



# GRADUAAT IN HET **PROGRAMMEREN**

## **Data Essentials**

Opleidingsonderdeel

Informatica | Programmeren

Afdeling

Patricia Briers

Auteur



**DE HOGESCHOOL  
MET HET NETWERK**

# INHOUD

<b>1</b>	<b>INLEIDING EN BASISBEGRIPPEN .....</b>	<b>4</b>
1.1	DATABASE.....	4
1.1.1	<i>Belang van databases .....</i>	<i>4</i>
1.2	DATAMODEL.....	5
1.3	COMPONENTEN VAN GEGEVENSVERZAMELINGEN .....	5
<b>2</b>	<b>SQL.....</b>	<b>7</b>
2.1	KENNISMAKING MET SQL.....	7
2.1.1	<i>Gegevensmanipulatietaal (DML) .....</i>	<i>7</i>
2.1.2	<i>Gegevensdefinitietaal (DDL) .....</i>	<i>7</i>
2.1.3	<i>Raadpleging.....</i>	<i>8</i>
2.1.4	<i>Beveiliging (DCL) .....</i>	<i>8</i>
2.2	BASISBEGRIPPEN.....	8
<b>3</b>	<b>SQL*PLUS.....</b>	<b>10</b>
3.1	KENNISMAKING MET SQL*PLUS.....	10
3.2	DE SQL-BUFFER .....	10
3.3	EXTERNE BUFFER .....	10
3.4	SQL*PLUS-EDITOR .....	11
3.5	COMMANDO'S BEWAREN .....	11
3.6	SQL*PLUS-INSTELLINGEN .....	12
3.7	OVERIGE SQL*PLUS-COMMANDO'S.....	13
<b>4</b>	<b>RAADPLEGING.....</b>	<b>14</b>
4.1	OVERZICHT VAN DE SELECT-COMPONENTEN.....	14
4.1.1	<i>De SELECT-component.....</i>	<i>14</i>
4.1.2	<i>De WHERE-component.....</i>	<i>15</i>
4.1.3	<i>De ORDER BY-component.....</i>	<i>17</i>
4.1.4	<i>CASE-expressie .....</i>	<i>17</i>
4.1.5	<i>NULL-waarden.....</i>	<i>18</i>
4.2	SUBQUERY'S .....	19
4.3	FUNCTIES .....	19
4.3.1	<i>Rekenfuncties .....</i>	<i>20</i>
4.3.2	<i>Tekstfuncties .....</i>	<i>21</i>
4.3.3	<i>Algemene functies.....</i>	<i>23</i>
4.3.4	<i>Datumfuncties.....</i>	<i>23</i>
4.3.5	<i>Conversiefuncties .....</i>	<i>25</i>
4.4	OEFENINGEN .....	28
4.4.1	<i>Select-operator: selecteren van rijen.....</i>	<i>28</i>
4.4.2	<i>Subquery.....</i>	<i>28</i>
4.4.3	<i>Functies .....</i>	<i>29</i>
<b>5</b>	<b>GEAVANCEERDE RAADPLEGING .....</b>	<b>31</b>
5.1	DE JOIN .....	31
5.1.1	<i>De equi-join .....</i>	<i>31</i>
5.1.2	<i>De auto-join.....</i>	<i>31</i>
5.1.3	<i>De outerjoin.....</i>	<i>32</i>
5.1.4	<i>De ANSI/ISO SQL-standaard .....</i>	<i>33</i>
5.1.5	<i>BREAK commando.....</i>	<i>33</i>
5.1.6	<i>De gepartitioneerde outerjoin.....</i>	<i>35</i>
5.2	GROEPSFUNCTIES.....	36
5.2.1	<i>Group by-component .....</i>	<i>36</i>
5.2.2	<i>Groepsfuncties.....</i>	<i>36</i>
5.2.3	<i>De having-component .....</i>	<i>38</i>
5.2.4	<i>BREAK en COMPUTE commando.....</i>	<i>39</i>
5.3	GECORRELEERDE SUBQUERIES .....	40
5.4	VERZAMELINGOPERATOREN .....	41
5.4.1	<i>Union .....</i>	<i>41</i>
5.4.2	<i>Minus.....</i>	<i>42</i>
5.4.3	<i>Intersect.....</i>	<i>42</i>

5.5	VIEWS .....	43
5.5.1	<i>Kennismaking met views.</i> .....	43
5.5.2	<i>Views in de datadictionary.</i> .....	44
5.5.3	<i>Views wijzigen.</i> .....	45
5.5.4	<i>Views verwijderen.</i> .....	45
5.6	TOEPASSINGSMOGELIJKHEDEN.....	45
5.7	OEFENINGEN .....	47
5.7.1	<i>Join-operator</i> .....	47
5.7.2	<i>Oefeningen groepsfuncties.</i> .....	49
5.7.3	<i>Oefeningen verzamelingsoperatoren</i> .....	50
5.7.4	<i>Oefeningen View</i> .....	51
6	DATA MANIPULATION LANGUAGE.....	52
6.1	<i>Gegevens invoeren</i> .....	52
6.2	<i>Gegevens wijzigen</i> .....	53
6.3	<i>Gegevens verwijderen</i> .....	54
6.4	<i>Transactieverwerking</i> .....	55
6.5	<i>Oefeningen</i> .....	56

# 1 Inleiding en basisbegrippen

## 1.1 Database

Een database, gegevensbank of databank is een digitaal opgeslagen archief, ingericht met het oog op flexibele raadpleging en gebruik. Databases spelen een belangrijke rol voor het archiveren en actueel houden van gegevens bij onder meer de overheid, financiële instellingen en bedrijven, in de wetenschap, en worden op kleinere schaal ook privé gebruikt.

Het woord *database* wordt voor verschillende begrippen gebruikt:

1. de opgeslagen gegevens als zodanig.
2. de wijze waarop de gegevens zijn opgeslagen ( datamodel).
3. de software waarmee databases kunnen worden aangemaakt en benaderd, (Database management systeem of DBMS).

Een database moet aan de volgende minimale voorwaarden voldoen om als database gezien te worden:

1. Gegevens moeten eenvoudig kunnen worden opgeslagen.
2. Gegevens moeten eenvoudig kunnen worden opgezocht en doorzocht.
3. Gegevens moeten gewijzigd kunnen worden.
4. Gegevens moeten verwijderd kunnen worden zonder dat de werking van dat systeem nadelig beïnvloedt.

Een database is meer dan een gedigitaliseerd archief, een essentiële toevoeging is dat de gegevens in een database zodanig zijn opgeslagen dat deze gegevens optimaal doorzoekbaar zijn. Deze toevoeging staat bekend onder de naam index

De doorzoekbaarheid van de gegevens wordt hier mee vergroot omdat bij zoekopdrachten als "toon alle personen met postcode tussen 3000 en 4000" alleen in het veld "postcode" hoeft te worden gezocht en alle andere gegevens niet geëvalueerd hoeven te worden.

### 1.1.1 Belang van databases

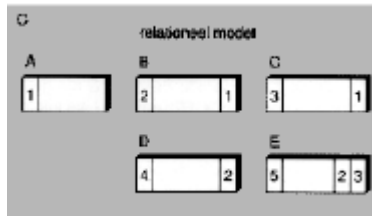
Databases zijn een essentieel onderdeel van de informatiemaatschappij, steeds meer gegevens worden in een database opgeslagen. Het functioneren van de overheid, bedrijven en wetenschap is tegenwoordig zonder databases ondenkbaar.

Ook zoekmachines maken gebruik van een database, door de pagina's op internet te indexeren. De gebruiker van een zoekmachine zoekt niet direct op internet, maar in de index die is aangemaakt. Vb Google

## 1.2 Datamodel

Met een **datamodel** (of datamodel of gegevensmodel) wordt beschreven hoe de gegevens in een informatiesysteem gestructureerd zijn.

### Relationeel model



Gegevens die in een relationeel model zijn gestructureerd, worden in een relationele database geïmplementeerd. Dit zijn eveneens een verzameling tabellen. Tabellen worden hier "relaties" genoemd, naar analogie van het concept van een wiskundige relatie. De kracht van dit model treedt aan de dag wanneer bevestigingen van verschillende tabellen

gecombineerd worden. De theorie achter het relationele model werd ontwikkeld door Ted Codd. Voor het bevestigen van de gegevens in een relationele database wordt de querytaal SQL gebruikt.

Anders dan bij netwerkdatabases worden de verbanden tussen tabellen niet expliciet gedefinieerd. In plaats daarvan duidt de aanwezigheid van gegevenselementen uit hetzelfde domein impliciet op een potentieel verband tussen twee eigenschappen, eventueel in verschillende tabellen. Bewerkingen worden uitgevoerd op basis van de relationele algebra. Als gevolg daarvan kan een relationele database flexibel gereorganiseerd worden en gebruikt op een manier die door de oorspronkelijke ontwerpers niet was voorzien. Veel van de huidige databases zijn gebaseerd op afleidingen van het relationele model, juist vanwege deze flexibiliteit.

Relationele databases worden meestal gebouwd met een genormaliseerd datamodel. Dit zorgt ervoor dat gegevens niet gedupliceerd worden opgeslagen (opslag optimalisatie). Normalisatie kan echter bijvoorbeeld bij rapportages tot een trage respons leiden. Om deze reden zijn alternatieve gedenormaliseerde datamodellen ontworpen in data warehouse oplossingen.

## 1.3 Componenten van gegevensverzamelingen

Gegevensverzamelingen bestaan uit **onderling gerelateerde gegevens**. Gegevens kunnen zijn opgebouwd uit andere gegevens. Bepaalde gegevens kunnen niet verder onderverdeeld worden zonder dat ze hun betekenis verliezen. Deze laatste soort noemen we **elementaire gegevens of kortweg gegevens**, de eerste soort samengestelde gegevens.

Een gegeven is een element dat relevantie en semantische waarde (betekenis) heeft. Met gegevens worden kenmerken van personen, zaken, handelingen e.d. uit de werkelijkheid beschreven. Samengestelde gegevens kunnen altijd vervangen worden door de samenstellende elementaire gegevens. Het feit dat een samengesteld gegeven op zich deel kan uitmaken van een ander samengesteld gegeven enz., levert bij de analyse van gegevensverzamelingen problemen op. Wij zullen verder maar één niveau van samenstelling onderkennen. Dit noemen we een groep. Een **groep is samengesteld uit elementaire**

**gegevens** en een groep kan geen deel uitmaken van een andere groep. Een gegeven kan wel in meerdere groepen voorkomen. Een groep bevat de gegevens van een logisch te onderkennen eenheid.

De relaties tussen de gegevens in een gegevensverzameling zijn nu gesplitst in de samenhang tussen de gegevens in één groep en de koppeling tussen de groepen onderling.

De hier gehanteerde begrippen "groep", "gegeven" en "koppeling" komt men in andere literatuur onder grote verscheidenheid aan namen tegen. In onderstaand overzicht wordt daarvan een indruk gegeven, zonder overigens de illusie te willen wekken compleet te zijn.

✓ **Groep**

- Tabel (relationele dB)
- Object of Entiteit (object georiënteerd datamodel)
- Entiteitsklasse: beschrijving van object (object georiënteerd datamodel)
- ...

✓ **Gegeven:**

- Veld (relationele dB)
- Kolom (relationele dB)
- Attribuut (relationele dB)
- ...

✓ **Koppeling**

- Relatie: verband tussen tabellen
- Parent-Child relationship (ouder- kindrelatie)
- ...

Voorbeeld componenten van een gegevensverzameling:

Groep	:	Medewerker	
Gegevens	:	Naam	Woonplaats Afdeling
Waarden	:	Els	Antwerpen Opleiding
		Pieter	Leuven Opleiding
		Patricia	Tongeren Analyse
		Rudi	Genk Analyse
		Gert	Hasselt Programmering

De verzameling waarden die betrekking heeft op één medewerker noemen we een *voorkomen* (*occurrence*) van de groep medewerker. Iedere groep heeft in het algemeen meerdere voorkomens. Een gegevensverzameling zal meestal uit meerdere onderling gekoppelde groepen bestaan.

Uit de informatie(behoefte)analyse moet blijken welke gegevens moeten worden vastgelegd. Door middel van gegevensanalyse worden de gegevens eenduidig benoemd, hun eigenschappen vastgelegd, bepaald in welke groepen de gegevens thuishoren en wat de koppeling tussen de groepen is. Dus het **bepalen van de structuur bepaalt de gegevens, deelt de groepen in en onderzoekt de koppeling tussen de groepen.**

## 2 SQL

### 2.1 Kennismaking met SQL

Een van de krachtigste kenmerken van een DBMS is het ondersteunen van een querytaal. Bij query's kan je 1 commando gebruiken om een groep records op te halen of te wijzigen. Query's kunnen ook gebruikt worden om de structuur van de database zelf te wijzigen. Vaak kan één enkele query een grote hoeveelheid programmacode vervangen. Ze voeren ook de taken sneller uit dan de programmacode.

SQL is een taal die in principe op verschillende manieren kan worden toegepast:

- ♦ Interactief: de SQL-commando's rechtstreeks ingeven waarbij je dadelijk het resultaat op het scherm krijgt;
- ♦ embedded: gebruik van SQL-commando's binnen een andere programmeertaal.

De taal SQL kan je onderverdelen in 4 groepen:

- 1 Gegevensmanipulatie of data manipulation language (DML)
- 2 Gegevensdefinitie of data definition language (DDL)
- 3 Raadpleging of query
- 4 Beveiliging of data control language (DCL)

#### 2.1.1 Gegevensmanipulatietaal (DML)

Verschaft de middelen om groepen rijen in een database in te voegen, te verwijderen, bij te werken en op te halen. Deze commando's werken met de *feitelijke gegevens* in de database. DML-commando's worden altijd opgevat als onderdeel van een transactie. Dat heeft onder meer tot gevolg dat alle daarmee aangebrachte wijzigingen in de database een voorlopig karakter krijgen, totdat de transactie wordt afgesloten.

Delete ... from	verwijderen van records
Insert ... into	toevoegen van records
Update ... set	wijzigen van kolomwaarden

Commit	bevestigt de wijzigingen van de huidige transactie
Rollback	annuleert de wijzigingen van de huidige transactie

#### 2.1.2 Gegevensdefinitietaal (DDL)

Met DDL-commando's kan je in één enkel commando een tabel maken, wijzigen of verwijderen. Je kan ook indexen maken of verwijderen. Deze commando's bewerken de *structuur* van de database.

Create table	tabel maken
Alter table	tabel wijzigen
Drop table	tabel verwijderen

### 2.1.3 Raadpleging

Het raadplegen van de databasegegevens gebeurt door selectiequery. Dit commando werkt op *tabelniveau* en levert als resultaat weer een tabel op. Zelfs als een bepaalde vraag precies één of helemaal geen rij als resultaat oplevert, dan nog is het resultaat een tabel (lege tabel).

