

Databanken 1

Hoofdstuk 1

Inleidende begrippen

cursusmateriaal

- ❑ cursus 'Databanken 1' blz. 5-44
- ❑ Deze powerpoint



Informatiebehoefte en informatiesysteem

❑ Wat is een informatiesysteem?



Bestandsverwerkingssystemen

Bestandsverwerkingssystemen versus databases

- ❑ Bestandsgericht informatiesysteem + nadelen
- ❑ Database gericht informatiesysteem + voordelen

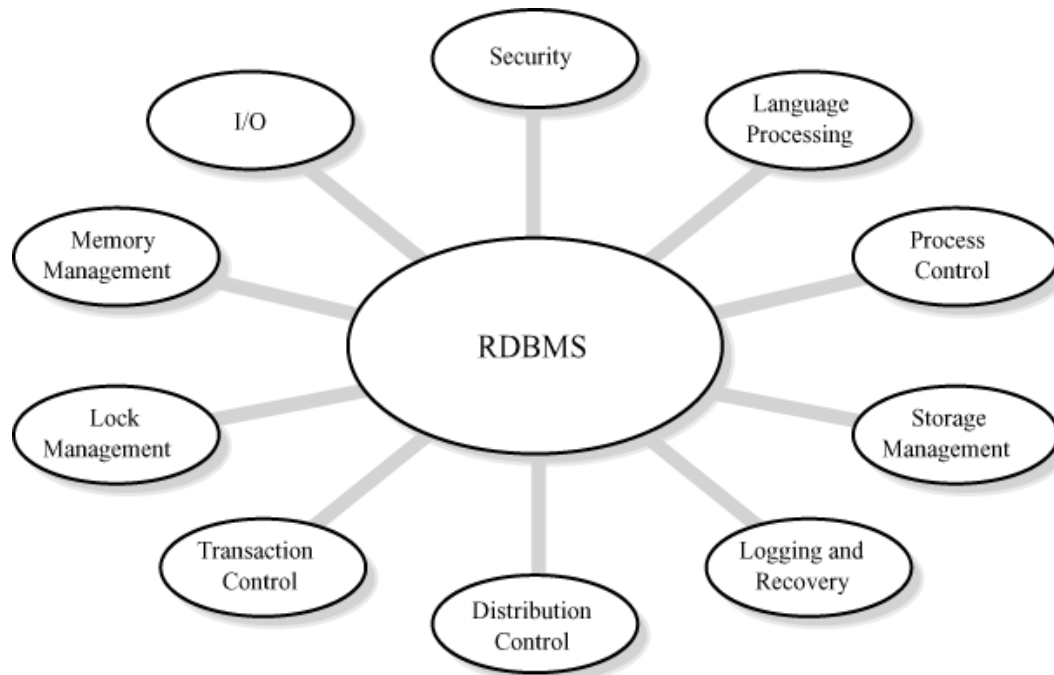


Database en database management systeem

❑ Wat is een database?

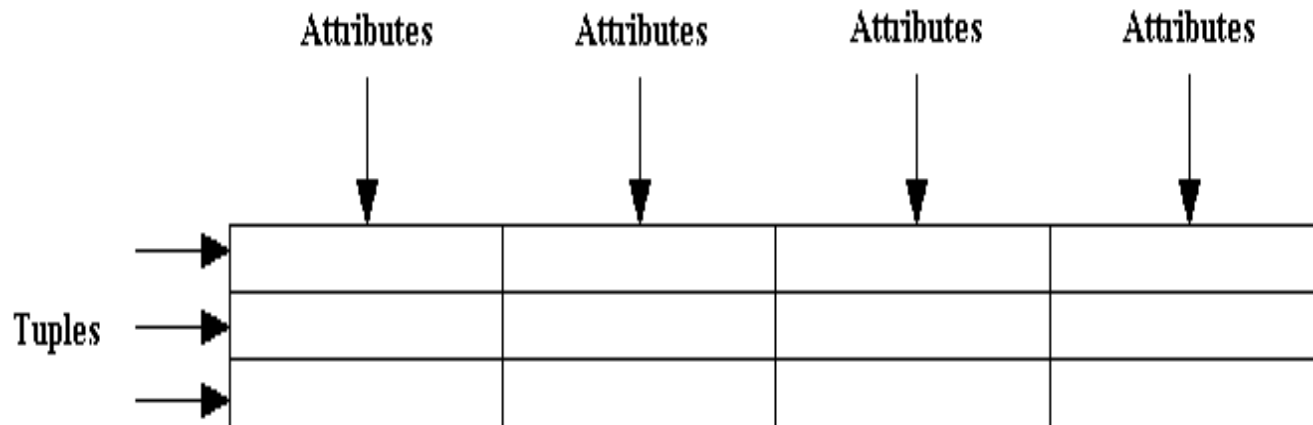


❑ Wat is een database management systeem?



Het relationele model

- ❑ Wat is een relationele tabel?
- ❑ Sleutelattributen?
- ❑ Integriteitsregels opgelegd door het relationele model?



