H1 Databanken gekaderd

Gegevensmanagement
DBMS
Historiek

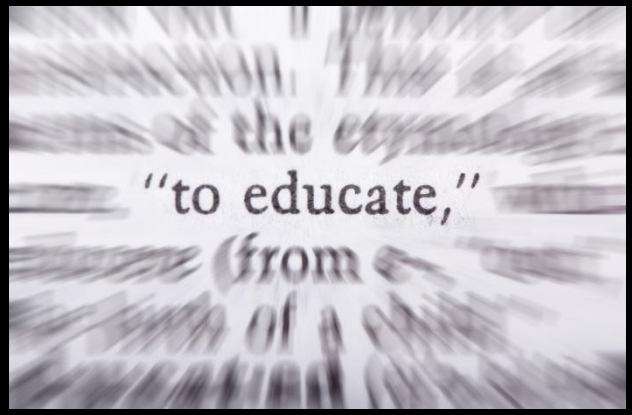
HO GENT

Definities

1. Database

2. Databasemanagementsysteem

3. Databaseprogramma's



HO
GENT

Definities

1. **Database** (databank)

- Een databank bestaat uit **verzamelingen van persistente gegevens** die gebruikt worden door de softwareapplicaties van een bedrijf en **beheerd** worden door een **DBMS** (J. Date).
- Een gedeelde verzameling van **logisch met elkaar verbonden gegevens** en hun **beschrijving**, ontworpen om aan de informatienoden van een organisatie te voldoen (O. Connolly).

We onthouden:

- verzameling van gegevens
- digitaal opgeslagen
- met onderlinge relaties
- met bepaalde betekenis
- voor een bepaalde groep van gebruikers

HO GENT

Definities

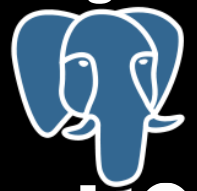
2. **Databasemanagementsysteem** (DBMS)

   = verzameling van programma's waarmee:

   – een DB wordt gecreëerd en beheerd

   – gegevens in de DB worden geladen, gewijzigd en opgevraagd

PostgreSQ

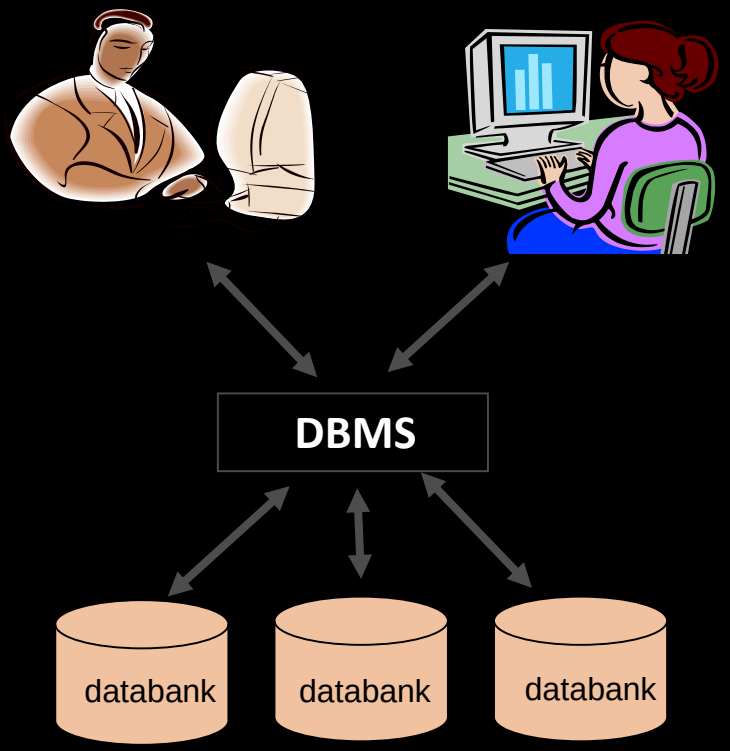


HO
GENT

Definities

2. **Databasemanagementsysteem** (DBMS)