Select .... from
------------------

### 2.1.4 Beveiliging (DCL)

SQL biedt allerlei commando's ten behoeve van beveiliging. De toegang tot de database wordt geregeld via een gebruikersautorisatie met behulp van wachtwoorden. Je kan met behulp van privileges de toegang tot de gegevens tot in detail regelen evenals de handelingen die mogen worden verricht.

Oracle onderscheidt twee soorten privileges:

#### Objectprivileges

Het recht om een bepaald object op een bepaalde manier te benaderen (bv. select, insert, alter, delete en update van een specifieke tabel).

#### Systeemprivileges

Het recht om bepaalde acties te ondernemen ( bv. mogen aanloggen, tabel aanmaken,...). Er zijn 80 verschillende systeemprivileges.

Create user	om nieuwe gebruikers te definiëren
Alter user	om bestaande gebruikers te wijzigen
Drop user	om gebruikers te verwijderen

Grant	bepaalde privileges verlenen
Revoke	bepaalde privileges ontnemen

## 2.2 Basisbegrippen

#### Constante

Numeriek	bv. 4, 8.75
Alfanumeriek	bv. 'Jansen'
Datum	bv. '12-05-2005'

#### Variabelen

SQL maakt een onderscheid tussen 2 soorten variabelen:

Kolomnamen	bv. &gesl
Systeemvariabelen of pseudokolommen	bv. sysdate, user



## Operatoren

De SQL-operatoren kunnen ingedeeld worden in 4 soorten:

### *Rekenkundige operatoren*

+	optellen
-	afrekken
*	vermenigvuldigen
/	delen

### *Alfanumerieke operator*

	concatenatie
--	--------------

### *Vergelijkingsoperatoren*

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

### *Logische operatoren*

AND	logische 'en'
OR	logische 'of'
NOT	logische ontkenning

### *Voorrangverwerking van operatoren*

- 1 rekenkundige operatoren
- 2 vergelijkingsoperatoren
- 3 logische operator

## Naamgeving van tabellen en kolommen

- ♦ Steeds met een letter beginnen;
- ♦ Max 30 karakters lang;
- ♦ Bevat letters, cijfers en \_ ;
- ♦ Geen verschil tussen kleine letters en hoofdletters;
- ♦ Geen gereserveerde woorden.

## Commentaar

In SQL-commando's kan eventueel commentaar worden opgenomen en wel op 2 manieren: tussen /\* en \*/ of na --

- Vb.    /\* dit is een gewone opmerking \*/  
         -- dit is een verhelderend voorbeeld

## 3 SQL\*Plus

### 3.1 Kennismaking met SQL\*Plus

De tool SQL\*plus is bij uitstek geschikt voor het interactief toepassen van SQL op een Oracle-database.

#### Log on

Om je te melden, moet een gebruikersnaam en een paswoord ingevuld worden.

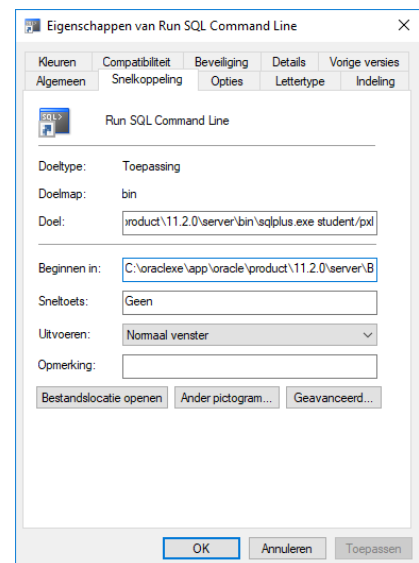
Gebruikersnaam: **student**

Paswoord: **pxl**

**SQL> CONNECT system/pxl** of gebruik de command line: **sqlplus student/pxl**

Met het commando Exit of Quit kan je SQL\*Plus verlaten.

Bij de eigenschappen van Run SQL Command Line kan je je user/paswoord toevoegen waardoor SQLPlus geconnecteerd start. NOLOG verbindt je enkel met de databank. Let wel dit bevat een veiligheidsrisico.



### 3.2 De SQL-buffer

Een centraal begrip binnen SQL\*Plus is de SQL-buffer. Hierin wordt steeds het laatst ingegeven SQL-commando bewaard. SQL-commando's beslaan meestal meer regels, die door SQL\*Plus voor ons tijdens het invoeren van regelnummers worden voorzien. Willen we een commando laten uitvoeren, dan moeten we de laatste regel afsluiten met een puntkomma(;). De puntkomma komt niet in de buffer terecht.

Vb. SQL> select \*  
2 from cat; *wordt dadelijk uitgevoerd*

SQL> select \*  
2 from cat *zit in buffer*

Een commando in de buffer uitvoeren.

SQL> / of r(un)

### 3.3 Externe buffer

Je kan op 2 manieren de inhoud van de buffer wijzigen door gebruik te maken van de externe editor of de editor van SQL\*Plus.

De standaard editor is Notepad met de default naam *afiedt.buf*. Je kan deze editor oproepen dmv het SQL-commando edit. Om verder te kunnen in SQL\*Plus volstaat het om de editor af te sluiten.

SQL> ed(it) *roept de externe editor op*

### 3.4 SQL\*Plus-editor

De SQL\*Plus-editor is een regeeditor, dwz dat slechts 1 regel actueel is waarop gewijzigd kan worden. Deze regel wordt op het scherm gemarkeerd met een \*.

Vb. SQL> select  
2\* form medewerkers;

Zoals je kan zien, zitten in ons commando 2 fouten. Zoek de 2 fouten en verbeter dmv van de volgende SQL\*Plus-commando's.

A(ppend) naam	toevoegen aan huidige regel
C(hange) /form/from	wijzigen
Del	verwijdert alle regels
Del 1	verwijdert regel 1
Del 1 5	verwijdert regel 1 t.e.m 5
I(nput) from afdelingen	toevoegen van nieuwe regel (na laatste)
L(ist)	geeft alle regels in de buffer
L 5 of 5	geeft regel 5
L 5 10	geeft regel 5 t.e.m 10
L*	geeft huidige regel

*Let op:* Je kan maar veranderen, toevoegen of verwijderen op de huidige gemarkeerde regel.

NB Bestaande commando's kunnen gemakkelijk worden verbeterd dmv de muistechiek (selecteren en met de rechter muisknop plakken)

### 3.5 Commando's bewaren

De inhoud van een buffer kan je bewaren met behulp van het SQL\*Plus-commando 'Save'. Met de optie Append of Replace kan je aan het bestand toevoegen of vervangen. Het bestand krijgt automatisch de extensie .sql.

Vb. SQL> save oefeningen append

Het bestand kan terug opgehaald worden in de buffer dmv het SQL\*Plus-commando 'Get'. Het SQL\*Plus-commando Start of @ haalt het bestand op en voert het ook dadelijk uit.

**Tip** Maak een eigen map aan waar je de bestanden en oefeningen kan bewaren.

Met het SQL\*Plus-commando Spool wordt de volledige schermuitvoer bewaard in een bestand (Oef .lst) in de standaarddirectory

Vb. SQL> spool Oef  
SQL> ...  
SQL> spool off

## 3.6 SQL\*Plus-instellingen

Het gedrag van SQL\*Plus als omgeving kan op talloze manieren worden beïnvloed met behulp van variabelen of instellingen (settings).

SQL> set pagesize 22	doseert uitvoer naar beeldscherm per pagina
SQL> set numwidth 10	standaardbreedte voor kolommen
SQL> set space 2	aantal spaties tussen de kolommen
SQL> set sqlprompt 'oef> '	prompt wijzigen
SQL> set pause on	dosering per pagina
SQL> set pause "Druk op een toets ..."	

De instellingen kunnen eveneens

*C:\oracle\app\oracle\product\11.2.0\server\sqlplus\admin\glogin.sql* geraadpleegd of gewijzigd worden. De instellingen blijven van kracht totdat we SQL\*Plus verlaten. Bestaande settings kunnen worden opgevraagd met het commando Show.

SQL> show pause	geeft instelling van pause
SQL> show all	geeft alle instellingen

Er is naast het SQL\*Plus-commando Set ook nog een SQL-commando om de omgeving aan te passen: Alter session. Hiermee kunnen enkele NLS (National Language Support) parameters worden geregeld.

SQL>	alter session	
2	set nls_date_format = "dd-mm-yyyy"	standaard datumformaat
3	nls_language=Dutch	taal voor boodschappen
4	nls_currency='EUR'	symbool voor geldbedrag
5	nls_numeric_characters=', '	' ' decimale scheiding en blanco: scheiding voor duizendtal

Je kan de NLS parameters opvragen: SQL> select \* from V\$NLS\_PARAMETERS

PARAMETER	VALUE
NLS_LANGUAGE	DUTCH
NLS_TERRITORY	BELGIUM
NLS_CURRENCY	€
NLS_ISO_CURRENCY	BELGIUM
NLS_NUMERIC_CHARACTERS	, .
NLS_CALENDAR	GREGORIAN
NLS_DATE_FORMAT	DD/MM/RR
NLS_DATE_LANGUAGE	DUTCH
NLS_CHARACTERSET	AL32UTF8
NLS_SORT	DUTCH
NLS_TIME_FORMAT	HH24:MI:SSXFF
NLS_TIMESTAMP_FORMAT	DD/MM/RR HH24:MI:SSXFF
NLS_TIME_TZ_FORMAT	HH24:MI:SSXFF TZR
NLS_TIMESTAMP_TZ_FORMAT	DD/MM/RR HH24:MI:SSXFF TZR
NLS_DUAL_CURRENCY	€
NLS_NCHAR_CHARACTERSET	AL16UTF16
NLS_COMP	BINARY
NLS_LENGTH_SEMANTICS	BYTE
NLS_NCHAR_CONV_EXCP	FALSE

### 3.7 Overige SQL\*Plus-commando's

#### Column

Als een resultaat niet op een schermregel past, kan het handig zijn om met het commando Column een of meer kolommen een beetje te versmallen. Normaal gesproken worden kolommen op het scherm even breed afgebeeld als ze zijn gedefinieerd in de database.

SQL> col omschrijving format a32	32 karakters breed
SQL> col toelage format 999.99	3cijfers.2cijfers
SQL> col naam justify L(ef), C(enter), R(ight)	uitlijning van kolomnaam
SQL> col afd format a20 heading afdeling	benaming van kolom
SQL> col naam clear	verwijdert kolominstellingen

#### Describe

Bij het formuleren van SQL-commando's is het soms handig om even een overzicht te krijgen van de *structuur* van een tabel. Het commando geeft naast de kolomnamen ook het datatype weer.

Bv.

```
SQL> desc medewerkers
```

#### Buffer en scherm leegmaken

Met Clear buffer kan de SQL-buffer worden leeggemaakt, terwijl met Clear screen het SQL\*Plus-venster wordt leeggemaakt.

Vb.

```
SQL> cl(ear) scr(een)
```

```
SQL> cl(ear) buffer
```

## 4 Raadpleging

### 4.1 Overzicht van de Select-componenten

Met het SELECT-commando kunnen we gegevens in een databank raadplegen.

```
SELECT [DISTINCT] select_expressie
FROM tabel_expressie
[WHERE conditie]
[GROUP BY expressie] [HAVING condition]
[ORDER BY {expressie } [ASC|DESC]]
```

De componenten moeten in deze volgorde worden verwerkt:

SELECT	Welke kolom(men) krijgt de resultaat tabel.
FROM	Welke tabel(len) worden geraadpleegd.
WHERE	Waaraan moeten de rijen voldoen.
GROUP BY	Waarop moet worden gegroepeerd.
HAVING	Waaraan moeten de groepen rijen voldoen.
ORDER BY	In welke volgorde wensen we het resultaat.

#### Opmerkingen

- De volgorde van de componenten ligt vast;
- SELECT en FROM zijn verplicht;
- WHERE, GROUP BY en ORDER BY zijn optioneel;
- HAVING komt niet voor zonder GROUP BY.

#### 4.1.1 De SELECT-component

*Alle gegevens van alle medewerkers.*

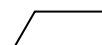
```
SQL> select *
      from medewerkers;
```

*Geef de naam en het afdelingsnummer van alle medewerkers.*

```
SQL> select naam, afd
      from medewerkers;
```

*Verfraai de kolomtitels van voorbeeld 2.*

```
SQL> select naam "Medewerkers", afd as afdeling
      from medewerkers;
```

 *kolomalias*

*Geef de afdelingen waar de medewerkers werken.*

```
SQL> select distinct afd
      from medewerkers;
```

*Geef naam en afdelingsnummer van alle medewerkers en verfraai.*

```
SQL> select naam, ' is werkzaam in afdeling ', afd
      from medewerkers;
```

*Werk in vorig voorbeeld de tabulators weg.*

```
SQL> select naam || ' is werkzaam in afdeling ' || afd as overzicht
      from medewerkers;
```

*Geef de huidige datum.*

```
SQL> select sysdate
      from dual;
```

*Geef het jaarsalaris (exclusief commissie) van iedere medewerker.*

```
SQL> select voorn, naam, 12*maandsal as jaarsalaris
      from medewerkers;
```

### 4.1.2 De WHERE-component

Met de relationele restrictie-operator WHERE kunnen we een *voorwaarde* of *conditie* specificeren waaraan de rijen moeten voldoen om in het resultaat te komen. We maken onderscheid tussen *enkelvoudige* en *samengestelde condities*.

*Enkelvoudige condities* zijn expressies waarin een vergelijkingsoperator (>, <, >=, ....) van SQL voorkomt. Samengestelde condities worden gebouwd met behulp van de logische operatoren AND, OR en NOT.

		AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

#### 4.1.2.1 Enkelvoudige conditie

*Geef de werknemers geboren na 1 mei 1977.*

```
SQL> select voorn, naam, gbdatum
      from medewerkers
      where gbdatum >= DATE '1977-05-01'    standaard datumformat: '01-MAY-77 '
```

*Geef de medewerkers van afdeling 30.*

```
SQL> select voorn, naam
      from medewerkers
      where afd=30;
```

#### 4.1.2.2 Operatoren Between, In, Like

*Geef de medewerkers van afdeling 10 of afdeling 30.*

```
SQL> select voorn, naam
      from medewerkers
      where afd in (30,10);           of where afd= 30 OR afd=10
```

*Geef de medewerkers die werken in afdeling 20 t.e.m afdeling 30*

```
SQL> select voorn, naam
      from medewerkers
      where afd between 20 and 30;
```

*Geef alle medewerkers in wiens naam een 'A' voorkomt.*

```
SQL> select voorn, naam
      from medewerkers
      where naam like '%A%';      Caspers, Allard, Martens,...
```

### **Betekenis Wildcards:**

Het onderlijningsteken ( \_ ) stelt precies 1 willekeurig karakter voor terwijl het %-teken nul, 1 of meer willekeurige karakters voorstelt.

