SQL & Oracle Databases

# Inleidende begrippen

Een **informatiesysteem** is een vereenvoudigde**,** representatieve voorstelling van de werkelijkheid binnen een organisatie. Deze wordt gebruikt om op het juiste moment over de juiste informatie te beschikken. Dit hoeft niet noodzakelijk geautomatiseerd te zijn (bv. Klantenfiches, Excel, …).

|  |  |
| --- | --- |
| **Bestandsgericht**: alle informatie uit bv. Excel, fiches, contracten, … worden in bestanden geplaatst dewelke door programma’s verwerkt worden | **Databasegericht**: alle informatie wordt in een database beheerd dewelke door meerde afdelingen simultaan gebruikt kan worden, verdeeld over entiteiten (= objecten van de DB) |
| * **Redundantie**: meerdere afdeliungen van een organisatie hebben dezelfde gegevens in eigen bestanden met gevoor voor inconsistentie, inefficiënt geheugengebruik en updates zijn omslachtig. | * **Shared data**: alle applicaties kunnen gebruik maken van dezelfde gegevens uit de DB. Simultaan uitlezen, voor aanpassingen wel rekening houden met elkaar. |
| * **Gescheiden geïsoleerde data**: vraagt een complexe programmatie om verschillende bestanden in een programma te synchroniseren. | * **Geïntegreerd en samenhangend**: alle gegevens zitten gecentraliseerd en gerelateerd. |
| * **Incompatibele bestanden**: mogelijks geprogrammeerd met andere syntax (andere bestandsstructuur). | * **Geringe redundatie**: gegevens zitten niet meer dubbel gestockeerd. |
|  | * **Data onafhankelijkheid**: de gegevens worden in de DB beschreven en niet in een applicatie. Een structuurwijziging van een DB object vraagt geen programmatorische aanpassing. |
|  | * **Integriteit bewaking**: weinig inconsistenties, constraints (automatisch ingebouwde controles) en triggers. |
|  | * **Veiligheidsbewaking**: makkelijker te beveiligen door centralisatie. |
|  | * **Vereenvoudigen** **van** **standaard** **oplegging**: omwille van de centralisatie is het makkelijker om afspraken te maken rond bv. Benamingen. |

** Definitie**: Een **database** is een geïntegreerde verzameling gegevens die eventueel door meerdere gebruikers simultaan kan gemanipuleerd worden en die voldoet aan de informatie behoeften van een organisatie. Deze bevat gegevens en metagegevens (zelfs geen documentatie nodig, elke wijziging wordt automatisch bijgehouden.

Een **Database Management Systeem (DBMS)** is de software dewelke nodig om de database te beheren en te gebruiken.

* ***Gebruiken****: opvragen van gegevens (queries), inhoud wijzigen, onderhoud.*
* ***Beheren****: structuur definiëren en wijzigen, beveiliging, transactiebeheer.*

### Gebruikersprofielen

|  |
| --- |
| **Data administrator** |
| **Database administrator** |
| **Toepassingsontwikkelaar** |
| **Gebruiker** |

|  |  |
| --- | --- |
|  | ***Hiërarchisch model*** |
| ***Netwerk model*** |
| ***Relationeel model*** |

## Het relationeel model

Ontwikkeld door E.F. Codd, gebaseerd op relationele algebra dewelke bestaat uit ***relaties***, ***operatoren*** ***om*** ***met*** ***relaties*** ***te*** ***werken*** en ***integriteitsregels*** ***om*** ***consistentie*** ***te*** ***garanderen***. Een relationele DB wordt gebouwd volgens het relationele model.

De relationele DB is een verzameling relaties/tabellen waarbij verbanden gelegd worden met vreemte sleutels (FK).