   = interface tussen gebruikersprogramma's en de databank

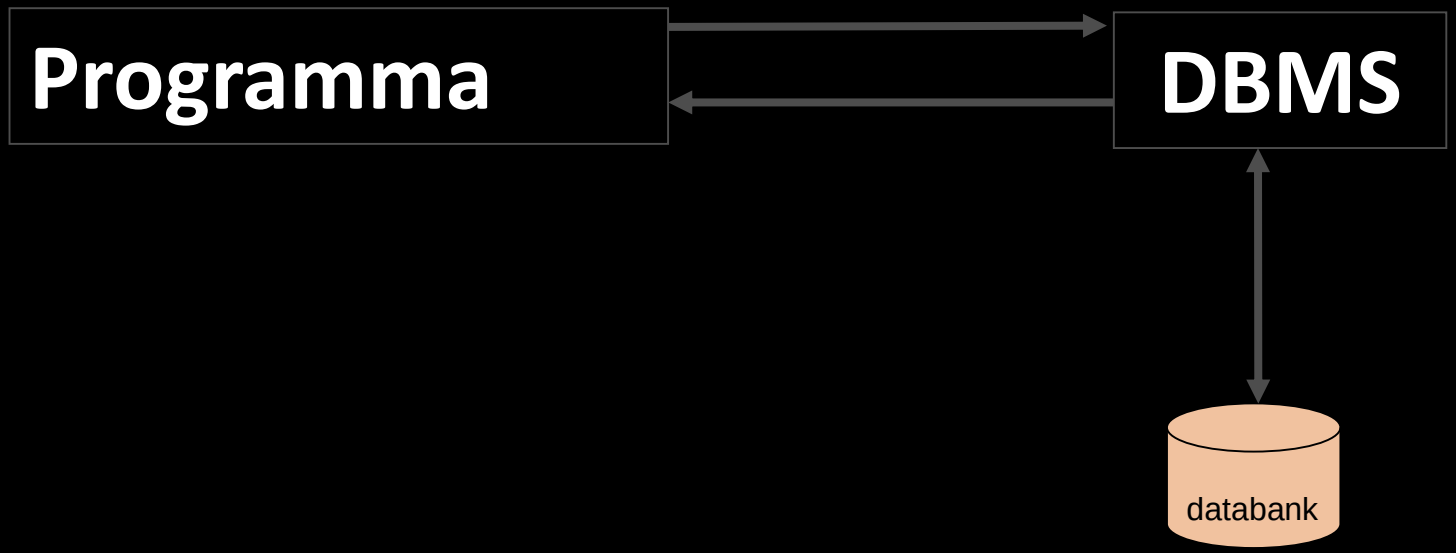


HO GENT

Definities

3. Databaseprogramma

= computerprogramma dat gegevens uitwisselt met de database (maar nooit rechtstreeks!).



HO GENT

Toepassingen

- Databases zijn niet meer weg te denken uit ons dagelijks leven.

- Wat je tegenwoordig ook doet, ergens kom je rechtstreeks of onrechtstreeks in aanraking met een database.

HO
GENT

Toepassingen

- Betalen aan kassa:
  - ➔ producten worden ingescand
  - ➔ prijs wordt ‘gelezen’
  - ➔ deze informatie komt uit een database

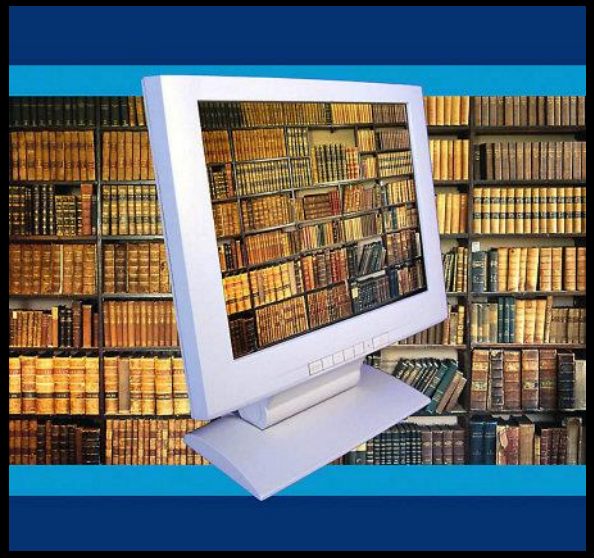
HO GENT

Toepassingen

- Bibliotheek:

