

GRADUAAT IN HET **PROGRAMMEREN**

Data Analysis & SQL

Opleidingsonderdeel

Informatica | Programmeren

Afdeling

Patricia Briers

Auteur



**DE HOGESCHOOL
MET HET NETWERK**

INHOUD

1	RELATIONEEL MODEL: BEGRIPPEN	5
1.1	ALGEMENE BEGRIPPEN	5
1.2	SOORTEN GEGEVENS	6
1.2.1	<i>Samengestelde/elementaire gegevens</i>	<i>6</i>
1.2.2	<i>Procesgegevens</i>	<i>6</i>
1.2.3	<i>Sleutels</i>	<i>7</i>
1.2.4	<i>Referentiële integriteit</i>	<i>9</i>
1.2.5	<i>Redundant</i>	<i>9</i>
2	RELATIONEEL DATAMODEL - ONTWERP	10
2.1	NORMALISATIE	10
2.1.1	<i>Eerste Normaalvorm</i>	<i>13</i>
2.1.2	<i>Tweede Normaalvorm</i>	<i>17</i>
2.1.3	<i>Derde Normaalvorm</i>	<i>19</i>
2.1.4	<i>Verdere Normaalvormen</i>	<i>22</i>
2.1.5	<i>Oefeningen</i>	<i>23</i>
	• <i>Werkbon</i>	<i>27</i>
	• <i>Factuur</i>	<i>29</i>
	• <i>Rekeninguittreksel</i>	<i>31</i>
	• <i>Klantenkaart</i>	<i>32</i>
	• <i>Artikelkaart</i>	<i>35</i>
2.2	ENTITY-RELATIONSHIP DIAGRAM	38
2.2.1	<i>ER diagram symbolen</i>	<i>38</i>
2.2.2	<i>Kardinaliteit in een relatie</i>	<i>39</i>
2.2.3	<i>Tips voor het ontwerp van ER-diagrammen in draw.io</i>	<i>41</i>
3	DATA DEFINITION LANGUAGE	42
3.1	SCHEMA'S EN GEBRUIKERS	42
3.2	TABELLEN MAKEN	42
3.3	DATATYPES	43
3.4	CONSTRAINTS	44
3.5	DE CASUSTABELLEN	45
3.6	DE DATADICIONARY	47
3.7	ALTER TABLE	48
3.8	INDEXEN	49
3.8.1	<i>Index maken</i>	<i>49</i>
3.8.2	<i>Index opvragen</i>	<i>50</i>
3.8.3	<i>Index verwijderen</i>	<i>50</i>
3.9	SEQUENCES	50
3.10	DROP TABLE	52
3.11	OVERIGE COMMANDO'S	52
3.11.1	<i>Truncate</i>	<i>53</i>
3.11.2	<i>Rename</i>	<i>53</i>
3.11.3	<i>Synoniemen</i>	<i>53</i>
3.11.4	<i>Comment on</i>	<i>53</i>
3.11.5	<i>Scripts</i>	<i>54</i>
3.12	OEFENINGEN	55
3.12.1	<i>Creatie database Teams</i>	<i>55</i>
3.12.2	<i>Creatie database Garage Gebr. Valkenborg</i>	<i>56</i>
3.12.3	<i>Creatie database Kinopolis</i>	<i>56</i>
3.12.4	<i>Creatie tabel HISTORIE in dB medewerkers</i>	<i>57</i>
4	DATA MANIPULATION LANGUAGE	61
4.1	GEGEVENS INVOEREN	61
4.2	GEGEVENS WIJZIGEN	62
4.3	GEGEVENS VERWIJDEREN	63

4.4	TRANSACTIEVERWERKING.....	64
4.5	OEFENINGEN	65
4.5.1	<i>Database Teams</i>	65
5	SQL SERVER MANAGEMENT STUDIO (SSMS).....	66
5.1	INSTALLEER SQL SERVER MANAGEMENT STUDIO.....	66
5.2	WAT KAN SQL SERVER MANAGEMENT STUDIO?.....	66
5.3	CONNECTIE.....	67
5.4	ORACLE SQL VERSUS SQL SERVER.....	67
	<i>Data Types</i>	67
	<i>SELECT Statement</i>	68
	<i>SQL Statements</i>	68
	<i>SQL*Plus Commands</i>	69
5.5	CREATIE VAN DATABASE EN TABELLEN.....	69
5.5.1	<i>DML met de SQL commando's</i>	69
5.5.2	<i>DML met de GUI van SQL Server</i>	72
5.6	DATA MANIPULATION LANGUAGE	76
5.6.1	<i>DML met de SQL commando's</i>	76
5.6.2	<i>DML met de GUI van SSMS</i>	78
5.7	RAADPLEGEN VAN GEGEVENS IN SQL SERVER	79
5.7.1	<i>Select</i>	79
5.7.2	<i>Select Top</i>	80
5.7.3	<i>Geavanceerde select</i>	81
5.8	SQL FUNCTIES	83
5.8.1	<i>Alle Functies</i>	83
5.8.2	<i>Algemene functies</i>	85
5.8.3	<i>Rekenkundige Functions</i>	85
5.8.4	<i>String Functions</i>	87
5.8.5	<i>Datetime functies</i>	89
5.8.6	<i>Conversie en Format Functies</i>	91
5.9	TOEPASSINGEN	95
5.9.1	<i>Data Definition Language</i>	95
5.9.2	<i>Data Manipulation Language</i>	95
5.9.3	<i>Het medewerkersvoorbeeld</i>	95

1 Relatieveel model: begrippen

1.1 Algemene begrippen.

We weten ondertussen dat een database een door een *database managementsysteem* (DBMS) beheerde **geïntegreerde verzameling van gegevens** is, waarbij aan de structuur van de gegevensverzameling bepaalde eisen worden gesteld.

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.

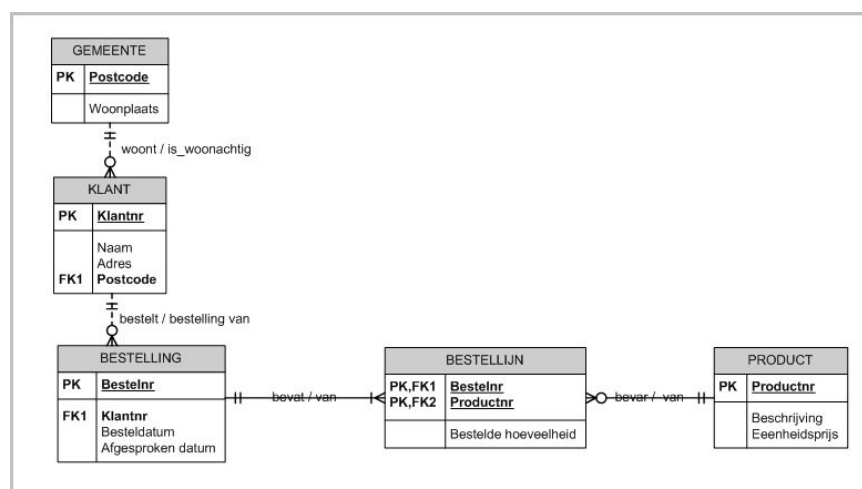
De structuur van een database wordt op 3 niveaus beschreven: het intern niveau, het conceptueel niveau en het extern niveau.

Het **intern** niveau geeft aan op welke manier de gegevens feitelijk in de database worden vastgelegd. Bv. op welke schijf moeten de gegevens komen, ruimte op de schijf, indexeringstechnieken,...

Het **conceptuele** niveau geeft op een logische manier aan:

- Welke gegevens in de database bijgehouden worden.
- Welke soorten gegevens samen kolommen/rijen vormen.
- Welke regels gelden (waardebereik, de relaties tussen de tabellen,...). De grafische vastlegging van het conceptuele datamodel gebeurt veelal in een Entity Relationship Diagram (ERD).

Voorbeelden ERD



kraaienpootnotatie

- Het **extern niveau** geeft aan hoe de eindgebruiker de gegevens ter beschikking krijgt. Vb. welke applicatie moet worden voorzien, welke gegevens mag een bepaald soort eindgebruiker raadplegen (beveiliging van gegevens),....

1.2 Soorten gegevens

1.2.1 Samengestelde/elementaire gegevens

Allereerst zijn we geneigd om bij elkaar horende gegevens te groeperen tot samengestelde gegevens. Zo spreken we vaak van NAW als we Naam, Adres en Woonplaats bedoelen. Dit kan een bron van onduidelijkheid vormen want wellicht bedoelen we met NAW titel, Voornaam, Familienaam, Straatnaam, Huisnummer, Postcode en Woonplaats. In een database mogen er echter enkel elementaire gegevens gebruikt worden. Dus de samengestelde gegevens moeten opgesplitst worden in elementaire gegevens.

Vb. samengesteld voornaam+familienaam → elementair: familienaam

1.2.2 Procesgegevens

Gegevens kunnen soms door berekening uit andere gegevens worden afgeleid. Zulke gegevens noemen we procesgegevens daar ze door middel van een proces kunnen bepaald worden.

Voorbeeld:

Aantal	Artikelnr	Omschrijving	Eenheidsprijs	Bedrag
2	dB bouwen		€ 29,95	€ 59,90
5	dB en SQL		€ 26,50	€ 132,50
			Totaal	€ 192,40
Btw : 6%			Btw	€ 11,54
			Te betalen	€ 203,94

Bedrag is een procesgegeven. De berekening is eenvoudig: de prijs vermenigvuldigd met het aantal stuks geeft het bedrag.

Zijn er nog procesgegevens?

Ja, *Totaal*, *Btw* en *Te betalen* zijn af te leiden uit *Aantal* en *Eenheidsprijs*.

In het algemeen kan men stellen dat als een gegeven wijzigt en er ten gevolge van die wijziging ook een wijziging bij een ander gegeven moet worden aangebracht, dit laatste gegeven een procesgegeven is. Deze afgeleide gegevens worden niet bewaard in een database behalve wanneer deze gegevens vaak worden geconsulteerd. Vb. boekhouddienst raadpleegt heel vaak het totaal van de facturen.

1.2.3 Sleutels

- **Primaire sleutel**

Een kandidaat sleutel is de combinatie van het minste aantal kolommen dat nodig is om een unieke rij in een tabel te identificeren in een databank . De waarden van andere attributen/kolommen zijn afhankelijk van de sleutel.

Een '*supersleutel*' is de combinatie van een kandidaat-sleutel met mogelijk aanwezige, andere attributen. Het aantal attributen van een kandidaat-sleutel is ten minste één en ten hoogste gelijk aan het totale aantal attributen van een tabel.

Voorbeeld studentnummer



The diagram shows a table with the title 'CURSIST' above it. To the left of the table is an orange box labeled 'PK' with an arrow pointing to the first column of the table. The table has four rows and three columns. The first column contains the values '001', '002', '003', and '004'. The second column contains the names 'Briers', 'Vandeput', 'De Cooman', and 'Reyskens'. The third column contains three dots '...' in each row.

CURSIST		
001	Briers	...
002	Vandeput	...
003	De Cooman	...
004	Reyskens	...

De **primaire sleutel** van een tabel is de eenvoudigste kandidaatsleutel die voorkomt in de tabel .

De *primaire sleutel* (*primary of prime key*) identificeert de rijgegevens uniek in de tabel. De primaire sleutel moet dan ook altijd **uniek** zijn!

Surrogaatsleutels

Een surrogaatsleutel is een kunstmatige kolom dat aan een tabel wordt toegevoegd om als de primaire sleutel te fungeren. Er wordt een surrogaatsleutel gebruikt als de primaire sleutel te groot of onhandig is.

Vb. VERHUUR_ITEM (Straat, Postcode, Gemeente, VerhuurTarief)

VERHUUR_ITEM (VerhuurID, Straat, Postcode, Gemeente, VerhuurTarief)

Door de surrogaatsleutel *VerhuurID* toe te voegen, wordt het aanmaken en opzoeken in een tabel efficiënter.

- **Secundaire sleutel**

Voor de sleutel kiest men bij voorkeur de kandidaat-sleutel met het kleinste aantal attributen en dan liefst de numerieke attributen omdat die eenvoudiger te vergelijken zijn dan alfanumerieke attributen.

Het grote nut van primaire sleutels (van sleutels in het algemeen, dus ook van secundaire sleutels) is natuurlijk dat we daarmee in een tabel de gewenste entiteit kunnen vinden en daarmee beschikken over alle andere attribuutwaarden van die tabel. Een attribuut dat geen primaire sleutel is, maar toch wordt gebruikt voor het zoeken van een attribuut met een bepaalde attribuutwaarde heet een **secundaire sleutel**. Vb. geboortedatum

- **Vreemde sleutel**

Door een sleutel van een tabel naar een andere tabel te verwijzen ontstaat tussen de tabellen een koppeling. Een zo gebruikt attribuut (of combinatie van attributen) heet een ***vreemde sleutel*** (*foreign key*). De vreemde sleutel verwijst naar de primaire sleutel van de andere tabel. Het kan echter ook naar een andere kolom verwijzen (best niet gebruiken).

Voorbeeld

ARTIKEL	<u>Artikelnr</u> , Omschrijving, Leveranciernr , ...
PRODUCENT	Leveranciernr , Adres, Telefoon,

In de tabel ARTIKEL kunnen we bij elk Artikelnr via het *vreemde-sleutelattribuut* **Leveranciernr** in de tabel PRODUCENT het adres van de leverancier van het desbetreffende artikel vinden.

Uit deze voorbeelden zien we dat een gegevensbank alleen dan **CONSISTENT** is wanneer de waarde van iedere vreemde sleutel ook als primaire sleutel in een (ander) bestand voorkomt. Anders is die vreemde sleutel zinloos!

Tenslotte nog enkele praktijkopmerkingen over sleutels!

Heel vaak willen we met de sleutel een zekere ordening of classificatie van entiteiten bereiken. Bijvoorbeeld door aan sommige cijfers van een numerieke sleutel een aparte betekenis te verbinden.

- Voor een artikelnummer van materialen kan zo de waarde van het eerste cijfer aangeven of het gaat om glas, steen, ijzer, koper, enzovoort.
- Het tweede cijfer zou kunnen uitmaken in welk toepassingsgebied dat materiaal wordt gebruikt.
- Het derde cijfer in welke vorm het materiaal wordt geleverd (staaf, plaat, kubus, korrel, enzovoort).

Als een sleutel op zo'n manier een bepaalde betekenis heeft, is het zaak om voor latere, nu nog onbekende aanvullingen de nodige ruimte te verschaffen door bijvoorbeeld al meer cijfers in de sleutel op te nemen dan direct nodig is. Een codering in een later stadium te moeten wijzigen is in de praktijk een ramp.

Om fouten uit te sluiten die ontstaan door menselijke vergissingen, gebruikt men soms - bijvoorbeeld in nummers van bankrekeningen - '**zichzelf controlerende**' sleutels.

Een heel eenvoudig systeem voor numerieke sleutels bestaat uit het toevoegen van een slotcijfer, waarmee de som van alle cijfers een tienvoud is. Uit de coderingstheorie zijn nog veel geraffineerder methoden bekend, maar dat zou ons hier te ver voeren.

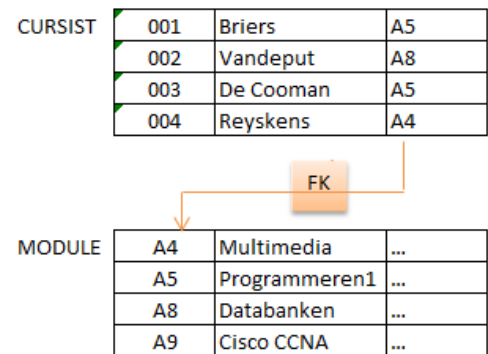
Bv. rekeningnr: restdeling 97 BE 41 0011 4689010

11468999		97
11468989		118237
		10

1.2.4 Referentiële integriteit

Referentiële integriteit is het uitgangspunt dat de **consistentie** tussen de verschillende tabellen binnen een database wordt gewaarborgd.

Dat betekent dat er altijd een primaire sleutel in een tabel bestaat als er in een sleutelveld in een andere tabel naar wordt verwezen.



Er zijn een paar mogelijke maatregelen die referentiële integriteit afdwingen.

- ✓ Bij het maken of wijzigen van een rij controleert het systeem of de foreign keys wel geldige waarden hebben.
- ✓ Daarnaast moet de databaseontwerper een keuze maken voor een delete:
 - Je mag een rij in de tabel pas verwijderen als er geen gerelateerde rijen meer zijn. Je zal als gebruiker dan ook een foutmelding krijgen dat de data nog in gebruik is. Bijvoorbeeld, je wil een klant verwijderen maar er bestaan nog bestellingen voor deze klant. Dit noemt met een *restricted delete*
 - Bij het verwijderen van een rij in een tabel gooit de database automatisch alle gerelateerde gegevens in de gerelateerde tabel weg. Dit het een *cascaded delete*. Bijvoorbeeld, je wil een bestelling verwijderen en automatisch worden ook alle onderlijnen verwijderen.

1.2.5 Redundant

In een goed ontworpen relationele databank bewaren we gegevens op **één plaats** in een tabel. Geven we informatie toch meermaals weer dan spreken we over gegevensovertolligheid of redundantie.

Vb. in tabel Studentenadministratie worden de gegevens van de student opgenomen maar ook in de tabel Resultaat wordt ook de gegevens bewaard. We hebben een inconsistente databank dat tot verwarring en fouten kan leiden. Mocht immers één van de gegevens veranderen dan moet dit in alle tabellen gewijzigd worden. Als dit slechts op één plaats gebeurt dan ontstaat data-inconsistentie waardoor het later niet meer duidelijk is welke van de gegevens juist is.

Procesgegevens zijn ook een vorm van dataredundantie.

2 Relatieve datamodel - Ontwerp

2.1 Normalisatie.

Met behulp van de normalisatiestappen van CODD, kortweg normaliseren genoemd, zijn we in staat om een willekeurige informatiebehoefte te verdelen in een aantal groepen, waarbij iedere groep voldoet aan de volgende definitie:

Alle attributen van de groep zijn functioneel afhankelijk van de volledige sleutel en tussen attributen onderling zijn géén functionele afhankelijkheden aanwezig.

Groepen die aan deze definitie voldoen noemen we genormaliseerde groepen. Zo'n genormaliseerde groep bevat altijd een vast aantal gegevens en kan dus als een tabel worden voorgesteld.

Medewerkers			
Nummer	Naam	Woonplaats	Afdeling
001	Els	Antwerpen	Opleiding
002	Pieter	Leuven	Opleiding
003	Patricia	Tongeren	Analyse
004	Rudi	Genk	Analyse
005	Gert	Hasselt	Programmering

In deze groep MEDEWERKERS vormt het gegeven 'Nummer' de sleutel. Alle attributen: Naam, Woonplaats en Afdeling zijn afhankelijk van het Nummer. Er zijn géén functionele afhankelijkheden aanwezig tussen de attributen onderling. De groep is dus 'genormaliseerd' en kan ook als volgt worden weergegeven:

Medewerker (Nummer, Naam, Woonplaats, Afdeling)

Hierbij worden per groep tussen de haken de gegevens opgesomd, waaruit de groep bestaat en wordt de **sleutel onderstreept**.

Bij een samengestelde sleutel zullen dus meerdere gegevens onderstreept zijn.

Het is de gewoonte om de sleutel het eerst te vermelden, maar noodzakelijk is dat niet.

In tegenstelling tot de voorstellingswijze in tabelvorm kunnen op deze manier niet de afzonderlijke voorkomens elk apart worden aangegeven. Deze laatste voorstellingswijze geeft dus alleen het *type* aan, terwijl met een *tabel* de *afzonderlijke voorkomens (entities)* getoond kunnen worden. Meestal is een voorstelling van de verschillende typen voldoende om aan te geven hoe een gegevensverzameling is opgebouwd. Maar soms is een voorstelling van de afzonderlijke voorkomens (entities) nodig om dit precies te begrijpen. We zullen zoveel mogelijk trachten beide voorstellingswijzen te vermelden.

Het normaliseren vindt plaats in drie stappen.

Vanuit een niet genormaliseerde informatiebehoefte worden de genormaliseerde groepen bepaald. De stappen vormen een recept, dat - indien goed opgevolgd - altijd hetzelfde eindproduct geeft. Het product is niet afhankelijk van de kok die het bereidt.

De drie stappen van het recept luiden:

- ✓ **Verwijder de zich herhalende deelverzamelingen**
- ✓ **Verwijder de attributen die functioneel afhankelijk zijn van slechts een gedeelte van de sleutel.**
- ✓ **Verwijder de attributen die ook functioneel afhankelijk zijn van andere (niet-sleutel) attributen.**

In al deze stappen wordt gesproken over 'verwijderen'.

Dit zou er op kunnen duiden dat gegevens echt verwijderd moeten worden en dus geen deel meer uitmaken van het geheel. Dat is natuurlijk niet de bedoeling, want dan zou met behulp van de genormaliseerde groep(en) nooit meer de oorspronkelijke informatiebehoefte vervaardigd kunnen worden.

Het 'verwijderen' in deze normalisatiestappen houdt in:

- het verwijderen uit de oorspronkelijke groep,
- maar het tegelijkertijd creëren van een nieuwe groep.

Er mag dus niets écht verwijderd worden.

Iedere stap heeft slechts betrekking op één groep.

Als er meerdere groepen zijn dan moet iedere stap voor alle groepen afzonderlijk worden uitgevoerd.

De drie stappen zullen nu eerst zonder veel toelichting getoond worden. Hiervoor is het inmiddels bekende projectoverzicht iets aangepast. Na deze korte behandeling wordt iedere stap afzonderlijk uitvoerig toegelicht in de hierna volgende paragrafen.

PROJECTOVERZICHT

<u>PROJECT</u>	<u>BUDGET</u>	<u>MEDEWERKER</u>	<u>AFDELING</u>	<u>CHEF</u>	<u>UREN</u>
001	VDU 1000	003 Patricia	Analyse	Johan	60
		005 Gert	Programmering	Ron	100
002	CRT 800	004 Rudi	Analyse	Johan	200
		005 Gert	Programmering	Ron	50
003	TTV			

Deze niet genormaliseerde informatiebehoefte kan als volgt worden voorgesteld:

Projectoverzicht:	Projectnummer
	Projectomschrijving
	Budget
	Medewerkersnummer

Naam
Afdeling
Chef
Uren

Stap 1: Verwijder de zich herhalende deelverzamelingen.

De groepen die ná deze eerste stap ontstaan worden wel aangeduid met "**eerste normaalvorm (1NV)**". Deze eerste normaalvorm van het projectoverzicht bestaat uit de volgende eerste normaalvormgroepen:

Project: Projectnummer
Projectomschrijving
Budget

Projectmedewerker: Projectnummer
Medewerkersnummer
Naam
Afdeling
Chef
Uren

De oorspronkelijke niet genormaliseerde groep is nu opgedeeld in twee groepen waarin géén deelverzamelingen meer voorkomen die zich herhalen. Per groep is de **sleutel** onderstreept.

Stap 2: Verwijder de attributen die functioneel afhankelijk zijn van slechts een gedeelte van de sleutel.

Ná deze stap wordt gesproken over de "**tweede normaalvorm (2NV)**".
Deze tweede normaalvorm bestaat uit de volgende groepen:

Project: Projectnummer
Projectomschrijving
Budget

Projectmedewerker: Projectnummer
Medewerkersnummer
Uren

Medewerker: Medewerkersnummer
Naam
Afdeling
Chef

Alle attributen zijn in élk van deze groepen functioneel afhankelijk van de **volledige sleutel**.

Stap 3: Verwijder de attributen die ook functioneel afhankelijk zijn van andere (niet-sleutel) attributen.

Ná deze stap wordt natuurlijk gesproken over de “**derde normaalvorm (3NV)**”.
Deze derde normaalvorm bestaat voor het projectoverzicht uit de volgende groepen.