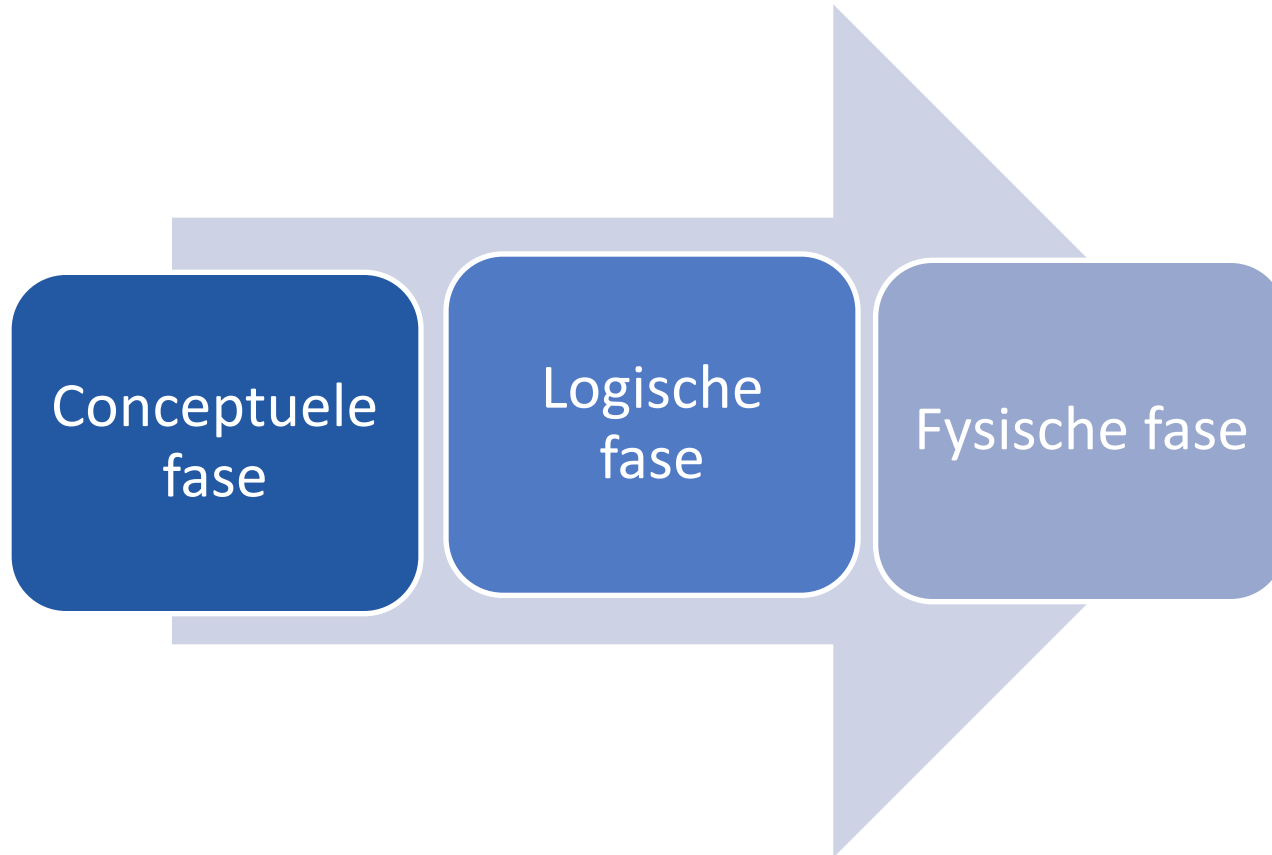
SQL (Structured Query Language)

- ☐ Wat is SQL?
- ☐ Welke eigenschappen heeft SQL?
- ☐ Hoe de SQL instructies indelen?



Database ontwerp

❑ Verloop van het database ontwerp



Wat is een informatiesysteem?

Een organisatie heeft doelstellingen



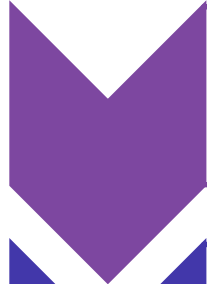
Om die doelstellingen te bereiken moeten beslissingen genomen worden



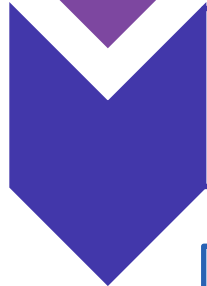
Om correcte beslissingen te kunnen nemen, moet men op de juiste moment over de juiste informatie beschikken: men heeft **een informatiesysteem** nodig.

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

Wat is een informatiesysteem?



- Een organisatie heeft doelstellingen



- Om die doelstellingen te bereiken moeten beslissingen genomen worden



- Om correcte beslissingen te kunnen nemen, moet men op de juiste moment over de juiste informatie beschikken: men heeft een **informatiesysteem** nodig.

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

Wat is een informatiesysteem?

Hoeft niet noodzakelijk geautomatiseerd te zijn

➔ fichebak kan perfect informatiesysteem zijn!

bv klantenbeheer in een kleine kledingzaak

- ☐ fiche per klant
- ☐ op fiche naam, adres, aankoopbedrag, datum
- ☐ fiches bv. gesorteerd op naam
- ☐ na 10 aankopen % korting berekenen op totaal bedrag aankopen en nieuwe fiche aanmaken



Bestandsgerichte benadering

In de wereld van informatiesystemen volgen we het verhaal van Mr. Peeters.

Mr. Peeters zoekt in zijn vrije tijd naar huurders voor de appartementen en villa's van zijn vrienden aan de kust.

informatiesysteem:

- ➡ 1) **fichebak** met per fiche informatie over appartement of villa
- ➡ 2) **excelsheet** met overzicht wanneer welk appartement of villa vrij is doorheen het kalenderjaar
- ➡ 3) **voorgedrukte verhuurcontracten**



perfect informatiesysteem **voor**
beperkt aantal woningen



Bestandsgerichte benadering

Stel: Mr. Peeters krijgt hoe langer hoe meer woningen te huur aangeboden.

Huidig informatiesysteem voldoet niet meer

- ➡ ☐ Mr. Peeters besluit een immo-verhuurkantoor DREAMHOME op te richten en neemt een informaticus onder de arm
- ➡ ☐ **er wordt besloten om over te stappen op een bestandsgericht informatiesysteem**

concreet:

- ☐ alle informatie uit de excelsheets, fiches en papieren verhuurcontracten worden in **bestanden** geplaatst.
De **bestanden worden door programma's verwerkt.**

Bestandsgerichte benadering

Verhuurafdeling

Houdt zich bezig met het verhuren van de woningen

KLANT

bevat potentiële klanten

(klantnr, naam, straat, nr, postnr, gemeente, type_woning, max_huur, ...)

EIGENDOM_TE_HUUR

bevat de eigendommen te huur

(eignr, straat, nr, postnr, gemeente, aantal_kamers, type_woning, huur, eigenaarsnr)

EIGENAAR

bevat de eigenaarsgegevens

(eignaarsnr, naam, straat, nr, postnr, gemeente, ...)

