# DATABANKEN 1

# Hoofdstuk 1: inleidende begrippen

## Wat is een informatiesysteem

Een **informatiesysteem** is een vereenvoudigde, representatieve voorstelling van de werkelijkheid binnen een organisatie.

Een organisatie heeft doelstellingen

🡪Om die doelstellingen te bereiken moeten er beslissingen genomen worden.

🡪Nood aan informatie

Hoeft niet per se geautomatiseerd te zijn. Het bijhouden van effectieve documenten ergens in een kast is ook een informatiesysteem.

### Bestandsgerichte benadering

Nadelen:

* Redundantie: Op verschillende documenten dezelfde informatie.
  + Risico op inconsistentie (Bij een verandering aan gegevens moet het op verschillende plaatsen aangepast worden
  + Inefficiënt geheugengebruik
* Gescheiden geïsoleerde data.
* Data afhankelijkheid
  + Een wijziging in de structuur van het bestand betekent dat het programma moet aangepast worden
* Incompatibele bestanden.

### Databasegerichte benadering

Voordelen:

* Gedeelde data
* Geïntegreerd en samenhangend
* Geringe redundantie
* Data onafhankelijkheid
* Veiligheidsbewaking

*Wie krijgt toegang tot db, welke handelingen mogen er uitgevoerd worden*

* Vereenvoudigen van het opleggen van standaarden.

## Database

**Database** = *Geïntegreerde verzameling gegevens die eventueel door meerdere gebruikers simultaan kan gemanipuleerd worden en die voldoet aan de informatiebehoeften van een organisatie. Ze bevat gegevens en metagegevens (structuur van de database).*

**Database-Management System (DBMS)** = de software die nodig is om een database te beheren en te gebruiken.

* Gebruiken:
  + Opvragen van gegevens
  + Wijzigen van de inhoud
  + Onderhoud van de db
* Beheren:
  + Definiëren en wijzigen van de structuur van de db
  + Beveiliging
  + Transactiebeheer

### Gebruikersprofielen

* Data-admins
* Database-admins
* Toepassingontwikkelaar
* Eindgebruiker

### Het relationeel model

Een database is een **geïntegreerde verzameling**. Tussen de entiteiten bestaan er verbanden. Die bestanden worden bepaald door het onderliggend datamodel.

De focus bij ons ligt op het relationeel model.

Het **relationeel model** bestaat uit:

* Relaties
* Operatoren om te werken met die relaties
* Integriteitsregels

De relationele db is een verzameling van relaties (tabellen). De verbanden tussen die tabellen worden gelegd door foreign keys.

Terminologie:

* Tabel
* Kolom of attribuut
  + Is atomair: op het snijpunt van een kolom en een rij kan maar 1 waarde staan
  + Kolomnamen zijn uniek en volgorden heeft geen betekenis
* Rij of tuple
  + Is uniek binnen een tabel
  + Geen specifieke volgorde
* Attribuutwaarde
* Populatie
* Domein
* Null-waarde

### Sleutelattributen

3 soorten sleutels:

* Primary key

*Uniciteit van een rij afgedwongen door primary key*

* Alternative key

*Meerdere attributen of een combinatie ervan vormen een primary key*

* Foreing key

*Verbanden leggen tussen 2 tabellen*

#### Integriteitsregels relationele model

* Key constraint

*De primary key moet uniek zijn en uniek blijven*

* Enitity integrity constraint

D*e primary key moet steeds een geldige waarde hebben (niet null)*

* Referential integrity constraint

*De foreign key moet naar een primary key verwijzen*

## Bewerkingen met relaties

### Unie

De unie voegt twee relaties samen tot 1 relatie.

### Intersectie

Neemt uit twee relaties met dezelfde attributenlijst de gemeenschappelijke rijen.

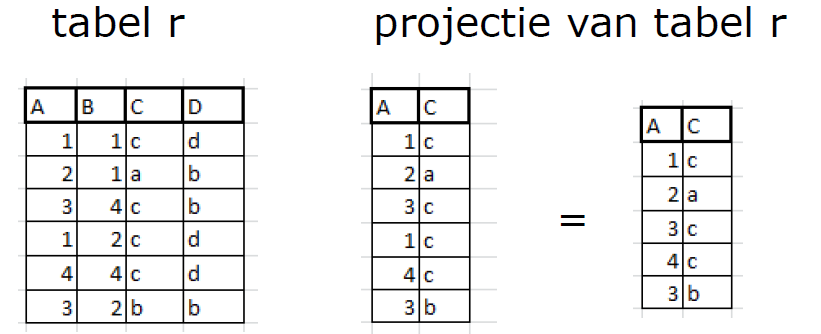
### Verschil

Bevat alle rijen uit relatie 1 die niet in relatie 2 zitten.