### Terminologie

|  |  |
| --- | --- |
| **Tabel** |  |
| **Kolom of attribuut** |  |
| **Rij of tuple** |  |
| **Attribuutwaarde** |  |
| **Populatie** | *Verzameling van concrete waarden van een attribuut* |
| **Domein** | *Bij elk attribuut hoort een verzameling van mogelijke waarden* |
| **NULL waarde** | *Ontbrekende of onbekende waarden* |

### Eigenschappen

|  |  |
| --- | --- |
| **Attribuut is atomair** | *Op het snijpunt van rij en kolom staat één waarde* |
| **Elke rij binnen 1 tabel is uniek** | *Geen dubbele rijen* |
| **Rijen hebben geen specifieke volgorde** |  |
| **Kolommen zijn uniek** | *Tevens geen specifiek volgorde* |

### Sleutels

|  |  |
| --- | --- |
| **Primair (PK)** | *Attribuut of combinatie van attributen dewelke een rij uit een tabel op een unieke wijze gaat identificeren* |
| **Alternatief** | *Meerdere attributen of combinaties van attributen zijn mogelijke primaire sleutels, deze worden alternatieve of kandidaatsleutels genoemd* |
| **Vreemde (FK)** | *Verband tussen twee tabellen, verwijzing naar een primaire sleutel of unieke attribuutcombinatie van een andere tabel.* |

### Integriteitsregels

|  |  |
| --- | --- |
| **Key constraint** | *De primaire sleutel moet uniek zijn en blijven!* |
| **Entity integrity contstraint** | *De primaire sleutel moet een geldige waarde hebben (!= NULL)* |
| **Referential integrity constraint** | *De populatie van de verwijssleutel moet een deelverzameling zijn van de populatie van de overeenkomstige primaire sleutel.* |

 Deze integretiteitsregels bouwen we in het **CREATE TABLE** statement zodat het DBMS automatische controle doet op deze regels.

### Types bewerkingen

|  |  |
| --- | --- |
| **UNIE** | *Voegt rijen van 2 relaties samen tot 1 relatie (mits dat ze dezelfde attributenlijst hebben!)* |
| **INTERSECTIE** | *Gemeenschappelijke rijen nemen* |
| **VERSCHIL** | *Het verschil van 2 relaties is een 3de relatie dewelke alle rijen uit de 1ste bevat dewelke niet in de 2de voorkomen* |
| **PRODUCT** | *Cartesiaans product; resulterende relatie bevat alle mogelijke rijen gevormd door de samenvoeging van een rij uit relatie 1 en een rije uit relatie 2.* |
|  |  |
| **PROJECTIE** | *Vormt een nieuwe relatie door slechts een deel van de attributen van de input over te houden, bij ontstaan van dubbele rijen worden deze uit het resultaat verwijdert.* |
|  |  |
| **SELECTIE** | *Deelverzameling van een relaties op basis van een voorwaarde.* |
| **JOIN** | *Combinatie van 2 relaties d.m.v. Cartesiaans product en selectie, beide relaties moeten een attribuut bevatten waarop vergelijking mogelijk is.* |
|  |  |

## SQL

**S**tructured **Q**uery **L**anguage, de taal om met een relationeel DBMS te communiceren.

### Eigenschappen

|  |  |
| --- | --- |
| **Declaratieve taal** | *WAT en niet HOE (⬄ procedurale taal)* |
| **Interactief gebruik** | *Instructie op het scherm, immediate response* |
| **Embedded gebruik** | *Ingebed in een programma* |

### Instructies

|  |  |
| --- | --- |
| **Data Retrieval** | *Ophalen van data uit de DB:*  ***SELECT*** |
| **Data Manipulation Language (DML)** | *Inhoud wijzigen:*  ***UPDATE, DELETE, INSERT*** |
| **Data Definition Language (DDL)** | *Objecten creëren:*  ***CREATE, ALTER, DROP*** |
| **Data Control Language (DCL)** | *Beveiliging:*  ***GRANT, REVOKE*** |
| **Transaction Controle** | *Wijzigingen dewelke zeker moete slagen/falen groeperen:*  ***COMMIT, ROLLBACK, SAVEPOINT*** |

## Database ontwerpen

Het eindresultaat van een DB ontwerp is een DB model (goed DB model == goed DB systeem). Zo’n DB model moet aan de noden van alle gebruikers voldoen, door de eindgebruiker begrepen worden en voldoende details bevatten zodat met die informatie een DB aangemaakt kan worden.