BESCHIKBAAR

bevat info over beschikbaarheid

Contractafdeling

Houdt zich bezig met de administratieve kant

KLANT

bevat klantgegevens

(klantnr, naam, straat, nr, postnr, gemeente...)

EIGENDOM

bevat de verhuurde eigendommen

(eignr, straat, nr, ..)

CONTRACT

bevat de contractgegevens

(eignr, klantnr, overeengekomen_huur, begin, einde...)

Nadelen van een bestandsgerichte benadering?

Redundantie

Beide afdelingen hebben een KLANT en een EIGENDOM bestand die voor een groot stuk dezelfde gegevens bevatten



- ❑ risico voor inconsistenties (tegenstrijdige informatie)
bv. klant stuurt adresverandering door die enkel in het KLANT bestand van de VERHUURAFDELING wordt aangepast
- ❑ inefficiënt geheugengebruik
- ❑ updates op verschillende plaatsen doorvoeren

Nadelen van een bestandsgerichte benadering?

Gescheiden geïsoleerde data b.v.:

- ☐ De KLANT gegevens staan in een afzonderlijk bestand.
- ☐ De EIGENDOM gegevens staan in een afzonderlijk bestand.
- ☐ Welke eigendommen komen voor welke klanten in aanmerking?

in programma beide **bestanden synchroniseren**
om antwoord op vraag te vinden = vrij **complexe**
programmatie

Verhuurafdeling

Houdt zich bezig met het verhuren van de woningen

KLANT

bevat potentiële klanten

(klatnr, naam, straat, nr, postnr, gemeente, type_woning, max_huur, ...)

EIGENDOM_TE_HUUR

bevat de eigendommen te huur

(eignr, straat, nr, postnr, gemeente, aantal_kamers, type_woning, huur, eigenaarsnr)

EIGENAAR

bevat de eigenaarsgegevens

(eignaarsnr, naam, straat, nr, postnr, gemeente, ...)

BESCHIKBAARHEID

Contractafdeling

Houdt zich bezig met de administratieve kant

KLANT

bevat klantgegevens

(klatnr, naam, straat, nr, postnr, gemeente...)

EIGENDOM

bevat de verhuurde eigendommen

(eignr, straat, nr, ..)

CONTRACT

bevat de contractgegevens

(eignr, klatnr, overeengekomen_huur, begin, einde...)

Nadelen van een bestandsgerichte benadering?

❑ Data afhankelijkheid

Bestand → programma → resultaat

De structuur van de bestanden staat in programma beschreven



```
01 klant  
  02 klantnr      pic 99999.  
  02 naam         pic x(30).  
  02 straat       pic x(50).  
  02 huisnr       pic x(4).  
  02 postnr       pic 9999.  
  02 gemeente     pic x(30).  
  02 type_woning  pic x(2).  
  02 max_huur     pic 9999.
```

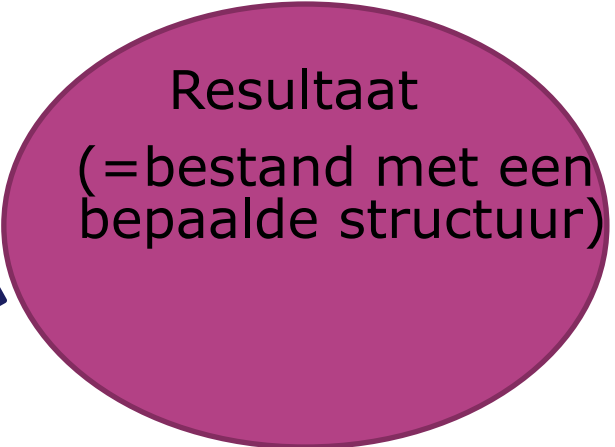
Een **wijziging** doorvoeren in de **structuur** van het **bestand** betekent dat het **programma moet aangepast worden**

Nadelen van een bestandsgerichte benadering?

❑ Incompatibele bestanden

Vb.

KLANT bestand → COBOL programma →



Resultaat
(=bestand met een
bepaalde structuur)

bevat een bestandsstructuur die C++
programma niet aanvaardt



verwerkt door een C++ programma



resultaat

Afhankelijk van de programmeertaal geldt een andere
bestandsstructuur.

Nadelen van een bestandsgerichte benadering?

- ❑ Gescheiden geïsoleerde data
- ❑ Redundantie
- ❑ Data afhankelijkheid
- ❑ Incompatibele bestanden

wanneer de nadelen van een bestandsgerichte benadering te fel doorwegen kan Mr. Peeters overwegen om over te stappen naar een database gerichte benadering.

Database gerichte benadering

- ❑ Mr Peeters vervangt zijn bestanden door een database die door beide afdelingen **tegelijkertijd** kan gebruikt worden:
- ❑ alle informatie die in de bestanden stond, wordt (door toepassing van bv. normalisatietechnieken) verdeeld over verschillende groepen = entiteiten.
- ❑ Deze entiteiten vormen de 'objecten' van de databank

database gerichte benadering

Mr Peeters vervangt zijn bestanden door een database die door beide afdelingen **tegelijkertijd** kan gebruikt worden :

VERHUURAFDELING

CONTRACTAFDELING



Voordelen van de database benadering?

❑ Gedeelde (=shared) data

Alle applicaties kunnen gebruik maken van de gegevens uit de databank (lezen zelfs simultaan – voor aanpassingen moeten ze met mekaar rekening houden).

❑ Geïntegreerd en samenhangend

Alle gegevens zitten gecentraliseerd en gerelateerd. Alle applicaties kunnen ze daar gaan halen.

Applicatie → DBMS → OS → DBMS → applicatie

Voordelen van de database benadering?

❑ Geringe redundantie

gegevens zitten niet meer dubbel gestockeerd
(via bepaalde technieken waaronder normalisatie komt men tot database objecten zonder of met weinig redundantie)

❑ Data onafhankelijkheid

De gegevens worden in de databank beschreven
(in repository / data dictionary) en niet in de applicatie.



structuurwijziging van een database object
vraagt geen aanpassing van de applicatie

Voordelen van de database benadering?