### Product

Het cartesiaans product van 2 relaties.

### 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

Een selectie nemen op basis van een voorwaarde.

### Join

Combineren van 2 relaties.

## Wat is SQL

**Structured Query Language**. Een taal om met een relationeel DBMS te communiceren

Subcategorieën:

* Data retrieval (select)
* Data manipulation language **DML** (update, insert, delete)
* Data definition language **DDL** (create, alter, drop,…)
* Data control language **DCL** (grant, revoke)
* Transaction control(commit, rollback savepoint)

### Database ontwerp

Het eindresultaat van een database ontwerp is een database model. Een goed model leidt tot een goed databasesysteem. Dit voldoet aan de noden van alle gebruikers wordt begrepen door de eindgebruiker en bevat voldoende details en specificaties zodat de db met die informatie fysisch kan aangemaakt worden.

Er zijn 2 benaderingen voor een db-ontwerp:

* **Bottom-up** benadering vertrek vanuit de attributen die door het analyseren van de verbanden tussen die attributen in relaties gegroepeerd worden. **Normalisatie** representeert een bottom-up benadering.
* **Top-down** gaat van algemeen naar meer specifiek. De analist vertrekt van wat er nodig is.

#### Fases

##### De conceptuele fase

Bouwen aan een model dat alle gegevens bevat die de organisatie gebruikt. Dit is de bouwsteen voor de logische fase.

##### De logische fase

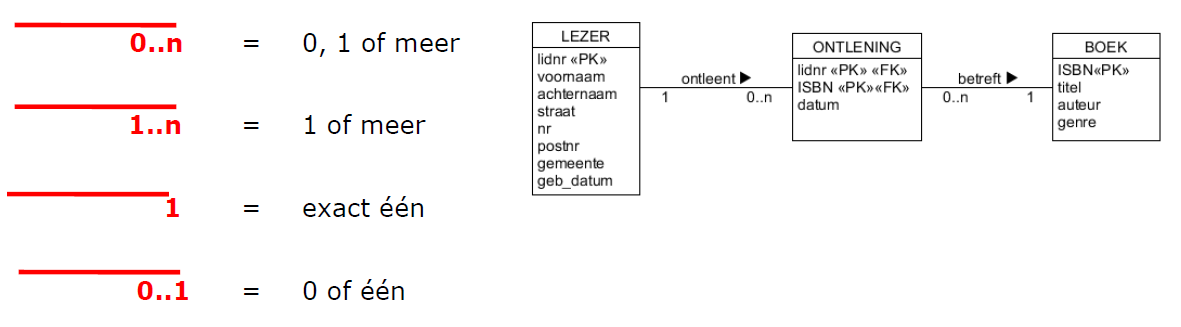
Het db model wordt aangepast aan het onderliggend datamodel dat gebruikt zal worden (relationeel, hiërarchisch, netwerk, …)

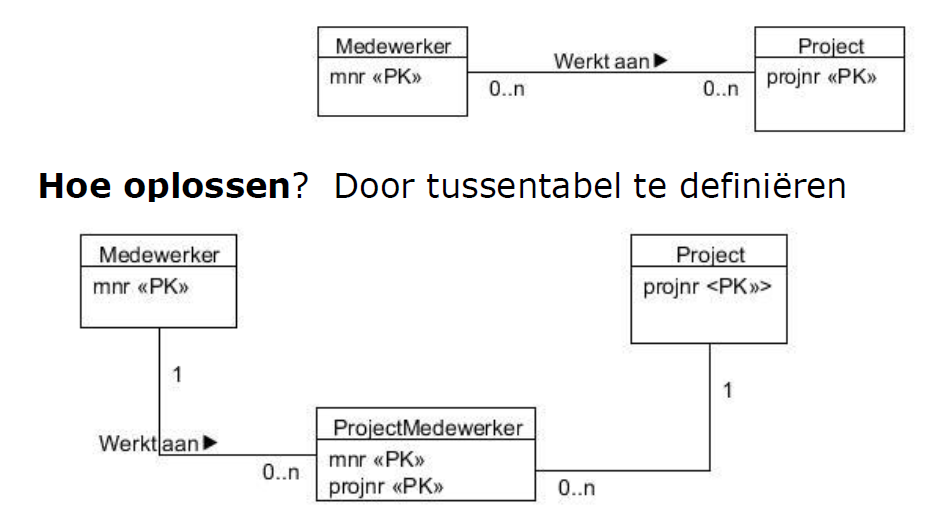
Er wordt dan regelmatig teruggekoppeld naar de eindgebruiker om aan de eisen te voldoen.