#### **4.1.2.3 Samengestelde conditie**

*Geef de medewerkers die niet in de afdeling 30 werken.*

```
SQL> select voorn, naam
      from medewerkers
      where NOT afd = 30;      of where afd <>30;
```

*Geef de medewerkers van afdeling 10 of afdeling 30.*

```
SQL> select voorn, naam
      from medewerkers
      where afd= 30 OR afd=10;
```

*Geef alle medewerkers behalve Briers of de medewerkers van afdeling 30.*

```
SQL> select *
      from medewerkers
      where NOT (naam = 'BRIERS' OR afd = 30 );      of where naam<> 'BRIERS' AND afd <>30
```

*Geef alle cursussen die geen goede waardering (goed: 3,4 en 5 ) hebben behaald.*

```
SQL> select *
      from inschrijvingen
      where evaluatie NOT IN (3,4,5);
```

**Of** where evaluatie NOT in (3,4,5)  
where NOT evaluatie in (3,4,5)  
where NOT (evaluatie = 3 OR evaluatie = 4 Or evaluatie = 5)  
where evaluatie <> 3 AND evaluatie <> 4 AND evaluatie <>5

### **Let op**

In samengestelde condities is het toepassen van haakjes altijd aan te raden om de volgorde af te dwingen. Kijk maar eens naar onderstaand voorbeeld.

Vb. where naam = 'BRIERS' OR naam like 'D%' AND afd = 30  
where naam = 'BRIERS' OR (naam like 'D%' AND afd = 30)  
where naam = ( 'BRIERS' OR naam like 'D%') AND afd = 30



### 4.1.3 De ORDER BY-component

De volgorde van de resultaatquery kan door middel van de ORDER BY- component worden afgedwongen. We kunnen meerdere sorteerspecificaties opgeven, onderling gescheiden door komma's. Iedere sorteerspecificatie bestaat uit kolomspecificatie, eventueel gevolgd door de toevoeging DECS (descending) als we aflopend willen sorteren. Zonder deze toevoeging wordt ASC (ascending) stijgend gesorteerd.

*Geef de gegevens van alle werknemers in volgorde van hun personeelsnummer.*

```
SQL> select *  
      from medewerkers  
      order by mnr desc;
```

*Volgorde in functie van kolomnummer 2 (kolom 2 in select-component).*

```
SQL> select *  
      from medewerkers  
      order by 2;
```

*Volgorde in stijgende volgorde van commissie en dalend voor naam.*

```
SQL> select *  
      from medewerkers  
      order by comm asc ,naam desc;    of order by comm,naam desc
```

### 4.1.4 CASE-expressie

Met behulp van de CASE-expressie kunnen vrij complexe procedurele problemen aangepakt worden.

<b>CASE</b> inputexpressie <b>WHEN</b> waarde <b>THEN</b> resultaat <b>[ELSE]</b> resultaat <b>END</b>
---

De inputexpressie en de WHEN-waarde moeten van hetzelfde datatype zijn, alsook het THEN-resultaat en het ELSE-resultaat.

```
SQL> select cursist, begindatum  
      , CASE evaluatie  
          WHEN 1 THEN 'slecht'  
          WHEN 2 THEN 'matig'  
          WHEN 3 THEN 'ok'           7499      12-APR-15      goed  
          WHEN 4 THEN 'goed'        7934      12-APR-15      zeer goed ...  
          WHEN 5 THEN 'zeer goed'  
          ELSE 'niet ingevuld'  
      END as beoordeling  
      from inschrijvingen  
      where cursus = 'SQL';
```

### Voorbeeld 2

```
SELECT cursist, begindatum
, CASE
    WHEN evaluatie = 0 THEN 'Slecht'
    WHEN evaluatie < 2 THEN 'Matig'
    WHEN evaluatie < 5 THEN 'Ok'
    WHEN evaluatie < 7 THEN 'Goed'
    WHEN evaluatie < 8 THEN 'Zeer goed'
    ELSE 'Uitstekend'
END as beoordeling
FROM inschrijvingen;
```

## 4.1.5 NULL-waarden

### 4.1.5.1 Werken met NULL-waarden

Als een kolom voor een bepaalde rij geen waarde bevat, dan noemt men dat een NULL-waarde. Een NULL-waarde geeft aan dat er ergens informatie ontbreekt. Dit ontbreken van informatie kan diverse oorzaken hebben.

Op het beeldscherm wordt een NULL-waarde meestal door niets weergegeven. NULL-waarden kunnen in SQL het beste worden weergegeven met behulp van het gereserveerde woord NULL. Voorzie via het menu set options de waarde NULL bij de instelling van NULL. De weergave van NULL-waarden kan bovendien op kolomniveau worden geregeld met behulp van het SQL\*Plus-commando Column.

Bij het werken met NULL-waarde onthoud je best de volgende stelling:

NULL = NULL	is onbeslist
NULL IS NULL	is waar

*Geef de medewerkers waarvan de commissie niet van toepassing is.*

```
SQL> col comm NULL "Onbekend" of show NULL NULL " "
SQL> Set NULL "NULL"
```

```
SQL> select voorn, naam, comm
      from medewerkers
      where comm is NULL;
```

*Volgorde in stijgende volgorde van commissie en dalend voor naam.*

```
SQL> select *
      from medewerkers
      order by comm asc ,naam desc;    of order by comm,naam desc
```

### **Let op**

Bij het dalend sorteren krijgt de NULL-waarde de hoogste waarde (staan bovenaan) en bij stijgend sorteren de laagste waarde (staan onderaan).

#### 4.1.5.2 NVL-functie

SQL beschikt over twee functies die specifiek bedoeld zijn om flexibel met NULL-waarden om te kunnen gaan: de nvl-functie en de NVL2-functie.

*Geef de maandsalarissen en de commissie (inclusief een verhoging van 10%).*

```
SQL> select maandsal, nvl(comm,0)*1.1  
      from medewerkers;
```

```
SQL> select maandsal, nvl2(comm,comm,0)*1.1  
      from medewerkers;
```

NVL(x,y) geeft y als x NULL is, anders x zelf  
NVL2(x,y,z) geeft y als x niet NULL is, anders z

## 4.2 Subquery's

*Geef alle medewerkers die meer verdienen dan Slechten.*

```
SQL> select voorn, naam, maandsal  
      from medewerkers  
      where maandsal > (select maandsal  
                        from medewerkers  
                        where naam = 'SLECHTEN');
```

*Geef alle medewerkersnummers die de bouw cursus (type BLD) gevolgd hebben.*

```
SQL> select distinct cursist  
      from inschrijvingen  
      where cursus in (select code  
                       from cursussen  
                       where type = 'BLD');
```

*Geef alle namen van medewerkers die de bouw cursus (type BLD) gevolgd hebben*

```
SQL> select voorn, naam  
      from medewerkers  
      where mnr in ( select distinct cursist  
                    from inschrijvingen  
                    where cursus in ( select code  
                                     from cursussen  
                                     where type = 'BLD'));
```

## 4.3 Functies

Functies zijn te herkennen aan het feit dat ze een naam hebben gevolgd door een of meer argumenten tussen haakjes. Functies mogen, vrijwel overal in queries worden toegepast: in de *select*-, de *where*- en in het *order by*-component.

Functies kunnen ingedeeld worden in 6 groepen.

Numerieke functies	toepasbaar op numerieke gegevens
Alfanumerieke functies	toepasbaar op alfanumerieke gegevens
Datumfuncties	toepasbaar op datums
Algemene functies	toepasbaar op ieder datatype
Conversiefuncties	conversie naar een andere datatype
Groepsfuncties	toepasbaar op een groep gegevens

### 4.3.1 Rekenfuncties

De belangrijkste functies om met cijfers te werken zijn:

ROUND( <i>n</i> [, <i>m</i> ])	rondt <i>n</i> af op <i>m</i> decimale posities
CEIL( <i>n</i> )	rondt <i>n</i> naar boven af op een geheel getal
FLOOR( <i>n</i> )	rondt <i>n</i> naar beneden af op een geheel getal
TRUNC( <i>n</i> [, <i>m</i> ])	kapt <i>n</i> af op <i>m</i> decimale posities
ABS( <i>n</i> )	de absolute waarde van <i>n</i>
SIGN( <i>n</i> )	-1, 0 of 1 als <i>n</i> negatief, nul of positief is
SQRT( <i>n</i> )	vierkantswortel uit <i>n</i>
POWER( <i>n</i> , <i>m</i> )	<i>n</i> tot de <i>m</i> -de macht
MOD( <i>n</i> , <i>m</i> )	rest na deling van <i>n</i> door <i>m</i>

In beide gevallen dat *m* optioneel is geldt 0 (nul) als default-waarde en zijn negatieve waarden voor *m* ook toegestaan.

#### Voorbeelden

SQL> select round(345.678) from dual	346 (standaard: geen cijfers na de komma)
SQL> select ceil(345.678) from dual	346
SQL> select floor(345.678) from dual	345
SQL> select round(345.678, 2) from dual	345.68
SQL> select round(345.678, -1) from dual	350 -2 (AFRONDEN OP 100-tal)
SQL> select trunc(345.678, 2) from dual	345.67
SQL> select abs(-123) , abs(0), abs(456) from dual	123/0/456

SQL> select sign(-13) , sign(0), sign(456) from dual	-1/0/1
SQL> select sqrt(16), sqrt(8), sqrt(4) from dual	4/2.8284271/2
SQL> select power(2, 3), power(-2,3) from dual	$2^3=8$ $-2^3=-8$
SQL> select mod(8,3), mod(13,0) from dual	2/13
SQL> select naam, maandsal, mod(maandsal,100) from medewerkers	0/0/50/75/....
SQL> select naam, maandsal, sign(maandsal-1600) from medewerkers	-1/0/-1/... (welke groter, kleiner dan 1600)
SQL> select * from medewerkers where mod (mnr, 2 ) = 0	even nummers...
SQL> select naam , floor ((sysdate-gbdatum)/7) as weken , floor (mod(sysdate-gbdatum,7)) as dagen from medewerkers	uitvoeren

### 4.3.2 Tekstfuncties

De voornaamste tekstfuncties van Oracle zijn:

LENGTH(t)	aantal karakters (lengte) van <i>t</i>
ASCII(t)	ascii-waarde eerste karakter van <i>t</i>
CHR(n)	karakter met ascii-waarde <i>n</i>
UPPER(t)	<i>t</i> in hoofdletters
LOWER(t)	<i>t</i> in kleine letters
INITCAP(t)	elke woord in <i>t</i> met beginhoofdletter
LTRIM(t,k)	verwijdert links van <i>t</i> de <i>k</i> -karakters
RTRIM(t,k)	verwijdert rechts van <i>t</i> de <i>k</i> -karakters
LPAD(t,n)	vult <i>t</i> links uit met spaties tot lengte <i>n</i>
LPAD(t,n,k)	idem, met <i>k</i> -karakter
RPAD(t,n)	vult <i>t</i> rechts aan met spaties tot lengte <i>n</i>
RPAD(t,n,k)	idem, met <i>k</i> -karakter
SUBSTR(t,n)	geeft deel van <i>t</i> vanaf positie <i>n</i> tot het einde
SUBSTR(t,n,m)	geeft deel van <i>t</i> vanaf positie <i>n</i> , <i>m</i> -karakters lang
INSTR(t,k)	positie eerste voorkomen van <i>k</i> in <i>t</i>
INSTR(t,k,n)	idem, op of na de <i>n</i> -de positie in <i>t</i>

INSTR(t,k,n,m)	het <i>m</i> -de voorkomen van <i>k</i> , op of na de <i>n</i> -de positie in <i>t</i>
REPLACE(t,v)	<i>verwijdt</i> uit <i>t</i> elk voorkomen van <i>v</i> (woorden)
REPLACE(t,v,w)	<i>vervangt</i> in <i>t</i> elk voorkomen van string <i>v</i> in <i>w</i> (woorden)
TRANSLATE(t,v,w)	vervangt alle <i>karakters</i> uit <i>v</i> die in <i>t</i> voorkomen door het corresponderende karakter uit <i>w</i>
CONCAT(t1,t2)	concateneert t1 en t2 (equivalent met   )

Posities in strings worden altijd geteld vanaf het eerste karakter (1).

#### Voorbeelden

SQL> select code, upper(omschrijving), lower(type) from cursussen	INTRODUCTIE... alg
SQL> select anr, naam, initcap(locatie) from afdelingen order by length(naam)	30 VERKOOP Genk
SQL> select * from medewerkers where lower(functie) = 'trainer '	SWINNEN TRAINER ...
SQL> select ascii( 'a' ), ascii( 'z' ), chr( 77 ) from dual	97/122/M
SQL> select substr( naam,4) , substr (naam,4,3) from afdelingen	fdkantoor fdk
SQL> select naam, instr (naam,'A') , instr (naam,'A',3) , instr (naam,'A',1,2) from medewerkers	ALLARD 1/4/4 JACOBS 2/0/0 DE COOMAN 8/8/0
SQL> select ltrim(naam,'SDAER') , rtrim(naam,'SDAER') from medewerkers	CASPERS CASP ALLARD ALL N RUYTER RUYT
SQL> select lpad (naam,8,'@') , rpad (naam,12,'=') from medewerkers	@@JACOBS JACOBS=====
SQL> select lpad(maandsal,4)    ' '    rpad('=',maandsal/100,'=') 800 from medewerkers	===== 1600 =====
SQL> select translate(code,'AESOL' , '12345') , replace(omschrijving,'SQL',' Visual C#') from cursussen	SQL INTRODUCTIE SQL 3Q5 INTRODUCTIE Visual C#

### 4.3.3 Algemene functies

GREATEST(a,b,...)	grootste waarde uit de argumenten
LEAST(a,b, ...)	kleinste waarde uit de argumenten
NULLIF(a,b)	geeft NULL als a=b, anders a
COALESCE(a,b,...)	Retourneert het eerste argument dat niet NULL is (en NULL als ze allemaal NULL zijn)
NVL(x,y)	geeft y als x NULL is, anders x zelf
NVL2(x,y,z)	geeft y als x niet NULL is, anders z
DECODE(x ,a1, b1	geeft : <b>b1</b> als x = <b>a1</b>
,a2, b2	<b>b2</b> als x = <b>a2</b>
...	...
,an, bn	<b>bn</b> als x = <b>an</b>
[,y]	en anders y (of default: NULL)

Enkel NULLIF() en COALESCE() zijn onderdeel van de ANSI/ISO-standaard, de overige functies zijn specifiek voor Oracle.

Bovendien zijn deze functies uit te drukken als een CASE-expressie. We gebruiken ze toch omdat ze een compactere code opleveren.