  opvragen of een bepaald boek aanwezig is of niet

  ➔ informatie van alle boeken zit in een database



HO GENT

Toepassingen

- Online een vlucht boeken:

  opvragen beschikbaarheid vlucht

  ➔ informatie van alle vluchten zit in een database

HO GENT

Toepassingen

Talloze andere:

- Voorraad checken

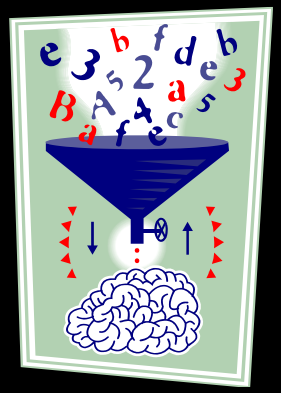
- Websites / Internet
- Betalingen met je bankkaart

- …

HO GENT

Gegevensmanagement

Gegevens worden informatie



A5

B12

C 21

Leeftijdscategorie	*Maximumleeftijd*
A	5
B	12
C	21

door er een <u>betekenis</u> aan te geven

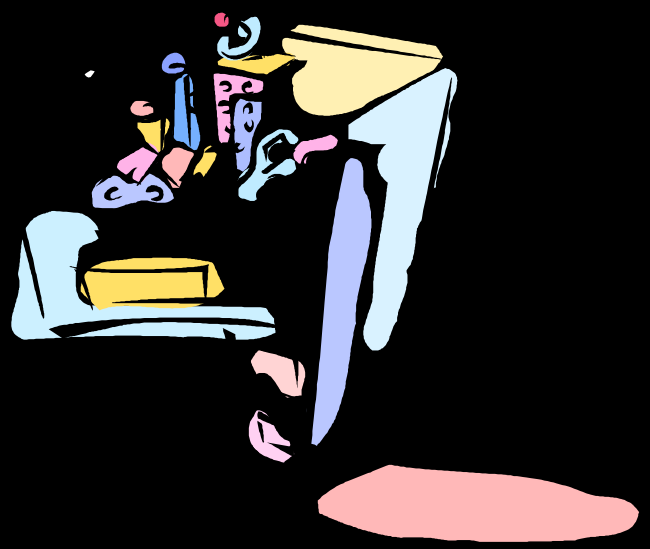
HO GENT

Gegevensmanagement

Gegevensmanagement =

- verzamelen
- opslaan
- terugvinden

} van gegevens



HO GENT

Gegevensmanagement

Gegevensmanagement **vroeger** :

- Stand-alone applicaties

- elke toepassing definieert en gebruikt zijn eigen bestanden

  ➔ Verschillende **bestanden** worden gedefinieerd en geïmplementeerd **zonder relaties** tussen beide. Het is in de applicatieprogramma's dat de relaties tussen de data uit de verschillende bestanden moeten worden gelegd/gevonden.

HO
GENT

Gegevensmanagement

Enkele nadelen van stand-alone applicaties :