Project:	<u>Projectnummer</u> Projectomschrijving Budget
Projectmedewerker:	<u>Projectnummer</u> <u>Medewerkersnummer</u> Uren
Medewerker:	<u>Medewerkersnummer</u> Naam Afdeling
Afdeling:	<u>Afdeling</u> Chef

Niet alle groepen worden in iedere stap nader opgedeeld.
Zo is de groep "Project" na de eerste stap niet meer gewijzigd.
Voor deze groep geldt dus dat de eerste normaalvorm gelijk was aan de tweede en de derde normaalvorm.

Uit deze waarneming vallen twee zaken af te leiden.

- Ten eerste, dat het normalisatieproces ergens kan stoppen; er is sprake van een zogenaamde “**laatste normaalvorm (LNV)**”, daar waar het normalisatieproces stopt voor een bepaalde groep.
- Ten tweede, dat een groep in de eerste normaalvorm kan voldoen aan de tweede en de derde normaalvorm; omgekeerd is het zo, dat iedere groep in 3NV altijd voldoet aan 2NV en iedere groep in 2NV altijd voldoet aan 1NV. De drie normaalvormen zijn deelverzamelingen van elkaar.

2.1.1 Eerste Normaalvorm

<u>Stap 1:</u> Verwijder de zich herhalende deelverzamelingen.
--

Dit is een stap, die vaak moeilijker gevonden wordt dan de overige stappen.
Het is ook een stap, die niet eenduidig is, dat wil zeggen dat het resultaat verschillend kan zijn. Het eindresultaat van de drie normalisatiestappen is wél eenduidig, maar de af te leggen weg kennelijk niet.

Om enige steun te bieden bij het vinden van de eerste normaalvorm, is ook hiervoor een recept ontwikkeld:

- 1.1 Inventariseer alle elementaire gegevens.
- 1.2 Verwijder alle procesgegevens.
- 1.3 Doe het volgende totdat er géén nieuwe groepen meer ontstaan:
- 1.4 Geef de sleutel van de groep aan.
- 1.5 Geef de deelverzameling aan die een herhaald aantal keren voorkomt.

- 1.6 Herhaal de sleutelgegevens van de oorspronkelijke groep samen met de gegevens van de zich herhalende deelverzameling als een nieuwe groep.
- 1.7 Verwijder de zich herhalende deelverzameling uit de oorspronkelijke groep.
- 1.8 Eind-doe.

Stap 1.1

Alleen elementaire gegevens mogen geïnterpreteerd worden.

Voor ieder aanwezig samengesteld gegeven moet dus minstens bekend zijn uit welke elementaire gegevens dit is samengesteld. Ieder elementair gegeven behoort een naam te krijgen, waarmee het zich van alle andere gegevens onderscheidt.

Stap 1.2

Procesgegevens moeten wel apart genoteerd worden, maar worden in de normalisatiestappen niet meegenomen mits aan de volgende voorwaarden is voldaan:

- Alle voor de berekening benodigde gegevens zijn aanwezig.
- De voor de berekening benodigde gegevens bevatten op het tijdstip waarop de berekening moet worden uitgevoerd nog de juiste waarden.

In het project voorbeeld zijn géén procesgegevens aanwezig.

Stap 1.3

Zich herhalende deelverzamelingen kunnen genest voorkomen.

D.w.z. dat er een zich herhalende deelverzameling bevindt binnen een andere deelverzameling, die ook een herhaald aantal keren voorkomt.

De eerste normalisatiestap moet zover worden doorgevoerd totdat alle zich herhalende deelverzamelingen zijn opgedeeld.

Stap 1.4

Deze stap is bepalend voor de wijze waarop de eerste normalisatiestap verloopt.

Afhankelijk van de keuze van de sleutel zullen er al of niet (geneste) herhalingen voorkomen. Deze wellicht wat cryptische woorden zullen we aan de hand van het voorbeeld nader toelichten.

Na de inventarisatie van de elementaire gegevens is het projectoverzicht als volgt voorgesteld:

Projectoverzicht:	Projectnummer
	Projectomschrijving
	Budget
	Medewerkersnummer
	Naam
	Afdeling
	Chef
	Uren

Er is blijkbaar gekozen voor het gegeven Projectnummer als sleutel.

Want de eerste normaalvorm wordt gevormd door twee groepen, Project en Medewerker. Men had ook de samengestelde sleutel Projectnummer en Medewerkersnummer kunnen kiezen, maar dan waren er geen zich herhalende deelverzamelingen meer aanwezig geweest.

De ervaring heeft geleerd dat het verstandig is om de meest uitgebreide sleutel te kiezen, daar dan de normalisatiestappen het eenvoudigst zijn.

We zullen nu verder het voorbeeld volgen.

Stap 1.5

Afhankelijk van de sleutelkeuze in stap 1.4 zullen nu nul, één of meer zich herhalende deelverzamelingen aanwezig zijn.

Projectoverzicht:	<u>Projectnummer</u>	
	Projectomschrijving	
	Budget	
	Medewerkersnummer	} komt een herhaald aantal keren voor
	Naam	
	Afdeling	
	Chef	
	Uren	

Stap 1.6

Nu wordt een nieuwe groep gevormd. Deze moet bestaan uit

de zich **herhalende** deelverzameling

+

de **sleutel** van de oorspronkelijke groep.

Deze laatste wordt in de nieuwe groep opgenomen om de koppeling met de oorspronkelijke groep in stand te houden.

Zo krijgen we de volgende twee groepen:

Projectoverzicht:	<u>Projectnummer</u>	
	Projectomschrijving	
	Budget	
	Medewerkersnummer	} herhalend
	Naam	
	Afdeling	
	Chef	
	Uren	

Projectmedewerker:	Projectnummer
	Medewerkersnummer
	Naam
	Afdeling
	Chef
	Uren

Let op:

- Op dit ogenblik is de oorspronkelijke groep nog compleet.
- In de nieuwe groep is nog géén sleutel bepaald.

Stap 1.7

Nu wordt pas de oorspronkelijke groep aangepast en vindt de eigenlijke '**verwijdering**' plaats. Door dit in deze stappen te doen kan men zich ervan overtuigen dat alle gegevens die verwijderd worden ook daadwerkelijk in de nieuwe groep zijn opgenomen. De sleutel, die in de nieuwe groep herhaald is, wordt natuurlijk niet verwijderd.

Project: Projectnummer
 Projectomschrijving
 Budget

Projectmedewerker: **Projectnummer**
 Medewerkersnummer
 Naam
 Afdeling
 Chef
 Uren

Stap 1.8

Deze stap, het einde van de DOE-loop, zorgt ervoor dat we weer bij stap 1.4 beginnen voor de nieuw gevormde groep.

Stap 1.4 (2-de doorgang)

De keuze van de samengestelde sleutel "**Projectnummer & Medewerkersnummer**" is misschien niet direct duidelijk.

Indien alleen Medewerkersnummer als sleutel gekozen was, dan kwamen Uren en Projectnummer nog een herhaald aantal keren voor.

Bij de nu gekozen sleutel zijn er géén zich herhalende deelverzamelingen meer aanwezig. Stap 1 van het normalisatieproces is voltooid en de eerste normaalvorm is bepaald.

Project: Projectnummer
 Projectomschrijving
 Budget

Projectmedewerker: **Projectnummer**
 Medewerkersnummer
 Naam
 Afdeling
 Chef
 Uren

1NV "Projectoverzicht".

2.1.2 Tweede Normaalvorm

Stap 2: Verwijder de attributen die functioneel afhankelijk zijn van slechts een gedeelte van de sleutel.

Om te komen van de eerste tot de tweede normaalvorm, moeten de attributen die slechts van een *gedeelte van de sleutel* afhankelijk zijn in een aparte groep worden opgenomen. Alleen groepen met een samengestelde sleutel komen hiervoor in aanmerking, want alleen bij een samengestelde sleutel kan een attribuut afhankelijk zijn van een gedeelte van de sleutel.

Eigenlijk is de sleutel van groepen, die nog niet in tweede normaalvorm zijn, geen goede sleutel voor alle attributen, want een sleutel mag volgens de definitie géén overbodige gegevens bevatten.

Voor sommige attributen, is dat echter wel het geval.

Het zijn dan ook deze attributen die samen met het deel van de sleutel dat voor hun geen overbodige gegevens bevat, een afzonderlijke groep gaan vormen.

Het recept voor de tweede normaalvorm luidt als volgt:

- 2.1 Geef de attributen aan die niet functioneel afhankelijk zijn van de volledige sleutel.
- 2.2 Vorm een aparte groep voor ieder deel van de sleutel waarvan attributen functioneel afhankelijk zijn.
- 2.3 Neem in iedere groep de attributen met het bijbehorende sleuteldeel op.
- 2.4 Verwijder deze attributen uit de oorspronkelijke groep.

Stap 2.1

Om dit te kunnen doen moet van ieder attribuut de vraag beantwoord worden:

"Welk gegeven of combinatie van gegevens identificeert dit attribuut op een eenduidige wijze?"

In het voorbeeld "Projectoverzicht" komt de *groep Project* niet in aanmerking, daar die groep niet beschikt over een samengestelde sleutel.

Binnen de *groep Projectmedewerker* zijn o.a. de volgende functionele afhankelijkheden te onderkennen:

Attribuut Naam	is functioneel afhankelijk van sleutel	Medewerkersnummer
Attribuut Afdeling	is functioneel afhankelijk van sleutel	Medewerkersnummer
Attribuut Chef	is functioneel afhankelijk van sleutel	Medewerkersnummer
Attribuut Uren	is functioneel afhankelijk van sleutel	Projectn° + Medewerkersn°

De attributen Naam, Afdeling en Chef zijn slechts functioneel afhankelijk van een deel van de sleutel, het Medewerkersnummer.

Stap 2.2

Het kan gebeuren dat een samengestelde sleutel in meerdere delen gesplitst kan worden en dat van ieder deel afzonderlijk attributen functioneel afhankelijk zijn.

Er moeten dan meerdere groepen gevormd worden.

In ons voorbeeld ontstaat slechts één nieuwe groep: Medewerker.

Stap 2.3

De in stap 2.2 geïnterpreteerde groepen moeten nu gevuld worden.

Dit moet zodanig gebeuren, dat iedere nieuwe groep voldoet aan de eisen van de tweede normaalvorm:

- **er mogen dus géén herhalingen aanwezig zijn**
- **en alle attributen moeten functioneel afhankelijk zijn van de volledige sleutel.**

In het voorbeeld ontstaat nu de groep Medewerker

- met als sleutel Medewerkersnummer,
- en als attributen Naam, Afdeling, Chef.

Na deze stap is de oorspronkelijke groep Projectmedewerker opgebouwd uit de volgende groepen:

Projectmedewerker: Projectnummer
Medewerkersnummer
Naam
Afdeling
Chef
Uren

Medewerker: Medewerkersnummer
Naam
Afdeling
Chef

Stap 2.4

Doordat de oorspronkelijke groep nog intact is, kunnen we heel zorgvuldig te werk gaan bij het vervangen van de attributen uit deze groep. De sleutel van de oorspronkelijke groep mag niet aangetast worden.

We hebben nu in totaal de volgende groepen gekregen:

Project: Projectnummer
Projectomschrijving
Budget

Projectmedewerker: Projectnummer
Medewerkersnummer
Uren

Medewerker: Medewerkersnummer
Naam
Afdeling
Chef

2NV "Projectoverzicht".

2.1.3 Derde Normaalvorm

Stap 3: Verwijder attributen die ook functioneel afhankelijk zijn van andere (niet-sleutel) attributen.

Bij de stap naar de derde normaalvorm moeten de attributen

- **die functioneel afhankelijk zijn van de volledige sleutel (2NV),**
- **maar ook nog functioneel afhankelijk zijn van andere attributen,**

in aparte groepen worden opgenomen.

Dit soort afhankelijkheden, tussen attributen onderling, wordt wel aangeduid als *transitieve afhankelijkheden*.

Uiteraard kunnen alleen groepen met meerdere attributen deze soort afhankelijkheden bevatten.

De afhankelijkheid tussen de attributen onderling moet wél een functionele afhankelijkheid zijn om over te gaan tot de vorming van een nieuwe groep.

Afhankelijkheden, waarbij het ene attribuut op de een of andere manier samenhangt met een ander attribuut, zijn hiervoor geen reden. Een voorbeeld van dit soort (losse, toevallige) afhankelijkheden is het feit dat de datum-in-dienst groter moet zijn dan de geboortedatum van een medewerker. Dit is echter géén functionele afhankelijkheid.

De *afhankelijkheden* die *transitief* genoemd worden zijn altijd van het type 'sleutel attribuut' en deze moeten in een aparte groep worden opgenomen.

Het recept voor de derde normaalvorm is het volgende:

- 3.1 Geef de attributen aan die ook functioneel afhankelijk zijn van andere (niet-sleutel) attributen.
- 3.2 Vorm een aparte groep voor ieder attribuut of combinatie van attributen, waar andere attributen functioneel van afhankelijk zijn.
- 3.3 Neem in iedere nieuwe groep de attributen met hun bijbehorende sleutel op.
- 3.4 Verwijder de attributen van de nieuwe groep(en) uit de oorspronkelijke groep.

Stap 3.1

Om dit te kunnen doen moet van ieder attribuut worden vastgesteld of er nog één of meer andere attributen zijn die het beschouwde attribuut uniek identificeren.

In het voorbeeld "Projectoverzicht" is in de groep Project geen functionele afhankelijkheid aanwezig tussen de attributen Projectomschrijving en Budget.

De groep Projectmedewerker bevat slechts één attribuut en komt dus niet in aanmerking. Maar *in de groep Medewerker is het attribuut Chef ook functioneel afhankelijk van het attribuut Afdeling*

Attribuut Chef is functioneel afhankelijk van attribuut Afdeling.

De attributen Naam en Afdeling zijn onderling onafhankelijk.

In feite is hier vastgesteld dat iedere medewerker één chef heeft en één afdeling.

Maar tevens dat iedere afdeling ook één chef heeft en dat dus de chef van de medewerker de chef is van de afdeling van die medewerker.

Stap 3.2

Er kunnen meerdere transitieve afhankelijkheden aanwezig zijn.

Alle unieke attributen of attribuutcombinaties, die als sleutel fungeren in transitieve afhankelijkheden, geven aanleiding tot de vorming van verschillende groepen.

In het voorbeeld ontstaat slechts één nieuwe groep: Afdeling.

Stap 3.3

De in stap 3.2 geïnventariseerde groepen moeten nu van sleutels en attributen worden voorzien.

De aldus nieuw gevormde groepen moeten wél aan de **definitie van de derde normaalvorm** voldoen:

- **zij mogen dus géén herhalingen bevatten**
- **én alle attributen moeten functioneel afhankelijk zijn van de volledige sleutel**
- **én onderling géén functionele afhankelijkheden bevatten.**

De nieuw gevormde groep Afdeling bestaat uit twee gegevens, "Afdeling" en "Chef".

Wat is nu de sleutel: Afdeling of Chef?

Een sleutel moet eenduidig identificeren en dus unieke waarden bevatten.

Er kunnen zich verschillende situaties voordoen. De gebruiker zal moeten aangeven welke situatie voor hem geldig is.

Situatie a:

Iedere afdeling heeft één chef en iedere chef is chef van één afdeling.

Afdeling	Chef
A	1
B	2
C	3

Door het tekenen van een tabel met afzonderlijke voorkomens is direct te zien dat er sprake is van twee kandidaat-sleutels.

Welke er gekozen wordt is niet belangrijk.

Situatie b:

Iedere afdeling heeft één chef, maar een chef kan van meerdere afdelingen chef zijn.

Afdeling	Chef
A	1
B	2
C	1

Het enige gegeven dat nu een unieke waarde bevat is Afdeling.

Dit is dus de sleutel in deze situatie.

Situatie c:

Iedere afdeling heeft meerdere chefs, maar iedere chef is slechts chef van één afdeling.

Afdeling	Chef
A	1
A	2
B	3
B	4

Nu is Chef de sleutel.

Situatie d:

Iedere afdeling heeft meerdere chefs en iedere chef kán chef zijn van meerdere afdelingen.

Afdeling	Chef
A	1
A	2
B	3
B	1

Geen van beide gegevens bevat nu nog unieke waarden.

De combinatie van beide gegevens is wél uniek.

Beide gegevens vormen een samengestelde sleutel.

De gebruiker heeft aangegeven dat in zijn omgeving situatie a geldt, maar dat situatie b niet uitgesloten wordt geacht. Om deze reden is gegeven Afdeling tot sleutel gekozen.

Na deze stap is de oorspronkelijk 2NV groep Medewerker nu opgebouwd uit de volgende groepen:

Medewerker: Medewerkersnummer
 Naam
 Afdeling
 Chef

Afdeling: Afdeling
 Chef

Stap 3.4

Uit de nu nog intact zijnde oorspronkelijke groep moeten de attributen van de nieuw ontstane groep(en) verwijderd worden.

+

De sleutel van de nieuwe groep blijft als attribuut in de oorspronkelijke groep gehandhaafd.

Dit geeft de volgende groepen, die elk op zich in de 3-de normaalvorm zijn:

Project:	<u>Projectnummer</u> Projectomschrijving Budget
Projectmedewerker:	<u>Projectnummer</u> <u>Medewerkersnummer</u> Uren
Medewerker:	<u>Medewerkersnummer</u> Naam Afdeling
Afdeling:	<u>Afdeling</u> Chef

3NV "Projectoverzicht".

Alle nieuw gevormde groepen hierboven voldoen immers aan de **definitie van de derde normaalvorm** :

- | |
|--|
| <ul style="list-style-type: none">➤ zij mogen géén herhalingen bevatten➤ alle attributen in elke groep op zich moeten functioneel afhankelijk zijn van de volledige sleutel van die groep➤ alle attributen in elke groep op zich mogen onderling géén functionele afhankelijkheden bevatten. |
|--|

2.1.4 Verdere Normaalvormen

Naast de drie te bespreken normaalvormen worden ook nog de "**Boyce Codd normal form (BCNF)**", de vierde **4NV** en de vijfde **5NV** onderscheiden.

Het praktische nut ervan is zeer gering, daar deze slechts sporadisch voorkomen en meestal vanzelf gevonden worden, indien men naast de regels ook het gezonde verstand blijft gebruiken.

Voor de volledigheid volgen hier zonder nadere verklaring de definities:

- BCNF:** een groep is in de Boyce Codd Normal Form wanneer iedere determinant een kandidaat-sleutel is
(een determinant is een gegeven met een identificerend karakter).
- 4NV:** een groep is in de vierde normaalvorm wanneer bij het voorkomen van een meerwaardige afhankelijkheid van A alle attributen ook functioneel afhankelijk zijn van A.
- 5NV:** een groep is in de vijfde normaalvorm als iedere samenvoegings afhankelijkheid geïmpliceerd wordt door de kandidaat-sleutels.

2.1.5 Oefeningen

2.1.5.1 Normaliseer onderstaande offerte.

Offerte					
Datum: 20-3-2017 Ordernr: 123456 Vervaldatum: 4/20/2018					
AAN Pascal Degrootte Vaartstraat 12 3500 Hasselt 011 77 51 00 Klantnr: 7546					
Aantal	Artikelnummer	Beschrijving	Prijs per eenheid	Korting	Totaal
1	987456	Televisie 71 cm	550,00	50,00	550,00
100	456123	Nylon Plug 5mm	3,36		336,00
10		Verdeelddoos Inbouw -Wit	1,99		19,90
Totale korting				50,00	905,90
				Subtotaal	855,90
				Btw	179,74
				Totaal	1035,64

2.1.5.2 Normaliseer onderstaande gegevens.

Een multinational heeft 1 of meerdere opslagplaatsen (in verschillende landen) met producten. Elke opslagplaats (depot) heeft een uniek nummer binnen de multinational. Eenmaal per jaar wil men per product de voorraden in elke vestiging kennen.

Overzicht:

Productnr	Productnaam	Productomschrijving	Depotnr	Landcode	Landomschrijving	Voorraad
1830	Canon EOS 400D	- Verwisselbare lens: Ja	1	65	België	2700
		- Focus handmatig/automatisch	6	66	Nederland	3000
		-...	9	77	Frankrijk	5000
1900	Nikon D40X	- Bestandsformaten: dpof, jpeg, raw	6	66	Nederland	2450
		- rode ogen reductie: Ja	12	65	België	3004
		-...	24	34	Spanje	2300

2.1.5.3 Normaliseer onderstaande lidkaart.

In de plaatselijke basketbalclub wensen ze de gegevens van de leden gestructureerd bij te houden. Een speler kan maar tot 1 ploeg behoren en een ploeg wordt aangeduid door een bepaalde minimumleeftijd en maximumleeftijd. Een speler neemt met zijn ploeg deel aan de competitie en speelt hierin verschillende wedstrijden.

Lidnummer : 123456789

Mark Branders
Roosterstraat 23
3500 Hasselt

Geslacht : M

Telefoon : 012/345678

GSM : 0497/935684

Geboortedatum : 11/11/1979

Leeftijd : 39

E-mail : mark.branders@hotmail.com

Ploeg : Senioren

Leeftijdscategorie : 18 - 99

Bedrag lidgeld : €150

Lidgeld betaald : ja

Datum betaling : 23/01/2018

Wedstrijdnr	Datum	Beginuur	Tegenspeler	Categorie	Uit/thuis	Aantal gescoorde punten	Score ploeg	Score tegenploeg
310203	04/09	09u00	Sint-Truiden	Senioren	Uit	13	104	11
307218	18/10	09u30	Peer	Senioren	Thuis	2	77	37
306104	18/09	12u30	Lummen	Senioren	Uit	29	51	81

2.1.5.4 Normaliseer tot de derde normaalvorm.

In het ziekenhuis wordt per patiënt de nodige gegevens bijgehouden. Bij een opname wordt er bijgehouden op welke kamer deze patiënt gelegen is. De patiënt kan tijdens zijn opname op verschillende kamers verblijven. Voorts wordt er ook bijgehouden welke dokters deze patiënt behandeld hebben. Het bedrag dat de dokter aanrekent aan zijn patiënt kan variëren van patiënt tot patiënt.

Patiëntnr : 123456789

Geert Wouters
Halveweg 12
3500 Hasselt

Geslacht : M

Telefoon : 012/345678

Geboortedatum : 11/11/1979

Leeftijd : 38

Kamer	Omschrijving	Aantal dagen	Prijs/Dag
423	4-persoonskamer	14	€25,49
401	Intensieve zorgen	2	€49

Totaal : €454,86

Dokter	Naam	Adres	Bedrag
987654	Vanderveken Jan	Roosterstraat 15 3500 Hasselt	€154,25
564578	Dedonder Greet	Dwarsweg 95 3500 Hasselt	€429,99

Totaal : €584,24

Algemeen totaal : €1039,10

2.1.5.5 Normaliseer tot de derde normaalvorm.

De Kinopolis Group beschikt over een groot assortiment aan films die gedraaid worden in al hun complexen (Brussel, Antwerpen, Hasselt,...). Elk complex heeft verschillende zalen waar films gespeeld worden. Een film kan in meerdere zalen worden gespeeld, maar in 1 zaal wordt slechts 1 film gespeeld. Het filmnummer is uniek en wordt door alle filmcomplexen op dezelfde manier gebruikt.

Overzicht:

Complex nr.	Complex naam	Complex adres	Film nr	Film naam	Film prijs	KT	Zaalnr.	Zaal naam	Aantal plaatsen
1	Kinopolis Hasselt	Via media 1	1	Revenant	10,35	Neen	1	Zaal 1	450
			2	Star Wars	10,35	Ja	2	Zaal 2	500
			1	Revenant	10,35	Neen	3	Zaal 3	340
2	Metropolis Antwerpen	Groenendaallaan 394	4	De Premier	10,35	Ja	1	Zaal 1	750
			2	Star Wars	10,35	Ja	3	Zaal 3	490
3

2.1.5.6 Normaliseer tot de derde normaalvorm.

Men wenst een overzicht van de reservaties per zaal per evenement. Tevens wil men verder inzoomen op een evenement en kijken welke toeschouwers tickets gereserveerd hebben.