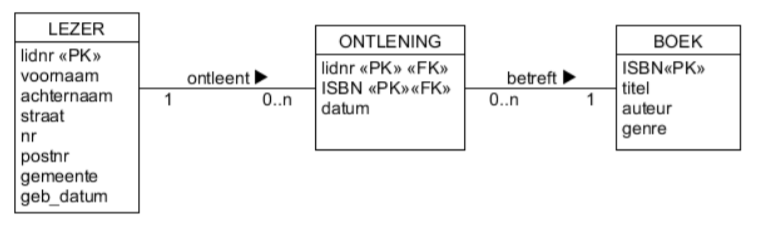
### Benaderingen

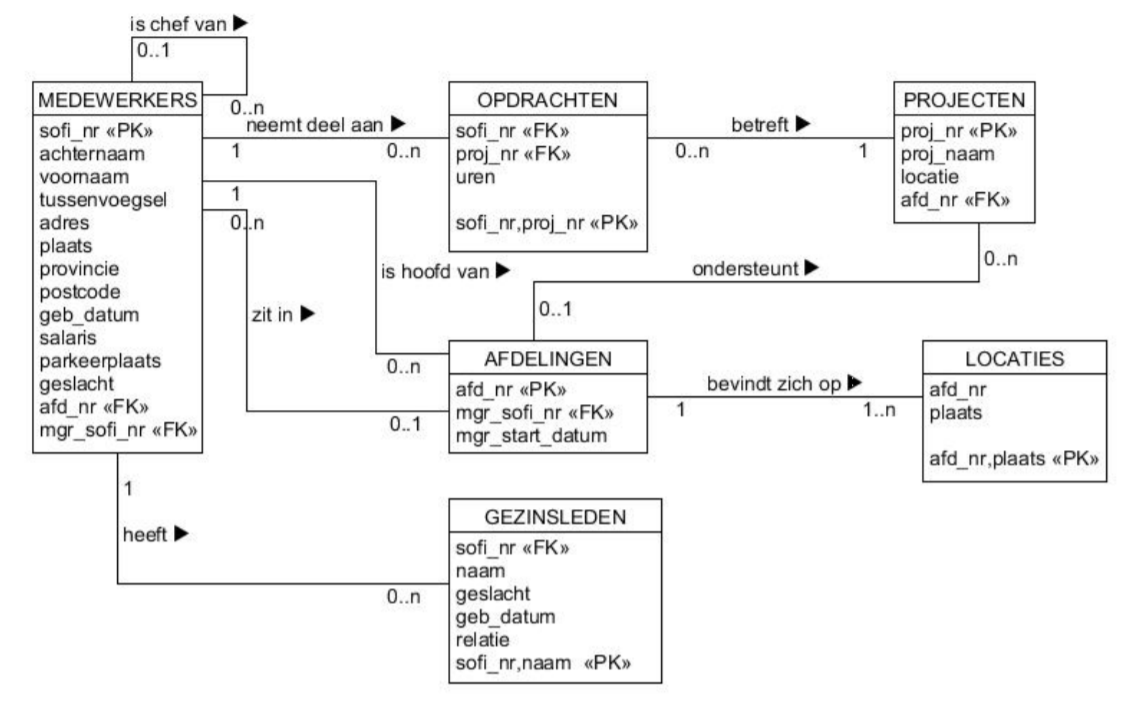
|  |  |
| --- | --- |
| **Bottom up** | *Vertrekt vanuit attributen die door het analyseren van verbanden tussen die attributen gegroepeerd kunnen worden (bv. Normalisatie)* |
| **Top down** | *Vetrekt van algemeen naar zeer specifiek.* |

### Fases in het ontwerpen

|  |  |
| --- | --- |
| **Conceptuele fase** | *Hierin bouwt men een model dat alle gegevens van de organisatie bevat, wat heeft de gebruiker nog nodig buiten een DBMS of een applicatie?* |
| **Logische fase** | *Hierin wordt het DB model aangepast aan het onderliggend data model dat gebruikt zal worden. Regelmatige terugkoppeling naar de eindgebruiker om te toetsen of aan alle eisen voldaan is + opstellen business rules.* |
| **Fysische fase** | *Implementatie en het nemen van beslissingen rond de performantie.* |

Een **ERD** of **E**ntity **R**elationship **D**iagram is een grafische voorstelling van zulk ontwerp. Het bevat de verschillende entiteiten van de te beschrijven realiteit, samen met de verbanden (relaties) tussen deze entiteiten.