❑ Integriteit (=correctheid) bewaken

- **Beperkte redundantie:** dus weinig inconsistenties
- **Constraints:** inbouwen waardoor er automatische controles worden ingebouwd (business rules laten respecteren) bv. nagaan of het max_huur binnen bepaalde grenzen blijft.
- **Triggers** bouwen waar constraints ontoereikend zijn (zie 2^e jaar)

Voordelen van de database benadering?

❑ Veiligheidsbewaking

gegevens zitten gecentraliseerd



makkelijker te beveiligen:

- ❑ wie krijgt toegang tot de database?
- ❑ wie mag welke handelingen uitvoeren op de database?
- ❑ wie mag met welke objecten werken?

Voordelen van de database benadering?

☐ Vereenvoudigen van het opleggen van standaarden

Vermits alles bij mekaar staat kan je gemakkelijk afspraken maken rond benamingen die je aan velden geeft

bv. adres is steeds: straat/ nr /postnr /gemeente

familienaam moet steeds in hoofdletters...

Definitie databank

Definitie databank en database management system

Database=

geïntegreerde verzameling gegevens die eventueel *door meerdere gebruikers simultaan* kan ***gemanipuleerd*** worden en die voldoet aan de informatie behoeften van een organisatie. Ze ***bevat*** gegevens en ***metagegevens*** (structuur van de database)



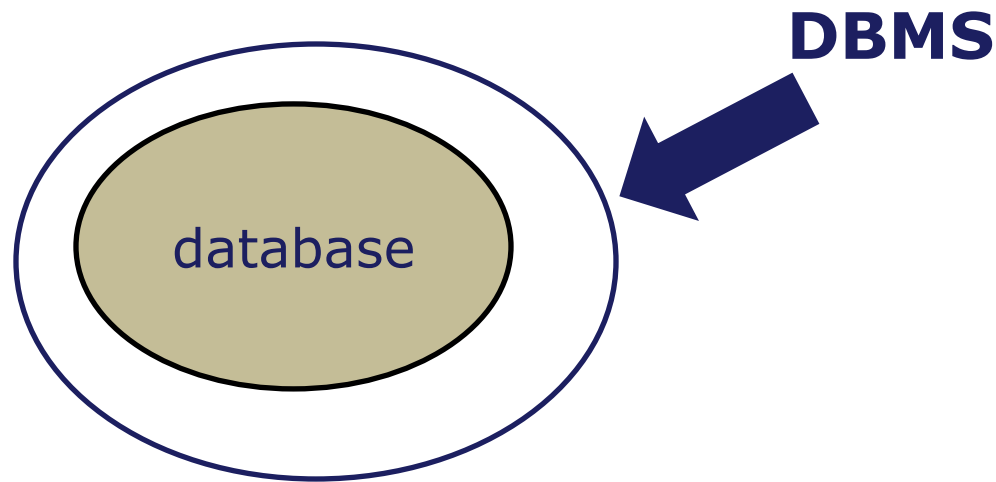
Je hoeft zelf geen documentatie over de database aan te leggen. Dit gebeurt automatisch; elke wijziging wordt ook automatisch bijgehouden.



Wat is een DBMS?

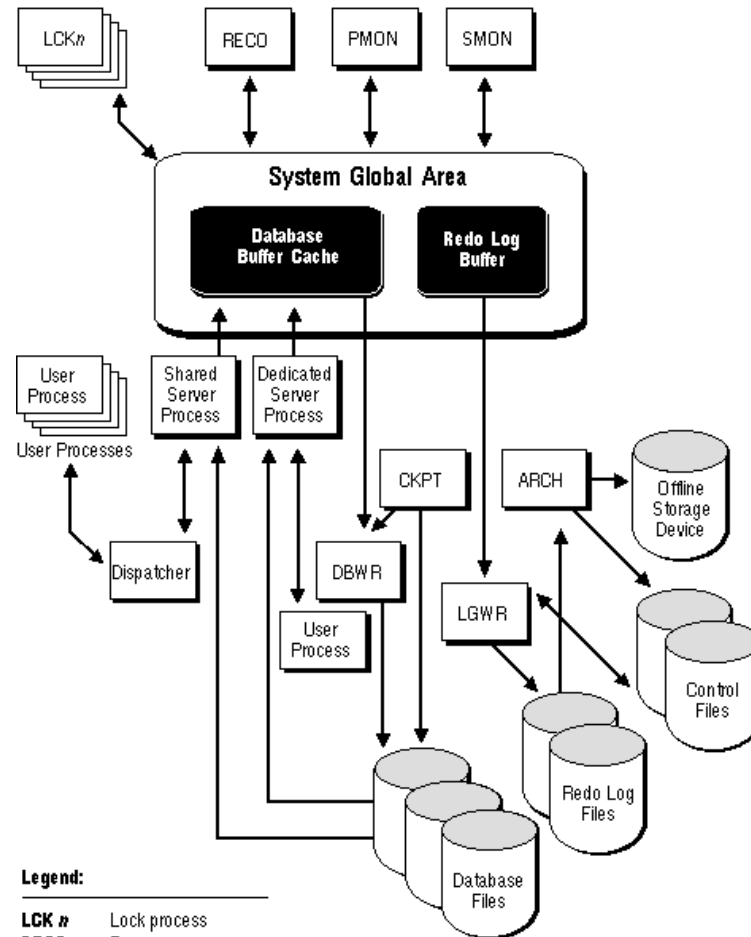
DBMS (= DataBase Management System)

= de **software** die nodig is om de **database** te **beheren** en te **gebruiken**



Wat is een DBMS?

Oracle DBMS



Legend:

LCK n	Lock process
RECO	Recoverer process
PMON	Process monitor
SMON	System monitor
CKPT	Checkpoint
ARCH	Archiver
DBWR	Database writer
LGWR	Log writer

DBMS

(= DataBase Management System) = de **software** die nodig is om de **database te gebruiken** en **te beheren**



- ☐ **Opvragen van gegevens** door het schrijven van queries
- ☐ Het **wijzigen van de inhoud** van de tabellen uit de data-base door het schrijven van DML instructies
- ☐ Het **onderhoud** van de database (de inhoud van de database moet beschikbaar zijn en blijven / performantie / backups...).