Rapport: Beheer van concertzalen

Zaal-code	Zaal-omschrijving	Aantal plaatsen	Optreden	Datum	Reservaties	Vrije plaatsen
S	Sportpaleis	15000	Editors	29/apr	14000	1000
				30/apr	14500	500
			Nick Cave and the Bad Seeds	24/nov	2000	13000
				25/nov	1200	13800
LA	Lotto Arena	8000	Dua Lipa	17/feb	7000	1000
				18/feb	7900	100
				19/feb	7950	50
			Billie Eilish	25/feb	6000	2000

Er moet ook een detail per lijn opgevraagd kunnen worden om over de reservatiegegevens een duidelijker beeld te verkrijgen.

Bijvoorbeeld voor het concert van *Oscar And The Wolf* in het Sportpaleis op 30 april:

Klatcode	Klantnaam	Aantal plaatsen
78652	Tuba Uzun	4
789412	Lynn Rabaut	3
419637	Rani Rutten	5
....		

Opmerkingen

- Een zaal heeft een vast aantal plaatsen die bezet kunnen worden.
- We gaan ervan uit dat we enkel de aantallen moeten bijhouden. Wie waar moet gaan zitten (genummerde plaatsen) houden we hier niet bij en behoort dus niet tot de scope van deze opdracht.
- Op een bepaalde datum kunnen er wel twee evenementen zijn, maar telkens in een andere zaal. In een zaal kan er maar 1 optreden op een bepaalde datum zijn.

2.1.5.7 Case technische dienst centrale verwarming.

De Technische dienst van een installateur van centrale verwarming krijgt telefonische oproepen voor herstellingen en onderhoudsbeurten. Na de technische interventie vraagt de technicus aan de klant een werkbon te ondertekenen.

Op het einde van de maand worden, uitgaande van de gegevens vermeld op de werkbonnen en de klant- en artikelkaarten, facturen gemaakt en verstuurd naar de klanten. De klant betaalt de factuur via zijn financiële instelling.

- **Werkbon**

[illegible]

Gegevensanalyse van de Werkbon

Gegevens	V (verplicht) F (facultatief)	V (vaste lengte) N (variabele lengte)	B (Berekening)	E of M (één of meerdere malen per document)	S (sleutel)	S (stockeren op extern geheugen)	K (compact stockeren)
Klantcode (1)	V	V 4		E		S	
Klantnaam	V	N 25		E		S	
Adres	V	N 30		E		S	
Plaats (Postcode+Gem)	V	N 30		E		S	
Tel.- of Gsmnr	F	V 12		E		S	
Firmanaam & adres	V	N 85		E		-	
Nummer werkbon	V	V 5 (2)		E	S	S	
Datum	V	V 6		E		S	
Herstelling	F	V 11		E		S	K (3)
Onderhoud	F	V 9		E		S	K (3)
Garantie	F	V 8		E		S	K (3)
Montage	F	V 7		E		S	K (3)
Omschrijving werken	V	N 90		E		S	
Aantal uren	V	V 2 (2)		E		S	
Materialen:							
Aantal uren	V	V 2 (2)		M		S	
Artikelnr	V	V 6		M		S	
Omschrijving	V	N 20		M		S	
Opmerkingen	F	N 60		E		- (4)	
Naam technicus	V	N 20		E		S	K (5)
Handtekening technicus	V	N		E		-	
Handtekening klant	V	N		E		-	

Bemerkingen:

Indien een gegeven een variabele lengte heeft, wordt steeds het maximum aantal karakters vermeld.

(1) Klantcode:

indien de technicus deze code niet kent dan wordt deze code door de Technische dienst aangebracht.

(2) Nummer werkbon, Aantal uren, Aantal:

indien plaats voorzien wordt voor spaties of niet-beduidende nullen kan men een vaste lengte voorzien.

(3) Herstelling, Onderhoud, Garantie, Montage:


- een keuze moet gemaakt worden uit deze types van technische interventie.
- compact stockeren is hier mogelijk door bv. één karakter te voorzien:
H (Herstelling), **O** (Onderhoud), **G** (Garantie), **M** (Montage).

(4) Opmerkingen:

60 karakters; daar dit gegeven niet op de factuur voorkomt is men dan ook niet verplicht dit gegeven te stockeren op extern geheugen.

(5) Naam technicus: i.p.v. de naam van de technicus kan men een code technicus stockeren met b.v. 2 karakters.

- Factuur



Adres

FACTUUR

Klantcode
 Naam
 Adres
 Plaats
 Btw.nr.

Datum
 Factuurnr.
 Betalen voor:

15-09
1111
15-10

Werkbon nr.
Datum uitvoering
Verplaatsingskosten
Werkloon

Omschrijving werken:

Materialen

Aantal	Artikelnr	Omschrijving	Eenheidsprijs	Bedrag
1	Artikelnummer 1			
5	Artikelnummer 2			
2	Artikelnummer 3			

Btw %: --

Totaal
 Btw
 Te betalen

Gegevensanalyse van de Factuur

Gegevens	V (verplicht) F (facultatief)	V (vaste lengte) N (variabele lengte)	B (Berekening)	E of M (één of meerdere malen per document)	S (sleutel)	S (stockeren op extern geheugen)	K (compact stockeren)
Firmanaam & adres	V	N 85		E		-	
Klantnaam	V	N 25		E		S	
Adres	V	N 30		E		S	
Plaats (Postcode+Gem)	V	N 30		E		S	
Factuurnr	V	V 4 (1)	B (2)	E	S	S	
Klantcode	V	V 4		E		S	
Factuurdatum	V	V 6		E		S	
Betalen voor	V	V 6	B (3)	E		- (3)	
Btw nr klant	F	V 9		E		S	
Omschrijving werken	V	N 90		E		- (4)	
Datum uitvoering	V	V 6		E		S	
Werkbonnummer	V	V 5 (1)		E		S	
Verplaatsingskosten	V	N 4 (1)		E		S	
Werkloon	V	V 4 (1)	B (5)	E		S	
Materialen:							
Aantal	V	V 2 (1)		M		S	
Artikelnr	V	V 6		M		S	
Omschrijving	V	N 20		M		S	
Eenheidsprijs	V	V 5 (1)		M		S	
Bedrag	V	V 6 (1)	B (6)	M		- (6)	
Totaal	V	V 6 (1)	B	E		S (7)	
Btw %	V	V 2		E		S	
Btw-bedrag	V	V 5 (1)	B	E		S (7)	
Te betalen	V	V 6 (1)	B	E		S (7)	

Bemerkingen:

- (1) Een vaste lengte kán worden genomen indien men plaats voorziet voor spaties of niet-beduidende nullen.
- (2) Het Factuurnummer moet een doorlopend nummer zijn (+1).
- (2) Betalen voor:
Indien de vervaldag kan afgeleid worden van de factuurdatum (bv. *factuurdatum + 30 dagen*) dan moet de vervaldag niet gestockeerd worden.
- (3) Omschrijving van de werken: Dit gegeven dat voorkwam op de werkbon kan afgedrukt worden op de factuur.
Het verder stockeren van dit gegeven op extern geheugen is, gezien het geringe belang ervan en het groot aantal posities, niet interessant.
- (5) Werkloon = aantal uren (werkbon) * uurloon.
Dit uurloon kan een constante zijn of afhankelijk zijn van het soort werk.
- (6) Bedrag = aantal * eenheidsprijs
- (7) Alhoewel de gegevens Totaal, Btw-bedrag en Te Betalen het resultaat zijn van een berekening kan het toch verantwoord zijn die gegevens te stockeren, gezien het beperkt aantal posities die nodig zijn en de toch eerder uitgebreide berekening in het geval van meerdere artikelen.
Of gedeeltelijk te stockeren, bv. 'Te Betalen', zodat dit bedrag onmiddellijk kan gehanteerd

worden bij het identificeren van betalingen, waarop de klant vergeten heeft zijn factuurn° of klantrn° te vermelden.

- **Rekeninguittreksel**

Rekeninguittreksel			
Rekeningnummer: Naam: Adres: Plaats:	Afschriftnr.		
Vorig saldo op: €			
Rekeningnr.: Mededeling Rekeningnr.: Mededeling	Bedrag: € Bedrag: €		
Nieuw saldo op: €			

Gegevensanalyse van het Rekeninguittreksel

Gegevens	V (verplicht) F (facultatief)	V (vaste lengte) N (variabele lengte)	B (Berekening)	E of M (één of meerdere malen per document)	S (sleutel)	S (stockeren op extern geheugen)	K (compact stockeren)
Rekeningnummer	V	V 12		E	S (2)	(3)	
Afschriftnummer	V	V 4 (1)		E	S (2)	(3)	
Naam	V	N	Firma gegevens				-
Adres	V	N					-
Woonplaats	V	N					-
Vorig saldo : datum	V	V 6		E		(3)	
Vorig saldo : bedrag	V	V 7 (1)		E		(3)	
Rekeningnummer	V	V 12		M		(3)	
Bedrag	V	V 6 (1)		M		(3)	
Mededeling	V	N 40		M		(3)	
Nieuw saldo : datum	V	V 6		E		(3)	
Nieuw saldo : bedrag	V	V 7 (1)		E		(3)	

Bemerkingen:

- **Klantenkaart**

Gegevensanalyse van de Klantenkaart

Gegevens	V (verplicht) F (facultatief)	V (vaste lengte) N (variabele lengte)	B (Berekening)	E of M (één of meerdere malen per document)	S (sleutel)	S (stockeren op extern geheugen)	K (compact stockeren)
Klantcode (1)	V	V 4		E	S	S	
Type klant (2)	V	V 1		E		S	K
Klantnaam	V	N 25		E		S	
Adres	V	N 30		E		S	
Plaats	V	N 30		E		S	
Telefoon - Gsm	F	V 12		E		S	
Btwnr	F	V 9		E		S	
Facturen:							
Nummer factuur	V	V 3 (3)		M		S	
Datum factuur	V	V 6		M		S	
Exclusief Btw	V	V 6 (3)		M		S	
Btw-bedrag	F	V 5 (3)		M		S	
Totaal	V	V 6 (3)	B	M		S	
Betalingen							
Datum	V	6		M		S	
Nummer factuur	V	V 3 (3)		M		S	
Wijze van ontvangst	V	V 1 (4)		M		S	K
Afschriftnr	V	V 4 (3)		M		S	
Bedrag	V	V 6 (3)		M		S	
Debet (5)	V	V 6 (3)	B	M		S	
Credit (5)	V	V 6 (3)	B	M		S	
Saldo (5)	V	V 6 (3)	B	M		S	
Code Dubieuze klant	F	V 1	B (6)	E		S	K (6)
Code Installatie	V	V 1 (7)	B (7)	E		S	K (7)

Bemerkingen:

(1) Klantcode:

De doelstelling van een code is het individualiseren van een bedrijf/persoon/voorwerp en KAN ook gebruikt worden om de karakteristieken van dit bedrijf/persoon/voorwerp te definiëren. De code wordt langer naargelang die meer beschrijvend wordt.

Voor een klantcode kan men opteren voor:

- een sequentiële code: een doorlopend nummer 1, 2, 3, ...
- een alfanumerieke code waarbij de eerste positie de éérste letter is van de klantnaam; de volgende posities zijn dan terug een sequentieel nummer.
- wil men een zuivere numerieke "alfabetische" code dan kan men de eerste positie van de alfanumerieke code vervangen door twee cijfers:
01 i.p.v. A, 02 i.p.v. B,... -->26 i.p.v. Z
- de eerste vier posities van de klantcode kunnen bestaan uit de postcode, voor de vijfde en een zesde positie kan men dan terug een sequentieel nummer nemen.
- ...

(2) Type klant:

een cijfer of een letter kan hiervoor worden genomen, bv.:

- P (of 1): privépersoon
- B (of 2): bedrijf (eventueel nog opsplitsen per soort bedrijf)
- S (of 3): school
- F (of 4): financiële instelling
- ...

Een andere mogelijkheid kan erin bestaan het gegeven 'type klant' op te nemen in de klantcode.

(3) Een vaste lengte kan worden genomen indien men plaats voorziet voor spaties of niet-beduidende nullen.

(4) Wijze van ontvangst:

Indien het bedrijf beschikt over rekeningen bij meerdere financiële instellingen kan via een code van bv. één positie genoteerd worden via welke instelling de betaling gebeurd is.

(5) De Debet-, Credit- en Saldogegevens kunnen meerdere malen voorkomen op de klantenkaart; dit is ook het geval voor de factuur- en de betalingsgegevens.

Nochtans is er een fundamenteel verschil:

factuur- en betalingsgegevens zijn VASTE gegevens: factuurnummer, datum,... blijven onveranderd.

Echter zijn de debet-, credit- en saldogegevens CUMULATIEVE gegevens, ze veranderen bij iedere factuur of betaling.

Belangrijk is hier ook dat het saldobedrag moet gelijk zijn aan het verschil tussen de som van de factuurbedragen en de som van de betalingsbedragen.

(6) Code Dubieuze klant:

via een één positie code, bv. **D**, kan men bijhouden welke klanten een belangrijke betalingsachterstand hebben.

Deze methode houdt nochtans wel risico's in:

indien het betaalgedrag van de klant verandert kan men vergeten deze code aan te passen.

Dit kan dan mogelijk invloed hebben op het toegestane kortingspercentage.

Een betere (maar complexere) methode bestaat erin dat de bepaling van een Dubieuze klant niet vervat zit in een STATISCH manueel in te vullen code, maar wél DYNAMISCH bepaald wordt door een algoritme dat uitgaat van FEITEN:

- hoeveel facturen zijn er nog niet betaald?
- wat is het totaalbedrag ervan?
- hoe belangrijk is de betalingsachterstand?

(7) Code Installatie:

Het toe te passen Btw-tarief kan afhankelijk zijn van het aantal jaren dat de centrale verwarmingsinstallatie in gebruik is.

Dit kan gestockeerd worden op twee manieren:

- het voorzien van een STATISCHE code:

Ingeval de installatie bv. meer dan 20 jaar oud is, kan men de letter **J** als code nemen.

Het probleem met een STATISCHE code is terug dat men, indien nodig, kan vergeten de code aan te passen.

- de oplossing ligt hier voor de hand:

i.p.v. een statische code te voorzien stockeert men de installatiedatum, zodat het aantal jaren DYNAMISCH kan bepaald worden.

- Artikelkaart

Artikelkaart

Artikelcode:

Type artikel:

Omschrijving:

Verkoopprijs 1:

Verkoopprijs 2:

Minimum stock:

Leverancier 1:

Leverancier 2:

....

Maateenheid:

Gewicht/Afmetingen:

Locatie in magazijn:

Datum:

Levertermijn:

Levertermijn:

Kostprijs:

Kostprijs:

Stockbeweging

Datum IN / UIT	Aantal IN	Lev. Code	Nr. bestelling	Aantal UIT	Tech. Code	Stock
9/09/2015	100	BRP	0014	20	SLR	80
9/09/2015			0023	5	NER	75

Gegevensanalyse van de Artikelkaart

Gegevens	V (verplicht) F (facultatief)	V (vaste lengte) N (variabele lengte)	B (Berekening)	E of M (één of meerdere malen per document)	S (sleutel)	S (stockeren op extern geheugen)	K (compact stockeren)
Artikelcode (1)	V	V 6		E	S	S	
Type artikel (2)	V	V 3		E		S	K
Omschrijving	V	N 20		E		S	
Maateenheid	V	V 2		E		S	K
Gewicht/Afmetingen	F	N 10		E		S	
Locatie (4)	V	N 6		E		S	K
Verkoopprijs 1	V	V 6 (3)		E		S	
Verkoopprijs 2 (5)	F	V 6 (3)		E		S	
Datum (5)	F	V 6		E		S	
Min. Stock	F	V 5 (3)		E		S	
Bestelhoeveelheid	F	V 5 (3)		E		S	
Leveranciers:							
Code Leveranciers	F	V 6		M		S	
Levertermijn	F	V 3 (3)		M		S	
Kostprijs	F	V 6 (3)		M		S	
Stockbewegingen:							
Datum beweging (IN/UIT)	V	V 6		M		S	
Aantal IN	V	V 5 (3)		M		S	
Leverancierscode	V	V 6		M		S	
Nr. Bestelling	V	V 4 (3)		M		S	
Aantal UIT	V	V 4 (3)		M		S	
Code technicus	V	V 2		M		S	
Stock (6)	V	V 5 (3)	B	M		S	

Bemerkingen:
Zoals reeds werd vermeld worden de bestellingen en de opvolging van de bestellingen hier niet behandeld.

- (1) Artikelcode:
 voor de artikelcode kan men opteren voor:
 - een sequentiële code: een doorlopend nummer 1,2,3,...
 - ofwel kan men een bestaand codeersysteem voor werkstukken gebruiken waarbij deze codes de vorm, het materiaal, de functies en/of de bewerkingen van een werkstuk beschrijven,
 bv. het DIN-codesysteem, het OPITZ-codesysteem, het MICLASS-codesysteem,... Het betreft hier dan *beschrijvende codes*; het aantal posities varieert naargelang het codesysteem.
- (2) Type artikel:
 één of meerdere cijfers/letters kunnen hiervoor worden genomen.
 Een andere mogelijkheid kan erin bestaan dit gegeven op te nemen in de artikelcode.
- (3) Een vaste lengte *kán* worden genomen indien men plaats voorziet voor spaties of niet-beduidende nullen.
- (4) Maateenheid en Locatie in het magazijn:
 hiervoor kan men terug een codesysteem ontwerpen.
- (5) De mogelijkheid van twee Verkoopprijzen is hier voorzien.
 Vanaf de versnelde datum is Verkoopprijs 2 van toepassing.

(6) Stock:

dit gegeven komt meerdere malen voor op de artikelkaart.

Het is een CUMULATIEF gegeven: bij iedere stockbeweging in plus of min verandert de stockhoeveelheid.

Optimalisatie

Indien er na normalisatie van de gegevens van de verschillende documenten, entiteiten met hetzelfde sleutelattribuut voorkomen, dan worden die entiteiten samengevoegd tot één entiteit.

- Vergelijkt men de entiteiten met als sleutel 'Werkbonnummer' en 'Factuurnummer';
daar in de opgave vermeld werd dat er een één-tot-één relatie bestaat tussen 'Werkbon' en 'Factuur', kunnen deze twee entiteiten herleid worden tot één entiteit.
- Vergelijkt men de entiteit met sleutel 'Factuurnummer-Artikelnummer' met de entiteit met sleutel 'Werkbonnummer-Artikelnummer'
én
daar één factuur overeenstemt met één werkbon,
kunnen deze entiteiten terug herleid worden tot één entiteit.

Ook kan men stellen dat de 'Werkbons' de mutaties zijn die gebruikt worden om de 'Facturen' op te stellen.

(Vergelijk dit met de gegevens van de rekeninguittreksels die ook niet verder dienen gestockeerd te worden).

De werkbongegevens vormen TIJDELIJKE, MUTATIE-entiteiten.

- De entiteit afgeleid van de 'Klantenkaart' met sleutel 'Factuurnummer' kan samengevoegd worden met de entiteit, afgeleid van de 'Factuur' met sleutel 'Factuurnummer'.
- De entiteiten met sleutel 'Klantcode' kunnen samengevoegd worden tot één entiteit.
- De entiteiten met sleutel 'Artikelcode' kunnen samengevoegd worden tot één entiteit.

2.2 Entity-relationship diagram.

Het entity-relationship model of entity-relationship diagram (ER-diagram) is een visuele weergave van de tabellen, relaties en regels. ER-diagrammen worden daarom veelal gebruikt om het ontwerp van een database te schetsen.

Er zijn twee redenen om een database-diagram te maken. U ontwerpt een nieuw schema of u moet uw bestaande structuur documenteren.

Als u een bestaande database hebt die u moet documenteren, maakt u een databasediagram met gegevens rechtstreeks uit uw database. U kunt uw databasestructuur exporteren als een CSV-bestand en vervolgens laat een programma de ERD automatisch genereren.

Een entiteit is een object dat bestaat en onderscheidbaar is van andere objecten. Bijvoorbeeld Anna Peeters met studentnummer 89204 is een entiteit omdat het op een unieke manier een specifieke persoon in het universum identificeert. Een entiteit kan concreet zijn, zoals een persoon of een boek, of abstract zoals een vakantiedag of een concept.

Een entiteitverzameling is een verzameling van entiteiten van hetzelfde type. De verzameling van alle personen die aan een bepaald instituut studeren, kan gedefinieerd worden als de entiteitverzameling student.

Mogelijke attributen voor de student entiteit zijn snaam, studnr, straat en woonplaats. Voor elk attribuut bestaat er een verzameling van toegelaten waarden, het domein van dat attribuut.

Voorbeeld van een entiteitverzameling met attributen:

- STUDENT met attributen snaam, studnr, straat en woonplaats;
- BIOGRAFIE met attributen geboorteplaats en geboortedatum;
- DOCENT met attributen dnaam, docnr en acadgraad;
- VAK met attributen vnaam, uren en vaknr;
- RESULTAAT met attributen percentage en vermelding.

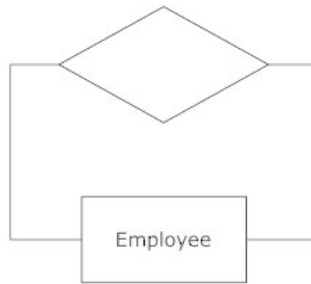
2.2.1 ER diagram symbolen

Een ER-diagram is een manier om te visualiseren hoe de informatie die een systeem produceert, verband houdt. Er zijn 5 hoofdonderdelen van een ERD:

- **Entiteiten** worden weergegeven door rechthoeken. Een entiteit is een object of concept waarover u informatie wilt opslaan. Een zwakke entiteit is een entiteit die moet worden gedefinieerd door een externe sleutelrelatie met een andere entiteit, omdat deze niet uniek kan worden geïdentificeerd door haar eigen attributen alleen.
- **Acties**, die worden weergegeven door ruitvormen, laten zien hoe twee entiteiten informatie delen in de database. In sommige gevallen kunnen entiteiten zelf gekoppeld zijn. Medewerkers kunnen bijvoorbeeld toezicht houden op



andere medewerkers.



- **Attributen** , die worden weergegeven door ovalen. Een belangrijk kenmerk is het unieke, onderscheidende kenmerk van de entiteit. Het rijksregisternummer van een werknemer kan bijvoorbeeld het belangrijkste kenmerk van de werknemer zijn.



- **Verbindingslijnen** , ononderbroken lijnen die attributen verbinden om de relaties van entiteiten in het diagram te tonen.
- **Kardinaliteit** geeft aan hoeveel instanties van een entiteit betrekking hebben op een instantie van een andere entiteit. Ordinaliteit is ook nauw verbonden met kardinaliteit. Terwijl kardinaliteit het voorkomen van een relatie specificeert, beschrijft ordinaliteit de relatie als verplicht of optioneel. Met andere woorden, kardinaliteit specificeert het maximale aantal relaties en ordinaliteit specificeert het absolute minimum aantal relaties.

Er zijn veel notatiestijlen die kardinaliteit uitdrukken. Wij hanteren de Chen stijl.

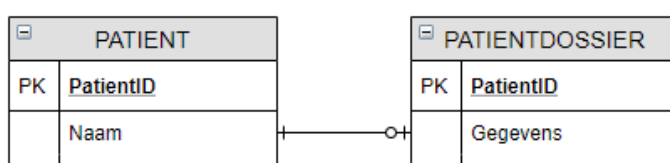
2.2.2 Kardinaliteit in een relatie

De kardinaliteit in een relatie zegt iets over de hoeveelheid van de ene tabel ten opzichte van de hoeveelheid van de andere tabel. Er zijn 3 vormen van relaties:

- 1-op-1
- 1-op-veel
- veel-op-veel

1-op-1 relatie

Dit type relatie komt niet zo vaak voor maar voor de volledigheid hoort het er toch bij. Je moet je afvragen of je de beide tabellen in de 1-op-1 relatie niet moeten worden samengevoegd.