##### De fysische fase

De implementatie.

### Database ontwerp – het ERD

**Entity Relationship Diagram** is een grafische voorstelling van de verschillende entiteiten van de te beschrijven realiteit, samen met de verbande tussen deze entiteiten.

De veel-kant staat altijd aan de kant van de entiteit met de foreign key. In een relationeel datamodel zijn veel op veel relaties niet toegelaten. Dit lossen we op door een tussentabel te definiëren.

# Inleiding DDL

Bij een relationele database zijn er enkele eigenschappen. Rijen zijn uniek, in een willekeurige volgorde, kolomnamen zijn uniek en attribuutwaarden zijn atomair.

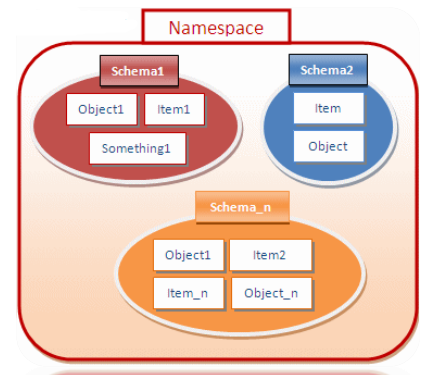
Er zij **primary en foreign keys**, met **key constraint** (unieke PK**), entity integrity constraint** (PK != null) **en referential integrity constraint** (FK verwijst naar PK).

Deze regels kunnen we inbouwen in het CREATE TABLE statement. De controle gebeurt auto;atisch door het DBMS.

## Begrippen gebruiker, schema, namespace

Overzicht van objecten:

*SELECT object\_type*

*FROM DBA\_OBJECTS;*

Een **gebruiker (user)** is iemand die aan de databse kan connecteren. Een **schema** is de container van objecten waarvan een gebruiker eigenaar is.

Een **namespace** definieert een groep van object types. Alle object types binnen een name space moeten een unieke naam hebben.

# Creatie van een tabel

Om een tabel te creëren moet je:

Bepalen hoe de **entiteiten en de attributen gaan noemen**, **hoe groot de attributen** mogen zijn en **welk soort informatie** ze mogen bevatten.

Belangrijk is ook om na te denken over de **beperkingen op bepaalde attributen (constraints)**.

*CREATE TABLE tabelnaam (*

*attribuutnaam1 gegevenstype [default waarde] [column constraint],*

*attribuutnaam2 gegevenstype [default waarde] [column constraint],*

*[table constraint]*

*);*

### Gegevenstype

#### Alfanumerieke attributen

* CHAR(n)

*Wordt gebruikt voor attributen die een vaste lengte zullen hebben, bv. een telefoonnummer.*

* VARCHAR2(n)

*Wordt gebruikt voor attributen met een variabele lengte.*

* CLOB (= Character Large Object)

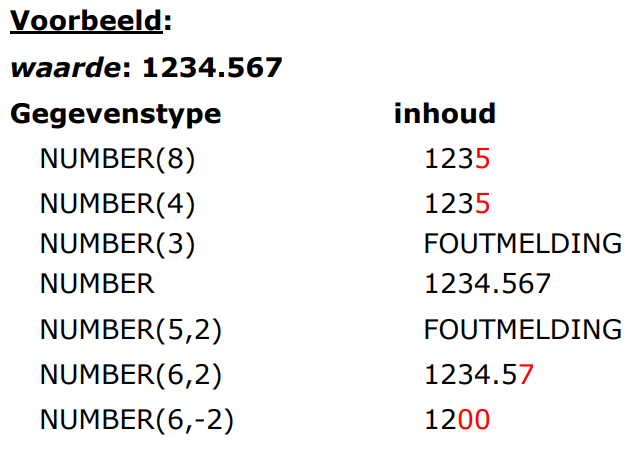
*Stockering van grote documenten.*

#### Numerieke attributen

* NUMBER(n)

*N 🡪 aantal gehelen waarvoor ruimte wordt voorzien.*

* NUMBER(n, m)

*N 🡪 totaal aantal tekens.*

*M 🡪 geeft het aantal decimalen weer.*

* NUMBER

#### Datumattributen

* DATE

*Standaard DD-MON-YYYY. Bevat* ***eeuw, jaar, maand, dag, uur, minuten en seconden****. SYSDATE is standaard de datum van vandaag.*

* TIMESTAMP

*Tot op fracties van seconden stockeren.*

#### Andere datatypes

**Binaire datatypes**

BLOB (Binary large objects)

**Bfile**

Bevat een pointer naar een binaire file in het operating system van de db server.

**ROWID**

Elke rij bevat een rowid. Deze bevat het fysieke adres van die rij in de db.