1. Verspreiding en isolatie van data
2. Redundantie
3. Incompatibiliteit
4. Fixed queries

➔ **Inconsistentie en inefficiëntie !**

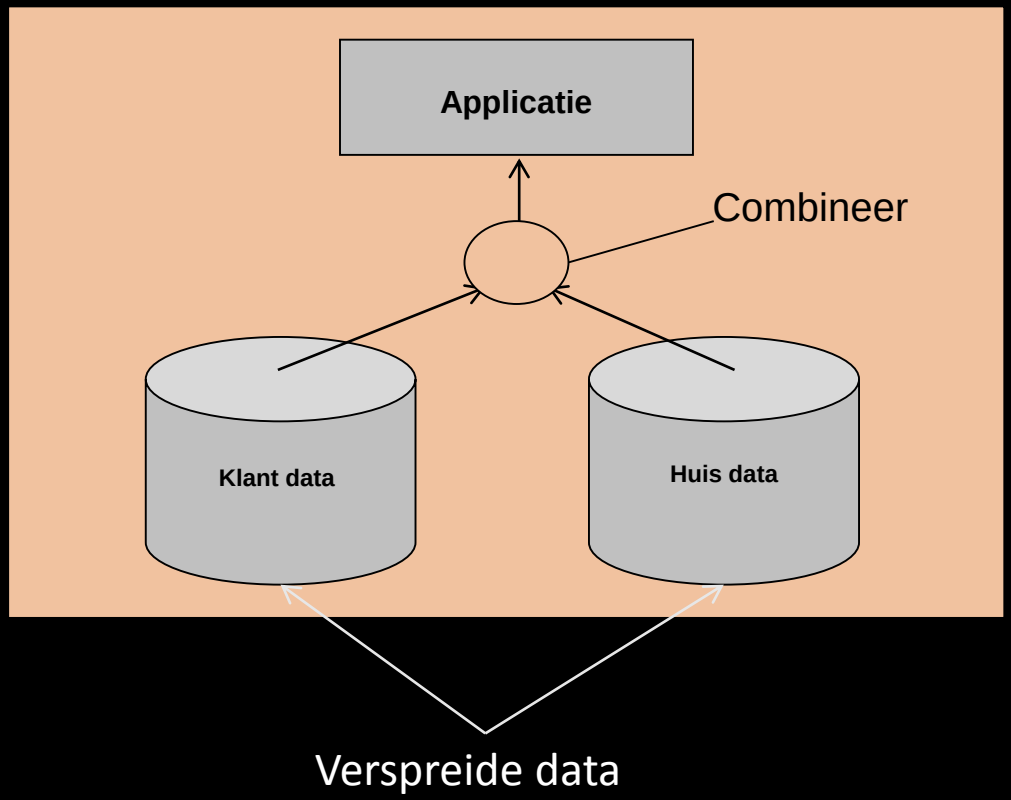
HO GENT

Gegevensmanagement

Nadeel 1: Verspreiding en isolatie van data

Voorbeeld: applicatie voor immobureau

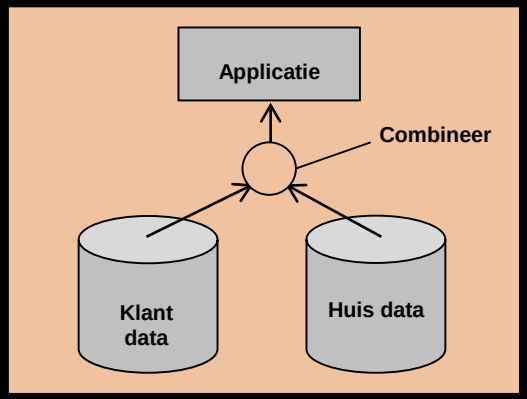
Doel: Per klant de huizen opvragen die bij zijn wensen passen



HO GENT

Gegevensmanagement

Nadeel 1: Verspreiding en isolatie van data



Hoe oplossen?

Een mogelijkheid:

1. Per klant een tijdelijk bestand aanmaken.

2. Voor elke klant: record per record de huisdata overlopen.

3. Elk passend record wegschrijven in het tijdelijk bestand van deze klant.

HO GENT

Gegevensmanagement

Nadeel 1: Verspreiding en isolatie van data

Gevolgen?