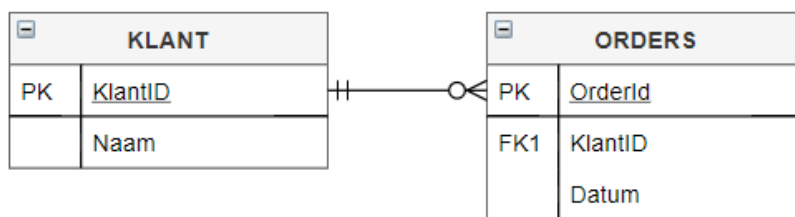


Kenmerkend voor de 1-op-1 relatie is de primary key van beide tabellen. Deze zijn namelijk gelijk aan elkaar. In dit voorbeeld is het PatientID in PATIENTDOSSIER een foreign key die wijst naar PatientID in de tabel PATIENT.

De relatie gelezen van rechts naar links: het dossier van een patient hoort bij één patient.
De relatie gelezen van links naar rechts: Een patient heeft nul of één patientdossier.

1-op-veel relatie

Dit type relatie komt samen met de veel-op-veel relatie het meest voor.

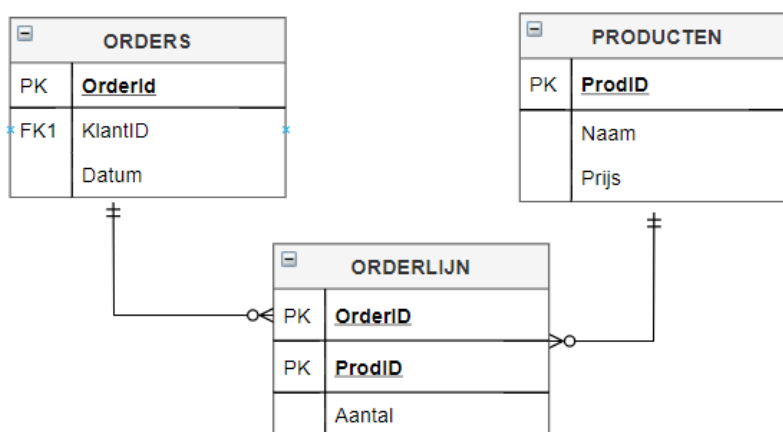


Aan de veel-kant, in dit geval bij de tabel ORDERS, wordt een foreign key KlantID toegevoegd die wijst naar KlantID in de tabel KLANT.

De relatie gelezen van links naar rechts: een klant heeft nul of meerdere orders.
De relatie gelezen van rechts naar links: Een order wordt geplaatst door één klant.

Veel-op-veel relatie

Met twee tabellen is het onmogelijk om de veel-op-veel relatie voor elkaar te krijgen. Een veel-op-veel relatie heeft altijd een zogenaamde koppeltabel.

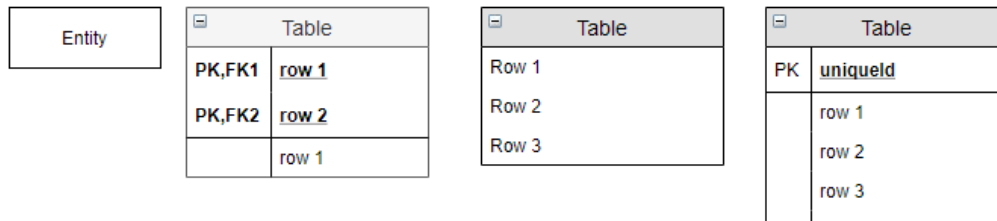


De tabel ORDERLIJN zorgt ervoor dat de relatie tussen ORDERS en PRODUCTEN veel-op-veel is. De tabel ORDERLIJN is de koppeltabel.

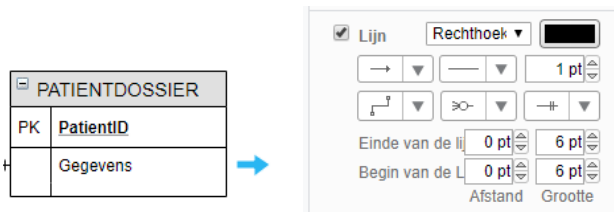
De relatie gelezen van links naar rechts: Een order bevat nul of meerdere producten.
De relatie gelezen van rechts naar links: Een product hoort bij nul of meerdere orders.

2.2.3 Tips voor het ontwerp van ER-diagrammen in draw.io.

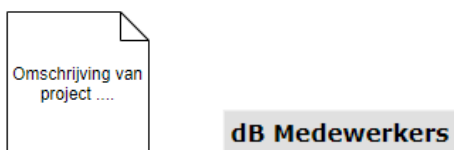
Identificeer de entiteiten. De eerste stap bij het maken van een ERD is het identificeren van alle entiteiten die u gaat gebruiken. Een entiteit is niets meer dan een rechthoek met een beschrijving van iets waarover uw systeem informatie opslaat. Dit kan een klant, een manager, een factuur, een planning, etc. zijn. Teken een rechthoek of gebruik een entiteitsmodel voor elke entiteit.



- **Identificeer relaties.** Kijk naar de relatie tussen de entiteiten. Zo ja, sleep de blauwe pijl om vanuit de entiteit met de vreemde sleutel naar de entiteit met de primaire sleutel. Selecteer de pijl en pas eventueel het begin- of eindpunt aan om de kardinaliteiten/ ordinaliteiten te specificeren.



- **Beschrijf de relatie.** Hoe zijn de entiteiten gerelateerd? Je kan een actiediamant tekenen tussen de twee entiteiten op de lijn die u zojuist hebt toegevoegd. Hierin wordt een korte beschrijving beschreven hoe de entiteiten gerelateerd zijn met elkaar. Wij passen dit niet toe.
- **Voeg attributen toe.** Je kan de sleutelkenmerken van entiteiten ook toevoegen met behulp van ovale symbolen. Wij passen dit niet toe.
- **Voltooi het diagram.** Zorg dat alle entiteiten verbonden zijn en de relaties (pijlen) tussen de entiteiten correct zijn. Voeg eventueel kleuren, notes, titels, tekst, links,.... toe.



3 Data Definition Language

3.1 Schema's en gebruikers

In een Oracle-database maken tabellen altijd onderdeel uit van een schema en een schema heeft over het algemeen een gebruiker als eigenaar. In een Oracle-database bestaat er over het algemeen een één-op-één-relatie tussen gebruikers en gelijknamige schema's. Een schema is een logische verzameling databaseobjecten waarmee een gebruiker een connectie kan leggen. Bij het aanloggen, wordt de gebruiker automatisch verbonden met het schema.

Wij zijn automatisch verbonden met het schema BOEK met de 7 casustabellen.

3.2 Tabellen maken

Het SQL-commando om tabellen te maken is CREATE TABLE. Om een tabel te kunnen maken moet voor die tabel een naam worden gekozen en vervolgens moeten alle kolommen gespecificeerd worden. Iedere kolomspecificatie wordt gevolgd door het datatype en een constraint.

```
CREATE TABLE tabelnaam
( kolomnaam      datatype      [kolomconstraint]
, kolomnaam      datatype      [kolomconstraint]
, ...
, [tabelconstraint]
) [AS query]
```

Het create table-commando bevat twee componenten: de *kolomspecificaties* en de *tabelspecificaties*.

Tabellen kunnen ook worden gecreëerd met behulp van een subquery in de AS-component. Hierbij wordt de tabel aangemaakt en de inhoud van de tabel gekopieerd. De constraints en de privileges worden niet mee gekopieerd.

Creatie van tabel personeel.

```
SQL> create table personeel
2   ( mnr          number(4)      constraint p_pk          primary key
3     ,            constraint p_mnr_chk    check(mnr>7000)
4     , naam       varchar2(22)   constraint p_naam_nn    not null
5     , functie    varchar2(10)   constraint p_funct_n    null
6     , chef       number(4)      constraint p_chef_fk     references personeel
7     , gbdatum    date           constraint p_gbdat_nn    not null
8     , salaris    number(6,2)    constraint p_mndsal_nn   not null
9     , afd        number(2)      default 10
10                                constraint p_afd_fk     references afdelingen
11  );
```

Creatie van tabel personeel dmv een query.

```
SQL> create table personeel
2     as select *
3     from scott.medewerkers;
```

Creatie van tabel personeel dmv een query (een deel wordt overgenomen).

```
SQL> create table personeel (pmnr,pnaam,pafd)
1     as select mnr,naam,afd
2     from scott.medewerkers;
```

Bij een tabel gecreëerd door middel van een query wordt enkel de inhoud gekopieerd en worden de constraints (en grants) niet overgenomen.

3.3 Datatypes

Oracle ondersteunt vele datatypes, sommige lijken op elkaar of zijn zelfs synoniem. Ze worden ondersteund vanwege de compatibiliteit van Oracle met andere database management systemen.

De voornaamste Oracle-datatypes zijn NUMBER, VARCHAR2 en DATE.

CHAR(n)	alfanumeriek met vaste lengte n
VARCHAR[2](n)	alfanumeriek met variabele lengte van max. n karakters
DATE	datum (4712 BC t.e.m. 4712 AD)
TIMESTAMP	tijdstip (met of zonder tijdzone-informatie)
INTERVAL	Tijdsinterval
RAW(n)	binaire gegevens met max. n bytes
NUMBER	geheel getal met een maximale precisie van 38 cijfers
NUMBER(n)	geheel getal van max. n cijfers
NUMBER(n,m)	precisie van n cijfers, waarvan m achter de decimale punt
BINARY_FLOAT	32-bits drijvende komma-getal
BINARY_DOUBLE	64-bits drijvende komma-getal
BLOB	ongestructureerde (binaire) gegevens van max. 4 GB
CLOB	grootte stukken tekst van max. 4 GB

BLOB (binary large object), CLOB (character large object) en BFILE (binary file) zijn in Oracle 8 geïntroduceerd ten behoeve van opslag van LOB's (Large Objects).

Sinds Oracle 7 betekenen VARCHAR en VARCHAR2 hetzelfde. Oracle adviseert echter om zoveel mogelijk gebruik te maken van VARCHAR2 omdat in de toekomst een verschil van behandeling tussen VARCHAR en VARCHAR2 zal worden gemaakt.

Sinds Oracle 10g worden ook drijvendekomma-getallen gebruikt. Ze zijn niet zo precies als NUMBER-getallen maar leveren wel een betere responstijd op bij intensieve berekeningen.

Ieder datatype heeft zijn eigen maximale breedte of precisie.

NUMBER	38 cijfers
CHAR	2000 karakters
VARCHAR[2]	4000 karakters

Voorbeelden

Voorletter	char(3)	Ch., P., ...
Naam	varchar2(25)	Briers , Vos, Slechten, ...
Getal	number(5)	23856, 99999, ...
Getal	number(6,2)	6583.50, 9999.99
Getal	number(6,-1)	748630 (tienvoud)
Getal	number(6,-3)	749000 (duizendvoud)
Getal	number	idem number(38,0)
Getal	number(*,5)	idem number(38,5)

3.4 Constraints

In het create table-commando kunnen constraint-definities worden opgenomen, hetzij als onderdeel van een kolomspecificatie (kolomconstraint of inline constraints) of als een afzonderlijk onderdeel (tabelconstraint). Tabelconstraints worden gebruikt bij het definiëren van constraints die betrekking hebben op meer dan één kolom of het definiëren van een samengestelde primaire sleutel of het later toevoegen van constraints.

Een optioneel onderdeel van iedere constraint-definitie is de specificatie van een naam. Het is beslist aan te raden om van deze mogelijkheid gebruik te maken. Als we het namelijk zelf niet doen, genereert Oracle een weinig informatieve naam voor ons, *SYS_Cnnnnn*, waarin *nnnnn* een willekeurig volgnummer is. Constraint-namen zijn nodig om constraints te kunnen raadplegen of te manipuleren (aanzetten, uitzetten, verwijderen). Ze komen ook in de foutmelding voor als ze geschonden worden. Een goed gekozen constraint-naam maakt de melding dan ook informatiever.

Kolomconstraint

CONSTRAINT [constraintnaam]	
• [NOT] NULL	moet (geen) waarde bevatten
• UNIQUE	moet uniek, maar niet ingevuld zijn
• PRIMARY KEY	primaire sleutel definiëren
• CHECK (voorwaarde)	conditie definiëren
• REFERENCES tabelnaam [(kolomnaam)][ON DELETE CASCADE]/[ON DELETE SET NULL]	vreemde sleutel

Tabelconstraint

CONSTRAINT [constraintnaam]

- UNIQUE moet uniek, maar niet ingevuld zijn
- PRIMARY KEY primaire sleutel definiëren
- CHECK (voorwaarde) conditie definiëren
- FOREIGN KEY (kolomnaam) REFERENCES ... (idem kolomconstraint) vreemde sleutel

Merk op

- De kolom functie met de Null-constraint wordt niet in de datadictionary opgenomen. Vandaar dat kolommen met een Null-constraint verder niet meer gedefinieerd worden.
- Het is wenselijk de constraint-namen betekenisvol te definiëren (max 30 karakters lang)

Afkorting tabelnaam_kolomnaam_soort constraint

(NOT) NULL	(N)N	UNIQUE	UN
PRIMARY KEY	PK	CHECK	CHK
REFERENCES	FK		

3.5 De casustabellen

```
SQL> create table medewerkers
2  ( mnr          number(4)      constraint m_pk      primary key
3    , naam        varchar2(15)  constraint m_mnr_chk check(mnr>7000)
4    , voorn       varchar2(12)  constraint m_naam_nn not null
5    , functie     varchar2(10)
6    , chef        number(4)      constraint m_chef_FK references medewerkers
7    , gbdatum     date          constraint m_gbdat_nn not null
8    , maandsal    number(6,2)   constraint m_mndsal_nn not null
9    , comm        number(6,2)
10   , afd         number(2)      default 10
11   , constraint m_verk_chk      check (decode(functie,'VERKOPER',0,1) +
12                                     decode(comm,NULL,0,1) = 1)
13
14 );
```

```
SQL> create table afdelingen
2  ( anr          number(2)      constraint a_PK      primary key
3    , naam        varchar2(20)  constraint a_anr_chk check (mod(anr,10)=0)
4    , locatie     varchar2(20)  constraint a_naam_nn not null
5    , hoofd       number(4)      constraint a_naam_un   unique
6    , constraint a_naam_chk      check(naam = upper(naam))
7    , constraint a_loc_nn       not null
8    , constraint a_loc_chk      check (locatie = upper(locatie))
9    , constraint a_hoofd_fk      references medewerkers
10 );
```

```
SQL> alter table medewerkers add
2  (constraint m_afd_fk foreign key (afd) references afdelingen);
```

```

SQL> create table schalen
2   ( snr          number(2)      constraint s_pk   primary key
3     , ondergrens number(6,2)    constraint s_onder_nn not null
4                                     constraint s_onder_chk check (ondergrens>=0)
5     , bovengrens number (6,2)   constraint s_boven_nn not null
6     , toelage    number (6,2)   constraint s_toelg_nn not null
7     , constraint s_onder_boven_chk check (ondergrens <= bovengrens)
8   );

```

```

SQL> create table cursussen
2   ( code          varchar2(4)    constraint c_PK   primary key
3     , omschrijving varchar2(50)  constraint c_omschr_nn not null
4     , type         char(3)       constraint c_type_nn not null
5     , lengte       number(2)     constraint c_lengte_nn not null
6     , constraint c_type_chk  check (type in ('ALG','BLD','DSG'))
7     , constraint c_code_chk  check (code = upper(code))
8   );

```

```

SQL> create table uitvoeringen
2   ( cursus        varchar2(4)    constraint u_cursus_nn not null
3                                     constraint u_cursus_fk references cursussen
4     , begintatum  date           constraint u_begin_nn not null
5     , docent      number(4)      constraint u_docent_fk references medewerkers
6     , locatie     varchar2(20)
7     , constraint u_pk primary key (cursus,begintatum)
8   );

```

```

SQL> create table inschrijvingen
2   ( cursist       number(4)      constraint i_cursist_nn not null
3                                     constraint i_cursist_FK references medewerkers
4     , cursus       varchar2(4)   constraint i_cursus_nn not null
5     , begintatum  date          constraint i_begindat_nn not null
6     , evaluatie   number(1)     constraint i_eval_chk check
7                                     (evaluatie in (1,2,3,4,5))
8     , constraint i_pk primary key (cursist,cursus,begintatum)
9     , constraint i_uitv_fk foreign key(cursus,begintatum)
10                                     references uitvoeringen
11   );

```

```

SQL> create table historie
2   ( mnr          number(4)      constraint h_mnr_nn not null
3                                     constraint h_mnr_FK references medewerkers
4                                     on delete cascade
5     , beginjaar   number(4)      constraint h_bjaar_nn not null
6     , begintatum  date          constraint h_begin_nn not null
7     , einddatum   date
8     , afd         number(2)     constraint h_afd_nn
9                                     constraint h_afd_fk references afdelingen
10     , maandsal    number(6,2)   constraint h_sal_nn not null
11     , opmerkingen varchar2(60)
12     , constraint h_pk          primary key (mnr,begintatum)
13     , constraint h_beg_eind check (begintatum < einddatum)
14   );

```

Oracle kent ook het SQL-commando CREATE SCHEMA, waarmee we in staat zijn om in één DDL-transactie een aantal tabellen tegelijk te creëren. Het voordeel daarvan is dat de actie als geheel slaagt of faalt. Bovendien moet er geen ALTER TABLE-commando worden gebruikt. De casustabellen zouden dan als volgt worden gecreëerd:

```

SQL> create schema authorization BOEK
2   create table medewerkers (...)
3   create table afdelingen (...)
4   create table cursussen (...)
5   create table uitvoeringen (...)
6   create table inschrijvingen (...)
7   create table historie (...)
8 );

```

3.6 De Datadictionary

Als wij bijvoorbeeld willen weten welke tabellen in de database aanwezig zijn, welke kolommen ze hebben, welke privileges ze hebben en meer van dat soort vragen, dan kunnen we daartoe de datadictionary raadplegen. De datadictionary is eigenlijk de interne administratie van Oracle, die gegevens over de metagegevens voortdurend bijhoudt. Vermits de datadictionary-gegevens ook in tabellen worden bijgehouden, kunnen we ze met behulp van SQL raadplegen.

De tabellen in de datadictionary zijn opgedeeld in 3 groepen:

USER_...	informatie over eigen objecten
ALL_...	informatie toegankelijk voor de gebruiker
DBA_...	alleen toegankelijk voor de databasebeheerder

De namen geven over het algemeen een duidelijke indicatie van de inhoud van de tabellen. Voor de veelvoorkomende tabellen (met lange namen) kunnen vaak synoniemen worden gebruikt.

DICT	overzicht van alle dictionary tabellen;
CAT	overzicht van onze eigen tabellen, indexen;...
TABS	overzicht van onze eigen tabellen;
COLS	overzicht van de kolommen van onze eigen tabellen
USER_CONSTRAINTS	overzicht van onze constraints
USER_TAB_PRIVS	overzicht toegekende en verleende privileges

Voorbeeld

```

SQL> select constraint_name,constraint_type,search_condition
2   from user_constraints
3   where table_name = 'MEDEWERKERS';

```

CONSTRAINT_NAME	C	SEARCH_CONDITION
-----	-	-----
M_NAAM_NN	C	NAAM IS NOT NULL
M_VOORN_NN	C	VOORN IS NOT NULL
M_GBDAT_NN	C	GBDATUM IS NOT NULL
M_MNDSAL_NN	C	MAANDSAL IS NOT NULL
M_PK	P	
M_MNR_CHK	C	mnr>7000
M_CHEF_FK	R	
M_AFD_FK	R	

3.7 Alter Table

Soms is het noodzakelijk om de bestaande tabelstructuur aan te passen.

```
ALTER TABLE tabelnaam  
[ADD (kolomspecificatie|tabelconstraint)]  
[MODIFY(kolomspecificatie)]  
[RENAME COLUMN oudnaam TO nieuwnaam]  
[DROP COLUMN kolomnaam]
```

```
ALTER TABLE tabelnaam  
[DROP CONSTRAINT constraintnaam]  
[DISABLE CONSTRAINT constraintnaam]  
[ENABLE CONSTRAINT constraintnaam]
```

Met de ADD-optie kan je:

- kolommen toevoegen;
- **tabel**constraints toevoegen (geen kolomconstraints!).

Met de MODIFY-optie kan je:

- datatype van een kolom wijzigen;
- size van een kolom wijzigen;
- default-waarde aan een kolom toevoegen/wijzigen;
- Not Null-kolomconstraint toevoegen;

Met RENAME-optie kan je de naam van een kolom wijzigen.

Met DROP COLUMN kan je de kolommen van tabellen verwijderen.

Met de optie voor CONSTRAINT-MANIPULATIE kan je constraints verwijderen, aan- of uitzetten.

Merk op

- Constraints kan je niet wijzigen, enkel verwijderen en vervolgens toevoegen. Een Null-constraint kan je echter niet verwijderen, vermits deze niet in de datadictionary wordt opgenomen.
- Het spreekt vanzelf dat je voor de bovenstaande opties een onderscheid moet maken tussen een lege tabel of een gevulde tabel (bevat reeds gegevens).
Je kan elke soort wijziging bij een lege tabel doorvoeren, terwijl voor een *gevulde tabel* kan je:
 - ✓ Enkel een Null-kolom toevoegen (ADD);
 - ✓ Enkel de size vergroten en niet verkleinen (MODIFY);
 - ✓ Enkel het datatype wijzigen van Not Null naar Null (MODIFY).

Toevoegen van een tabelconstraint.

```
SQL> alter table personeel
```

```
2      add (constraint p_afd_fk foreign key (afd) references afdelingen);
```


De size van de kolom Afd wordt vergroot en Not Null wordt toegevoegd.

```
SQL> alter table personeel  
3      modify afd number(5) not null;
```

De constraint p_afd_fk wordt uitgezet.

```
SQL> alter table personeel  
1      disable constraint p_afd_fk;
```

3.8 Indexen

3.8.1 Index maken

De rijen binnen een tabel hebben vaak een willekeurige volgorde. Als een tabel veel rijen (vb. 30 000) bevat en we zouden de medewerkers moeten zoeken wiens naam met een 'B' begint dan moeten alle rijen doorlopen worden en zullen we even op de resultaatquery moeten wachten.

Een index zou hier uitkomst kunnen bieden omdat er dan een gesorteerde lijst wordt bijgehouden en het doorzoeken veel sneller gaat. De responstijd voor de resultaatquery zal aanzienlijk korter zijn.

Het is duidelijk dat queries veel voordeel kunnen hebben van indexen maar anderzijds zal de datamanipulatie veel trager verlopen omdat iedere wijziging ook doorgevoerd moet worden in de indexen. Bovendien kosten indexen ook extra opslagruimte.

Indexen worden dan ook enkel op bepaalde kolommen ingesteld:

- indexen op vreemde sleutels;
- indexen op kolommen die vaak in de WHERE-restrictie voorkomen;
- indexen op kolommen waarop vaak gesorteerd wordt.

```
CREATE [UNIQUE] INDEX indexnaam  
ON tabelnaam (kolom1 [ASC | DESC] , [kolom2...])  
[TABLESPACE tablespacenaam]
```

Voorbeeld 1

```
SQL> CREATE INDEX idx_geboortedatum  
2   ON medewerkers (gbdatum);
```

Index created.

Voorbeeld 2

```
SQL> CREATE INDEX idx_volnaam  
2   ON medewerkers ( voorn, naam)
```

Index created.

Voorbeeld 3

```
SQL> CREATE INDEX idx_uppernaam  
2   ON medewerkers (UPPER(naam))
```

Index created.