*Bepaal het jaarlijks inkomen van elke medewerker.*

```
SQL> select naam, (maandsal*12) + nvl(comm,0) as salaris      CASPERS      21600
      from medewerkers                                     ALLARD      22200...
```

**of**     select naam, (maandsal\*12) + nvl2(comm,comm,0) as salaris  
      from medewerkers

**of**     select naam, (maandsal\*12) + coalesce(comm,0) as salaris  
      from medewerkers

*Sorteer de medewerkers op basis van de functie.*

```
SQL> select naam, functie
      from medewerkers
      order by decode(functie , 'DIRECTEUR' , 1
                           , 'MANAGER'   , 2
                           , 'VERKOPER'  , 3
                           , 'TRAINER'   , 4 )      Boekhouden is "Else"
```

### 4.3.4 Datumfuncties

Alvorens de diverse datumfuncties te bespreken bekijken we eerst de manier waarop we in SQL tijdgerelateerde constanten kunnen weergeven:

DATE 'yyyy-mm-dd'

TIMESTAMP 'yyyy-mm-dd hh24:mi:ss.ff'

INTERVAL 'expr' <qualifier>

*Voorbeelden*

DATE '2016-09-25'

TIMESTAMP '2013-09-25 23:59:59.99999' AT TIME ZONE 'CET'

```

INTERVAL '1' YEAR
INTERVAL '1 2:3' DAY TO MINUTE
SELECT interval '1' YEAR + sysdate
FROM medewerkers
WHERE gbdatum < DATE '1984-01-20';

```

Als we alleen maar een alfanumerieke string ingeven, moeten we vertrouwen op een impliciete conversie door Oracle. Of deze impliciete conversie slaagt of faalt is afhankelijk van de NLS\_DATE\_FORMAT en NLS\_TIMESTAMP\_FORMAT parameterinstellingen. Je kan steeds het datumformaat, de taal en de munteenheid via de parameters van het National Language Support of NLS aanpassen. Je kan deze parameters als volgt opvragen:

```

SQL> select * from nls_session_parameters;
SQL> alter session
      set nls_date_format="dd-mm-yyyy"      'Voorbeeld 18-02-2016
        nls_language=Dutch
        nls_currency=' EUR'
        nls_numeric_characters=', '        'komma voor decimale scheiding en
                                           spatie voor de scheiding voor duizendtal

```

De belangrijkste datumfuncties van Oracle zijn:

ADD_MONTHS(d,n)	datum <i>d</i> plus <i>n</i> maanden
MONTHS_BETWEEN(d,e)	maanden verschil tussen <i>d</i> en <i>e</i>
LAST_DAY(d)	laatste dag van de maand waarin <i>d</i> valt
NEXT_DAY(d,wkdag)	de eerste weekdag (ma, di, ...) na <i>d</i>
NEXT_TIME (d,z1,z2)	converteert datum/tijd van tijdzone1 naar tijdzone2
ROUND(d[,fmt])	<i>d</i> afgerond op <i>fmt</i> (default middernacht)
TRUNC(d[,fmt])	<i>d</i> afgekapt op <i>fmt</i> (default middernacht)
EXTRACT(c FROM d)	extraheert component <i>c</i> uit expressie <i>d</i>

#### Voorbeelden

```

SQL> select naam,months_between(sysdate,gbdatum)      CLERCKX      393.14705
      from medewerkers                                DE KONING     543.88899

SQL> select gbdatum, add_months(gbdatum,13)          17-DEC-1985 17-JAN-1987 17-SEP-1985
      ,      add_months(gbdatum,-3)                  ....
      from medewerkers

SQL> select add_months(date '2015-01-29',1)          28-FEB-2015
      ,      add_months(date'2016-01-,29',1)        29-FEB-2016 Schrikkeljaar
      from dual

SQL> select next_day(sysdate,'sat')                  20-FEB-2016
      ,      last_day (sysdate)                      29-FEB-2016
      from dual

SQL> select extract(year from gbdatum)                1972
      ,      extract(month from gbdatum)              11

```



```
,      extract(day from gbdatum)
from medewerkers
where naam = 'DE KONING'
```

17

De onderstaande tabel geeft de datumformaten (fmt) die door de datumfuncties **round** en **trunc** worden ondersteund. Het standaardformaat is 'DD' hetgeen afronden of afkappen op middernacht tot gevolg heeft.

CC,SCC	eeuw,met of zonder minteken (BC)
[S]YYYY,[S]Year,YYY,YY,Y	jaar (in alle gedaantes)
IYYY,IYY,IY,I	ISOjaar (van maandag tot zondag)
Q	kwartaal
MONTH,MON,MM,RM	maand(voluit,afgekort,getal,Romeins getal)
IW,WW	(ISO) weeknummer (weken volgens ISOjaar)
W	dag van de week
DDD,DD,J	dag(jaar/maand/Juliaans)
DAY,DY,D	dichtstbijzijnde zondag
HH,HH12,HH24	uur
MI	minuut

#### Voorbeeld

```
SQL> select round(date '2016-04-26','MONTH')      01-MAY-2016
,      trunc(date '2016-04-26','MONTH')          01-APR-2016
from dual;
```

### 4.3.5 Conversiefuncties

De belangrijkste conversiefuncties van Oracle zijn:

TO_CHAR(n[,fmt])	zet getal <i>n</i> om naar een string
TO_CHAR(d[,fmt])	zet datum <i>d</i> om naar een string
TO_NUMBER(t)	zet string <i>t</i> om naar een getal
TO_DATE(t[,fmt])	zet <i>t</i> om naar een datum

Het probleem met het datatype DATE is dat het op papier of op het scherm uitsluitend kan worden weergegeven als een string. Andersom kunnen we ook geen datums via een toetsenbord invoeren. We moeten een string invoeren, samen met een opmaakformaat hoe deze string als datum moet worden opgevat. Als we het standaardformaat hanteren, kan het expliciet aangeven van dit opmaakformaat achterwege gelaten worden.

#### Voorbeelden

```
SQL> select to_char(123)                        123
2 ,      to_char(123,'09999.99')                00123.00
3 ,      to_char(1236,'9G999D9')                1 236,0 (G=duizendtal, D=decimal)
4 ,      to_char(123,'0999.0L')                 0123.0 EUR (nls_currency='EUR')
5 ,      to_number('123')                       123
5 from dual;
```

```
SQL> select sysdate
2 , to_char(sysdate,'hh24:mi:ss')
3 , to_char(to_date('26-03-2016','dd-mm-yyyy'),' "valt op" Day')
4 from dual;
```

26-03-2011  
11:34:31  
valt op Zaterdag

```
SQL> select to_char (sysdate,'yyyy')
2 , to_char (sysdate, 'yy')
3 , to_char (sysdate,'y')
4 , to_char (sysdate,'year')
5 from dual;
```

2016  
16  
6  
TWENTY SIXTEEN

```
SQL> select to_char (sysdate,'ad')
2 from dual;
```

n.Chr.

```
SQL> select to_char (sysdate,'Q')
2 from dual;
```

2

```
SQL> select to_char (sysdate,'mm')
2 , to_char (sysdate,'month')
3 , to_char (sysdate,'mon')
4 from dual;
```

04  
april  
apr

```
SQL> select to_char (date'2016-01-13','ddd')
2 , to_char (date'2016-01-13','dd')
3 , to_char (date'2016-01-13','d')
4 , to_char (date'2016-01-13','day')
5 , to_char (date'2016-01-13','Dy dy')
6 from dual;
```

013  
13  
4  
wednesday  
Wed wed

```
SQL> select to_char (sysdate,'hh:mi:ss AM')
2 , to_char (sysdate,'hh24:mi:ss')
3 , to_char (sysdate,'sssss')
4 from dual;
```

01:19:15 PM  
13:19:15  
47955

*Vraag de weekdag waarop je geboren bent.*

```
SQL> select decode (to_char(to_date('&gbdatum','ddmmyyyy'),'d')
, 1, 'zondag'
, 2, 'maandag'
, 3, 'dinsdag'
, 4, 'woensdag'
, 5, 'donderdag'
, 6, 'vrijdag'
, 7, 'zaterdag') geboortedag
from dual;
```

```
SQL> Enter value for gbdatum: 15041973
```

GEBOORTEDAG

-----  
zondag

Raadpleging

Met betrekking tot de conversiefuncties *to\_char* en *to\_date* zijn volgende formaten mogelijk:

[S]CC	eeuw,S voor het minteken (BC)
[S]YYYY	jaar, met of zonder minteken
[S]Year	jaartal uitgespeld, met minteken (S)
YYY,YY,Y	jaar (laatste 3,2 of 1 getal)
BC,AD	BC/AD indicator
Q	kwartaal(1,2,3,4)
MM	maand (01–12)
MONTH	maandnaam, met spaties uitgevuld tot lengte 9
MON	maand(afgekort)
IW,WW	(ISO)weeknummer (01-52)
W	weeknummer van de maand (1-5)
DDD	dagnummer van het jaar (1-366)
DD	dagnummer van de maand (1-31)
D	dagnummer van de week (1-7)
J	Juliaans datum, dagnummer sinds 01/01/4712 BC
DAY	dagnaam, met spaties uitgevuld tot lengte 9
DY	afkorting van de dag
AM,PM	AM/PM indicator
HH[12]	uur van de dag (01-12)
HH24	uur van de dag (00-23)
MI	minuut (00-59)
SS	seconden
SSSS	seconden na middernacht (0-86399)
/.,	deze leestekens letterlijk in de datum
"..."	string wordt eveneens weergegeven

Verder zijn er nog enkele toevoegingen mogelijk:

TH	ordinaal getal (4th)
SP	uitgespeld getal (four)
THSP, SPTH	uitgespeld ordinaalgetal (fourth)
FM	fill mode (voorlooptnullen en spaties onderdrukken – met FM fillmechanisme in- en uitschakelen)

#### Voorbeeld

```
SQL> select to_char(sysdate,'fmDay, ddth "of" month yyyysp')
       from dual;           Friday, 13th of january two thousand sixteen
```

## 4.4 Oefeningen

### 4.4.1 Select-operator: selecteren van rijen

- 1 Geef het nummer en de naam van de medewerkers.
- 2 In welke locaties zijn er afdelingen gevestigd?
- 3 Geef de code en de omschrijving van elke cursus.
- 4 Geef de naam en functie van alle medewerkers die verkoper zijn.
- 5 Geef de naam en functie van alle medewerkers die geen boekhouders zijn.
- 6 Geef alle medewerkers die meer dan € 5000 verdienen en geen manager zijn.
- 7 Welke medewerkers verdienen minder dan € 2500?
- 8 Geef de naam van de afdeling waarvan medewerker 7698 de baas is.
- 9 Geef de verschillende functies van de medewerkers in de onderneming.
- 10 Geef het nummer en de naam van elke medewerker die trainer is.
- 11 Geef de naam, de functie en het salaris van elke medewerker die in afdeling 30 werkt en meer dan € 2500 verdient.
- 12 Geef de locatie van de afdeling die door medewerker 7566 geleid wordt?
- 13 Geef de namen van de medewerkers die in afdeling 10 of 20 werken.
- 14 Geef van elke medewerker de naam en jaarsalaris (incl. commissie).
- 15 Geef de namen van de medewerkers wiens naam eindigt op 'N'.
- 16 Geef de namen van alle medewerkers waarin een dubbele 'o' voorkomt.
- 17 Alle medewerkers krijgen een opslag van 10%. Geef een lijst waarop de naam van de medewerkers staat, zijn huidig salaris en zijn toekomstig salaris. Zorg voor verzorgde attribuu tkoppen in een realistisch formaat!
- 18 Geef de namen en het salaris van de medewerkers van groot naar klein gesorteerd op salaris.
- 19 Geef de namen en het salaris van alle medewerkers die tussen de € 2000 en € 4000 verdienen, maar niet gelijk is aan € 3850, noch aan € 2600. Druk het resultaat volgens dalend salaris af.
- 20 Geef de namen en het salaris van de medewerkers, gesorteerd op functie en binnen de functie op naam.
- 21 Bepaal het jaarlijks inkomen van elke medewerker. Los de null-problematiek op dmv de CASE-expressie.

### 4.4.2 Subquery

- 1 Geef de naam en het salaris van de medewerkers die meer verdienen dan Clerckx.
- 2 Geef de namen van de medewerkers die in dezelfde afdeling werken als Allard.
- 3 Geef de naam van alle medewerkers die zich voor een cursus hebben ingeschreven (in volgorde van de naam).

- 4 Geef de medewerkers (namen) die jonger zijn dan hun collega E Jacobs.
- 5 Geef de cursisten die een cursus in Maaseik gevolgd hebben.
- 6 Geef de cursusnamen (en het type) die de medewerkers van de afdeling Verkoop hebben gevolgd.
- 7 Geef naam en voornaam van iedereen die ooit bij J. Caspers een cursus heeft gevolgd.
- 8 Geef de namen van de cursussen waarvan de docent niet gekend is.
- 9 Geef de namen van de medewerkers van de afdeling verkoop wiens commissie niet gekend is.
- 10 Geef de namen van de medewerkers die zich inschreven voor een cursus waarvan de docent niet gekend is.

### 4.4.3 Functies

#### Zorg voor mooie geformatteerde kolommen.

- 1 Maak een lijst met achter elke medewerker het precieze uurloon, het uurloon dat op twee decimalen is afgerond, en het uurloon dat op twee decimalen is afgekapt. Het uurloon wordt berekend op basis van een kwartaal:  $(\text{maandsalaris} \times 3) / (38 \times 13)$
- 2 Maak een lijst waarin de verschillende waarden van FUNCTIE elk op drie manieren wordt weergegeven: met alleen aan het begin een hoofdletter, volledig in hoofdletters en volledig in kleine letters.
- 3 Geef van elke cursus de naam en de lengte van de naam.
- 4 Doorzoek de namen van de medewerkers op de letter 'e'  
Wordt de letter 'e' gevonden dan drukt men in het resultaat het positienummer.  
Laat op drie manieren zoeken:
  - zoek vanaf de eerste positie naar de eerste letter 'e'
  - zoek vanaf de vierde positie naar de eerste letter 'e'
  - zoek vanaf de eerste positie naar de tweede letter 'e'
- 5 Maak een histogram met als maatstaf de lengte van de namen van de medewerkers.  
Zorg dat de kolom histogram 25 karakters breed is.

NAAM	LENGTE	HISTOGRAM
BRIERS	6	*****
DE KONING	9	*****

- 6 Maak een lijst met de namen van de medewerkers, met hun namen zonder vooraan de letters 'de' en met hun namen zonder achteraan de letters 'sen' indien ze voorkomen.
- 7 Geef een lijst met de namen van de afdelingen, de nummers en de nummers voorafgegaan door de letters 'AF'.
- 8 Geef een lijst met de namen van alle cursussen waarbij de letters 'WC#' vervangen worden door 'ZK-'. Zowel de hoofdletters als de kleine letters vervangen.
- 9 Sorteert de namen van de medewerkers zonder rekening te houden met eventuele blanco's.
- 10 Druk enkel de medewerkers af waarvan de naam uit 2 delen bestaat. Het eerste gedeelte wordt in kleine letters en tussen haakjes na het tweede gedeelte (in hoofdletters) van de naam afgedrukt.  
Vb. KONING (de)

- 11 Bepaal voor elke medewerker wiens naam een 'e' bevat wat groter is: zijn salaris\*2, zijn MNR of het MNR van zijn chef? Naast de naam en het grootste element wordt aangegeven welk element de grootste waarde heeft (chef, salaris of mnr). Gebruik hiervoor de Decode().
- 12 Bepaal voor elke medewerker wiens naam een 'e' bevat wat kleiner is: zijn salaris\*2, zijn MNR of het MNR van zijn chef? Naast de naam en het kleinste element wordt aangegeven welk element de kleinste waarde heeft (chef, salaris of mnr). Gebruik hiervoor de case.
- 13 Geef een lijst van de medewerkers waarbij de functie 'TRAINER' voorgesteld wordt als 'LERAAR' en 'DIRECTEUR' als 'HOOFD'. Alle overige functies worden als 'MEDEWERKER' voorgesteld. Los op met decode() en de case.
- 14 Welke waarden is het grootst: het MNR-2000 of de 2<sup>de</sup> macht van de eerste letterwaarde van de medewerkernaam ? Geef aan welke waarde het grootste is.
- 15 Geef een lijst van de medewerkers waarbij de kolommen NAAM, FUNCTIE en OVERZICHT worden afgedrukt.  
In de kolom OVERZICHT wordt voor een trainer zijn/haar salaris afgedrukt, voor een verkoper wordt zijn/haar commissie afgedrukt en in alle andere gevallen drukt men 'onbelangrijk' af.
- 16 Geef voor alle managers de dag van de week, de maand, de dag en het jaar in vier cijfers waarop ze geboren zijn.  
Vb. JACOBS donderdag 02 april 1987
- 17 Hoeveel dagen ben je oud?
- 18 Geef voor alle trainers de geboortedatum voluit geschreven.  

