statement FALSE oplevert, dan wordt een tabel gemaakt zonder data.

De syntax:

SELECT-clausule
[INTO tabelnaam]
FROM-clausule
[WHERE-clausule]
[GROUP BY-clausule]
[HAVING-clausule]

Maak een nieuwe tabel Boeken2 met boekid, auteurnaam, titel en jaar.

```
select boekid, familienaam, voornaam, titel, jaar
into boeken2
from boeken
join auteurs on boeken.auteurid = auteurs.auteurid
```

Ter controle:

```
select *
from boeken2
```

Bekijk ook de structuur van Boeken2.

Maak een nieuwe tabel Boeken3 met de structuur van de tabel Boeken, maar zonder data.
select * into boeken3 from boeken where 1 = 2
Ter controle: select * from boeken3
Bekijk ook de structuur van Boeken3.