### Default waarde

*Status char(1) default ‘Y’*

### Constraints

* Primary key constraint (key constraint + entity integrity constraint)
* Not null constraint
* Check constraint (UPPER() of LOWER(), IN, BETWEEN … AND … , <, >, =, …)
* Unique constraint
* Foreign key constraint (referential integrity constraint)

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

**Column constraint** is een beperking op 1 kolom.

**Table constraint** is een beperking waarbij **meer dan 1 attribuut uit eenzelfde tabel betrokken is**.

|  |  |  |
| --- | --- | --- |
|  | Column constraint | Table constraint |
| Not null | Achternaam VARCHAR(25) CONSTRAINT nn\_achternaam NOT NULL | KAN NOOIT ALS TABLE CONSTRAINT |
| Primary key | Sofi\_nr CHAR(9) CONSTRAINT pk\_medewerkers PRIMARY KEY | CONSTRAINT pk\_opdrachten PRIMARY KEY (sofi\_nr, proj\_nr) |
| Check | Salaris NUMBER (7,2) CONSTRAINT c\_salaris CHECK (salaris <= 25000) | CONSTRAINT c\_salaris CHECK (salaris + commissieloon <= 25000) |
| Unique | Nr\_identiteitskaart CHAR(9) CONSTRAINT u\_nr\_id UNIQUE | CONSTRAINT u\_sofi\_nr\_parkeerplaats UNIQUE (sofi\_nr, parkeerplaats) |
| Foreign key | Afd\_nr NUMBER(2) CONSTRAINT fk\_med\_afd REFERENCES afdelingen(afd\_nr) | CONSTRAINTfk\_opdrachten FOREIGN KEY (sofi\_nr, proj\_nr) REFERENCES opdrachten(sofi\_nr, proj\_nr) |
| *FK met ON DELETE* | *CONSTRAINT constraintnaam REFERENCES tabel (attribuut) [ON DELETE CASCADE /SET NULL]* | *CONSTRAINT constraintnaam FOREIGN KEY(attr1,attr2,…) REFERENCES tabel (attri,attrj…) [ON DELETE CASCADE/SET NULL ]* |

**ON DELETE CASCADE**: Als er rijen verwijderd worden, zullen de FK rijen mee verwijderd worden.

**ON DELETE SET NULL:** De fk’s worden op null gezet.

**ON DELETE RESTRICT (default):** Bij het verwijderen van de rij kan het zijn dat er geweigerd zal worden omdat er nog afhankelijke rijen zijn.

# Wijzigen van de tabelstructuur

## Verwijderen van een tabel

### Alter table

Add

Toevoegen attributen en table constraints.

*ALTER TABLE medewerkers ADD adres VARCHAR2 (50) CONSTRAINT c\_adres CHECK (adres = UPPER (adres));*

*ALTER TABLE opdrachten* *ADD constraint pk\_opdrachten PRIMARY KEY (sofi\_nr, proj\_nr)*;

#### Modify

Wijzijngen aan te brengen in bestaand attributen (datatype wijzigen, default waarde toevoegen, column constraints toevoegen).

*ALTER TABLE medewerkers MODIFY parkeerplaats CONSTRAINT nn\_parkplaats NOT NULL;*

*ALTER TABLE medewerkers MODIFY salaris DEFAULT 0;*

*ALTER TABLE medewerkers MODIFY salaris NUMBER (9, 2):*

#### Drop

Attribuut of constraint verwijderen

*ALTER TABLE medewerkers DROP COLUMN salaris;*

*ALTER TABLE medewerkers DROP constraint nn\_parkplaats;*

**Wat als je een PK wil verwijderen?**

* *ALTER TABLE projecten DROP CONSTRAINT pk\_project CASCADE;*

Of *ALTER TABLE projecten DROP PRIMARY KEY CASCADE;*

🡪 De FK constraints die hiernaar verwijzen worden mee verwijderd.

**Wat als je een bestaand constraint wilt wijzigen?**

Eerst droppen van constraint en dan **ADD (voor table constraint)** of **MODIFY (voor column constraint)**.

#### Set unused column

*ALTER TABLE afdelingen SET UNUSED COLUMN mgr\_start\_datum;*

#### Rename column

*ALTER TABLE afdelingen RENAME COLUMN afd\_naam TO naam\_afd;*

#### Read only

*ALTER TABLE afdelingen READ ONLY;*

#### Drop table

Je kan een table maar verwijderen wanneer er geen FK verwijzen naar de betreffende table, zelfs bij ON DELETE CASCADE/SET NULL.

* *DROP TABLE projecten CASCADE CONSTRAINTS;*