NAAM	GEBOORTEDATUM
VISSER	The Twenty-fifth of December One Thousand Nine Hunderd Sixty-five
...	...
- 19 Geef het weeknummer, het kwartaal en het Juliaans dagnummer van de geboortedata van Wouters en Swinnen?
- 20 Maak een lijst waarin voor elke medewerker aangegeven staat op welke uur van de dag hij/zij geboren is. De tijd die in het voorbeeld wordt afgedrukt is de tijd die in het attribuut GBDATUM werd opgeslagen. Standaard is deze tijd 12:00 AM omdat het niet meegedeeld is aan de databank.
- 21 Geef voor alle medewerkers de geboortedatum en de datum 6 maanden later.
- 22 Geef naam, de vierkantswortel van het salaris en het aantal jaren (afgerond op 2 cijfers na de komma) dat de medewerker oud is.
- 23 Druk de leeftijd af in maanden, dagen en jaren (zonder cijfers na de komma) voor elke medewerker ouder dan 35 jaar.
- 24 Druk voor alle medewerkers de laatste dag van de maand af waarop zij geboren zijn.
- 25 Druk van elke medewerker de datum af van de eerste vrijdag volgend op de 3000 dagen na de geboortedatum.
- 26 Geef de functie en naam van alle medewerkers die voor 1984 geboren zijn. Vermeld per medewerker de inkomensklasse. Wanneer het maandelijks salaris kleiner is dan 2500 wordt "goedkoop" vermeld anders is het "duur". Sorteert de resultaat tabel op basis van de functie. Eerst de directeur, dan de managers, vervolgens de verkopers en tenslotte de trainer.

## 5 Geavanceerde raadpleging

### 5.1 De Join

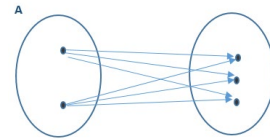
#### 5.1.1 De equi-join

We kunnen in de FROM-component van een query meerdere tabellen aanspreken, waardoor we informatie uit verschillende tabellen kunnen opvragen.

*Geef de namen van alle medewerkers en de naam van de afdeling waar ze werken.*

```
SQL> select  m.voorn, m.naam, a.naam
      from    afdelingen a
      ,       medewerkers m;
```

→ 56 rijen nl. 14 medewerkers X 4 afdelingen



Door gebruik te maken van een *tupelvariabele* of een *tabel-alias* weet SQL welke kolomnamen we gebruiken. Het resultaat van onze query is een cartesiaans product van beide tabellen dat resulteert in 56 rijen nl. 14 medewerkers maal 4 afdelingen geeft 56 mogelijke combinaties. Dmv de where-component verkrijgen we de juiste rij-combinaties.

```
SQL> select  m.voorn, m.naam, a.naam
      from    afdelingen a
      ,       medewerkers m
      where   m.afd = a.anr;
```

VOORN	NAAM	NAAM
JANA	CASPERS	OPLEIDINGEN
NELE	ALLARD	VERKOOP
THOMAS	DEFOUR	VERKOOP
EMMA	JACOBS	OPLEIDINGEN

Merk op dat het steeds belangrijker wordt de SQL-commando's zo *overzichtelijk* mogelijk te formuleren, zowel vanwege de leesbaarheid als vanwege de onderhoudbaarheid. Het commando wordt zoveel mogelijk over verschillende regels verdeeld.

```
SQL> select  m.voorn
      ,       m.naam
      ,       a.naam
      from    afdelingen a
      ,       medewerkers m
      where   m.afd = a.anr;
```

#### 5.1.2 De auto-join

We kunnen in SQL een tabel ook joinen met zichzelf. Hoewel er geen essentieel verschil is met de equi-join, wordt deze join de self-join of auto-join genoemd.

*Geef een overzicht van alle medewerkers met de naam van hun chef.*

```
SQL> select  m.naam as medewerker
      ,       mm.naam as chef
      from    medewerkers m
      ,       medewerkers mm
      where   m.chef = mm.mnr;
```

MEDEWERKER	CHEF
CASPERS	DE COOMAN
ALLARD	BRIERS
DEFOUR	BRIERS
JACOBS	DE KONING
MARTENS	BRIERS
BRIERS	DE KONING
CLERCKX	DE KONING
SWINNEN	JACOBS
DEN RUYTER	BRIERS
SLECHTEN	SWINNEN
JACOBS	BRIERS
DE COOMAN	JACOBS
WOUTERS	CLERCKX

### 5.1.3 De outerjoin

In onze toepassing van de auto-join krijgen we als resultaat maar 13 medewerkers terwijl er 14 medewerkers zijn. In het resultaat ontbreekt echter De Koning. Als de tupelvariabele m.naam De Koning als waarde heeft, dan is er geen enkele rij mm te vinden (NULL-waarde) zodat aan de where-conditie wordt voldaan. Dit probleem kan verholpen worden door de outerjoin. Door toevoeging van (+) kunnen de rijen wel vergeleken worden met NULL-waarden.

```
SQL> select    m.naam as medewerker
,             mm.naam as chef
from          medewerkers m
,             medewerkers mm
where         m.chef = mm.mnr(+);
```

Voor alle duidelijkheid nog een voorbeeld.

Medewerkers

mnr	naam	afd
7001	Thommis	10
7002	Schepers	20
7003	Claesen	30
7004	Brams	20

Afdelingen

anr	naam
10	Boekhouden
20	Expeditie
30	Personeelszaken
40	Verkoop

```
SQL> select m.mnr, a.anr, a.naam
2   from afdelingen a
3   ,     medewerkers m ;
```

```
SQL> select m.mnr, a.anr,a.naam
2   from afdelingen a
3   ,     medewerkers m
4   where m.afd = a.anr;
```

Resultaatstabel (met WHERE)

mnr	anr	naam
7001	10	Boekhouden
7002	20	Expeditie
7003	30	Personeelszaken
7004	20	Expeditie

Resultaatstabel (zonder WHERE)

mnr	anr	naam
7001	10	Boekhouden
7001	20	Expeditie
7001	30	Personeelszaken
7001	40	Verkoop
7002	10	Boekhouden
7002	20	Expeditie
7002	30	Personeelszaken
7002	40	Verkoop
7003	10	Boekhouden
7003	20	Expeditie
7003	30	Personeelszaken
7003	40	Verkoop
7004	10	Boekhouden
7004	20	Expeditie
7004	30	Personeelszaken
7004	40	Verkoop



```
SQL> select m.mnr, a.anr,a.naam
2   from  afdelingen a
3   ,      medewerkers m
4   where m.afd (+) = a.anr;
```

Resultaatstabel (met outerjoin)

mnr	anr	naam
7001	10	Boekhouden
7002	20	Expeditie
7003	30	Personeelszaken
7004	20	Expeditie
NULL	40	Verkoop

### 5.1.4 De ANSI/ISO SQL-standaard

De ANSI/ISO SQL-standaard ondersteunt ook een andere manier om joins te maken. Het ziet er beter uit doordat in het FROM-component de join gespecificeerd is en het WHERE-component de niet-join condities bevat.

#### Voorbeelden

```
select m.naam as medewerker
      ,      mm.naam as chef
      from    medewerkers m
              JOIN medewerkers mm
              ON m.chef = mm.mnr
where m.gbdatum > date '1985-01-01'
order by medewerker;
```

```
select a.anr, a.locatie
      ,      m.naam, m.voorn
      from    medewerkers m
              RIGHT OUTER JOIN
              afdelingen a
              ON m.afd = a.anr
order  by a.anr, m.naam ;
```

We gebruiken de RIGHT OUTER JOIN omdat we aan de rechterkant (afdelingen) de aanwezigheid van rijen vermoeden die geen corresponderende rijen aan de linkerkant (tabel medewerker) hebben. Als de twee tabellen andersom in de FROM-component staan was de LEFT OUTER JOIN nodig geweest. Oracle ondersteunt ook de FULL OUTER JOIN (beide kanten hebben lege rijen).

### 5.1.5 BREAK commando

Met behulp van een BREAK-commando kunnen we in een rapport kolomwaarden onderdrukken, een regel overslaan of een extra regel toevoegen. Op die plaatsen kunnen ook subtotalen berekend worden met behulp van het COMPUTE-commando (zie 9.2.4).

#### Voorbeeld 1

```
SQL> BREAK ON afd SKIP 2      (onderbreking op afdeling en 2 regels toevoegen)
```

```
SQL> BREAK ON afd PAGE        (onderbreking op afdeling met pauzestop)
SQL> SET PAUSE ON
```

```
SQL> select afd, functie, mnr, naam, maandsal, comm
2   from medewerkers
3   order by afd,functie;
```

AFD	FUNCTIE	MNR	NAAM	MAANDSAL	COMM
10	BOEKHOUDER	7934	WOUTERS	2300	
	DIRECTEUR	7839	DE KONING	7000	
	MANAGER	7782	CLERCKX	3450	
20	MANAGER	7566	JACOBS	4975	
	TRAINER	7369	CASPERS	1800	
	TRAINER	7902	DE COOMAN	4000	
	TRAINER	7876	SLECHTEN	2700	
	TRAINER	7788	SWINNEN	4000	
30	BOEKHOUDER	7900	JACOBS	2800	
	MANAGER	7698	BRIERS	5850	
	VERKOPER	7521	DEFOUR	2250	5000
	VERKOPER	7844	DEN RUYTER	2500	0
	VERKOPER	7499	ALLARD	1600	3000
	VERKOPER	7654	MARTENS	2250	3400

### Voorbeeld 2

```
SQL> BREAK ON afdeling SKIP 1 ON functie
```

```
SQL> select a.naam as afdeling
2   ,      m.functie
3   ,      m.naam as medewerker
4   from medewerkers m
5   right outer join
6   afdelingen a
7   on a.anr=m.afd
8   order by a.naam, m.functie;
```

AFDELING	FUNCTIE	MEDEWERKER
HOOFDKANTOOR	BOEKHOUDER	WOUTERS
	DIRECTEUR	DE KONING
	MANAGER	CLERCKX
OPLEIDINGEN	MANAGER	JACOBS
	TRAINER	CASPERS
		SWINNEN
		DE COOMAN
		SLECHTEN
PERSONEELSZAKEN		
VERKOOP	BOEKHOUDER	JACOBS
	MANAGER	BRIERS
	VERKOPER	DEFOUR
		DEN RUYTER
		ALLARD
		MARTENS

- De break-instellingen verwijderen.

```
SQL> clear breaks
```

### 5.1.6 De gepartitioneerde outerjoin

Met behulp van BREAK is het resultaat van de query in voorbeeld 9.1.5 overzichtelijker geworden.

We kregen echter geen rijen voor alle functies omdat er geen medewerkers zijn in de afdeling personeelszaken. Met behulp van PARTITION BY dwingt je de vermelding van alle functies af.

```
SQL> select a. naam as afdeling
2      ,      m.functie
3      ,      m. naam as medewerker
4      from medewerkers m
5           PARTITION BY (functie)
6           right outer join
7           afdelingen a
8           on a.anr=m.afd
9      order by a. naam, m.functie;
```

AFDELING	FUNCTIE	MEDEWERKER
HOOFDKANTOOR	BOEKHOUDER	WOUTERS
	DIRECTEUR	DE KONING
	MANAGER	CLERCKX
	TRAINER	
	VERKOPER	
OPLEIDINGEN	BOEKHOUDER	
	DIRECTEUR	
	MANAGER	JACOBS
	TRAINER	SWINNEN
		SLECHTEN
		CASPERS
		DE COOMAN
	VERKOPER	
PERSONEELSZAKEN	BOEKHOUDER	
	DIRECTEUR	
	MANAGER	
	TRAINER	
	VERKOPER	
VERKOOP	BOEKHOUDER	JACOBS
	DIRECTEUR	
	MANAGER	BRIERS
	TRAINER	
	VERKOPER	DEN RUYTER
		DEFOR
		MARTENS
		ALLARD

## 5.2 Groepsfuncties

### 5.2.1 Group by-component

Tot nu toe hebben we uitsluitend queries gezien waarbij informatie werd verwacht met betrekking tot individuele rijen uit onze tabellen. Het komt echter regelmatig voor dat we geïnteresseerd zijn in geaggregeerde informatie (op verzameling van rijen in plaats van op afzonderlijke rijen).

Veronderstel dat we een overzicht van het aantal medewerkers per afdeling willen. Bij dit soort queries hebben we de group by-component van het select-commando nodig.

```
SQL> select      m.afd          as afdeling
      ,          count(m.mnr)    as aantal_medewerkers
      from        medewerkers    m
      group by    m.afd;
```

AFDELING	AANTAL_MEDEWERKERS
10	3
20	5
30	6

Door de *group by*-optie worden de rijen gesorteerd en gegroepeerd per afdeling en de groepsfunctie *count()* telt het aantal medewerkers per afdeling.

Groeperen kan ook worden toegepast op meerdere kolommen. Onderstaande query geeft een overzicht van het aantal inschrijvingen per cursus.

```
SQL> select      i.cursus, i.begindatum
      ,          count(i.cursist)
      from        inschrijvingen i
      group by    i.cursus, i.begindatum;
```

CURS	BEGINDATU	COUNT(I.CURSIST)
WEB	17-DEC-15	5
WIN	04-FEB-16	2
SQL	08-OCT-15	3
SQL	17-DEC-15	2
ORG	10-AUG-15	3
CIS	11-SEP-16	3
WEB	05-FEB-16	3
SQL	16-APR-15	4
ORG	27-SEP-16	1

### 5.2.2 Groepsfuncties

De groepsfuncties werken op een verzameling waarden, waarna ze één waarde als resultaat retourneren. Vandaar dat groepsfuncties vaak in combinatie met de GROUP BY-component en met de HAVING-component worden toegepast.