- ☐ **Definiëren en wijzigen** van de **structuur** van de database
dmv DDL instructies
(CREATE, ALTER, DROP)
- Het DBMS volgt daarin het ANSI/SPARC model
- ☐ **Beveiliging**
- ☐ **Transactiebeheer**

Gebruikersprofielen

Voor gebruikers van databases worden gebruikersprofielen gemaakt. Belangrijke gebruikersprofielen zijn:

- ☐ de data-administrator,
- ☐ de database-administrator,
- ☐ toepassingsontwikkelaar
- ☐ eindgebruiker.

Het relationeel model

Een database is een **geïntegreerde** verzameling.

Tussen de entiteiten bestaan er verbanden.

Die verbanden worden bepaald door het onderliggende datamodel.

De belangrijkste datamodellen zijn :

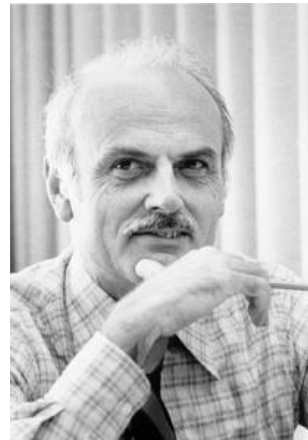
- ☐ hiërarchisch model
- ☐ netwerk model
- ☐ **relationeel model**

Het relationeel model

E.F. Codd ontwikkelde het **relationele model gebaseerd op de relationele algebra**.

Het relationele model bestaat uit:

- ❑ **relaties**
- ❑ een aantal **operatoren** om met die relaties werken
- ❑ **integriteitsregels** om de consistentie (=correctheid) van de data te garanderen
- ❑ Een relationele database wordt gebouwd volgens het relationele model.



De relationele database

De relationele database is een **verzameling van relaties (= tabellen)**. De verbanden tussen die tabellen worden gelegd door **vreemde sleutels**.

Terminologie:

Tabel Projecten

- ☐ tabel
- ☐ kolom of attribuut
- ☐ rij of tuple
- ☐ attribuutwaarde
- ☐ populatie
- ☐ domein
- ☐ null-waarde

PROJ_NR	PROJ_NAAM	LOCATIE	AFD_NR
1	Orderverwerking	Oegstgeest	7
2	Salarisadministratie	Groningen	7
3	Magazijn	Eindhoven	7
10	Inventaris	Maastricht	3
20	Personeelszaken	Eindhoven	1
30	Debiteuren	Maastricht	3

De relationele database

- ❑ Kolom of attribuut
- ❑ Populatie: de verzameling van concrete waarden van een attribuut.
- ❑ Domein: bij elke attribuut hoort een verzameling van mogelijke waarden voor dat attribuut.
- ❑ Null-waarde: **Null** is een term die vaak gebruikt wordt om een ontbrekende (of onbekende) waarde aan te duiden.

In SQL heeft NULL een speciale betekenis, en hoe ermee om te gaan is niet triviaal.

Eigenschappen van een relationele tabel

- ❑ **Een attribuut is atomair**= op het snijpunt van een kolom en een rij kan maar één waarde staan (je kan dus voor een project niet meer dan 1 afd_nr opgeven)
- ❑ **Elke rij binnen een tabel is uniek.** Een tabel kan dus geen dubbele rijen bevatten
- ❑ **De rijen binnen een tabel hebben geen specifieke volgorde.** De volgorde van de rijen is van geen belang om de tabel te kunnen begrijpen.
- ❑ **De kolomnamen zijn uniek** en de **volgorde** van de kolommen heeft **geen betekenis**.

Sleutelattributen

We onderscheiden in een relationele database **3 soorten sleutels**:

- ☐ primaire sleutel
- ☐ alternatieve sleutel
- ☐ vreemde sleutel

Sleutelattributen

Primaire sleutel (primary key)

- ❑ eigenschap tabel → rijen uniek
- ❑ uniciteit afgedwongen door primaire sleutel

Een primaire sleutel is een attribuut of een combinatie van attributen dat/die een rij uit de tabel op een unieke wijze gaat identificeren.

Sleutelattributen

Primaire sleutel

tabel Projecten

PROJ_NR	PROJ_NAAM	LOCATIE	AFD_NR
1	Orderverwerking	Oegstgeest	7
2	Salarisadministratie	Groningen	7
3	Magazijn	Eindhoven	7
10	Inventaris	Maastricht	3
20	Personeelszaken	Eindhoven	1
30	Debiteuren	Maastricht	3

tabel Opdrachten

SOFI_NR	PROJ_NR	UREN
999111111	1	31,4
999111111	2	8,5
999333333	3	42,1
999888888	1	21
999888888	2	22
999444444	2	12,2
999444444	3	10,5
999444444	1	(null)
999444444	10	10,1
999444444	20	11,8
999887777	30	30,8
999887777	10	10,2
999222222	10	34,5
999222222	30	5,1
999555555	30	19,2
999555555	20	14,8
999666666	20	(null)

Sleutelattributen

Alternatieve sleutel

Soms zijn er meerdere attributen of combinaties van attributen kandidaat om primaire sleutel te zijn.

Deze attributen of combinaties van attributen noemt men **alternatieve sleutels** of **kandidaatsleutels**.