1. Veel programmeerwerk
2. Foutgevoelig
3. Het kost veel tijd om dit uit te voeren (execution time)

HO GENT

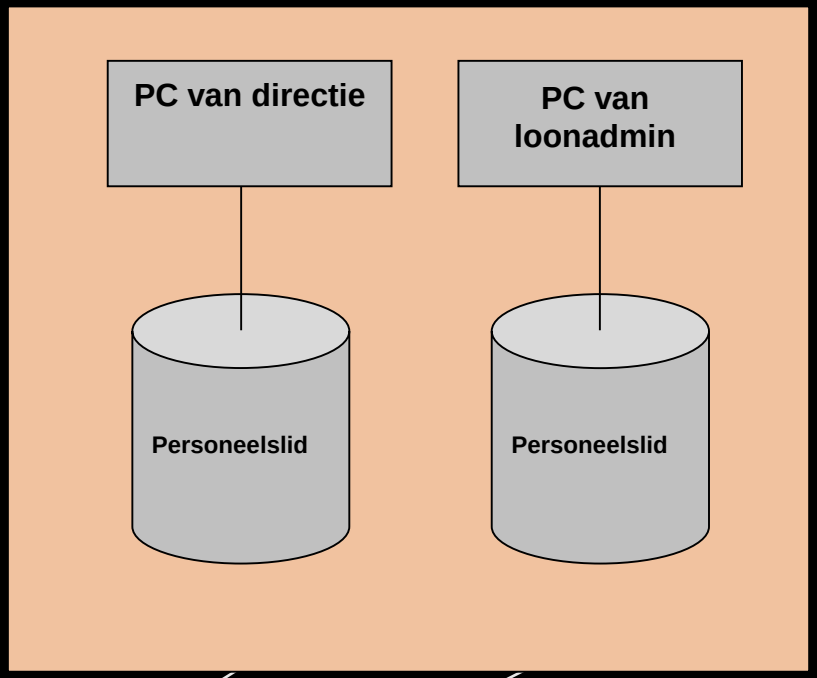
Gegevensmanagement

Nadeel 2: Redundantie = duplicatie van data

Voorbeeld:

➔ PC van de directie heeft een bestand Personeelslid

➔ PC van de loonadministratie heeft een kopie van het bestand Personeelslid



Zelfde gegevens verspreid over meerdere bestanden!

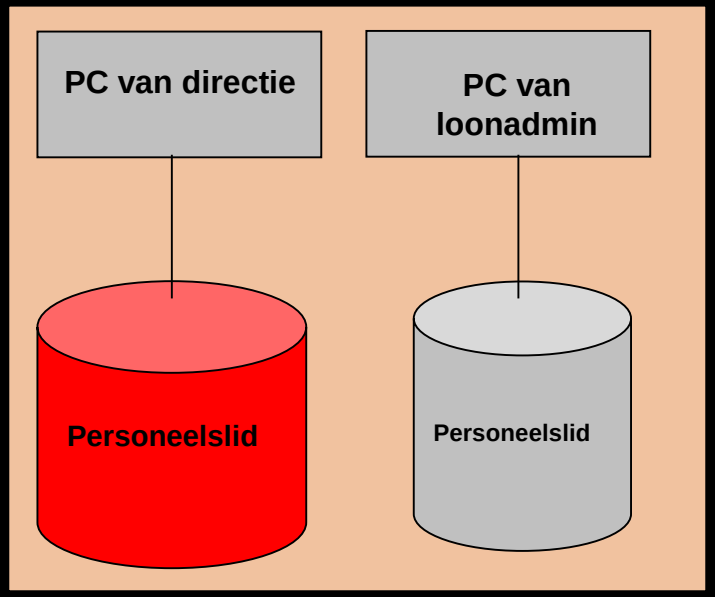
HO GENT

Gegevensmanagement

Nadeel 2: Redundantie = duplicatie van data

➔ Wat indien de directeur aan personeelslid Pieter Jansens 20 € opslag geeft?

➔ Loonsopslag zal opgenomen zijn in bestand van de directie maar niet in bestand van de loonadministratie.



Inconsistentie !

HO GENT

Gegevensmanagement

Nadeel 2: Redundantie = duplicatie van data

Hoe oplossen?

Mogelijkheid 1: Manueel

➔ Directie geeft elke wijziging door aan de loonadministratie, zodanig dat zij dit in hun bestand kunnen aanpassen en vice versa.