### ERD lezen

|  |
| --- |
| MEDEWERKERS heeft meerdere ***0..n*** GEZINSLEDEN |
| GEZINSLEDEN heeft exact één ***1*** MEDEWERKER |
| AFDELING bevindt zich op één of meerdere ***1..n*** LOCATIES |
| LOCATIES verwijst naar exact 1 ***1*** AFDELING |
| AFDELINGEN ondersteunen 0, 1 of meerdere ***0..n*** PROJECTEN |
| PROJECTEN verwijzen naar 0 of 1 ***0..1*** AFDELINGEN |

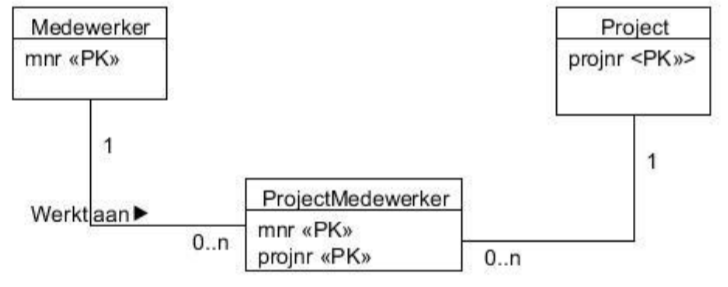
### Cardinaliteit

|  |  |
| --- | --- |
| **0..n** of **\*** | *0, 1 of meer* |
| **1..n** | *1 of meer* |
| **1** | *Exact 1* |
| **0..1** | *1 of 1* |

De veel-kant staat altijd aan de kant van de entiteit met de vreemde sleutel, in een relationeel datamodel zijn  **veel-op-veel relaties niet toegestaan**.



Indien er noodzaak ontstaat voor derlijke relatie kan men werken met een **tussentabel**:



# Inleiding DDL Statements (Data Definition Language)

## Algemene begrippen

|  |  |
| --- | --- |
| **USER** | *Is iemand die aan de DB kan connecteren.* |
| **schema** | *Is een container van objecten waarvan een gebruiker eigenaar is. Bij creatie van een gebruiker wordt het schema gecreërd (initieel leeg). Het schema kan leeg blijven als de gebruiker geen rechten heeft om objecten te creëren. In verschillende schema’s kunnen objecten dezelfde naam hebben.* |
| **namespace** | *Definieert een groep van object types (alle objecten behorend tot een bepaalde namespace horen binnen éénzelfde schema een unieke naam te hebben.* |

## Regels voor objecten

|  |
| --- |
| *Een naam mag van 1 tot 30 karakters bevatten.* |
| *Gereserveerde namen zijn niet toegelaten.* |
| *Alle namen beginnen met een letter.* |
| *Geen spaties toegelaten, enkel: letters, cijfers, \_, $, #* |

 Als een naam tussen “ “ staat mogen al deze regels overtreden worden, af te raden!

## Create table

Voor elk van de entiteiten uit het ERD moeten we bepalen:

|  |
| --- |
| *Hoe deze entiteiten gaan noemen.* |
| *Hoe de attributen van deze entiteiten gaan noemen.* |
| *Hoe groot elk van deze attributen moet zijn.* |
| *Welke informatie (gegevenstype) deze attributen mogen bevatten* |
| *Welke beperkingen (CONSTRAINTS) we de attributen gaan opleggen.* |

 Een tabelnaam moet uniek zijn binnen één schema.

 We zetten de naam van een tabel altijd in meervoudsvorm.

**CREATE TABLE** tabelnaam (attribuutnaam gegevenstype [DEFAULT WAARDE][COLUMN CONSTRAINT], …, [TABLE CONSTRAINT], …);

### Gegevenstype

#### Alfanumeriek

|  |  |
| --- | --- |
| **CHAR(n)** | *Voor attributen waarvan de inhoud een vaste lengte heeft (bv. telefoonnummer, artikelnummer, …). Bij CHAR(9) zal er achteraan aangevuld worden met blanco spaties indien er bv. slechts 7 karakters opgegeven worden.* |
| **VARCHAR2(n)** | *Voor attributen met variabele lengte (bv. straat, achternaam, …). Bij VARCHAR2(50) zullen er slechts 20 posities gebruikt worden als de attribuutwaarde maar 20 posities heeft.* |
| **CLOB** | ***C****haracter* ***L****arge* ***Ob****ject, voor grote documenten tot 4GB.* |
| **LONG** | *Kan tot 2GB aan karakters bevatten*  Depricated! |