Sleutelattributen

alternatieve sleutel

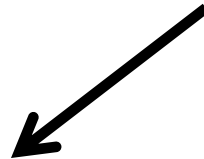
tabel Afdelingen

⚡ AFD_NR	⚡ AFD_NAAM	⚡ MGR_SOFI_NR	⚡ MGR_START_DATUM
7	Productie	999444444	22-05-2008
3	Administratie	999555555	01-01-2011
1	Hoofdvestiging	999666666	19-06-1991

Sleutelattributen

Vreemde sleutel (synoniemen: verwijssleutel, refererende sleutel, foreign key)

In een relationele database wordt het verband tussen 2 tabellen gelegd door de **vreemde sleutel**



deze verwijst naar de primaire sleutel (of uniek attribuutcombinatie) van een andere tabel

Sleutelattributen

Vreemde sleutel

Vb wat is de naam van de afdeling die project 10 ondersteunt?

tabel Afdelingen

⚡ AFD_NR	⚡ AFD_NAAM	⚡ MGR_SOFI_NR	⚡ MGR_START_DATUM
7	Productie	999444444	22-05-2008
3	Administratie	999555555	01-01-2011
1	Hoofdvestiging	999666666	19-06-1991

tabel Projecten

⚡ PROJ_NR	⚡ PROJ_NAAM	⚡ LOCATIE	⚡ AFD_NR
1	Orderverwerking	Oegstgeest	7
2	Salarisadministratie	Groningen	7
3	Magazijn	Eindhoven	7
10	Inventaris	Maastricht	3
20	Personeelszaken	Eindhoven	1
30	Debiteuren	Maastricht	3

Integriteitsregels relationele model

❑ *Key constraint:*

De primaire sleutel moet uniek zijn en uniek blijven

❑ *Entity integrity constraint:*

De primaire sleutel moet steeds een geldige waarde hebben (=waarde \neq NULL).

❑ *Referential integrity constraint:*

de populatie van een verwijssleutel moet een deelverzameling zijn van de populatie van de overeenkomstige primaire sleutel.

Integriteitsregels

tabel Afdelingen

AFD_NR	AFD_NAAM	MGR_SOFI_NR	MGR_START_DATUM
7	Productie	999444444	22-05-2008
3	Administratie	999555555	01-01-2011
1	Hoofdvestiging	999666666	19-06-1991

tabel Projecten

PROJ_NR	PROJ_NAAM	LOCATIE	AFD_NR
1	Orderverwerking	Oegstgeest	7
2	Salarisadministratie	Groningen	7
3	Magazijn	Eindhoven	7
10	Inventaris	Maastricht	3
20	Personeelszaken	Eindhoven	1
30	Debiteuren	Maastricht	3

Voorbeelden waarbij al dan niet gezondigd wordt tegen de integriteitsregels...

Bewerkingen met relaties

We bekijken aan de hand van voorbeelden:

- ☐ unie
- ☐ intersectie
- ☐ verschil
- ☐ product
- ☐ projectie
- ☐ selectie
- ☐ join

Bewerkingen met relaties: UNIE

- ❑ De unie voegt de rijen van twee relaties samen tot één relatie.
- ❑ Voor de inputrelaties geldt de beperking dat ze dezelfde attributenlijst moeten hebben

tabel r

A	B
a	1
a	2
b	1

tabel s

A	B
a	2
b	3

$r \cup (=unie) s$

A	B
a	1
a	2
b	1
b	3

Bewerkingen met relaties: INTERSECTIE

- ❑ De intersectie neemt uit twee relaties met dezelfde attributenlijst de gemeenschappelijke rijen.

Notatie:

$$\text{RELATIE3} = \text{RELATIE1} \cap \text{RELATIE2}$$

Tabel r

A	B
a	1
a	2
b	1

tabel s

A	B
a	2
b	3

$r \cap s$

A	B
a	2

Bewerkingen met relaties: VERSCHIL

- Het verschil van twee relaties RELATIE1 en RELATIE2 is de relatie RELATIE3, die alle rijen uit RELATIE1, die niet in RELATIE2 zitten bevat.

Notatie:

$$\text{RELATIE3} = \text{RELATIE1} \setminus \text{RELATIE2}$$

tabel r

A	B
a	1
a	2
b	1

tabel s

A	B
a	2
b	3

$r \setminus s$

A	B
a	1
b	1

Beide relaties hebben dezelfde attributenlijst!

Bewerkingen met relaties: **PRODUCT**

Het **cartesiaans product** van twee relaties wordt bekomen door de samenvoeging van de attributenlijsten van de twee relaties. De resulterende relatie bevat alle mogelijke rijen gevormd door samenvoeging van een rij uit R en een rij uit S.

tabel r tabel s cartesiaans product $r \times s$

A	B
a	1
a	2
b	1

C	D
a	2
b	3

A	B	C	D
a	1	a	2
a	1	b	3
a	2	a	2
a	2	b	3
b	1	a	2
b	1	b	3

Bewerkingen met relaties: PROJECTIE

- ❑ Projectie vormt een nieuwe relatie door slechts een deel van de attributen van de input relatie over te houden. Indien hierdoor dubbele rijen ontstaan, dan worden de overbodige rijen uit het resultaat verwijderd.
- ❑ Doel: een aantal kolommen uit een tabel halen

tabel r

A	B	C	D
1	1	c	d
2	1	a	b
3	4	c	b
1	2	c	d
4	4	c	d
3	2	b	b

projectie van tabel r

A	C
1	c
2	a
3	c
1	c
4	c
3	b

=

A	C
1	c
2	a
3	c
4	c
3	b

Bewerkingen met relaties: **SELECTIE**

- Een selectie neemt een deelverzameling van een relatie op basis van een voorwaarde

A	B	C	D
1	1	c	d
2	1	a	b
3	4	c	b
1	2	c	d
4	4	c	d
3	2	b	b

- We selecteren enkel de rijen met waarde 1 voor A:

A	B	C	D
1	1	c	d
1	2	c	d

Bewerkingen met relaties: JOIN

Join combineert twee relaties via een combinatie van cartesisch product en selectie. De enige vereiste is dat in beide relaties een attribuut voorkomt waartussen vergelijking ($<$, $>$, $=$, $\sim=$, \geq , \leq) mogelijk is.