Grote kans op fouten (mensen zijn geen machines!)

HO
GENT

Gegevensmanagement

Nadeel 2: Redundantie = duplicatie van data

Hoe oplossen?

Mogelijkheid 2: Kopie

➔ Het bestand Personeelslid op de pc van de directie wordt dagelijks gekopieerd naar de pc van de loonadministratie.

Geen up-to-date informatie bij de loonadministratie!
Wat indien de loonadministratie gegevens wijzigt?

HO GENT

Gegevensmanagement

Nadeel 2: Redundantie = duplicatie van data

Hoe oplossen?

Mogelijkheid 3: Synchroniseren

➔ Synchronisatieprogramma schrijven om gegevens van beide bestanden te vergelijken en up te daten.

1) Geen up-to-date informatie in beide bestanden tot de synchronisatie opgestart wordt.

HO GENT

Gegevensmanagement

Nadeel 2: Redundantie = duplicatie van data

Hoe oplossen?

Mogelijkheid 3: Synchroniseren

➔ Synchronisatieprogramma schrijven om gegevens van beide bestanden te vergelijken en up te daten.

2) Wat indien een gegeven in beide bestanden een verschillende waarde heeft ➔ wat is de juiste waarde?

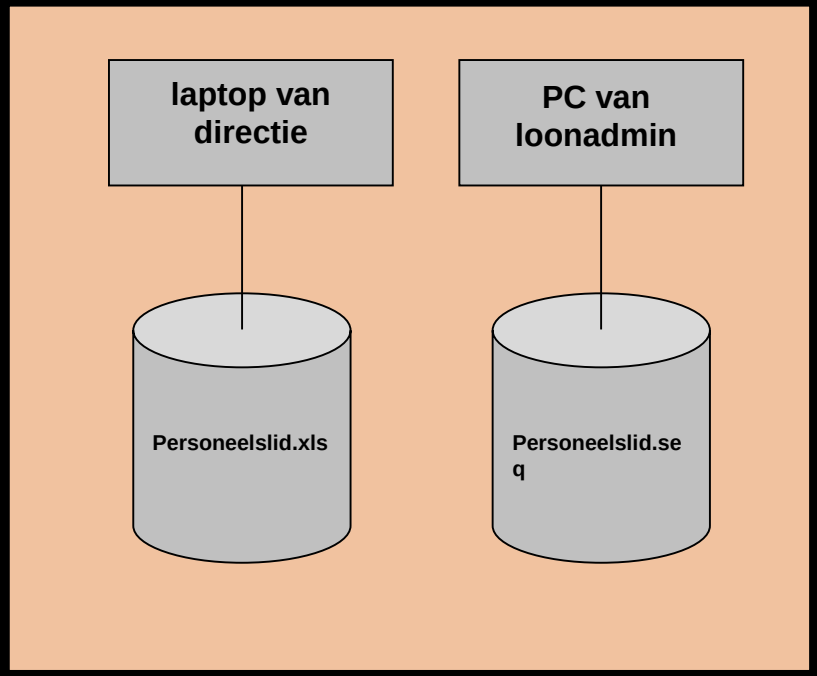
HO GENT

Gegevensmanagement

Nadeel 3: Incompatibiliteit

Voorbeeld:

- PC van de loonadministratie heeft een programma dat bestand Personeelslid.seq gebruikt (sequentieel bestand)
- Laptop van de directeur heeft enkel office-programma's. De gegevens van de personeelsleden moeten van het sequentieel bestand omgezet worden naar een excel-file



Conversieprogramma nodig!

HO GENT

Nadeel 4: Fixed queries

- De manier om de gegevens op te vragen ligt vast in een programma.
- Indien een gebruiker de data op een andere manier wil opvragen moet er een nieuw programma geschreven worden.

Weinig flexibiliteit!

HO GENT

Gegevensmanagement

Conclusie?

- De traditionele bestandsgebaseerde oplossing is verre van perfect.
- Het brengt veel extra problemen en werk met zich mee.