Om zeker te zijn dat Oracle deze index gebruikt, moet je ook onderstaande query gebruiken:

```
SQL> SELECT voorn, naam
2      FROM medewerkers
3      WHERE UPPER(naam) IS NOT NULL
4      ORDER BY UPPER(naam)
```

Voorbeeld 4

```
SQL> CREATE INDEX idx_jaarsalaris
2* ON medewerkers (12*maandsal + coalesce(comm,0))
```

Index created.

3.8.2 Index opvragen

```
SQL> select index_name, uniqueness, table_name
2 from user_indexes
```

INDEX_NAME	UNIQUENES	TABLE_NAME
U2_PK	UNIQUE	UITVOERINGEN
OP_PK	UNIQUE	OPLEIDING
M2_PK	UNIQUE	MEDEWERKERS
IDX_FAMILIENAAM	NONUNIQUE	MEDEWERKERS
IDX_VOLNAAM	UNIQUE	MEDEWERKERS
IDX_GEBOORTEDATUM	NONUNIQUE	MEDEWERKERS
IDX_JAARSALARIS	NONUNIQUE	MEDEWERKERS
I2_PK	UNIQUE	INSCHRIJVINGEN
C2_PK	UNIQUE	CURSUSSEN
A2_PK	UNIQUE	AFDELINGEN
A2_NAAM_UN	UNIQUE	AFDELINGEN

3.8.3 Index verwijderen

We kunnen indexen verwijderen met behulp van het commando DROP INDEX.

DROP INDEX indexnaam

3.9 Sequences

In informatiesystemen komt het vaak voor dat unieke volgnummers worden gehanteerd. Bijvoorbeeld voor orders of factuurnummers die elkaar strikt opvolgen. Het gebruik van sequences kan hier uitstekend voor gebruikt worden.

Sequences kunnen respectievelijk gecreëerd, gewijzigd of verwijderd worden met de volgende onderstaande SQL-commando's.

```
CREATE SEQUENCE sequencenaam  
ALTER SEQUENCE sequencenaam  
DROP SEQUENCE sequencenaam
```

CREATE SEQUENCE schema_name.sequence_name	
[INCREMENT BY interval]	stapgrootte
[START WITH first_number]	startnummer
[MAXVALUE max_value NOMAXVALUE]	maximum
[MINVALUE min_value NOMINVALUE]	minimum
[CYCLE NOCYCLE]	stapgrootte bij overschrijding max./min.
[CACHE cache_size NOCACHE]	geheugenruimte
[ORDER NOORDER]	niet van toepassing in deze cursus

Voorbeeld 1

```
SQL> CREATE SEQUENCE id_seq  
2 INCREMENT BY 10  
3 START WITH 10  
4 MINVALUE 10  
5 MAXVALUE 100  
6 CYCLE  
7 CACHE 2;
```

Sequence created.

```
SQL> SELECT id_seq.NEXTVAL, id_seq.CURRVAL --Geeft volgende en huidige waarde  
2* FROM dual
```

```
SQL> SELECT  
2 id_seq.NEXTVAL  
3 FROM  
4 dual  
5 CONNECT BY level <= 9; -- 9 waardes worden er aangemaakt.
```

NEXTVAL

```
-----  
30  
40  
50  
60  
70  
80  
90  
100  
10 -- Vermits max = 100 + CYCLE geactiveerd, wordt er terug 10 gegenereerd.
```

9 rows selected.

Voorbeeld 2

```
SQL> CREATE SEQUENCE seq_afd
  2 INCREMENT BY 10
  3 START WITH 50;
```

```
SQL> insert into afdelingen
  2 values (seq_afd.Nextval, 'ADMINISTRATIE', 'ZONHOVEN', 7698)
```

```
SQL> UPDATE medewerkers
  2 SET afd= seq_afd.currval
  3 WHERE mnr = 7698
```

3.10 Drop Table

DROP TABLE tabelnaam [CASCADE CONSTRAINTS]

Met het Drop-commando wordt de tabel uit de database verwijderd. Het effect van dit commando is onherroepelijk, er is geen rollback mogelijk. Deze eigenschap geldt overigens voor alle DDL-commando's van SQL.

Met de tabel worden ook allerlei daarmee samenhangende zaken uit de database verwijderd, zoals privileges op de tabel die aan bepaalde gebruikers waren verleend. Dat betekent dat bij een reorganisatie van de database, het niet zal volstaan met het opnieuw creëren van de tabel, maar dat ook de hele structuur eromheen opnieuw zal moeten worden opgezet.

Het kan voorkomen dat een tabel niet zomaar kan worden verwijderd omdat er vanuit een andere tabel een refererende sleutel op is gedefinieerd. Dit probleem kan worden opgelost met de optie *Cascade Constraints*, waardoor impliciet alle constraints worden verwijderd die van de tabel afhankelijk zijn.

Omdat het verwijderen van tabellen toch wel eens per ongelijk kan gebeuren, is in Oracle het concept van een prullenmand RECYCLE_BIN geïntroduceerd. Alle tabellen die worden verwijderd kunnen terug uit de prullenmand worden gehaald als inmiddels de prullenmand niet leeg (door administrator of impliciet door Oracle) is.

Prullenmand bekijken.

```
SQL> select * from user_recyclebin;
```

Uit prullenmand tabel halen.

```
SQL> flashback table afdelingen to before drop [rename to <nieuwe naam>];
```

Tabel definitief verwijderen (komt niet meer in prullenmand).

```
SQL> drop table afdelingen purge;
```

3.11 Overige commando's

3.11.1 Truncate

```
TRUNCATE tabelnaam
```

Met het Truncate-commando kan je grote tabellen op efficiënte wijze leeg maken zonder ze te verwijderen. Bovendien gaan geen privileges verloren en is dit commando aanzienlijk sneller dan het Delete-commando.

3.11.2 Rename

```
RENAME oude_tabelnaam TO nieuwe_tabelnaam
```

Met het Rename-commando kan je de tabel een nieuwe naam geven.

3.11.3 Synoniemen

Met behulp van het CREATE SYNONYM-commando kunnen synoniemen worden gedefinieerd voor tabelnamen. Synoniemen kunnen worden toegepast als tabelnamen er lang zijn. Vooral als men regelmatig tabellen van andere gebruikers raadpleegt zijn synoniemen interessant, anders moet steeds de naam van de eigenaar voor de tabelnaam (bv. scott.medewerkers) worden gespecificeerd.

Voor de belangrijkste tabellen van de datadictionary maken we trouwens geregeld gebruik van synoniemen.

DICT	synoniem voor DICTIONARY
CAT	synoniem voor USER_CATALOG
TABS	synoniem voor USER_TABLES
COLS	synoniem voor USER_TAB_COLUMNS
SYN	synoniem voor USER_SYNONYMS

Creatie van een synoniemnaam med voor de tabelnaam scott.medewerkers.

```
SQL> create synonym med  
2 for scott.medewerkers;
```

3.11.4 Comment on

Met behulp van het comment-commando kan je verklarende tekst bij tabellen en/of kolommen opslaan in de datadictionary.

```
COMMENT ON  
[ TABLE tabelnaam | COLUMN kolomnaam ] IS 'commentaar'
```

Commentaar bij de tabel Schalen.

```
SQL> comment on table schalen is 'salarisschalen en netto toelages';
```

Commentaar op de kolom comm van de tabel medewerkers.

SQL> comment on column medewerkers.comm is 'alleen voor verkopers';

Je kan het commentaar opvragen bij de tabellen *user_tab_comments* en *user_col_comments*.

```
SQL> select comments from user_tab_comments
      2   where table_name = 'SCHALEN';
```

COMMENTS

salarisschalen en netto toelages

```
SQL> select comments from user_col_comments
      2   where table_name = 'MEDEWERKERS'
      3   and column_name = 'COMM';
```

COMMENTS

alleen voor verkopers

3.11.5 Scripts

Indien we een aantal gerelateerde SQL-commando's samen willen bewaren, kunnen we dit vastleggen in een bestand. Dit soort bestanden zijn SQL*Plus-scripts en hierin kunnen we ook nog SQL*Plus-commando's in opnemen.

Voorbeelden

Zie CRECASE2015.sql en VULCASE205.sql

3.12 Oefeningen

3.12.1 Creatie database Teams

- 1 Creëer volgende 3 tabellen met de nodige constraints. Zorg voor betekenisvolle constraintnamen. Voeg achter de tabelnamen en constraintnamen je initialen toe (vb. spelersbrp).

SPELERS

spelersnr	numeriek 5 lang	primaire sleutel
naam	alfanum. 20 lang	verplicht
voorletter	alfanum. 3 lang	
geslacht	alfanum. 1 lang	verplicht en standaard « M »
gebdatum	datumveld	moet ingevuld worden
ploegnr	numeriek 3 lang	moet voorkomen in de tabel TEAMS

TEAMS

teamnr	numeriek 3 lang	primaire sleutel
teamnaam	alfanum. 20 lang	moet in hoofdletters
kapitein	numeriek 5 lang	moet voorkomen in de tabel SPELERS moet ingevuld zijn

BOETES

boetenr	numeriek 3 lang	primaire sleutel
spelersnr	num. 5 lang	moet voorkomen in de tabel spelers
datum	datumveld	verplicht
bedrag	numeriek 5 cijfers voor en 2 na de komma	verplicht

- 2 Vul deze tabellen ook op met enkele gegevens.

S P E L E R S	SPNR	NAAM	VOO	G	GEBDATUM	PLGNR
	10000	Haesen	JK	M	12-FEB-89	3
	90030	Eerdekens	DMV	M	25-APR-91	1
	10010	Lahon	G	M	01-JAN-95	2
	10020	Kusters	P	M	15-MAR-90	5
	10030	Max	BR	M	4-MAY-89	5
	90020	Noten	AP	M	23-JUN-91	3
	90050	Peels	V	M	05-AUG-87	2
	10040	Colijn	AB	M	11-JUL-90	3
	90010	Slingers	L	M	27-SEP-83	5
	90005	Pauwels	JK	M	17-NOV-92	4
	10045	Knevels	T	M	26-OCT-86	4
	10050	Vossen	CG	M	07-MAY-84	

B O E T E S	BNR	SPNR	DATUM	BEDRAG
	1	10020	15-JAN-16	5000
	2	10045	29-MAR-16	13500.5
	3	10020	07-NOV-16	4550

T E A M S	TNR	NAAM	KAPITEIN
	1	TEAM SPIRIT	90020
	2	DUCAMPS	90030
	3	A.A.C SANA	90050
	4	TEAM APOLLO	90010
	5	A.M.P. BRITEL	90005

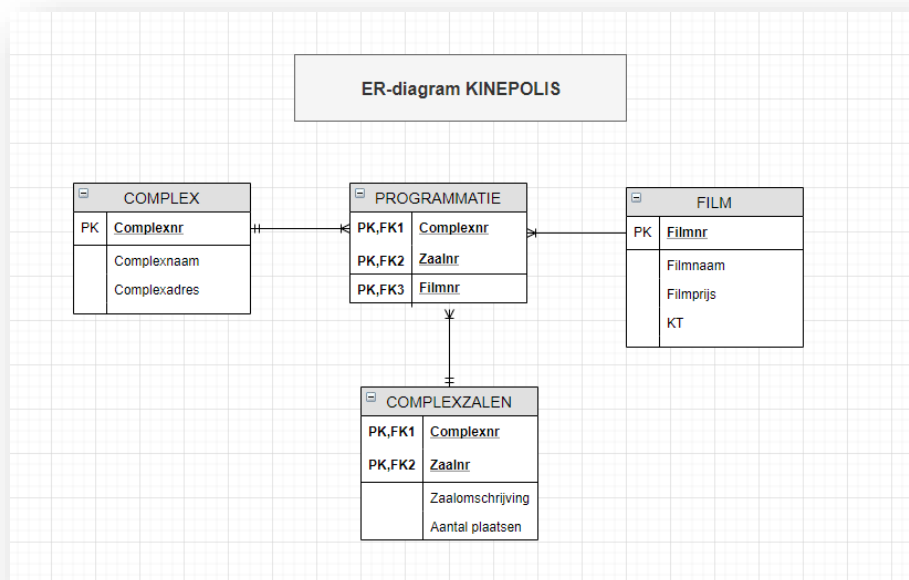
- 3 Zet de constraint uit die er voor zorgt dat het veld datum moet ingevuld worden in de tabel boetes.
- 4 Voeg aan de spelersrelatie een attribuut met de naam woonplaats toe.
- 5 Verwijder de kolom voorletter uit de tabel spelers.
- 6 Verwijder de boetes-tabel.

3.12.2 Creatie database Garage Gebr. Valkenburg

- 1 Maak een script dat db Valkenburg definieert op basis van bijgevoegd schema (zie volgende bladzijde).
- 2 Voeg gegevens toe in de tabellen.

3.12.3 Creatie database Kinopolis

- 1 Maak een script dat dB Kinopolis maakt op basis van onderstaand ERD. Formuleer correcte en zinvolle datatypes voor alle attributen.



3.12.4 Creatie tabel HISTORIE in dB medewerkers.

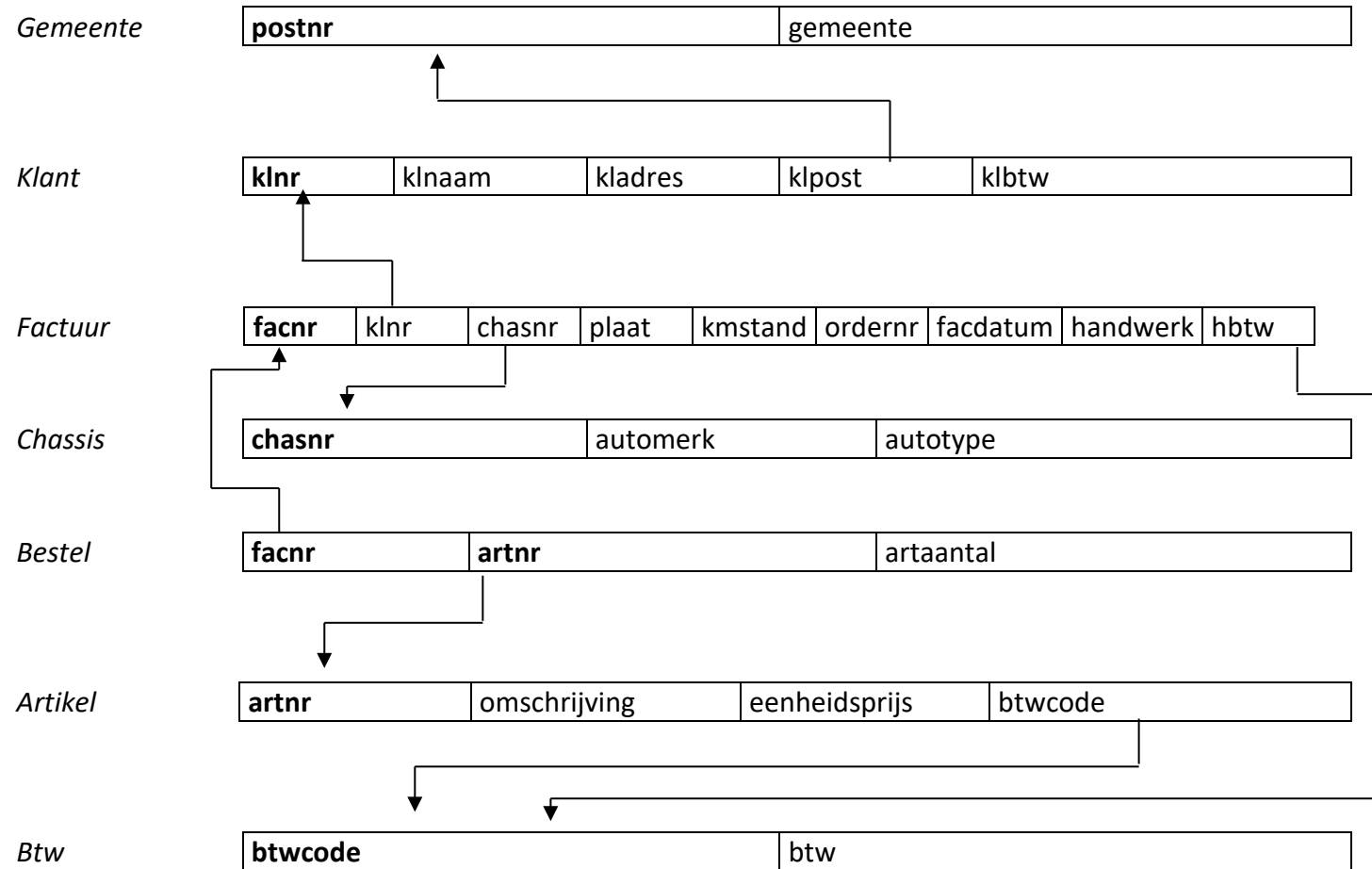
In de HISTORIE-tabel worden gegevens vastgelegd met betrekking tot het arbeidsverleden van de medewerkers. Vanaf de indiensttreding wordt iedere wijziging van afdeling/maandsalaris geregistreerd. De huidige afdeling en het huidige maandsalaris wordt ook opgeslagen waarbij het attribuut EINDATUM leeg wordt gelaten. Er is ook ruimte voor opmerkingen.

HISTORIE

mnr	numeriek 4 lang	ref. naar tabel medewerkers + primaire sleutel
beginjaar	numeriek 4 lang	verplicht
begindatum	datum	verplicht + primaire sleutel
einddatum	datum	
afd	numeriek 2 lang	verplicht + ref. naar afdelingen
maandsal	numeriek (6,2)	verplicht
opmerkingen	alfanumeriek 60 lang	

Bijkomende restrictie: begindatum moet liggen voor de einddatum.

Relaties database case-study garage Gebr. Valkenborgh



Gegevens Case-study Garage Gebr. Valkenborgh

Gemeente

POSTNR GEMEENTE

1000 BRUSSEL
2000 ANTWERPEN
2800 MECHELEN
3500 HASSELT
3590 DIEPENBEEK
3600 GENK
3680 MAASEIK
3740 BILZEN
3910 NEERPELT
8500 KORTRIJK
8660 DE PANNE
9000 GENT

Klant

KLNR	KLNAAM	KLADRES	KLPOST	KLBTW
-----	-----	-----	-----	-----
110680	Acaciahof	Zonneweg 52	3910	400454008
190447	Patricia Briers	Luikerpoort 23	3680	
190666	Dhr Gert Vos	Lazarijstraat 61	3500	
230545	Café De zoete inval	Dorpstraat 45	3590	696171770
445788	Automaatschappij nv	Kortrijksesteenweg 377	8500	419596561
558637	R.Slechten bvba	Minderbroederstraat 55	3600	416667656

BTW

BTWCODE	BTW
-----	-----
0	0
1	6
2	19
3	21

Artikel

ARTNR	OMSCHRIJVING	EENHEIDSPRIJS	BTWCODE
-----	-----	-----	-----
A0060517424	V. Riem	385	3
A0060710303	Remolie	560	3
A0070526314	Waterpomp	4520	3
A0080641345	Batterij	4570	3
A0080658456	Uitlaat	2680	3
AM504511200	Antivries	189	3

Chassis

CHASNR	AUTOMERK	AUTOTYPE
-----	-----	-----
ZAR905A1005263268	ALFA ROMEO	ALFA 33 1,3
QSR638V896778514	TOYOTA	AVENSIS 1,8
KLU235TR7612583	VOLKSWAGEN	PASSAT TD 1,2
MD12GN7812556398	FORD	FOCUS BREAK

Bestel

FACNR	ARTNR	ARTAANTAL
-----	-----	-----
2108	A0060517424	1
2108	A0060710303	.3
2108	AM504511200	.5
2230	A0070526314	1
2365	A0080658456	1
2365	AM504511200	1

Factuur

FACNR	KLNR	CHASNR	PLAAT	KMSTAND	ORDERNR	FACDATUM	HANDWERK	HBTW
2108	190666	ZAR905A1005263268	GNT953	98232	612	06-DEC-19	985	3
2230	230545	MD12GN7812556398	KCC955	88123	785	06-DEC-19	2250	3
	23650447	KLU235TR7612583	BNV778	45336	714	12-JAN-20	1780	3

4 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.

4.1 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
3 where snr <= 4;
```

4.2 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');
```

4.3 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..... ;

4.4 Transactieverwerking

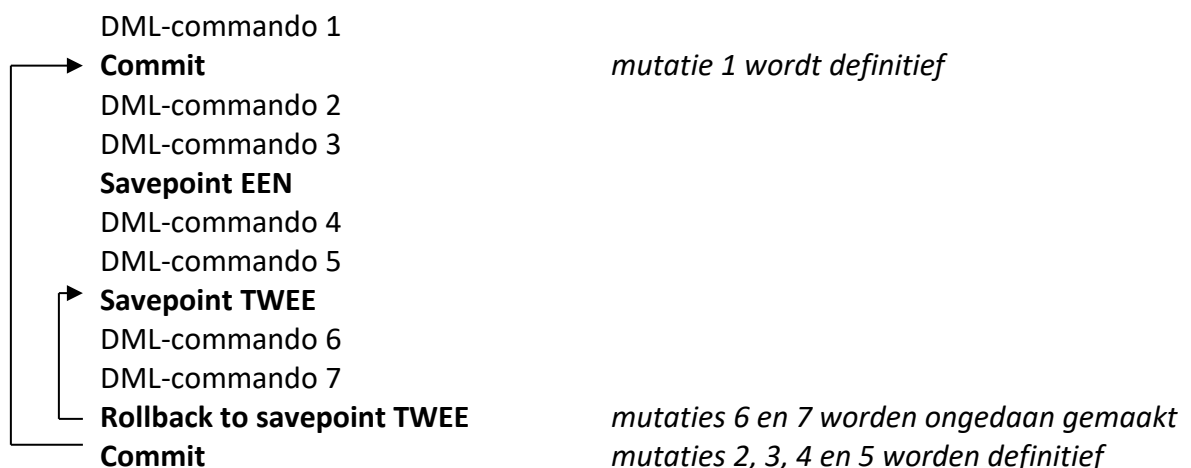
Wijzigingen in de inhoud van de database krijgen in eerste instantie een voorlopig karakter. Dat houdt onder meer in dat we de door ons gewijzigde waarden zelf kunnen bekijken, vooraleer we de wijzigingen definitief doorvoeren.

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

4.5 Oefeningen

4.5.1 Database Teams

- 1 Verwijder de spelers met spelersnr tussen 1 en 4.
- 2 Wijzig de naam van een speler uit de spelers tabel.
- 3 Voer de gegevens in tabel Boetes: boetenr. 4, € 2 500,00 boete op 15/11/2017 voor Pauwels. De naam wordt in de tabel Spelers eveneens geselecteerd.

5 SQL Server Management Studio (SSMS)

SQL Server Management Studio (SSMS) is een geïntegreerde gratis omgeving voor het beheer van elke SQL-infrastructuur, van SQL Server tot Azure SQL Database. Visual Studio maakt volop gebruik van de SQL Server om te werken met databases.