A	B	C	D
1	1	c	d
2	1	a	b
3	4	c	b
1	2	c	d
4	4	c	d
3	2	b	b

A	E
1	e
2	f
3	g

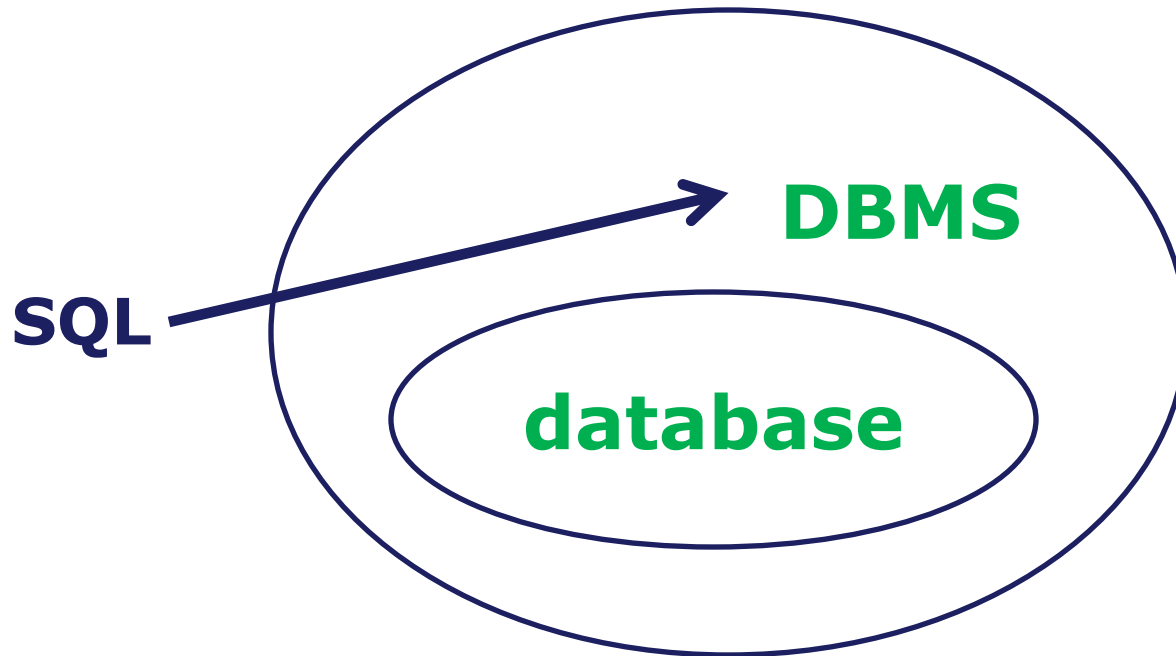
Leidt tot:

A	B	C	D	E
1	1	c	d	e
1	2	c	d	e
2	1	a	b	f
3	4	c	b	g
3	2	b	b	g

Wat is SQL?

SQL staat voor **S**tructured **Q**uery **L**anguage

SQL is een taal om met een **relationeel** DBMS te communiceren



Eigenschappen van SQL

- ❑ SQL is een **declaratieve** taal :

je geeft in je instructie aan wat je wil bereiken en niet de stappen die moeten gezet worden om je doel te bereiken (↔ procedurele taal)

- ❑ Je kan SQL **interactief** gebruiken:

je geeft op het scherm je instructie en krijgt onmiddellijk je antwoord

- ❑ Of je kan SQL **embedded** gebruiken:

in dat geval is de instructie ingebed in een programma (in COBOL of C of VB.net of...)

Subcategorieën SQL

❑ **data retrieval (periode 1-2)**

het ophalen van informatie uit de database (SELECT statement)

❑ **data manipulation language (periode 1-2)**

instructies om de inhoud van de tabellen te wijzigen (UPDATE,DELETE,INSERT)

❑ **data definition language (beperkt in periode 1-2)**

instructies om objecten in de databank te creëren (CREATE, ALTER, DROP..)

❑ **data control language (periode 1)**

instructies die mee instaan voor de beveiliging van de databank (GRANT EN REVOKE)

❑ **transaction control (periode 1)**

instructies die ervoor zorgen dat wijzigingen die zeker samen moeten slagen of falen gegroepeerd worden (COMMIT,ROLLBACK, SAVEPOINT)

Database ontwerp

Het eindresultaat van een database ontwerp is een database model.

- ❑ goed database model → goed database systeem
- ❑ het database model moet
 - ❑ aan de noden van alle gebruikers voldoen
 - ❑ door de eindgebruiker begrepen worden
 - ❑ voldoende details en specificaties bevatten zodat de database met die informatie fysisch kan aangemaakt worden

Database ontwerp

Voor een database **ontwerp zijn er 2 benaderingen:**

1.

De bottom-up benadering vertrekt vanuit attributen die door het analyseren van de verbanden tussen die attributen in relaties gegroepeerd worden.

Normalisatie representeert een bottom up benadering voor database design.

2.

De **top-down benadering** gaat van algemeen naar meer specifiek. De analist vertrekt van wat algemeen nodig is. Door vraagstelling aan de eindgebruiker worden de details toegevoegd aan het model.

Fases in het database ontwerp

Database ontwerp is opgebouwd uit 3 belangrijke fases:

- ☐ de conceptuele fase
- ☐ de logische fase
- ☐ de fysische fase

Fases in het database ontwerp

de **conceptuele fase**

In deze fase bouwt men een model dat alle gegevens bevat die de organisatie gebruikt.

Wat heeft de gebruiker nodig los van een DBMS of applicatie?

Deze fase is de voorbereiding voor de logische fase.

Fases in het database ontwerp

De **logische fase**

In de logische fase wordt het database model aangepast aan het onderliggend data model dat gebruikt zal worden (relationeel, hiërarchisch, netwerk, object georiënteerd..)

Er wordt regelmatig teruggekoppeld naar de eindgebruiker om te toetsen of aan zijn eisen wordt voldaan en er worden business rules vastgelegd.

Het logisch model zorgt voor de nodige informatie voor de 3^e fase: de fysische fase

Fases in het database ontwerp

de **fysische fase** = implementatie

Hier wordt voor het eerst het doel DBMS geïdentificeerd (Oracle, SQL server...).

In de fysische fase worden er beslissingen genomen rond de performantie van het systeem en dat kan de structuur van het logische data model beïnvloeden.