COUNT ( )	geeft aantal waarden	alle datatypes
SUM ( )	som van de waarden	numeriek
MIN ( )	minimumwaarde	alle datatypes
MAX ( )	maximumwaarde	alle datatypes
AVG ( )	gemiddelde waarden	numeriek
STDEV ( )	standaarddeviatie	numeriek
VARIANCE ( )	variantie	numeriek

*Hoeveel bedraagt het laagste maandsalaris?*

```
SQL> select min(maandsal)
      from medewerkers;
1600
```

*Wie heeft het laagste salaris? ALLARD*

```
SQL> select naam
      from medewerkers
     where maandsal =
      (select min(maandsal)
       from medewerkers);
of
SQL> select m.naam
      from medewerkers m
     JOIN
      (select min(maandsal) laag
       from medewerkers) mm
     ON m.maandsal = mm.laag
```

*Hoeveel medewerkers telt de onderneming en in hoeveel verschillende afdelingen werken ze?*

```
SQL> select count(*), count(distinct afd)
      from medewerkers;
14      3
```

De count-functie accepteert naast kolomnamen als argument ook een asterisk(\*), waardoor niet langer een aantal waarden wordt geteld, maar het aantal rijen van de groep worden geteld. De asterisk werkt in grote databanken trager. Dus best de primaire sleutel gebruiken.

*Hoeveel verschillende functies zijn er in elke afdeling?*

```
SQL> select afd, count(funcitie), count(distinct functie)
      from medewerkers
     group by afd
     order by afd;
10      3      3
20      5      2
30      6      3
```

*Hoeveel personen ontvangen een commissie en hoeveel bedraagt het gemiddelde?*

```
SQL> select count(comm), avg(comm), avg(nvl(comm,0))
      from medewerkers;
4      550      157.14286
```

### **Merk op**

Zoek de veel voorkomende fouten en verbeter.

```
SQL> select afd, count(*)
      from medewerkers;
.....
```

```
SQL> select m.afd, a.naam, count(*)
      from medewerkers m, afdelingen a
     where m.afd = a.anr
     group by m.afd;
.....
```

```
SQL> select a.anr, count(*)
      from medewerkers m, afdelingen a
      where m.afd(+) = a.anr
      group by a.anr;
```

.....

### 5.2.3 De having-component

Met behulp van de *having*-component kunnen we restricties op groepsniveau leggen. De *having*-component kan in SQL alleen voorkomen na een *group by*-constructie. De *where*-component is de restrictie-operator voor de rijen.

*Hoeveel medewerkers werken er in afdeling 10?*

```
SQL> select afd, count(*)
      from medewerkers
      group by afd
      having afd = 10
```

10	3
----	---

**of**

```
SQL> select afd, count(*)
      from medewerkers
      where afd = 10
      group by afd
```

10	3
----	---

*In welke afdeling zijn er meer dan 4 medewerkers?*

```
SQL> select afd, count(mnr)
      from medewerkers
      group by afd
      having count(*) >= 4 ;
```

20	5
30	6

*Welke functies in afdeling 10 of 20 hebben een gezamenlijk inkomen groter dan 5000?*

```
SQL> select functie, sum(maandsal)
      from medewerkers
      where afd in (10,20)
      group by functie
      having sum(maandsal) > 5000
      order by sum (maandsal) desc
```

MANAGER	5425
TRAINER	7900

**Zoek de fout en verbeter.**

*Geef de medewerkers die een hoger salaris hebben dan het gemiddelde salaris.*

```
SQL> select naam, voorn
      from medewerkers
      where maandsal > avg(maandsal);
```

.....

.....

```
SQL> select m.naam, m.voorn, m.maandsal
      from medewerkers m
      where m.maandsal > (select avg(n.maandsal)
                          from medewerkers n
                          where n.afd = m.afd);
```

#### 5.2.4 BREAK en COMPUTE comando

```
SQL> break on afd (Break altijd samen instellen met COMPUTE)
SQL> col afd format 999999999999999999999999
SQL> compute sum label TOTAAL of maandsal on afd (SUM op maandsal met BREAK op
afd)
SQL> COMPUTE count LABEL 'AANTAL MET COMMISSIE' number label 'AANTAL WERKNEMERS' of comm on afd

SQL> select afd, functie, mnr, naam, maandsal, comm
2  from medewerkers
3  order by afd,functie;
```

	AFD	FUNCTIE	MNR	NAAM	MAANDSAL	COMM
	10	BOEKHOUDER	7934	WOUTERS	2300	
		DIRECTEUR	7839	DE KONING	7000	
		MANAGER	7782	CLERCKX	3450	
*****						
AANTAL MET COMMISSIE						0
AANTAL WERKNEMERS						3
TOTAAL					12750	
	20	MANAGER	7566	JACOBS	4975	
		TRAINER	7369	CASPERS	1800	
			7902	DE COOMAN	4000	
			7876	SLECHTEN	2700	
			7788	SWINNEN	4000	
*****						
AANTAL MET COMMISSIE						0
AANTAL WERKNEMERS						5
TOTAAL					17475	
	30	BOEKHOUDER	7900	JACOBS	2800	
		MANAGER	7698	BRIERS	5850	
		VERKOPER	7521	DEFOUR	2250	5000
			7844	DEN RUYTER	2500	0
			7499	ALLARD	1600	3000
			7654	MARTENS	2250	3400
*****						
AANTAL MET COMMISSIE						4
AANTAL WERKNEMERS						6
TOTAAL					17250	

De toegestane functies van COMPUTE zijn:

<b>AVG</b>	gemiddelde
<b>COUNT</b>	het aantal not null-waarden in een kolom
<b>MAX</b>	maximum
<b>MIN</b>	minimum
<b>NUMBER</b>	aantal rijen
<b>STD</b>	standaardafwijking
<b>SUM</b>	som
<b>VAR</b>	variantie

- Compute-instellingen verwijderen.

SQL> Clear Computes

*Returnwaarde van GROUPING-ID en GROUPING.*

```
SQL> select afd, functie
2      ,      grouping_id (afd,functie)
3   from   medewerkers
4   group by cube(afd, functie);
```

AFD	FUNCTIE	RETURNWAARDE
		3
	MANAGER	2
	TRAINER	2
	VERKOPER	2
	DIRECTEUR	2
	BOEKHOUDER	2
10		1
10	MANAGER	0
10	DIRECTEUR	0
10	BOEKHOUDER	0
20		1
20	MANAGER	0
20	TRAINER	0
30		1
30	MANAGER	0
30	VERKOPER	0
30	BOEKHOUDER	0

```
SQL> select afd, functie
2      ,      grouping (functie)
3   from   medewerkers
4   group by cube(afd, functie);
```

AFD	FUNCTIE	RETURNWAARDE
		1
	MANAGER	0
	TRAINER	0
	VERKOPER	0
	DIRECTEUR	0
	BOEKHOUDER	0
10		1
10	MANAGER	0
10	DIRECTEUR	0
10	BOEKHOUDER	0
20		1
20	MANAGER	0
20	TRAINER	0
30		1
30	MANAGER	0
30	VERKOPER	0
30	BOEKHOUDER	0

### 5.3 Gecorreleerde subqueries

Veronderstel dat je alle medewerkers moet zoeken, die een hoger salaris hebben dan het gemiddelde salaris van hun afdeling. We kunnen de subquery niet meer zelfstandig uitvoeren want we moeten de subquery zien in context van de hoofdquery. In de subquery verwijzen we dus via een tupelvariabele naar de hoofdquery.

```
SQL> select m.naam, m.voorn, m.maandsal
      from medewerkers m
      where m.maandsal > (select avg(n.maandsal)
                          from medewerkers n
                          where n.afd = m.afd);
```

Voor elke rij *m* wordt het gemiddelde salaris berekend voor die afdeling.

We kunnen subqueries niet alleen in de where-component gebruiken, maar ze kunnen ook worden toegepast in de having- en de from-component. Bovenstaande subquery kan ook als volgt opgelost worden.

```
SQL> select m.naam, m.voorn, m.maandsal, g.gem
      from   medewerkers m
      ,      (select n.afd, avg(n.maandsal) gem
              from medewerkers n
              group by n.afd) g
      where  m.afd = g.afd
```



```

and m.maandsal > g.gem;
Wie is de 4de jongste medewerker?
SQL> select m.naam
      from medewerkers m
      where 4 = (select count(*)
                from medewerkers n
                where n.gbdatum >= m.gbdatum);

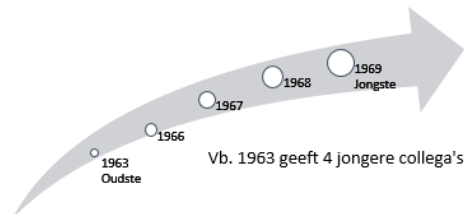
```

Wie zijn de 3 jongste medewerkers?

```

SQL> select m.naam
      from medewerkers m
      where 3 >= (select count(*)
                  from medewerkers n
                  where n.gbdatum >= m.gbdatum);

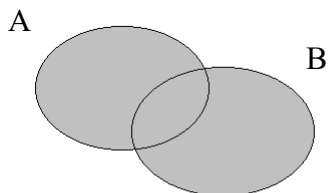
```



## 5.4 Verzamelingoperatoren

Met behulp van de verzamelingoperatoren *union*, *minus* en *intersect* kunnen we de resultaten van twee queries combineren tot één resultaat. Ze onderscheiden zich daarmee van de andere SQL-operatoren in die zin dat ze niet op basistabellen van de database werken, maar op resultaatstabellen die uit queries voorkomen.

### 5.4.1 Union



Vereniging  
 $A = \{ a, b, c, d, e \}$   
 $B = \{ b, d, g, h \}$   
 $A \cup B = \{ a, b, c, d, e, g, h \}$

Geeft alle rijen die in de 2 resultaatstabellen voorkomen.

*Geeft alle locaties (vestigingen en cursusplaatsen).*

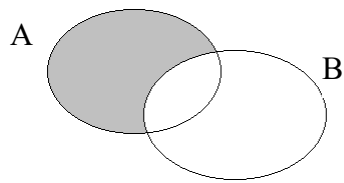
```

SQL> select u.locatie from uitvoeringen u
      2 union
      3 select a.locatie from afdelingen a

```

Merk op dat de union-operator automatisch een distinct doorvoert. **Union all** geeft wel alle rijen.

### 5.4.2 Minus



Verschil

$A = \{ a, b, c, d, e \}$

$B = \{ b, d, g, h \}$

$A \setminus B = \{ a, c, e \}$

$B \setminus A = \{ g, h \}$

De rijen van de resultaatstabel A met uitzondering van de rijen van de resultaatstabel B.

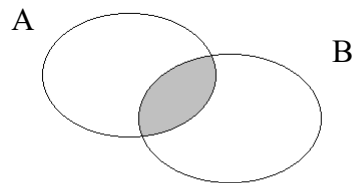
*In welke plaats vinden wel cursussen plaats, maar zijn geen afdelingen gevestigd?*

```
SQL> select u.locatie from uitvoeringen u
      2 minus
      3 select a.locatie from afdelingen a
```

*Welk resultaat verkrijg je als je de resultaatstabellen verwisselt?*

```
SQL> select a.locatie from afdelingen a
      2 minus
      3 select u.locatie from uitvoeringen u
```

### 5.4.3 Intersect



Doorsnede

$A = \{ a, b, c, d, e \}$

$B = \{ b, d, g, h \}$

$A \cap B = \{ b, d \}$

Gemeenschappelijke rijen van de resultaatstabellen.

*In welke vestigingen worden er ook cursussen gegeven.*

```
SQL> select u.locatie from uitvoeringen u
      2 intersect
      3 select a.locatie from afdelingen a
```

De operatoren stellen enkele eisen aan de queries waarop ze worden toegepast. De twee tussenresultaten van Q1 en Q2 moeten namelijk met elkaar te combineren zijn tot één resultaatstabel, hetgeen leidt tot de volgende voor de hand liggende eisen:

- Q1 en Q2 moeten evenveel kolommen selecteren;
- de datatypes moeten overeenstemmen.

### Merk op

- De resultaatstabel erft de kolomnamen/aliassen van Q1;
- Q1 mag geen order by-component bevatten;
- Een eventuele order by-component na Q2 geldt voor het geheel.

## 5.5 Views

### 5.5.1 Kennismaking met views.

Views vormen een belangrijk onderdeel van SQL. Je kan immers aan een ingewikkelde of veel gebruikte query opslaan als viewquery en daarmee werken.

Een view is een *virtuele* tabel met als inhoud het resultaat van een vastgelegde viewquery die bij iedere benadering opnieuw wordt uitgevoerd. Views zijn gedefinieerd in de vorm van een viewqueries die in de datadictionary worden opgeslagen. We spreken ook wel van 'stored queries'. Telkens er een beroep gedaan wordt op de 'inhoud' van de view wordt de viewquery opgehaald en uitgevoerd.

Views worden gecreërd met het SQL-commando CREATE VIEW. Met de optie OR REPLACE kunnen we een viewdefinitie vervangen.

**CREATE [OR REPLACE]**  
**VIEW** viewnaam [kolomalias]  
**AS** query [**WITH READ ONLY**] [**WITH CHECK OPTION**]

#### Opmerking

- Kolomnamen voor de view worden van de query overgenomen. Indien er in de query dezelfde kolomnamen staan of expressies, moet je kolomaliassen gebruiken.
- With Check Option wordt gebruikt bij datamanipulatie om te voorkomen dat rijen toegevoegd worden die niet beantwoorden aan de integriteitsvoorwaarden (zie later).

Veronderstel dat we onderstaand query in een view willen opslaan.

```
SQL> select m.naam,a.naam as afdeling, a.locatie, h.naam hoofd
2   from medewerkers m
3   JOIN afdelingen a
4   ON a.anr = m.afd
5   JOIN medewerkers h
6   ON h.mnr= a.hoofd;
```

```
SQL> CREATE view mah as
2   select m.naam,a.naam as
afdeling, a.locatie, h.naam
hoofd
3   from medewerkers m
4   JOIN afdelingen a
5   ON a.anr = m.afd
6   JOIN medewerkers h
7*  ON h.mnr= a.hoofd
```

NAAM	AFDELING	LOCATIE	HOOFD
CASPERS	OPLEIDINGEN	HASSELT	JACOBS
ALLARD	VERKOOP	GENK	BRIERS
DEFOUR	VERKOOP	GENK	BRIERS
JACOBS	OPLEIDINGEN	HASSELT	JACOBS
MARTENS	VERKOOP	GENK	BRIERS
BRIERS	VERKOOP	GENK	BRIERS
CLERCKX	HOOFDKANTOOR	MAASMECHELEN	CLERCKX
SWINNEN	OPLEIDINGEN	HASSELT	JACOBS
DE KONING	HOOFDKANTOOR	MAASMECHELEN	CLERCKX
DEN RUYTER	VERKOOP	GENK	BRIERS
SLECHTEN	OPLEIDINGEN	HASSELT	JACOBS
JACOBS	VERKOOP	GENK	BRIERS
DE COOMAN	OPLEIDINGEN	HASSELT	JACOBS
WOUTERS	HOOFDKANTOOR	MAASMECHELEN	CLERCKX

View created.