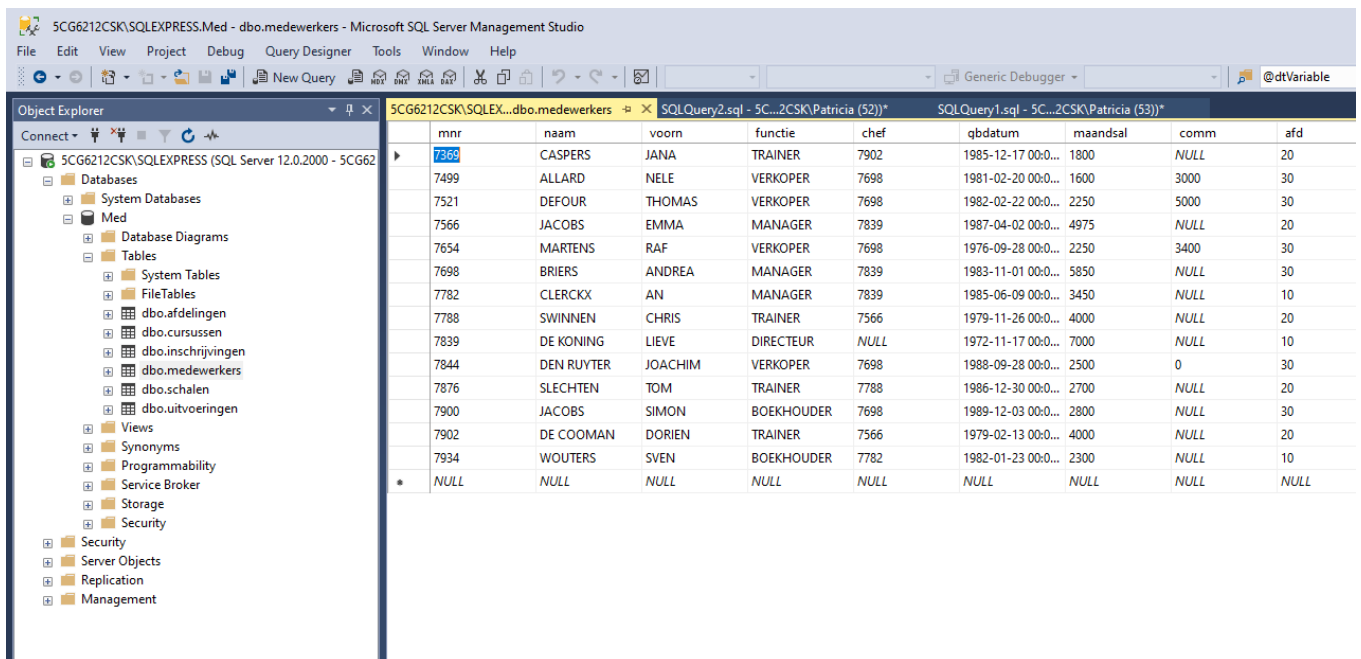
5.1 Installeer SQL Server Management Studio

Microsoft stelt SSMS gratis beschikbaar via Microsoft's eigen website. Als je SQL Server Management Studio nog nooit hebt geïnstalleerd, kies dan voor de downloadlink *Download SQL Server Management Studio*. Eventueel is er ook een download beschikbaar om je al geïnstalleerde versie van SSMS bij te werken naar de nieuwste versie.

Helaas is er op dit moment geen Nederlandstalige download beschikbaar van SQL Server Management Studio.

5.2 Wat kan SQL Server Management Studio?

Via SQL Server Management Studio kan je een groot aantal veel voorkomende databasetaken eenvoudig uitvoeren. Je kan de tabellen in je database bekijken, wijzigingen aanbrengen in de structuur van je database en de data via de grafische interface bewerken. Daarnaast kan je ook rechtstreeks query's uitvoeren op de database met behulp van het *New Query* venster.



mntr	naam	voorn	functie	chef	qbdatum	maandsal	comm	afd
7369	CASPERS	JANA	TRAINER	7902	1985-12-17 00:00...	1800	NULL	20
7499	ALLARD	NELE	VERKOPER	7698	1981-02-20 00:00...	1600	3000	30
7521	DEFOUR	THOMAS	VERKOPER	7698	1982-02-22 00:00...	2250	5000	30
7566	JACOBS	EMMA	MANAGER	7839	1987-04-02 00:00...	4975	NULL	20
7654	MARTENS	RAF	VERKOPER	7698	1976-09-28 00:00...	2250	3400	30
7698	BRIERS	ANDREA	MANAGER	7839	1983-11-01 00:00...	5850	NULL	30
7782	CLERCKX	AN	MANAGER	7839	1985-06-09 00:00...	3450	NULL	10
7788	SWINNEN	CHRIS	TRAINER	7566	1979-11-26 00:00...	4000	NULL	20
7839	DE KONING	LIEVE	DIRECTEUR	NULL	1972-11-17 00:00...	7000	NULL	10
7844	DEN RUYTER	JOACHIM	VERKOPER	7698	1988-09-28 00:00...	2500	0	30
7876	SLECHTEN	TOM	TRAINER	7788	1986-12-30 00:00...	2700	NULL	20
7900	JACOBS	SIMON	BOEKHOUDER	7698	1989-12-03 00:00...	2800	NULL	30
7902	DE COOMAN	DORIEN	TRAINER	7566	1979-02-13 00:00...	4000	NULL	20
7934	WOUTERS	SVEN	BOEKHOUDER	7782	1982-01-23 00:00...	2300	NULL	10
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

5.3 Connectie

Zie uitleg WPL2.

5.4 Oracle SQL versus SQL Server

Er wordt een vergelijking gemaakt tussen Oracle SQL en SQL Server.

	Oracle		SQL Server
1	 Operator	String concatenation	+ Operator and CONCAT function
2	+ and - Operators for date	Datetime arithmetic	+ and - Operators for datetime

Data Types

	Oracle		SQL Server	
1	DATE	Date and time with seconds	DATETIME	DATETIME2(0)
2	INTERVAL YEAR(p) TO MONTH	Date interval	VARCHAR(30)	
3	INTERVAL DAY(p) TO SECOND(s)	Day and time interval	VARCHAR(30)	
4	TIMESTAMP(p)	Date and time with fraction	DATETIME2(p)	
5	TIMESTAMP(p) WITH TIME ZONE	Date and time with fraction and time zone	DATETIMEOFFSET(p)	

	Oracle		SQL Server
1	BFILE	Pointer to binary file, ≤ 4G	VARCHAR(255)
2	BINARY_FLOAT	32-bit floating-point number	REAL
3	BINARY_DOUBLE	64-bit floating-point number	DOUBLE PRECISION
4	BLOB	Binary large object, ≤ 4G	VARBINARY(max)
5	CHAR(n), CHARACTER(n)(*)	Fixed-length string, 1 ≤ n ≤ 2000	CHAR(n), CHARACTER(n)
6	CLOB	Character large object, ≤ 4G	VARCHAR(max)
7	DECIMAL(p,s), DEC(p,s)	Fixed-point number	DECIMAL(p,s), DEC(p,s)
8	DOUBLE PRECISION	Floating-point number	FLOAT
9	FLOAT(p)	Floating-point number	FLOAT
10	INTEGER, INT	38 digits integer	DECIMAL(38)
11	LONG	Character data, ≤ 2G	VARCHAR(max)

12	LONG RAW	Binary data, $\leq 2\text{G}$	VARBINARY(max)
13	NCHAR(n)	Fixed-length UTF-8 string, $1 \leq n \leq 2000$	NCHAR(n)
14	NCHAR VARYING(n)	Varying-length UTF-8 string, $1 \leq n \leq 4000$	NVARCHAR(n)
15	NCLOB	Variable-length Unicode string, $\leq 4\text{G}$	NVARCHAR(max)
16	NUMBER($p,0$), NUMBER(p)	8-bit integer, $1 \leq p < 3$	TINYINT (0 to 255)
		16-bit integer, $3 \leq p < 5$	SMALLINT
		32-bit integer, $5 \leq p < 9$	INT
		64-bit integer, $9 \leq p < 19$	BIGINT
		Fixed-point number, $19 \leq p \leq 38$	DECIMAL(p)
17	NUMBER(p,s)	Fixed-point number, $s > 0$	DECIMAL(p,s)
18	NUMBER, NUMBER(*)	Floating-point number	FLOAT
19	NUMERIC(p,s)	Fixed-point number	NUMERIC(p,s)
20	NVARCHAR2(n)	Varying-length UTF-8 string, $1 \leq n \leq 4000$	NVARCHAR(n)
21	RAW(n)	Variable-length binary string, $1 \leq n \leq 2000$	VARBINARY(n)
22	REAL	Floating-point number	FLOAT
23	ROWID	Physical row address	CHAR(18)
24	SMALLINT	38 digits integer	DECIMAL(38)
25	UROWID(n)	Logical row addresses, $1 \leq n \leq 4000$	VAR CHAR(n)
26	VARCHAR(n)	Variable-length string, $1 \leq n \leq 4000$	VARCHAR(n)
27	VARCHAR2(n)	Variable-length string, $1 \leq n \leq 4000$	VARCHAR(n)

5

SELECT Statement

	Oracle		SQL Server	
1	DUAL table	A single row, single column dummy table	FROM clause can be omitted, DUAL removed	
2	FROM (SELECT ...)	Optional alias for subquery	FROM (SELECT ...) s	Alias required

SQL Statements

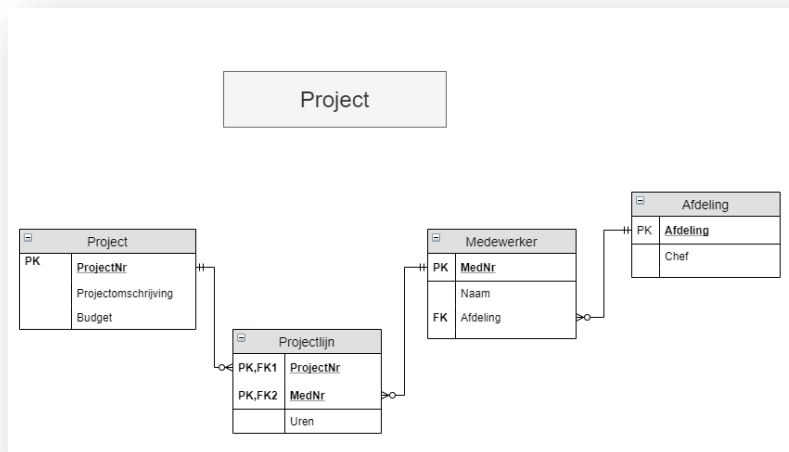
	Oracle	SQL Server
1	COMMENT ON COLUMN <i>schema.tab.col</i> IS ' <i>text</i> '	execute sp_addextendedproperty 'MS_Description', ' <i>text</i> ', 'user', ' <i>schema</i> ', 'table', ' <i>tab</i> ', 'column', ' <i>col</i> '
2	CREATE PUBLIC SYNONYM	CREATE SYNONYM

SQL*Plus Commands

	Oracle		SQL Server
1	PROMPT <i>text</i>	Output a text message	PRINT ' <i>text</i> '
2	REM REMARK <i>text</i>	Single line comment	-- <i>text</i>
3	&variable	Substitution variable in a script	\$(variable)

5.5 Creatie van database en tabellen.

We willen onderstaande database PROJECT maken. Je kan in SQL Server tabellen maken, gegevens toevoegen, raadplegen,... op 2 manieren. Ofwel werken we met de SQL-commando's of je kan ook werken met de grafische interface (GUI) van SSMS.



5.5.1 DML met de SQL commando's.

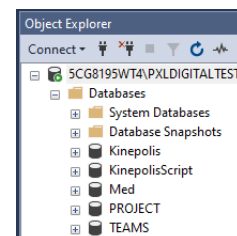
5.5.1.1 Create database

Je kan een database aanmaken waarin je de tabellen kan creëren.

```
CREATE DATABASE databasenaam
```

Create database PROJECT en druk op **Execute/F5**

In de Object Explorer vind je de nieuwe database terug.



5.5.1.2 Create Table

```
CREATE TABLE [databasenaam.][schemaanaam.]tabelnaam
( kolomnaam datatype [kolomconstraint]
, kolomnaam datatype [kolomconstraint]
, ...
, [tabelconstraint]
) [AS query]
```

```
-- Naar database PROJECT navigeren om onderstaande tabellen aan te maken.
use project
```

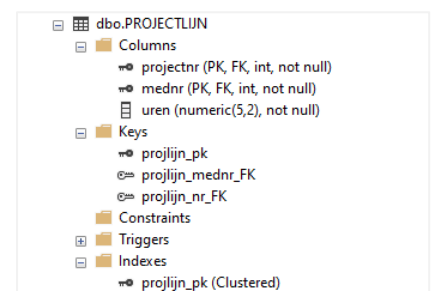
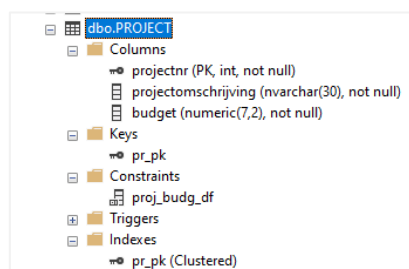
```
-- Tabel PROJECT
create table PROJECT
( projectnr int constraint pr_pk primary key
, projectomschrijving nvarchar(30) not null
, budget numeric (7,2) constraint proj_budg_df default 5000.00 not null
)
```

```
-- Tabel AFDELING
create table AFDELING
( afdeling nchar(5) constraint afdel_pk primary key
, chef nvarchar(20) not null
)
```

```
-- Tabel MEDEWERKER
create table MEDEWERKER
( mednr int constraint medw_pk primary key
, naam nvarchar(20) not null
, constraint medw_nm_chk check(naam = upper(naam))
, afdeling nchar(5) references afdeling
)
```

```
-- Tabel PROJECTLIJN
create table PROJECTLIJN
( projectnr int constraint projlijn_nr_FK references project
, mednr int constraint projlijn_mednr_FK references medewerker
, uren numeric(5,2) not null
, constraint projlijn_pk primary key(projectnr,mednr)
)
```

In de Object Explorer vind je o.a. nu:



Opgelet:

- De datatypes worden nu op een andere manier gedefinieerd. Zie hiervoor punt 7.4.1.
- De primaire sleutel definieer je best met datatype INT. Je hebt min. INT nodig als je op de sleutel een autonummering wilt maken.

Autonummering bij Create van tabel .

In Oracle konden we met CREATE SEQUENCE een autonummering maken. Hier kan je dit dadelijk bij de Create doen.

```
CREATE TABLE Persoon
(
    PersId int IDENTITY(1,1) NOT NULL primary key, -- start met 1 en stapgrootte 1
    Naam nvarchar(25) NOT NULL,
    Voornaam nvarchar(15),
    Leeftijd smallint,
)
```

5.5.1.3 Alter Table

```
-- Kolom verwijderen.
ALTER TABLE afdeling
DROP COLUMN chef

-- Kolom toevoegen.
ALTER TABLE afdeling
ADD chef nvarchar(20) not null

-- Kolom wijzigen.
ALTER TABLE afdeling
ALTER COLUMN chef nvarchar(15)

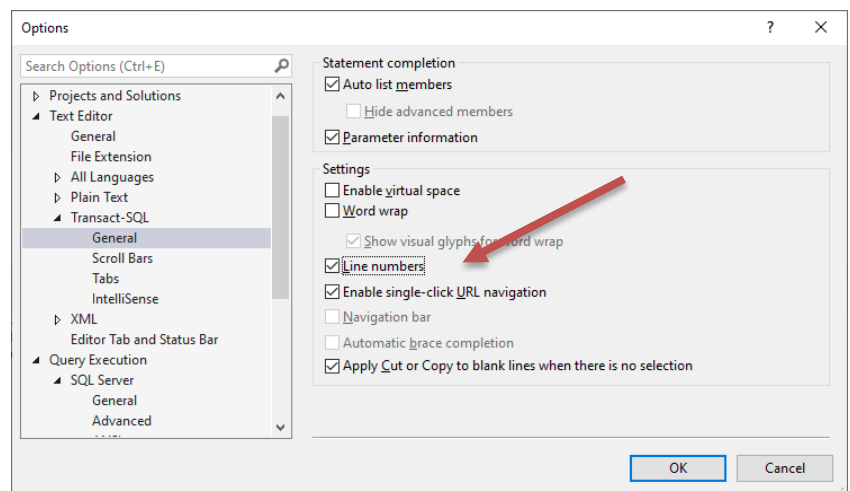
-- Constraint uitschakelen.
ALTER TABLE medewerkers
NOCHECK CONSTRAINT MED_AFD_FK

-- Constraint inschakelen.
ALTER TABLE medewerkers
CHECK CONSTRAINT MED_AFD_FK
```

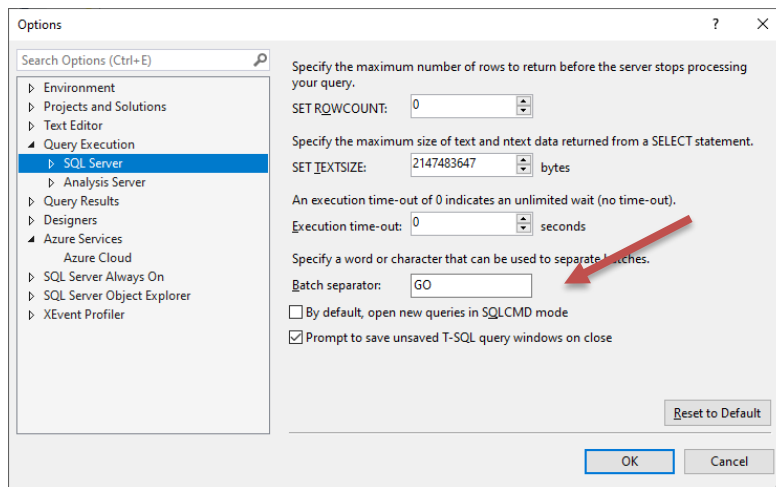
Opgelet:
SQL Server kent geen MODIFY in de Alter Table.

5.5.1.4 Tools options

Het is zeker aangenaam om je lijnnummers in te schakelen want de GUI duidt bij een fout het lijnnummer aan.



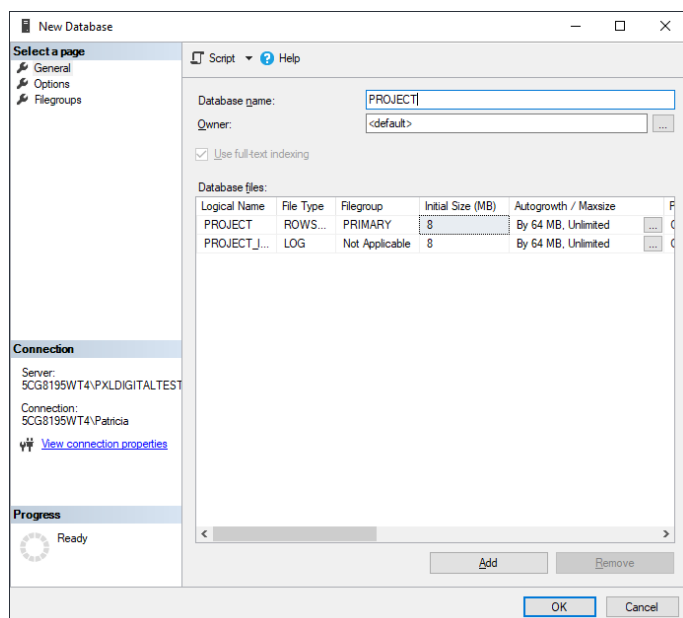
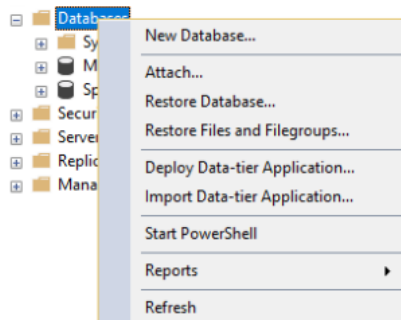
In script zie je vaak de batch separator GO staan. De GO zorgt ervoor dat elk commando dadelijk wordt uitgevoerd in aparte zelfstandige batch-bestand zodat de rest van de commando's foutloos doorlopen kan worden. Je kan in Tool-Options een karakter of iets anders definiëren maar GO is de standaard.



5.5.2 DML met de GUI van SQL Server

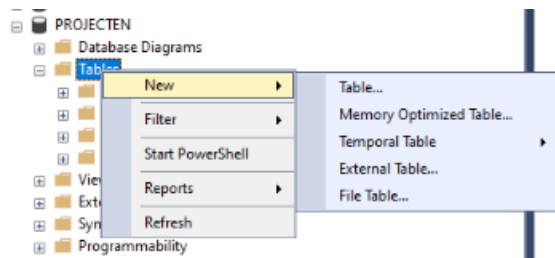
5.5.2.1 Create Database

Klik met je rechter muisknop klikken op de Database node en kies **New Database...** In het volgende venster geef je de naam op. De opties laten we ongemoeid.



5.5.2.2 Create Table

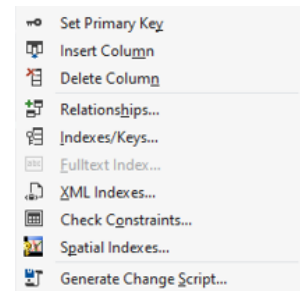
Klik met je rechter muisknop klikken op de Tables node en kies **New - Table...**



Vul alle kolommen in en bij het afsluiten geef je de tabel de gewenste naam. Vb. PROJECT Met de rechter muisknop op de tabel kan je **Design** kiezen en de structuur wijzigen. De primaire sleutel kies je ook met de rechter muisknop (Set Primary Key)

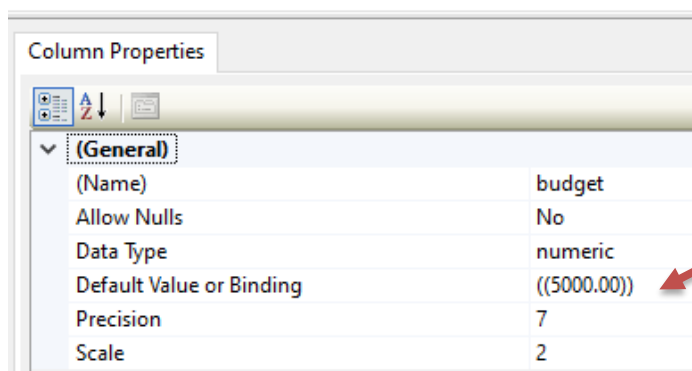
Tabel MEDEWERKERS

	Column Name	Data Type	Allow Nulls
🔑	mednr	int	<input type="checkbox"/>
	naam	nvarchar(20)	<input type="checkbox"/>
	afdeling	nchar(5)	<input checked="" type="checkbox"/>



Tabel PROJECT

	Column Name	Data Type	Allow Nulls
🔑	projectnr	int	<input type="checkbox"/>
	projectomschrijving	nvarchar(30)	<input type="checkbox"/>
▶	budget	numeric(7, 2)	<input type="checkbox"/>
			<input type="checkbox"/>



Met behulp van het venster **Column Properties** kan je bijvoorbeeld ook default settings ingeven. Vb. standaardwaarde voor Budget is 5000,00.