Voor een relationeel model betekent dit concreet:

- ☐ een set van relationele **tabellen creëren** met bijbehorende constraints, die afgeleid kunnen worden uit het logisch data model.
- ☐ de **storage structuren vastleggen**, bepalen hoe de database op een performante manier benaderd kan worden (indexen plaatsen ..).
- ☐ het database systeem **beveiligen**.

Database ontwerp - het ERD

hoe het resultaat van het ontwerp grafisch voorstellen?

ERD staat voor **E**ntity **R**elationship **D**igram.

Het is een grafische voorstelling van de verschillende entiteiten van de te beschrijven realiteit, samen met de verbanden tussen deze entiteiten.

Database ontwerp - het ERD

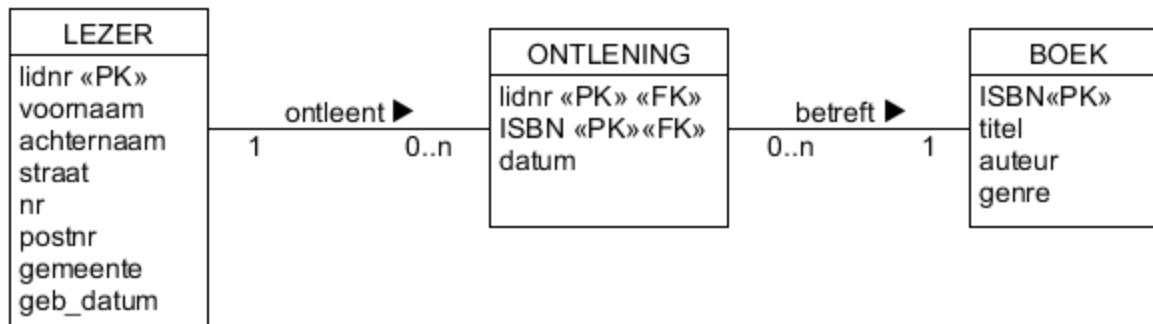
Database BIBLIOTHEEK

LEZER (**lidnr**,naam,adres,gebdat)

BOEK(**ISBN**,titel,auteur)

ONTLENING(**lidnr**,**ISBN**,datum) (*)

FK FK



(*) We onderstellen dat een boek door een lid maar 1 x kan ontleend worden

Database ontwerp - het ERD

Een entiteit wordt weergegeven door een rechthoek

De naam van de entiteit wordt in de rechthoek geschreven

De lijnen tussen de rechthoeken geven de relaties weer tussen de entiteiten

de relatie wordt kort beschreven (bij voorkeur werkwoord in enkelvoudsvorm)

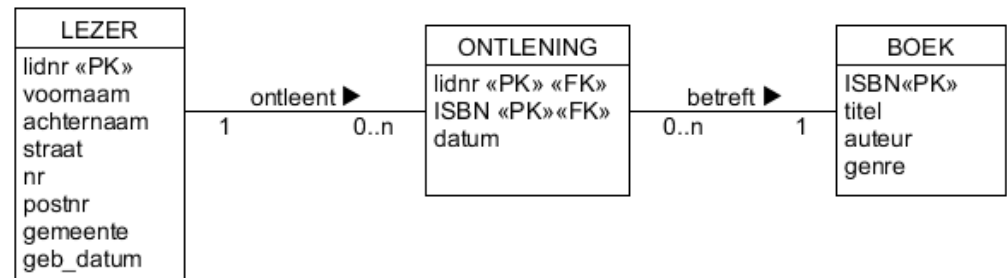
Cardinaliteit:

0..n = 0, 1 of meer

1..n = 1 of meer

1 = exact één

0..1 = 0 of één



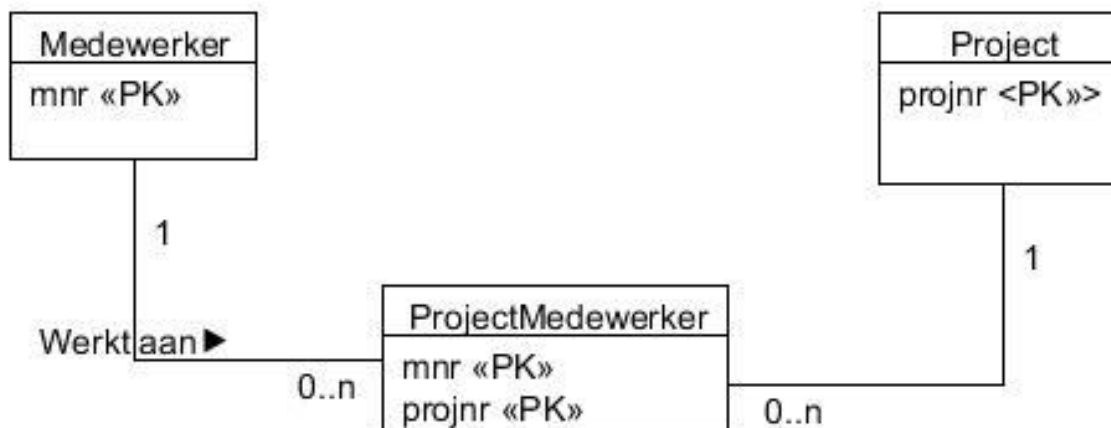
Database ontwerp - het ERD

Bemerkingen:

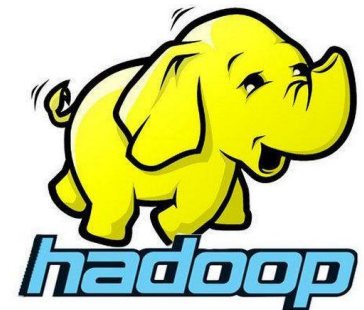
- ❑ De veel-kant staat altijd aan de kant van de entiteit met de vreemde sleutel
- ❑ In een relationeel datamodel zijn veel op veel relaties niet toegelaten:



Hoe oplossen? Door tussentabel te definiëren



Database en database management systeem



Databases en Database systems

- ❑ Databanken zijn complex geworden
- ❑ Databanken moeten aanpasbaar zijn aan de nieuwe informatiebehoeften:

Mobile development
Computing

Cloud