Deze view is nu toegevoegd aan databaseobjecten en kan je raadplegen in de datadictionary.

**SQL> select \* from tab;**

TNAME	TABTYPE	CLUSTERID
AFDELINGEN	TABLE	
CURSUSSEN	TABLE	
INSCHRIJVINGEN	TABLE	
MAH	VIEW	
MEDEWERKERS	TABLE	
SCHALEN	TABLE	
UITVOERINGEN	TABLE	

**SQL> desc mah**

Name	Null?	Type
NAAM	NOT NULL	VARCHAR2(15)
AFDELING	NOT NULL	VARCHAR2(20)
LOCATIE	NOT NULL	VARCHAR2(20)
HOOFD	NOT NULL	VARCHAR2(15)

**SQL> select \*  
2 from mah  
3 where hoofd = 'BRIERS';**

NAAM	AFDELING	LOCATIE	HOOFD
JACOBS	VERKOOP	GENK	BRIERS
DEN RUYTER	VERKOOP	GENK	BRIERS
BRIERS	VERKOOP	GENK	BRIERS
MARTENS	VERKOOP	GENK	BRIERS
DEFOUR	VERKOOP	GENK	BRIERS
ALLARD	VERKOOP	GENK	BRIERS

6 rows selected.

## 5.5.2 Views in de datadictionary.

We kunnen de viewdefinitie opvragen in de datadictionary.

**SQL> set long 999 -- LONG kolommen op 999 karakters zetten**

**SQL> col text format a40 word\_wrapped -- eventueel ook tekstterugloop regelen.**

**SQL> select view\_name, text  
2 from user\_views**

VIEW_NAME	TEXT
MAH	select m.naam,a.naam as afdeling, a.locatie, h.naam hoofd from medewerkers m JOIN afdelingen a ON a.anr = m.afd JOIN medewerkers h ON h.mnr= a.hoofd

## Opmerking

- Als views worden gedefinieerd met een query die begint met SELECT \* dan wordt het sterretje uitgewerkt met alle kolommen in de query.

```
SQL> create view afd20 as
```

```
2  select *
3  from medewerkers
4  where afd=20;
```

View created.

```
SQL> select view_name, text
```

```
2  from user_views;
```

VIEW_NAME	TEXT
AFD20	select "MNR","NAAM","VOORN","FUNCTIE","C HEF","GBDATUM","MAANDSAL","COMM","AFD" from medewerkers where afd=20

### 5.5.3 Views wijzigen.

We wensen in onze view afd20 ook de cursussen op te nemen die de medewerkers volgen.

```
SQL> CREATE OR REPLACE view afd20 as
```

```
2  select m.voorn || ' ' || m.naam naam, i.cursus,  
i.begindatum  
3  from medewerkers m  
4  join inschrijvingen i  
5  on i.cursist = m.mnr  
6* where afd=20
```

NAAM	CURS	BEGINDATUM
EMMA JACOBS	CIS	11-09-2016
EMMA JACOBS	WEB	05-02-2016
CHRIS SWINNEN	SQL	08-10-2015
CHRIS SWINNEN	WEB	17-12-2015
CHRIS SWINNEN	WEB	05-02-2016
TOM SLECHTEN	CIS	11-09-2016
TOM SLECHTEN	SQL	16-04-2015
TOM SLECHTEN	WEB	17-12-2015
DORIEN DE COOMAN	ORG	10-08-2015
DORIEN DE COOMAN	SQL	08-10-2015
DORIEN DE COOMAN	SQL	17-12-2015

### 5.5.4 Views verwijderen.

Views kunnen ook verwijderd worden.

```
SQL> drop view mah;
```

View dropped.

## 5.6 Toepassingsmogelijkheden

Views worden gebruikt omwille van verschillende mogelijkheden:

- Vereenvoudiging: complexe queries kan je stap voor stap opzetten.
- Veel gebruikte queries (standaardquery): queries die vaak worden opgeroepen.

- Views vormen ook nog een krachtig middel bij beveiliging van gegevens. Door middel van views kunnen bepaalde gegevens voor gebruikers worden afgeschermd. De view laat immers maar een bepaald aantal kolommen zien waarop gebruikers verder kunnen werken.
- Bij grote tabellen kan er enig performance-verlies zijn maar dat is verwaarloosbaar klein omdat er op de onderliggende tabellen indexen worden gelegd die al performance-winst kunnen opleveren. Je mag wel geen views op views maken!

### Voorbeeld

Geef een overzicht van alle medewerkers die meer cursussen hebben gevolgd dan het algemeen gemiddelde.

We zoeken eerst op hoeveel dagen iedereen cursus heeft gevolgd.

```
SQL> select m.mnr, m.naam
2 , sum(c.lengte) as dagen
3 from inschrijvingen i
4 join cursussen c
5 on c.code = i.cursus
6 join medewerkers m
7 on m.mnr = i.cursist
8 group by m.mnr, m.naam;
```

MNR	NAAM	DAGEN
7839	DE KONING	8
7499	ALLARD	11
7566	JACOBS	5
7788	SWINNEN	12
7844	DEN RUYTER	1
7900	JACOBS	3
7698	BRIERS	12
7934	WOUTERS	4
7521	DEFOUR	1
7876	SLECHTEN	9
7902	DE COOMAN	9
7782	CLERCKX	4

12 rows selected.

```
SQL> create view cursusdagen as
2 select m.mnr, m.naam
3 , sum(c.lengte) as dagen
4 from inschrijvingen i
5 join cursussen c
6 on c.code = i.cursus
7 join medewerkers m
8 on m.mnr = i.cursist
9 group by m.mnr, m.naam;
```

Nu kunnen we vergelijken met het gemiddeld aantal dagen.

```
SQL> select *
  2 from cursUSDagen
  3 where dagen > (select avg(dagen)
  4               from cursUSDagen);
```

MNR	NAAM	DAGEN
7839	DE KONING	8
7499	ALLARD	11
7788	SWINNEN	12
7698	BRIERS	12
7876	SLECHTEN	9
7902	DE COOMAN	9

6 rows selected.

## 5.7 Oefeningen

### 5.7.1 Join-operator

- 1 Geef van elke medewerker het nummer en de naam van de afdeling waarvoor hij/zij werkt.
- ~~2 Geef de naam en het salaris van elke medewerker die meer verdient dan Clerckx.~~
- 3 Geef de namen van de medewerkers die in dezelfde afdeling werken als Wouters.
- 4 Geef het nummer van elke medewerker die in de afdeling verkoop werkt.
- 5 Geef het nummer van elke medewerker gevolgd door de naam van zijn of haar baas.
- 6 Geef van elk medewerker het medewerkersnummer en de naam van de afdeling waarvoor hij of zij werkt en maak een overzicht per afdeling mogelijk.

AFDELING	MEDEWERKERS
HOOFDKANTOOR	7782 AN CLERCKX
	...
OPLEIDINGEN	7566 EMMA JACOBS
	7902 DORIEN DE COOMAN
	...
VERKOOP	7521 THOMAS DEFOUR
	...

- 7 Welke medewerkers verdienen meer dan hun baas wanneer het salaris van de baas met 1000 EUR verminderd wordt?
- 8 Geef de namen van de chef (ook van De Koning) en hun ondergeschikten. Voorzie een overzicht per chef.

BAAS	ONDERGESCHIKTEN
BRIERS	DEN RUYTER
	JACOBS
	DEFOUR
	MARTENS
	ALLARD ...

- 9 Geef de namen van alle afdelingen en de naam van het afdelingshoofd.

- 10 Geef de naam en voornaam van de medewerkers die een cursus volgen waar een afdelingsafdeling gevestigd is.
- 11 Geef de naam en voornaam van de medewerkers die een cursus volgen waar hun eigen afdeling gevestigd is.
- 12 Geef de namen van de medewerkers die een cursus gevolgd hebben die hun chef ook gevolgd heeft.
- 13 Geef de namen van de docenten die cursussen SQL, ORG of WEB doceren.
- 14 Geef van alle inschrijvingen de naam van de docent, de naam van de cursus en de naam van de cursist. Geef een overzicht per docent en per omschrijving.

DOCENT	OMSCHRIJVING	DEELNEMER
CASPER	Introductie SQL en databanken	BRIERS DE COOMAN DE KONING SWINNEN
	Windows Server	ALLARD JACOBS
DE COOMAN	Introductie SQL en databanken	ALLARD BRIERS SLECHTEN WOUTERS ...

- 15 Geef de namen van alle cursussen samen met de docent die deze cursussen doceren, evenals de begintdata. De kolom omschrijving mag maar 35 karakters breed zijn en bovendien moet je voor een lege waarde bij docent '==onbekend==' weergeven. Geef een overzicht per cursusnaam.

OMSCHRIJVING	BEGINDATU DOCENT
Cisco CCNA	11-SEP-16 SWINNEN
IT Organisatie	10-AUG-15 JACOBS 27-SEP-15 DE COOMAN
Introductie SQL en databanken	08-OCT-15 CASPERS 17-DEC-15 CASPERS 16-APR-15 DE COOMAN
Linux OS	13-JAN-17 ==ONBEKEND==
Ontwikkeling website	17-DEC-15 JACOBS 05-FEB-16 SLECHTEN
Programmeren1 Visual C#	17-FEB-17 ==ONBEKEND==
Web applicaties	24-FEB-17 SWINNEN
Windows Server	18-SEP-16 ==ONBEKEND== 04-FEB-16 CASPERS

- 16 Geef van alle cursisten hun naam, de naam van hun chef, de naam van de afdeling waar ze werken, de namen van de cursussen waarvoor ze zich eventueel inschreven en de namen van de docenten die deze cursussen geven.

DEELNEMER	CHEF	AFDELING	OMSCHRIJVING	DOCENT
ALLARD,NELE	BRIERS	VERKOOP	Cisco CCNA Introductie SQL en databanken Ontwikkeling website Windows Server	CHRIS SWINNEN DORIEN DE COOMAN EMMA JACOBS JANA CASPERS
BRIERS,ANDREA	DE KONING	VERKOOP	Introductie SQL en databanken Introductie SQL en databanken Ontwikkeling website	DORIEN DE COOMAN JANA CASPERS TOM SLECHTEN
CLERCKX,AN	DE KONING	HOOFDKANTOOR	Ontwikkeling website	EMMA JACOBS ...



## 5.7.2 Oefeningen groepsfuncties

- 1 Geef het aantal medewerkers per afdeling.
- 2 Geef het gemiddelde salaris per afdeling.
- 3 Geef het aantal medewerkers per afdeling, maar enkel voor de afdelingen waar in de afdelingsnaam een "OO" voorkomt.
- 4 Geef het aantal medewerkers voor alle afdelingen behalve de afdeling 10.
- 5 Geef het maximum en het minimum salaris en de afdelingsnaam per afdeling.
- 6 Geef de naam en het salaris van de medewerker die het meest verdient.
- 7 Geef per afdeling en per functie het aantal medewerkers en het gemiddelde salaris.
- 8 Geef per afdeling het gemiddelde salaris voor die afdelingen waar meer dan 3 medewerkers werken.
- 9 Geef per afdeling de naam van de afdeling en de naam van de medewerker(s) die het meest verdient.
- 10 Geef de namen van de medewerkers die een hoger salaris hebben dan het gemiddelde salaris.
- 11 Geef de namen van de medewerkers die een hoger salaris hebben dan het gemiddelde salaris van hun afdeling.
- 12 Geef het aantal cursussen die elke medewerker gevolgd heeft (stijgend gesorteerd).
- 13 Wat is het gemiddelde salaris van de medewerkers die in dezelfde afdeling werken als 'Den Ruyter'?
- 14 Hoeveel medewerkers verdienen minder dan het gemiddelde salaris plus 200?
- 15 Hoe heten de drie hoogst betaalde medewerkers?
- 16 Wat zijn de namen en de salarissen van de vijf laagst betaalde medewerkers?
- 17 Idem vraag 16 maar sorteer het resultaat dalend op salaris.
- 18 Hoe heten de 3 meest verdienende verkopers?
- 19 Welke is van alle gemiddelde maandsalarissen per afdeling het hoogste gemiddelde maandsalaris?
- 20 Welke afdeling (naam) heeft het hoogste gemiddelde maandsalaris?
- 21 Op welke datum is voor het laatst een cursus gepland?
- 22 Geef de 2 volgende resultaatqueries:

AFD	FUNCTIE	COUNT(MNR)	AVG(MAANDSAL)
10	BOEKHOUDER	1	2300
	DIRECTEUR	1	7000
	MANAGER	1	3450
*****			
GEMIDDELDE			4250
AANTAL		3	

20	MANAGER	1	4975
	TRAINER	4	3125
*****			
	GEMIDDELDE		4050
	AANTAL	5	
30	BOEKHOUDER	1	2800
	MANAGER	1	5850
	VERKOPER	4	2150
*****			
	GEMIDDELDE		3600
	AANTAL	6	

AFD	FUNCTIE	COUNT(MNR)	AVG(MAANDSAL)
10	BOEKHOUDER	1	2300
	DIRECTEUR	1	7000
	MANAGER	1	3450
		3	4250
*****			
	GEMIDDELDE		4250
	AANTAL	6	
20	MANAGER	1	4975
	TRAINER	4	3125
		5	3495
*****			
	GEMIDDELDE		3865
	AANTAL	10	
30	BOEKHOUDER	1	2800
	MANAGER	1	5850
	VERKOPER	4	2150
		6	2875
*****			
	GEMIDDELDE		3418.75
	AANTAL	12	
		14	3391.07143
*****			
	GEMIDDELDE		3391.07143
	AANTAL	14	

### 5.7.3 Oefeningen verzamelingsoperatoren

**Union-operator: samenvoegen van relaties.**

**Difference-operator: verschil tussen relaties**

**Intersect-operator: doorsnee van relaties**

- 1 Geef een lijst van de medewerkersnamen en de afdelingsnamen.
- 2 Geef een lijst met namen van afdelingen en namen van cursussen die respectievelijk in Hasselt gevestigd zijn of daar doorgaan.
- 3 Geef de medewerkers die geen manager zijn.
- 4 Geef de locaties waar wel cursussen doorgaan maar waar geen afdelingen gevestigd zijn.
- 5 Geef de naam en begindatum van elke cursus die alleen door de medewerker 7844 gevolgd is.
- 6 Geef de cursuscode van de cursussen die E. Jacobs niet gevolgd heeft.
- 7 Geef de namen van de medewerkers die geen cursussen gevolgd hebben.

- 8 In welke plaatsen is minstens één afdeling gevestigd en wordt minstens één cursus gepland.
- 9 Geef de nummers van de medewerkers die de cursus 'Windows Server' gevolgd hebben en in de afdeling Verkoop werken.