Tabel AFDELING

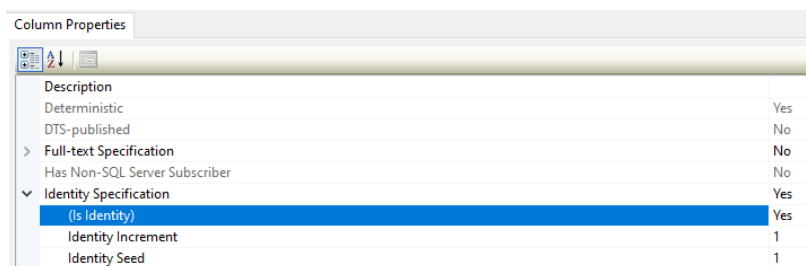
	Column Name	Data Type	Allow Nulls
🔑	afdeling	nchar(5)	<input type="checkbox"/>
	chef	nvarchar(15)	<input checked="" type="checkbox"/>

Tabel PROJECTLIJN

	Column Name	Data Type	Allow Nulls
🔑	projectnr	int	<input type="checkbox"/>
🔑	mednr	int	<input type="checkbox"/>
	uren	numeric(5, 2)	<input type="checkbox"/>

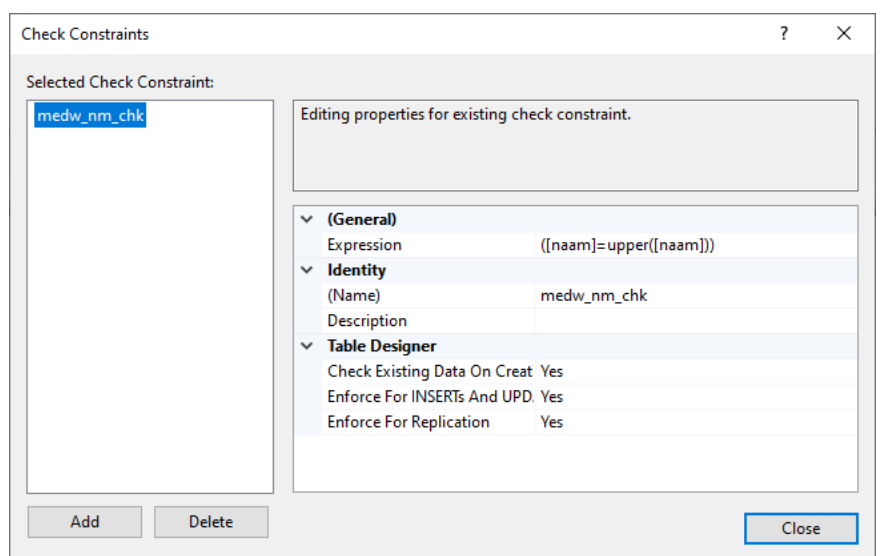
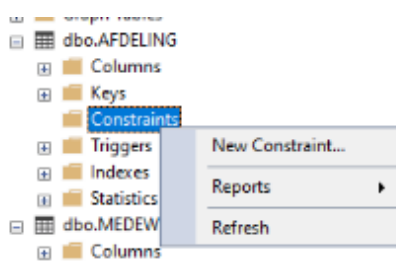
Vermits je maar éénmaal een PK in een tabel kan instellen moet je uiteraard beide kolommen selecteren en dan vervolgens de PK instellen.

Als je gebruik wilt maken van autonummering bij PK dan moet het datatype op zijn minst INT zijn of hoger. Ga naar **Properties** van PK en zet **Identity Specification** op.

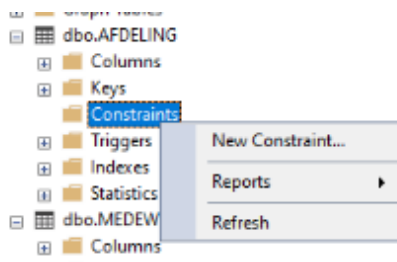


5.5.2.3 Constraints toevoegen of verwijderen.

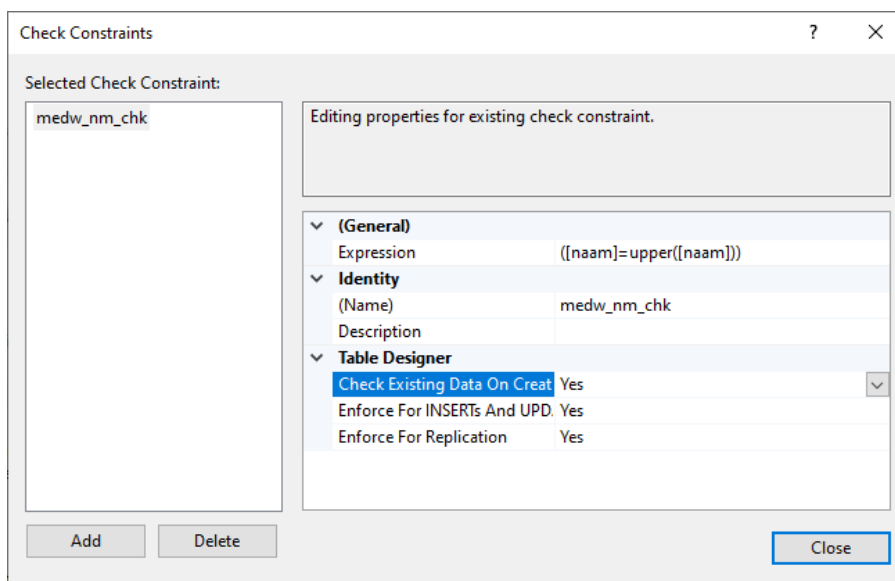
Als je met de rechter muisknop op tabel of kolom klikt dan kan je sleutels, constraints, kolommen, relaties,.... toevoegen of verwijderen. Vb. afdwingen van hoofdletters in NAAM.



Als je met de rechter muisknop op **tabel** klikt dan kan je sleutels, constraints, kolommen, relaties,... toevoegen of verwijderen. Vb. afdwingen van hoofdletters in NAAM.

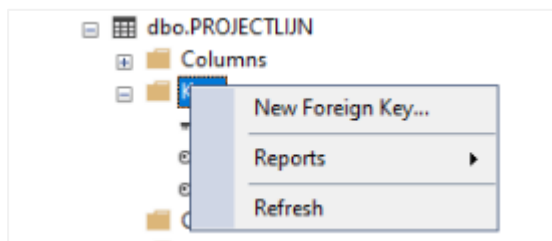


Wens je de constraint te wijzigen klik in de **Object Explorer** op de constraint en kies **Modify** om de constraint te wijzigen.

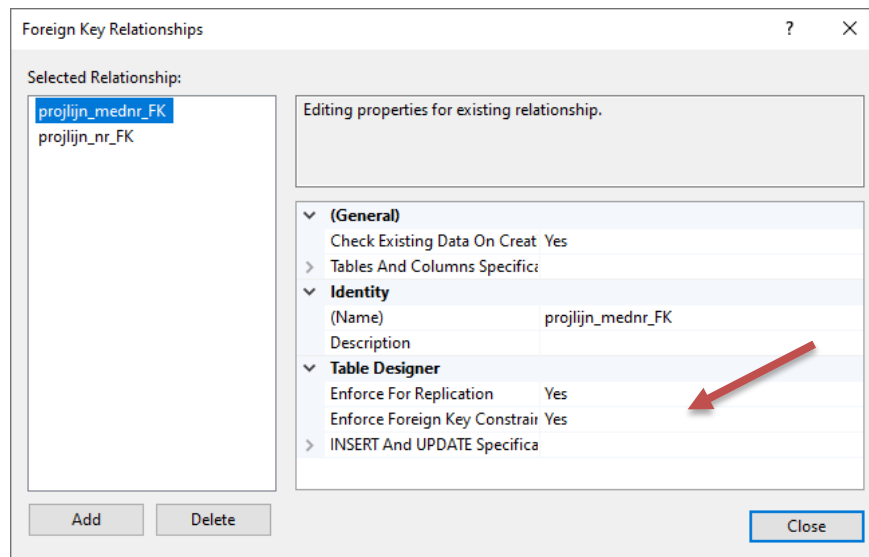


5.5.2.4 Relaties leggen tussen de tabellen

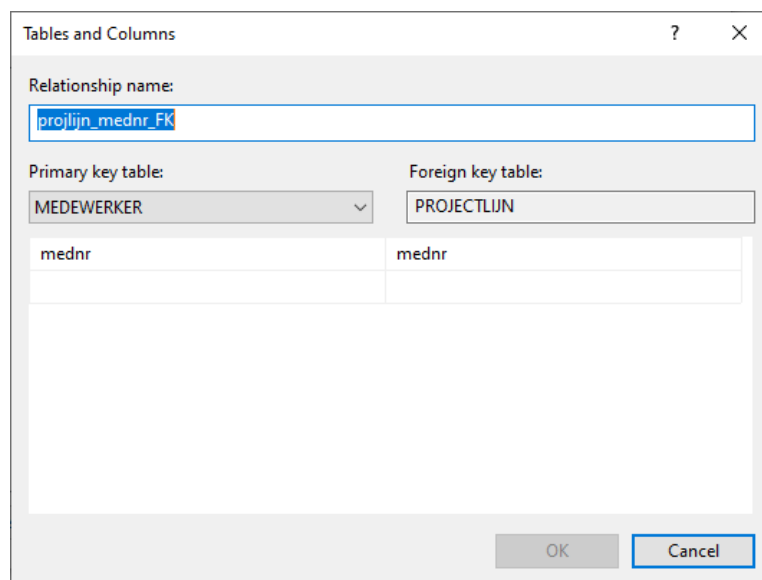
Klik met de rechter muisknop op **Keys** in de **Object Explorer** en kies **New Foreign Key...** om een vreemde sleutel of Foreign Key toe te voegen.



Geef de relatie een naam en definieer de relatie tussen de verschillende tabellen in *Tables And Columns Specifications*.



Je kan met de optie **Enforce Foreign Key Constraint** de constraint in- of uitschakelen. Of later met **Modify**.



5.6 Data Manipulation Language

Hoe kunnen we gegevens gaan invoeren, wijzigen of verwijderen?

5.6.1 DML met de SQL commando's.

5.6.1.1 Gegevens invoeren.

```
INSERT INTO afdelingen VALUES (50, 'DEPARTEMENT COVID', 'HASSELT', 7782)
```

```
INSERT INTO boetes VALUES (4, 10045, GETDATE(), 2500) --- Getdate() voegt huidige datum in.
```

Gegevens toevoegen in beperkte(enkel NOT NULL-kolommen) of willekeurige volgorde:

```
INSERT INTO inschrijvingen VALUES (7788, 'WBA', '2017-02-24', NULL)
```


```
INSERT INTO inschrijvingen (cursist, cursus, begintdatum)
VALUES (7934, 'WBA', '2017-02-24')
```

Merk op dat een NULL-kolom niet opgenomen wordt en dat het attribuut automatisch de waarde NULL krijgt.

Meerdere rijen tegelijk toevoegen.

```
INSERT INTO boetes
VALUES (5, 90020, getdate(), 760.50),
      (6, 10030, '2020-04-01', 1500),
      (7, 90020, getdate(), 280.30)
```

Gegevens invoeren vanuit een extern bestand (csv file in excel of in Notepad)

 BulkINSERTBoetes.csv - Kladblok

Bestand Bewerken Opmaak Beeld Help

1;10020;2016-01-15;5000
2;10045;2016-03-29;13500
3;10020;2016/03/9;4550
4;10045;2020/03/31;2500
5;90020;2020/03/31;760
7;10020;2020/04/25;2570
8;90020;2020/04/01;1320
9;90030;2020/02/12;870.25
10;10010;2020/01/15;120.75

	A	B	C	D	E
1	1	10020	2016-01-15	5000	
2	2	10045	2016-03-29	13500	
3	3	10020	2016-03-09	4550	
4	4	10045	2020-03-31	2500	
5	5	90020	2020-03-31	760	
6	7	10020	2020-04-25	2570	
7	8	90020	2020-04-01	1320	
8	9	90030	2020-02-12	870.25	
9	10	10010	2020-01-15	120.75	
10					
11			Zorg voor het juiste datumformaat!		

```
BULK INSERT boetes
FROM 'C:\SCHOOL\DATA ADVANCED\7 SQL Server\bulkininsertboetes.CSV'
WITH (firstrow = 1, --- Vb. firstrow =2 start op 2de rij (1ste rij = kolomnamen)
      fieldterminator = ';', -- standaard vanuit Excel is het scheidingsteken ;
      rowterminator='\n' -- einde van lijn
      )
```

5.6.1.2 Gegevens wijzigen.

Hierin verschilt het update-commando ook weer niet van Oracle SQL.

```
UPDATE boetes
SET bedrag = 750.50
WHERE boetenr = 5
```

```
UPDATE spelers
SET geslacht = default -- set de standaardwaarde
WHERE naam = 'Slingers'
```

5.6.1.3 Gegevens verwijderen.

Hierin verschilt het delete-commando ook niet echt van Oracle SQL.

```
DELETE boetes
WHERE spelersnr =10020    -- 3 rijen verwijdert

DELETE top(3) boetes    -- eerste 3 rijen worden verwijderd.
```

5.6.1.4 Transactieverwerking

Je kunnen ook in SQL Server DML-transacties ongedaan maken.

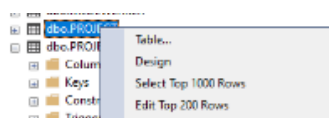
```
BEGIN transaction
    DELETE top(3) boetes
ROLLBACK
```

```
BEGIN transaction
    DELETE top(3) boetes
COMMIT
```

5.6.2 DML met de GUI van SSMS.

5.6.2.1 Gegevens invoeren.

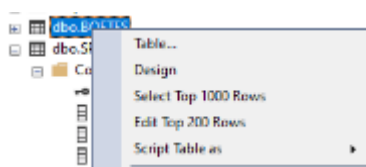
Klik met de rechter muisknop op de tabel en selecteer in het snelmenu **Edit Top 200 Rows**. Je kan in het grid je gegevens invoeren.



	afdeling	chef
▶*	NULL	NULL

5.6.2.2 Gegevens wijzigen.

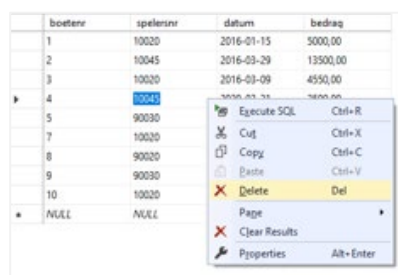
Klik met de rechter muisknop op de tabel en selecteer opnieuw in het snelmenu Edit Top 200 Rows. Je kan in het grid gegevens wijzigen (zie rij 5 waarbij spelersnr gewijzigd wordt).



	boetenr	spelersnr	datum	bedrag
	1	10020	2016-01-15	5000,00
	2	10045	2016-03-29	13500,00
	3	10020	2016-03-09	4550,00
	4	10045	2020-03-31	2500,00
✓	5	90030	2020-03-31	760,00
	7	10020	2020-04-25	2570,00
	8	90020	2020-04-01	1320,00
	9	90030	2020-02-12	870,25
	10	10010	2020-01-15	120,75

5.6.2.3 Gegevens Verwijderen.

Je kan met de rechter muisknop de betreffende rij verwijderen met de optie Delete. De optie Clear Results verwijdert de hele tabel.



5.7 Raadplegen van gegevens in SQL Server

Het raadplegen van gegevens is in vergelijking met de Oracle Server nauwelijks verschillend. De grote verschillen zitten vooral bij de functie dat we in het volgende punt bekijken.

5.7.1 Select

Hieronder opnieuw enkele voorbeelden ter illustratie.

```
--Geef het jaarsalaris (exclusief commissie) van iedere medewerker.
select voorn, naam, 12*maandsal as jaarsalaris
from medewerkers

-- Geef de huidige datum en tijd.
select getdate()

--Geef de werknemers geboren na 1 mei 1985.
select voorn, naam, gbdatum
from medewerkers
where gbdatum >= '1985-05-01'

--Geef de medewerkers van afdeling 10 of afdeling 30.
select voorn, naam, afd
from medewerkers
where afd in (30,10)

--Geef de medewerkers die werken in afdeling 20 t.e.m afdeling 30
select mnr, voorn, naam
from medewerkers
where mnr between 7000 and 7700;

- -Geef alle medewerkers in wiens naam een 'A' voorkomt. WILDCHARTS idem
select voorn, naam
from medewerkers
where naam like '_A%';

--Geef de medewerkers die niet in de afdeling 30 werken.
select voorn, naam, afd
from medewerkers
where NOT afd = 30; --of where afd <>30;

--Geef de medewerkers van afdeling 10 of afdeling 30.
select voorn, naam, afd
from medewerkers
where afd= 30 OR afd=10;

--Geef de naam en jaarsalaris (zonder comm) in dalende volgorde van hun jaarsalaris.
select voorn + ' ' + naam as naam, maandsal *12 as salaris
from medewerkers
order by salaris desc; -- je kan altijd op een kolomalias sorteren

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

```

order by 2;

--Volgorde in stijgende volgorde van commissie en dalend voor naam.
select *
from medewerkers
order by comm asc ,naam desc;      -- of order by comm,naam desc

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

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;

--Geef de medewerkers waarvan de commissie niet van toepassing is.
select voorn, naam, comm
from medewerkers
where comm is NULL;

```

5.7.2 Select Top

Wenst je het aantal rijen te beperken dan kan je de SELECT TOP gebruiken.

```

SELECT TOP (expression) [PERCENT] [WITH TIES]
FROM table_name
ORDER BY column_name;

```



```
--Geeft de gegevens van de eerste 10 inschrijvingen in de tabel.
select top(10) *
from inschrijvingen
```

	cursist	opleiding
1	7934	WBA
2	7934	SQL
3	7902	SQL
4	7902	SQL
5	7902	ORG

```
--Geeft de laatste 5 inschrijvingen.
select top(5) cursist, cursus as opleiding
from INSCHRIJVINGEN
order by cursist desc
```

```
--Geeft de eerste 2 inschrijvingen en geeft overeenkomend alle
rijen die overeenkomen met de laatste rij in de top
```

```
SELECT TOP 2 WITH TIES
    cursist, cursus
FROM inschrijvingen
order by cursist
```

	cursist	cursus
1	7499	CIS
2	7499	SQL
3	7499	WEB
4	7499	WIN

Top 2

Ties met 7499

```
SELECT TOP 3 WITH TIES
    cursist, cursus
FROM inschrijvingen
order by cursist desc
```

	cursist	cursus
1	7934	WBA
2	7934	SQL
3	7902	SQL
4	7902	SQL
5	7902	ORG

Top 3

Ties met 7902

```
-- Geef alle cursussen die cursist 7788 heeft gevolgd.
```

```
SELECT TOP 1 WITH TIES
    cursist, cursus
FROM inschrijvingen
where cursist = 7788
order by cursist
```

	cursist	cursus
1	7788	SQL
2	7788	WBA
3	7788	WEB
4	7788	WEB

```
-- Geef 1 percent van de gegevens. Vermits we hier een heel kleine tabel (28 rijen)
hebben, krijgen we 1 rij (10% van 28 = 0,28) te zien. Wordt steeds afgerond naar boven.
```

```
SELECT TOP 10 PERCENT
    cursist, cursus
FROM inschrijvingen
order by cursist
```

	cursist	cursus
1	7499	CIS
2	7499	SQL
3	7499	WEB

	cursist	cursus	begindatum	evaluatie
1	7499	CIS	2016-09-11	NULL
2	7499	SQL	2015-04-16	4
3	7499	WEB	2015-12-17	2
4	7499	WIN	2016-02-04	5
5	7521	ORG	2015-08-10	4
6	7566	CIS	2016-09-11	NULL
7	7566	WEB	2016-02-05	3
8	7698	SQL	2015-04-16	4
9	7698	SQL	2015-12-17	NULL
10	7698	WEB	2016-02-05	5
11	7782	WEB	2015-12-17	5
12	7788	SQL	2015-10-08	NULL
13	7788	WBA	2017-02-24	NULL
14	7788	WEB	2015-12-17	5
15	7788	WEB	2016-02-05	4
16	7839	SQL	2015-10-08	3
17	7839	WEB	2015-12-17	4
18	7844	ORG	2016-09-27	5
19	7876	CIS	2016-09-11	NULL
20	7876	SQL	2015-04-16	2
21	7876	WEB	2015-12-17	5
22	7900	ORG	2015-08-10	4
23	7900	WIN	2016-02-04	4
24	7902	ORG	2015-08-10	5
25	7902	SQL	2015-10-08	4
26	7902	SQL	2015-12-17	NULL
27	7934	SQL	2015-04-16	5
28	7934	WBA	2017-02-24	NULL

5.7.3 Geavanceerde select

- GECORRELEERDE SUBQUERY

```
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;
```

• JOIN

-- JOIN: Geef de namen van de medewerkers (geboren na 1 jan. 1980) en van de chef.

```
SELECT m.naam as MEDEWERKER_  
,      mm.naam as CHEF_  
FROM   medewerkers m_  
INNER JOIN medewerkers mm_  
ON     m.chef = mm.mnr_  
WHERE  m.gbdatum > '1985-01-01'
```

-- OUTER JOIN: Geef een overzicht van alle medewerkers(ook van de lege afdeling 40) en de locatie waar ze werken.

```
SELECT  a.anr, m.naam, a.locatie  
FROM    medewerkers m  
RIGHT OUTER JOIN afdelingen a  
ON      m.afd = a.anr;
```

-- JOIN: Geef de namen van de medewerkers die een cursus volgen in één van de bedrijfsafdelingen (dus geen cursuslocaties).

```
SELECT distinct m.naam, m.voorn, u.locatie  
FROM   medewerkers m  
JOIN   inschrijvingen i  
ON     m.mnr=i.cursist  
JOIN   uitvoeringen u  
ON     (i.cursus=u.cursus and i.begindatum=u.begindatum)  -- GEEN USING in SQL SERVER!  
JOIN   afdelingen a  
ON     a.locatie = u.locatie;
```

Opmerking:

- LEFT OUTER JOIN (alle rijen van de linker tabel)
- RIGHT OUTER JOIN (alle rijen van de rechter tabel)
- FULL OUTER JOIN (alle rijen van beide tabellen)

• GROUP BY

-- GROUP BY: Geef het aantal medewerkers per afdeling.

```
SELECT m.afd as AFDELING_  
, COUNT(m.mnr) as AANTAL_MEDEWERKERS_  
FROM   medewerkers m_  
GROUP BY      m.afd
```

-- GROEPSFUNCTIES: Wie heeft het laagste inkomen?

```
SELECT  m.naam, m.maandsal  
FROM    medewerkers m  
JOIN    (SELECT MIN(maandsal) laagste  
        FROM medewerkers) mm  
ON      m.maandsal = mm.laagste;
```

-- HAVING: Welke functies in afdeling 10 of 20 hebben een totaal inkomen groter dan 5000?

```
SELECT      functie, SUM(maandsal)  
FROM        medewerkers  
WHERE       afd in (10,20)  
GROUP BY   functie  
HAVING      SUM(maandsal) > 5000  
ORDER BY   SUM (maandsal)
```

Groepsfuncties:

- 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

• VERZAMELINGOPERATOREN

```
-- Geeft alle locaties (vestigingen en cursusplaatsen).
select u.locatie from uitvoeringen u
union
select a.locatie from afdelingen a

--In welke vestigingen worden er ook cursussen gegeven.
select u.locatie from uitvoeringen u
intersect
select a.locatie from afdelingen a

--In welke plaats vinden wel cursussen plaats, maar zijn geen afdelingen gevestigd?
select u.locatie from uitvoeringen u
except
select a.locatie from afdelingen a
```

Opgelet! In SQL Server wordt de minus-operator vervangen door de except-operator.

5.8 SQL Functies

In tegenstelling tot de Oracle Server zijn er wel duidelijke verschillen tussen de Oracle Server en de SQL Server. Hier wordt dan ook dieper op ingegaan.

5.8.1 Alle Functies

	Oracle		SQL Server
1	ADD_MONTHS	Add specified number of months	DATEADD(month, n, datetime)
2	CAST	Convert one built-in data type into another	
3	DECODE	Evaluate a list of conditions	CASE Expression
4	EMPTY_BLOB	Create an empty BLOB value	0x Constant (Empty binary string)
5	EMPTY_CLOB	Create an empty CLOB or NCLOB value	" (Empty string)
6	EXTRACT for Datetime	Extract day, month, year etc from datetime	

7	INITCAP	Capitalize the first letter of each word	User-defined function	
8	INSTR	Find position of substring in string	CHARINDEX	First occurrence only, different parameter order ⚠
9	LAST_DAY	Get last date of month	EOMONTH	Since SQL Server 2012 ⚠
10	LENGTH	Get string length in characters	LEN	CHAR handled differently, excludes trailing spaces ⚠
11	LOWER	Convert string to lowercase	LOWER	
12	LPAD	Left-pad string to the specified length	Expression using REPLICATE, RIGHT and LEFT	
13	MOD	Get the remainder of division of one number by another	% Operator	
14	MONTHS_BETWEEN	Get number of months between two dates	User-defined function	
15	NVL	Replace NULL with expression	ISNULL	
16	REPLACE	Replaces all occurrences of string with another string	REPLACE	
17	SIGN	If value is positive return 1, if negative then -1, if zero then 0	SIGN	
18	SUBSTR	Return a substring from string	SUBSTRING	Negative start position is not allowed, length must be specified ⚠
19	TO_CHAR for Datetime	Convert datetime to string	CONVERT(VARCHAR(n), datetime, style)	
20	TO_DATE	Convert string to datetime	CONVERT(DATETIME, string, style)	
21	TRANSLATE	One-to-one single-character substitution	Expressions using REPLACE or User-defined function	
22	TRIM	Trim leading or trailing characters	LTRIM and RTRIM	
23	TRUNC for Datetime	Truncate datetime	Expressions using CONVERT	

5.8.2 Algemene functies

NULLIF(a,b)	geeft NULL als a=b, anders a
COALESCE(a,b,...)	retourneert het eerste argument dat niet NULL is
CASE... WHEN... THEN	if-structuur
ISNULL	retourneert 2de parameter als null is

```
MS SQL> SELECT naam, comm, (maandsal*12)
+ COALESCE(comm,0) as salaris
FROM medewerkers
CASPER NULL 21600 ...