#### Numeriek

|  |  |
| --- | --- |
| **NUMBER(n)** | *Voor attributen met gehele getallen (decimalen worden afgerond).* |
| **NUMBER(n,m)** | *Voor attributen met decimale getallen (n = totaal, m = aantal decimalen).* |
| **NUMBER** | *Voor attributen waarvan de grootte niet in te schatten is (onbeperkt aantal gehelen, max 38 decimalen).* |

Bv. 1234.567

**NUMBER**(8) --1235  
**NUMBER**(4) --1235  
**NUMBER**(3) --FOUTMELDING (aantal gehele getallen past niet)  
**NUMBER** --1234.567  
**NUMBER**(5,2) –-FOUTMELDING (aantal impliceert 3 gehele getallen)  
**NUMBER**(6,2) –-1234.57  
**NUMBER**(6,-2)--1200

#### Datum

|  |  |  |
| --- | --- | --- |
| **DATE** | *Standaard als DD-MON-YYYY (bevat eeuw, jaar, maand, dag, uur, min & sec). Indien geen tijd opgegeven DEFAULT 0 (middernacht).* | |
| **SYSDATE** | *Standaard datum van vandaag.* | |
| **TIMESTAMP** | *Tevens fracties van seconden (tot 6 decimale posities voor seconden).* | |
| **INTERVAL YEAR TO MONTH** | | *Toont verschil tussen 2 datums.* |
| **INTERVAL YEAR TO SECONDS** | | *Toont verschil tussen 2 datums.* |

#### Binair (binaire gegevens zoals foto, video, …)

|  |  |
| --- | --- |
| **BFILE** | *Bevat een pointer naar een binaire file in het OS van de DB server.* |
| **ROWID** | *Bevat fysiek adres van die rij in de DB (pseudokolom).* |

## Constraints

Door middel van constraints kan je extra beperkingen opleggen aan attributen.

* Aan één attribuut == COLUMN CONSTRAINT
* Combinatie van attributen == TABLE CONSTRAINT

Een **TABLE CONSTRAINT** is een beperking waarbij meer dan één attribuut uit éénzelfde tabel betrokken is. Deze constraints worden niet op attribuut niveau gedefinieerd maar worden pas op het einde van de tabeldefinitie gedefinieerd.

### Soorten constraints

|  |  |
| --- | --- |
| **PRIMARY KEY** constraint | *Key constraint (unieke waarde) + Entity integrity constraint (not NULL)* |
| **NOT NULL** constraint |  |
| **CHECK** constraint | *Bereik van attribuutwaarden (domein)* |
|  *bv. stel dat commisieloon en salaris 2 attributen zijn uit dezelfde tabel. Als we de beperking opleggen dat commissiloon + salaris maximum 6000 mag zijn, dan is dat een tabel constraint omdat er 2 attributen bij betrokken zijn!* | |
| **UNIQUE** constraint |  |
| **FOREIGN KEY** constraint | *Referential integrity constraint (door deze constraint op te leggen zal het DBMS automatisch controlen of de waarden van de FK als PK waarden voorkomen in de tabel waar men naar verwijst).* |

#### COLUMN CONSTRAINT

**CONSTRAINT** constraintnaam **REFERENCES** tabel (attribuut) [ON DELETE CASCADE/SET NULL]

#### TABLE CONSTRAINT

**CONSTRAINT** constraintnaam **FOREIGN** **KEY** (attr1, attr2,…) **REFERENCES** tabel (attr1, attr2,…) [ON DELETE CASCADE/SET NULL]

Default is het **ON DELETE RESTRICT** wat checkt op afhankelijke rijen, indien die er zijn zal de **DELETE** actie geweigerd worden.

|  |  |
| --- | --- |
| **ON DELETE CASCADE** | *Afhankelijke rijen worden mee verwijderd.* |
| **ON DELETE SET NULL** | *De waarden van de verwijssleutels in de betrokken afhankelijke rijen worden op NULL gezet.* |

Tussen meerdere constraints zetten we **geen** komma, gezien een komma een attribuutbeschrijving impliceert.

achternaam **VARCHAR2**(20) **CONSTRAINT** c\_achternaam **CHECK**(achternaam=**UPPER**(achternaam)) **CONSTRAINT** nn\_achternaam **NOT** **NULL**