#### 5.7.4 Oefeningen View

- 1 Maak een view BAAS dat het nummer van elke medewerker geeft, gevolgd door de naam van zijn of haar baas.
- 2 Wijzig de view zodat de gegevens stijgend gesorteerd op de naam van de baas worden weergegeven.
- 3 Zoek de view op in de datadictionary en geef de viewdefinitie.
- 4 Maak een view JAARSAL dat de voornamen, de namen, afdelingsnaam en het jaarsalaris (incl. commissie) van alle medewerkers berekent.

VOORN	NAAM	AFDELING	JAARSALARIS
JANA	CASPERS	OPLEIDINGEN	21600
NELE	ALLARD	VERKOOP	22200
THOMAS	DEFOR	VERKOOP	32000
EMMA	JACOBS	OPLEIDINGEN	59700
RAF	MARTENS	VERKOOP	30400
ANDREA	BRIERS	VERKOOP	70200
AN	CLERCKX	HOOFDKANTOOR	41400
CHRIS	SWINNEN	OPLEIDINGEN	48000
LIEVE	DE KONING	HOOFDKANTOOR	84000
JOACHIM	DEN RUYTER	VERKOOP	30000
TOM	SLECHTEN	OPLEIDINGEN	32400
SIMON	JACOBS	VERKOOP	33600
DORIEN	DE COOMAN	OPLEIDINGEN	48000
SVEN	WOUTERS	HOOFDKANTOOR	27600

- 5 Gebruik de view om enkel de medewerkers van de afdeling opleidingen te selecteren.
- Verwijder je laatste view.

## 6 Data Manipulation Language

In een productieomgeving wordt datamanipulatie vaak gepleegd via applicaties, zeker als het om grote hoeveelheden gegevens gaat. Dit soort applicaties wordt over het algemeen gebouwd met behulp van tools zoals Oracle Forms. Toch is het soms wel eens handig om datamanipulatie in SQL uit te voeren als het gaat over globale updates.

### 6.1 Gegevens invoeren

Met behulp van het insert-commando kunnen rijen aan een tabel worden toegevoegd.

```
INSERT INTO tabelnaam [(kolomnaam,...)]  
VALUES (expressie,...)
```

```
INSERT INTO tabelnaam [(kolomnaam,...)]  
SELECT [kolomnaam,...]  
FROM tabelnaam
```

Je kan gegevens toevoegen op twee manieren:

- via de values-component, door een verzameling kolomwaarden op te geven.
- met behulp van een subquery, gebruik maken van bestaande gegevens.

Met de *eerste* methode kan per uitvoering van het insert-commando slechts één rij aan een tabel worden toegevoegd. Als men de fysieke kolomvolgorde van een tabel kent (zoals het describe-commando ze weergeeft) hoeven de kolomnamen niet te worden vermeld. Het is dan noodzakelijk dat er precies genoeg waarden (in de juiste volgorde) worden verstrekt. Met behulp van het gereserveerde woord NULL kan een kolom van een NULL-waarde worden voorzien.

De *tweede* methode vult een tabel met behulp van een query. Het is de snelste manier om een tabel op te vullen met gegevens. Zorg hierbij dat het aantal vermelde kolommen overeenstemmen met het aantal kolommen in de query.

Vaak wordt in de values-component constanten gebruikt, maar met het gebruik van substitutievevariabelen kan je de gegevens achtereenvolgens manueel toevoegen. De substitutievevariabelen verwijzen naar waarden van variabelen. Bij het uitvoeren van het SQL-commando wordt dan om een waarde gevraagd. Het karakter waarmee SQL\*Plus verwijzingen naar een substitutievevariabele herkent, is de ampersand (&) ook wel define-karakter genoemd.

*Met behulp van constanten een rij toevoegen aan de tabel personeel*

```
SQL> insert into personeel  
2 values ( 7955, 'NIJS', 'P', 'TRAINER', 7566, date '1997-02-19', 2860, NULL, 20);
```

*Gegevens toevoegen in willekeurige volgorde.*

```
SQL> insert into personeel (naam,voorn,mnr,maansal,gbdatum)
2 values( 'Vos', 'G', 7956, 2860, date '1983-05-22');
```

**Merk op** dat alleen de gegevens in NOT NULL-rijen worden toegevoegd en dat het attribuut afdeling automatisch de default-waarde 10 krijgt.

*Met behulp van een substitutievevariabele rijen toevoegen*

```
SQL> insert into personeel
2 values ( &mnr,&naam','&voorn',&functie,&chef,to_date('&gbdatum','dd-mm-yyy'),
        &maansal,&comm, &afd);
Enter value for mnr: 7666
Enter value for naam: NIJS
Enter value for voorn: P
Enter value for functie: null
Enter value for chef: 7698
Enter value for gbdatum: 24-09-1997
Enter value for maansal: 1980
Enter value for comm: 200
Enter value for afd: 30
old 2: values ( &mnr,&naam','&voorn',&functie,&chef,to_date('&gbdatum','dd-mm-
        yyyy'),&maansal,&comm, &afd)
new 2: values ( 7666, 'NIJS', 'P', null, 7698, to_date('24-09-1997','dd-mm-yyyy'), 1980, 200,
30)
```

**Merk op** dat bij alfanumerieke waarden en datums de expressies tussen "worden geplaatst. Om te vermijden dat bij de kolom functies de waarde NULL eveneens tussen "geplaatst wordt, wordt de expressie zonder "gebruikt.

*Op basis van een query worden 4 rijen toegevoegd.*

```
SQL> insert into schalen
2 select snr+5, ondergrens + 2300, bovengrens + 2300, 500
3 from schalen
1 where snr <= 4;
```

## 6.2 Gegevens wijzigen

Met het update-commando kunnen kolomwaarden van bestaande rijen in een tabel worden gewijzigd.

UPDATE tabelnaam SET kolomnaam = waarde, .... WHERE voorwaarde
--

Het commando heeft drie componenten met de volgende functie:

UPDATE ... welke tabel er gewijzigd moet worden;  
SET ... om welke wijziging het gaat;  
WHERE ... op welke rijen de wijziging van toepassing is.

Als de where-component wordt weggelaten zal de wijziging op alle rijen van de tabel worden uitgevoerd. Het update-commando werkt in principe op tabelniveau en niet op rijniveau, de where-component werkt ook hier als de relationele restrictieoperator.

*Meerdere kolommen van rij 7666 in de tabel medewerkers wijzigen.*

```
SQL> update medewerkers
2 set      functie  = 'TRAINER'
3 ,        maandsal = 5030
4 ,        comm     = 330
5 ,        afd      = 20
6 where mnr = 7666;
```

*Alle medewerkers krijgen een loonsverhoging van 10%.*

```
SQL> update medewerkers
2 set maandsal = maandsal *1.1;
```

*De werknemers van het hoofdkantoor krijgen een loonsverhoging van 10%.*

```
SQL> update medewerkers
2 set maandsal = maandsal *1.1
3 where afd = ( select anr
4              from afdelingen
5              where naam = 'HOOFDKANTOOR');
```

## 6.3 Gegevens verwijderen

Het eenvoudigste datamanipulatie-commando is het delete-commando.

DELETE FROM tabelnaam WHERE voorwaarde
---

Ook dit commando werkt op tabelniveau. Met de where-component wordt de restrictie tot bepaalde rijen van de tabel gerealiseerd. Als men de where-component achterwege laat zal het resultaat van het delete-commando een lege tabel zijn.

Let op het verschil tussen het commando drop table en delete. Het drop table-commando verwijdert niet alleen de inhoud van de tabel, maar ook de gegevens en de daarmee samenhangende structuren (indexen, privileges, vreemde sleutels,...). Bovendien is het een DDL-commando. Het delete-commando verandert niets aan de structuur van de database, maar verwijdert enkel de inhoud. Het is een DML-commando.

### Tip

Vergeet je constraints niet bij het verwijderen, toevoegen of wijzigen van gegevens. Je kunt immers geen gegevens toevoegen die niet aan voorwaarden voldoen of gegevens verwijderen die verwijzen naar een andere tabel.

SQL > ALTER TABLE med  
disable constraint..... ;

## 6.4 Transactieverwerking

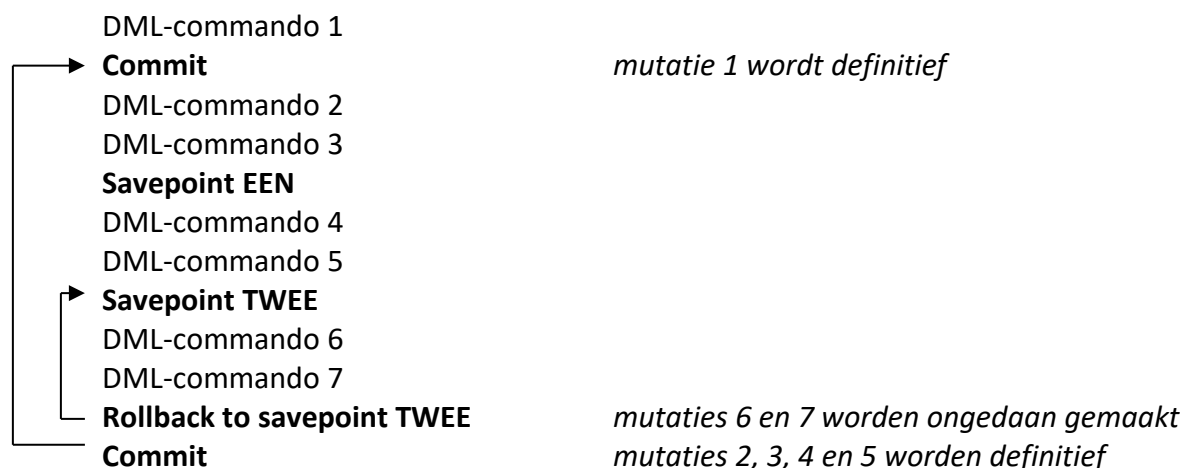
Wijzigingen in de inhoud van de database krijgen in eerste instantie een voorlopig karakter. Dat houdt onder meer in dat we de door ons gewijzigde waarden zelf kunnen bekijken, vooraleer we de wijzigingen definitief doorvoeren.

COMMIT  
ROLLBACK [(TO SAVEPOINT savepointnaam)]  
SAVEPOINT naam

Het commando om wijzigingen in de database definitief te maken is het commit-commando, terwijl met het rollback-commando wijzigingen ongedaan kunnen gemaakt worden.

Dit laat ons toe om verschillende mutaties door te voeren en ze bevestigen met commit om zo weer enkele mutaties te verrichten,... Met behulp van Savepoint kan men markeringspunten plaatsen tussen de verschillende mutaties.

### Voorbeeld



Vergeeten we echter om onze mutaties te “committen” dan lopen we het risico dat bij een systeemstoring als ons werk verloren gaat. Bovendien zijn de mutaties niet zichtbaar voor de andere databasegebruikers. Oracle hanteert een impliciete commit bij het beëindigen van SQL\*Plus.

**Merk op** dat alle DDL-commando's (create table, drop table, alter table) en de DCL-commando's (grant en revoke) een impliciete commit hebben. Dwz dat deze transacties onmiddellijk definitief zijn. SQL\*Plus kent een autocommit waarmee het mogelijk is om ook DML\_commando onmiddellijk te laten "committen". Het gevolg is namelijk dat er geen rollback meer mogelijk is na vergissingen.

Gebruik voor de autocommit de opties of het SQL\*Plus-commando set.  
SQL> set autocommit on

## 6.5 Oefeningen

### 6.5.1 Voer script CreCaseDML.sql uit.

### 6.5.2 Voeg de gegevens uit de case-tabellen toe naar de nieuwe tabellen:

- SchalenDML
- CursussenDML
- MedewerkersDML
- AfdelingenDML
- UitvoeringenDML
- InschrijvingenDML

### 6.5.3 Zoek de fout bij het toevoegen van gegevens uit de tabel Inschrijvingen.

### 6.5.4 Wijzig de gegevens in de tabel Uitvoeringen en voer de bewerking definitief door.

### 6.5.5 Voer onderstaande wijzigingen uit:

- Wijzig de naam van medewerker met mnr 7876 in Boonen.
- Verander de locatie van afdeling 10 in Tongeren.
- Verwijder cursus LIN.
- Alle medewerkers van de afdeling Verkoop krijgen 10% opslag.
- Voeg aan de tabel MEDEWERKERSDML de gegevens toe van een nieuwe medewerker: 7999, Willem Revis, 21/01/1983, boekhouder, salaris € 2950, chef 7782.
- Voer de gegevens in van nog een medewerker: Polien Dox, 7989, trainer, chef 7902 en geboren op de 350<sup>ste</sup> dag van 1980, om 3:30 's nachts.

### 6.5.6 Maak een script en voeg onderstaande gegevens toe:

Voeg in tabel HistoriekDML de volgende rijen toe:

MNR	BEGINJAAR	BEGINDATUM	EINDDATUM	AFD	MAANDSAL	OPMERKINGEN
7369	2015	1/01/2015		20	1800	
7499	2010	1/06/2010	1/11/2014	30	1000	
7499	2014	1/11/2014		30	1600	Targets gehaald.
7521	1999	1/08/1999		30	2250	Blijft liefs op afdeling.
7566	2011	1/12/2011	1/03/2019	20	3500	Niet geschikt al docent, wel leiderschapskwaliteiten.
7566	2019	1/03/2019		20	4975	Promotie!
7654	2019	1/01/2019		30	2250	Senior verkoper, op te volgen.
7698	2007	1/01/2007	1/01/2012	30	3000	Goede start als verkoper
7698	2012	1/01/2012	1/04/2019	30	4350	Promotie!



7698	2019	1/04/2019		30	5850	Hoofd van afdeling verkoop.
7782	2015	15/10/2015		10	3450	Aangenomen als manager voor hoofdkantoor.
7788	2004	10/07/2004		20	4000	
7839	1996	1/01/1996	1/01/2010	20	2500	Oprichter bedrijf
7839	2010	1/01/2010		10	7000	Afsplitsen naar hoofdkantoor
7844	2008	7/08/2008		30	2500	
7876	2004	20/05/2004	14/09/2016	20	2000	Junior medewerker, mentor toegewezen
7876	2016	14/09/2016		20	2700	Salaris verhoging wegens goed project.
7900	2015	1/01/2015		30	2800	
7902	1998	1/12/1998		20	4000	
7934	2005	15/09/2005	15/11/2017	30	1980	
7934	2017	15/11/2017		10	2300	Overplaatsing naar hoofdkantoor

### 6.5.7 Case Saldo

- Voer script CreCaseSaldo.sql uit.
- Voeg volgende gegevens toe in de tabel Saldo:

1	Jo	Lambrichts	800
1	Kristof	Palmaers	1300

Waarom krijg je een fout bij het toevoegen van de 2<sup>de</sup> rij?

- Voer volgende verrichtingen uit:  
Opname van € 300,00 voor dhr. Lambrichts  
Storting van € 300,00 voor dhr. Palmaers
- Bekijk alle gegevens in de tabel en voer de bewerking definitief door.
- Voer opnieuw volgende verrichtingen uit maar zorg ervoor dat je de bewerking ongedaan kan maken :  
Opname van € 300,00 voor dhr. Lambrichts  
Storting van € 300,00 voor dhr. Palmaers