MS SQL> SELECT naam, comm, (maandsal*12)
+ ISNULL(comm,0) as salaris
FROM medewerkers
CASPER NULL 21600 ...

MS SQL> SELECT NULLIF('SQL Server', 'SQL SERVER')
NULL

MS SQL> SELECT naam, comm, ISNULL(comm, maandsal)
FROM medewerkers
CASPER NULL 1800
MARTENS 3400 3400...
```

Sorteer de medewerkers op basis van de functie.

```
MS SQL> SELECT naam, functie
FROM medewerkers
ORDER BY CASE functie
WHEN 'DIRECTEUR' THEN 1
WHEN 'MANAGER' THEN 2
WHEN 'VERKOPER' THEN 3
WHEN 'TRAINER' THEN 4
ELSE 5
END
```

5.8.3 Rekenkundige Functions

	Oracle	SQL Server
1 MOD	Get the remainder of division of one number by another	% Operator
2 SIGN	If value is positive return 1, if negative then -1, if zero then 0	SIGN

De belangrijkste functies om met cijfers te werken zijn:

ROUND(n,m)	rondt n af op m decimale posities
CEILING(n)	rondt n naar boven af op een geheel getal
FLOOR(n)	rondt n naar beneden af op een geheel getal
ABS(n)	de absolute waarde van n
SIGN(n)	-1, 0 of 1 als n negatief, nul of positief is
SQRT(n)	vierkantswortel uit n
POWER(n,m)	n tot de m -de macht

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

Voorbeelden

ORACLE> select round(345.678) from dual	346
MS SQL > SELECT ROUND (345.678,0)	346
ORACLE> select ceil(345.678) from dual	346
MS SQL > SELECT CEILING (345.678)	346
ORACLE> select floor(345.678) from dual	345
MS SQL > SELECT FLOOR (345.678)	345
ORACLE> select round(345.678, 2) from dual	345.68
MS SQL > SELECT ROUND (345.678, 2)	345,680
ORACLE> select round(345.678, -1) from dual	350 -2
MS SQL > SELECT ROUND (345.678, -1)	350,000
ORACLE> select trunc(345.678, 2) from dual	345.67
MS SQL > SELECT ROUND (345.678, 2,1) 3de parameter <>0 is afkappen!	345,670
ORACLE> select abs(-123) , abs(0), abs(456) from dual	123/0/456
MS SQL > SELECT ABS (-123) , ABS (0), ABS (456)	123/0/456
ORACLE> select sign(-13) , sign(0), sign(456) from dual	-1/0/1
MS SQL > SELECT SIGN (-13) , SIGN (0), SIGN (456)	-1/0/1
ORACLE> select sqrt(16), sqrt(8), sqrt(4) from dual	4/2.8284271/2
MS SQL > SELECT SQRT (16), SQRT (8), SQRT (4)	4/2,82842712474619/2
ORACLE> select power(2, 3), power(-2,3) from dual	2 ³ =8 -2 ³ =-8
MS SQL > SELECT POWER (2, 3), POWER (-2,3)	8/-8
ORACLE> select mod(8,3), mod(13,0) from dual	2/13
MS SQL > SELECT 8%3, 13%0	2/Divide by zero error encountered.

5.8.4 String Functions

Oracle			SQL Server	
1	INITCAP	Capitalize the first letter of each word	User-defined function	
2	INSTR	Find position of substring in string	CHARINDEX	First occurrence only, different parameter order ⚠
3	LENGTH	Get string length in characters	LEN	CHAR handled differently, excludes trailing spaces ⚠
4	LOWER	Convert string to lowercase	LOWER	
5	LPAD	Left-pad string to the specified length	Expression using REPLICATE, RIGHT and LEFT	
6	REPLACE	Replaces all occurrences of string with another string	REPLACE	
7	SUBSTR	Return a substring from string	SUBSTRING	Negative start position is not allowed, length must be specified ⚠
8	TO_CHAR for Datetime	Convert datetime to string	CONVERT	
9	TRANSLATE	One-to-one single-character substitution	Expressions using REPLACE or User-defined function	
10	TRIM	Trim leading or trailing characters	LTRIM(RTRIM(<i>string</i>))	
11	CHR(<i>code</i>)	Get character from ASCII code	CHAR(<i>code</i>)	

De voornaamste tekstfuncties van Oracle zijn:

LEN(<i>t</i>)	aantal karakters (lengte) van <i>t</i>
ASCII(<i>t</i>)	ascii-waarde eerste karakter van <i>t</i>
CHAR(<i>n</i>)	karakter met ascii-waarde <i>n</i>
CHARINDEX(<i>t</i> , <i>k</i>)	positie eerste voorkomen van <i>k</i> in <i>t</i>
UPPER(<i>t</i>)	<i>t</i> in hoofdletters
LEFT(<i>t</i> , <i>n</i>)	verwijdert links een aantal karakters in <i>t</i>
LOWER(<i>t</i>)	<i>t</i> in kleine letters
LTRIM(<i>t</i> , <i>k</i>)	verwijdert links blanco's
RTRIM(<i>t</i> , <i>k</i>)	verwijdert rechts blanco's
LPAD(<i>t</i> , <i>n</i>)	vult <i>t</i> links uit met spaties tot lengte <i>n</i>
RPAD(<i>t</i> , <i>n</i>)	vult <i>t</i> rechts aan met spaties tot lengte <i>n</i>
SUBSTRING(<i>t</i> , <i>n</i>)	geeft deel van <i>t</i> vanaf positie <i>n</i> tot het einde
REPLACE(<i>t</i> , <i>v</i>)	<i>verwijdert</i> uit <i>t</i> elk voorkomen van <i>v</i> (woorden)
REPLICATE(<i>k</i> , <i>n</i>)	dupliceert <i>k</i> met opgegeven aantal <i>n</i>
RIGTH(<i>t</i> , <i>n</i>)	verwijdert rechts een aantal karakters

Voorbeelden

ORACLE> select code, upper(omschrijving), lower(type) from cursussen	INTRODUCTIE...	alg
MS SQL > SELECT CODE, UPPER (omschrijving), LOWER (type) FROM CURSUSSEN	idem	
ORACLE> select anr, naam, initcap(locatie) from afdelingen order by length(naam)	30	VERKOOP Genk
MS SQL > SELECT anr, naam, LEFT (locatie,1)+ LOWER (SUBSTRING (locatie,2, LEN (locatie))) FROM afdelingen order by LEN (naam)		
ORACLE> select * from medewerkers where lower(funcitie) = 'trainer'	SWINNEN	TRAINER ...
MS SQL > SELECT * FROM medewerkers WHERE LOWER (funcitie) = 'trainer'	idem	
ORACLE> select ascii('a'), ascii('z'), chr(77) from dual	97/122/M	
MS SQL > SELECT ASCII ('a'), ASCII ('z'), CHAR (77)	idem	
ORACLE> select substr(naam,4) , substr (naam,4,3) from afdelingen	fdkantoor	fdk
MS SQL > SELECT SUBSTRING (naam,4, LEN (naam)) , SUBSTRING (naam,4,3) FROM afdelingen	idem	
ORACLE> select naam, instr (naam,'A') , instr (naam,'A',3) , instr (naam,'A',1,2) from medewerkers	ALLARD JACOBS DE COOMAN	1/4/4 2/0/0 8/8/0
MS SQL > SELECT naam, CHARINDEX ('A',naam) , CHARINDEX ('A', naam,3) , CHARINDEX ('A', naam, CHARINDEX ('A' ,naam)+1) FROM medewerkers	idem	
ORACLE> select ltrim(naam,'SDAER') , rtrim(naam,'SDAER') from medewerkers	CASPERS ALLARD N RUYTER	CASP ALL RUYT
Enkel voor verwijderen van blanco's.		
MS SQL > SELECT ' ' + RTRIM ('AB ') + ' '	AB	
ORACLE> select lpad (naam,8,'@') , rpad (naam,12,'=') from medewerkers	@@JACOBS JACOBS=====	
Bestaat niet bij SQL Server.		


```

MS SQL> SELECT RIGHT(REPLICATE('@',10)+naam,10)                idem
        , LEFT(naam + REPLICATE('=',10),10)
        FROM medewerkers

ORACLE> select translate(code,'AESOL', '12345')                SQL    INTRODUCTIE SQL
        ,      replace(omschrijving,'SQL',' Visual C#')        3Q5    INTRODUCTIE Visual C#
        from cursussen

MS SQL> SELECT REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(code,'A','1'),'E','2'),'S','3')
        , 'O','4'),'L','5')
        ,      REPLACE(omschrijving,'SQL',' Visual C#')
        FROM cursussen                idem

```

5.8.5 Datetime functies

	Oracle		SQL Server
1	ADD_MONTHS	Add specified number of months	DATEADD
2	EXTRACT for Datetime	Extract day, month, year etc from datetime	
3	LAST_DAY	Get last date of month	EOMONTH
4	MONTHS_BETWEEN	Get number of months between two dates	
5	TO_CHAR for Datetime	Convert datetime to string	CONVERT
6	TO_DATE	Convert string to datetime	CONVERT
7	TRUNC for Datetime	Truncate datetime	Expressions using CONVERT
	EXTRACT(YEAR FROM <i>datetime</i>)	Get the year from <i>datetime</i>	YEAR(<i>datetime</i>)
	EXTRACT(MONTH FROM <i>datetime</i>)	Get the month from <i>datetime</i>	MONTH(<i>datetime</i>)
	EXTRACT(DAY FROM <i>datetime</i>)	Get the day from <i>datetime</i>	DAY(<i>datetime</i>)

Gebruik de CONVERT() om een string om te zetten naar een datumformaat.

	Oracle	SQL Server
1	TO_DATE('2020-07-18', 'YYYY-MM-DD')	CONVERT(DATETIME, '2020-07-18')
2	TO_DATE('2020/07/18', 'YYYY/MM/DD')	CONVERT(DATETIME, '2020/07/18')
3	TO_DATE('2020-07-18 13:27:18', 'YYYY-MM-DD HH24:MI:SS')	CONVERT(DATETIME, '2020-07-18 13:27:18')
4	TO_DATE('07/18/2020 13:27:18', 'MM/DD/YYYY HH24:MI:SS')	CONVERT(DATETIME, '07/18/2020 13:27:18')
5	TO_DATE('17-FEB-2020', 'DD-MON-YYYY')	CONVERT(DATETIME, '17-FEB-2020')

De belangrijkste datumfuncties in SQL Server zijn:

DATEADD(interval,n,d)	datum <i>d</i> plus <i>n</i> maanden/dagen/weken/....
DATEDIFF(interval,d1,d2)	maanden/dagen/weken/... verschil tussen <i>d1</i> en <i>d2</i>
DATEPART(interval,d)	extraheert component (interval) uit expressie <i>d</i>
EOMONTH(d)	laatste dag van de maand waarin <i>d</i> valt

Tijd/datum interval voor DATEADD/DATEDIFF/DATEPART:

- year, yyyy, yy = jaar
- quarter, qq, q = kwartaal
- month, mm, m = maand
- dayofyear = dag van het jaar
- day, dy, y = dag
- week, ww, wk = week
- weekday, dw, w = weekdag
- hour, hh = uur
- minute, mi, n = minuut
- second, ss, s = seconde
- millisecond, ms = milliseconde

Voorbeelden

ORACLE> select naam,months_between(sysdate,gbdatum) from medewerkers	CLERCKX	408,864208
MS SQL > SELECT naam, DATEDIFF(MONTH, gbdatum, GETDATE()) FROM medewerkers	DE KONING CLERCKX DE KONING	565,864208... 409 566
ORACLE> select gbdatum, add_months(gbdatum,13) 1985 , add_months(gbdatum,-3) from medewerkers	17-12-1985	17-12-1987 17-09-1985
MS SQL > SELECT GBDATUM, DATEADD(MONTH,13,gbdatum) , DATEADD(MONTH,-3,gbdatum) FROM MEDEWERKER	1985-12-17 00:00:00.000 1987-01-17 00:00:00.000 1985-09-17 00:00:00.000	
ORACLE> select add_months(date '2015-01-29',1) , add_months(date'2016-01-,29',1) from dual	28-FEB-2015 29-FEB-2016	Schrikkeljaar
MS SQL > SELECT DATEADD(MONTH,1, '2015-01-29') , DATEADD(MONTH,1, '2016-01-29')	2015-02-28 00:00:00.000 2016-02-29 00:00:00.000	
ORACLE> select next_day(sysdate,'sat') , last_day (sysdate) from dual	18-1-2020 31-1-2020	
MS SQL > SELECT EOMONTH(sysdatetime()) Equivalent Next_Day bestaat niet.	2020-01-31	

```
ORACLE> select extract(year from gbdatum)          1972
,          extract(month from gbdatum)            11
,          extract(day from gbdatum)              17
from medewerkers
where naam = 'DE KONING'
```

```
MS SQL > SELECT DATEPART(year,gbdatum)            idem
,          DATEPART(month,gbdatum)
,          DATEPART(day,gbdatum)
FROM medewerkers
WHERE naam = 'DE KONING'
```

```
MS SQL > SELECT DATEPART(YY, getdate()) AS Year,
DATEPART(QQ, getdate()) AS Quarter,
DATEPART(WK, getdate()) AS Week,
DATEPART(DY, getdate()) AS dayofYear,
DATEPART(MM, getdate()) AS Month,
DATEPART(DD, getdate()) AS Date,
DATEPART(hour, getdate()) AS Hour,
DATEPART(minute, getdate()) AS Minute,
DATEPART(second, getdate()) AS Second,
DATEPART(millisecond, getdate()) AS Millisecond,
DATEPART(microsecond, getdate()) AS Microsecond,
DATEPART(nanosecond, getdate()) AS Nanosecond;
```

Year	Quarter	Week	dayofYear	Month	Date	Hour	Minute	Second	Millisecond	Microsecond	Nanosecond
2020	1	3	12	1	12	20	10	16	703	703000	703000000

5.8.6 Conversie en Format Functies

	Oracle	SQL Server
1	CAST	Convert one built-in data type into another
2	TO_CHAR for Datetime	Convert datetime to string
3	TO_DATE	Convert string to datetime

De belangrijkste conversiefuncties van SQL Server zijn:

	Oracle	SQL Server
Syntax	TO_CHAR(datetime, format)	CONVERT(VARCHAR(n), datetime, style)
		CAST(datetime as VARCHAR(n))
Default Format and Style	Specified by NLS_DATE_FORMAT	Mon DD YYYY HH12:MI

Conversiefunctie CAST()

The CAST() function converteert een waarde (elk soort type) naar een bepaald gegevenstype .

CAST(expression AS datatype [length])

Datatype Kan het volgende zijn: bigint, int, smallint, tinyint, bit, decimal, numeric, money, smallmoney, float, real, datetime, smalldatetime, char, varchar, text, nchar, nvarchar, ntext, binary, varbinary, or image

Length Optioneel, de lengte van het datatype voor char, varchar, nchar, nvarchar, binary and varbinary.

Voorbeelden

SELECT CAST(25.650 AS varchar)	25.650
SELECT CAST('2020-08-25' AS date)	2020-08-25
SELECT CAST(25.650 AS int)	25 (altijd truncate)
SELECT CAST(25.650 AS numeric)	26 (vanaf 0.50 naar boven afgerond)

Conversiefunctie CONVERT

	Oracle TO_CHAR Format	SQL Server CONVERT Data Type	SQL Server CONVERT Style
1	YYYY-MM-DD	VARCHAR(10)	20, 21, 120, 121, 126 and 127
2	YYYY-MM-DD HH24:MI:SS	VARCHAR(19)	20, 21, 120 and 121
3	YYYYMMDD	VARCHAR(8)	112
4	YYYYMM	VARCHAR(6)	112
5	YYMM	VARCHAR(4)	12
6	YYYY	VARCHAR(4)	112
7	YYYY/MM/DD	VARCHAR(10)	111
8	HH24:MI	VARCHAR(5)	8, 108, 14 and 114
9	HH24:MI:SS	VARCHAR(8)	8, 108, 14 and 114

Voorbeelden conversie

	Oracle	SQL Server
1	TO_CHAR(SYSDATE, 'YYYY-MM-DD')	CONVERT(VARCHAR(10), GETDATE(), 20)
2	TO_CHAR(SYSDATE, 'YYYY-MM-DD HH24:MI:SS')	CONVERT(VARCHAR(19), GETDATE(), 20)
3	TO_CHAR(SYSDATE, 'YYYYMMDD')	CONVERT(VARCHAR(8), GETDATE(), 112)

4	TO_CHAR(SYSDATE, 'YYYYMM')	CONVERT(VARCHAR(6), GETDATE(), 112)
5	TO_CHAR(SYSDATE, 'YYMM')	CONVERT(VARCHAR(4), GETDATE(), 12)
6	TO_CHAR(SYSDATE, 'YYYY')	CONVERT(VARCHAR(4), GETDATE(), 112)
		CONVERT(VARCHAR, DATEPART(YEAR, GETDATE()))
		CONVERT(VARCHAR, YEAR(GETDATE()))
7	TO_CHAR(SYSDATE, 'YYYY/MM/DD')	CONVERT(VARCHAR(10), GETDATE(), 111)
8	TO_CHAR(SYSDATE, 'HH24:MI')	CONVERT(VARCHAR(5), GETDATE(), 8)
9	TO_CHAR(SYSDATE, 'HH24:MI:SS')	CONVERT(VARCHAR(8), GETDATE(), 8)

Voorbeelden

Je kan zoveel de CAST als de CONVERT gebruiken om te converteren. De CAST gebruik je best wanneer je standaardwaarde wenst te gebruiken. Met de CONVERT heb je ook opmaakmogelijkheden bij de verschillende datumformaten.

```
MS SQL> SELECT CAST('2020-01-13' AS date) 2020-01-13

MS SQL> SELECT CONVERT(VARCHAR(10), GETDATE(), 120) 2020-01-13
MS SQL> SELECT CONVERT(VARCHAR(19), GETDATE(), 120) 2020-01-13 12:41:09
MS SQL> SELECT CONVERT(VARCHAR(8), GETDATE(), 112) 20200113
MS SQL> SELECT CONVERT(VARCHAR(4), GETDATE(), 12) 202001
MS SQL> SELECT CONVERT(VARCHAR(4), GETDATE(), 112) 2020
MS SQL> SELECT CONVERT(VARCHAR(10), GETDATE(), 111) 2020/01/13
MS SQL> SELECT CONVERT(VARCHAR(5), GETDATE(), 8) 12:51
MS SQL> SELECT CONVERT(VARCHAR(8), GETDATE(), 8) 12:51:48

ORACLE> select sysdate 13-01-2020
2 , to_char(sysdate,'hh24:mi:ss') 13:02:09
3 , to_char(to_date('26-03-2016','dd-mm-yyyy'),' "valt op" Day') valt op Zaterdag
4 from dual;

MS SQL> SELECT GETDATE() 2020-01-13 13:02:09.797
, CONVERT(varchar(8), GETDATE(), 8) 13:02:09
Met de CASE kan je eventueel de dagen voorstellen.

ORACLE> select to_char(sysdate,'yyyy') 2016
2 , to_char(sysdate,'yy') 16
3 , to_char(sysdate,'y') 6
4 , to_char(sysdate,'year') TWENTY SIXTEEN
5 from dual;

MS SQL> select DATEPART(year, cast('2020-05-15' as date)) 2020
, DATEPART(YYYY, cast('2020-05-15' as date)) 2020
, DATEPART(yy, cast('2020-05-15' as date)) 2020
, DATEPART(Y, cast('2020-05-15' as date)) 136

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

```

MS SQL > select DATEPART(QUARTER, cast('2020-05-15' as date))      2
ORACLE> select to_char(sysdate,'mm')                                04
      2 , to_char(sysdate,'month')                                april
      3 , to_char(sysdate,'mon')                                apr
      4 from dual;
MS SQL > select DATEPART(month, cast('2020-05-15' as date))      5
      , DATEPART(mm, cast('2020-05-15' as date))                5
      , DATEPART(M, cast('2020-05-15' as date))                  5

ORACLE> select to_char(date'2016-01-13','ddd')                    013
      2 , to_char(date'2016-01-13','dd')                        13
      3 , to_char(date'2016-01-13','d')                          4
      4 , to_char(date'2016-01-13','day')                        wednesday
      5 , to_char(date'2016-01-13','Dy dy')                      Wed wed
      6 from dual;
MS SQL > select DATEPART(DAY, cast('2020-05-15' as date))        15
      , DATEPART(DD, cast('2020-05-15' as date))                15
      , DATEPART(DY, cast('2020-05-15' as date))                136
      , DATEPART(DAYOFYEAR, cast('2020-05-15' as date))          136

ORACLE> select to_char(sysdate,'hh:mi:ss AM')                     01:19:15 PM
      2 , to_char(sysdate,'hh24:mi:ss')                          13:19:15
      3 , to_char(sysdate,'sssss')                               47955
      4 from dual;
MS SQL > select DATEPART(hour, getdate())                         14
      , DATEPART(minute, getdate())                              42
      , DATEPART(SECOND, getdate())                             16
      , CONVERT(varchar(8), getdate(),8)                        14:42:16

```

Vraag de weekdag waarop je geboren bent.

```

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

```

```

c
MS SQL > select case (datepart(WEEKDAY, cast('1995-01-13' as date)))
      when 1 then 'zondag'
      when 2 then 'maandag'
      when 3 then 'dinsdag'
      when 4 then 'woensdag'
      when 5 then 'donderdag'
      when 6 then 'vrijdag'
      when 7 then 'zaterdag'
      end as geboortedag                                          zondag

```

5.9 Toepassingen

5.9.1 Data Definition Language

- Maak een script om de database van Kinepolis te maken (zie gedetailleerde opgave 3.12.3).
- Wijzig de script CRECASE MEDEWERKERS zodat je de database Medewerkers in SQL Sever kan installeren.
- Maak een script om de database van Garage Grb. Valkenborg te installeren en werk ook uit met de GUI van SQL Server (zie gedetailleerde opgave 3.12.2).
- Maak een script om de database van Teams te installeren en werk ook uit met de GUI van SQL Server (zie gedetailleerde opgave 3.12.1).

5.9.2 Data Manipulation Language

5.9.2.1 Database Teams

- Verwijder de spelers met spelersnr tussen 10000 en 10030.
- Wijzig de naam van een speler uit de spelers tabel.
- Voer de gegevens in tabel Boetes: boetenr. 4, € 2 500,00 boete op 15/11/2017 voor Pauwels. De naam wordt in de tabel Spelers eveneens geselecteerd.

5.9.3 Het medewerkersvoorbeeld

Onderstaande wijzigingen mogen NIET definitief doorgevoerd worden, dus werk met een Transactie-blok (Rollblack)

- 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 MEDEWERKERS 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^{ste} dag van 1980, om 3:30 's nachts.

Onderstaande wijzigingen mogen definitief doorgevoerd worden. Je werkt hier het best met de GUI van SQL Server.

Voeg in tabel HISTORIEK 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. Aangenomen als manager voor hoofdkantoor.
7782	2015	15/10/2015		10	3450	
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