Is er een betere oplossing?

HO GENT

Gegevensmanagement

Bemerking!

Bijna alle problemen komen voort uit :

- Datadefinitie wordt bewaard in de applicaties.
- Alle controle van de datatoegang en datamodificatie bevindt zich in de applicaties.

Een nieuwe oplossing moet minstens deze problemen aanpakken!

HO GENT

Gegevensmanagement

Een oplossing voor de problemen bij een bestandsgebaseerde oplossing vond men in een **DBMS (DataBase Management System)**.

Deze systemen worden vandaag overal gebruikt.

HO GENT

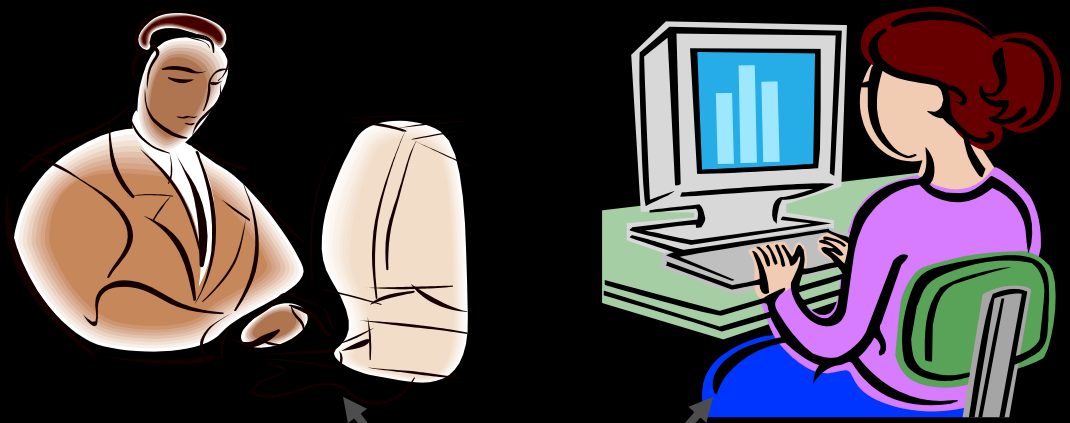
DBMS

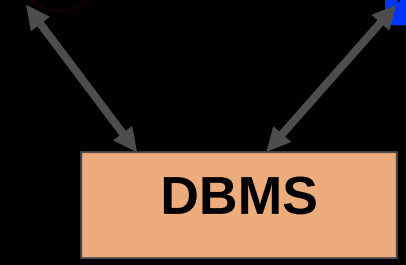
In volgende delen zullen we verduidelijken waaruit een DBMS bestaat en hoe het de problemen van bestandsgebaseerde applicaties oplost:

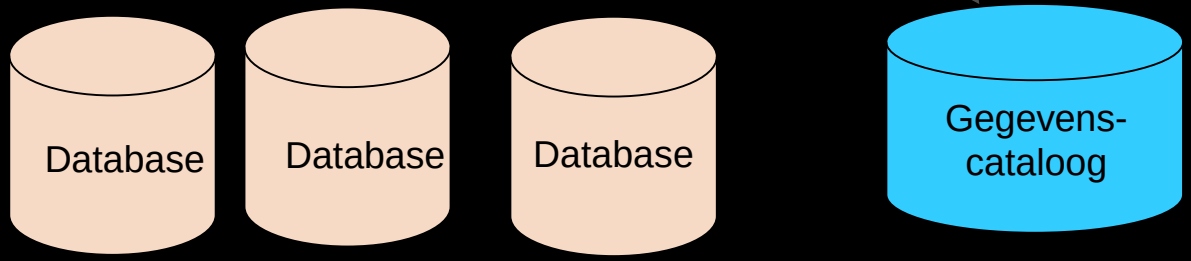
1. Elementen van een DB-Systeem
2. Modellen
3. Rollen binnen een DBMS
4. Voor-en nadelen

HO GENT

DBMS: Onderdelen van een DBMS-systeem



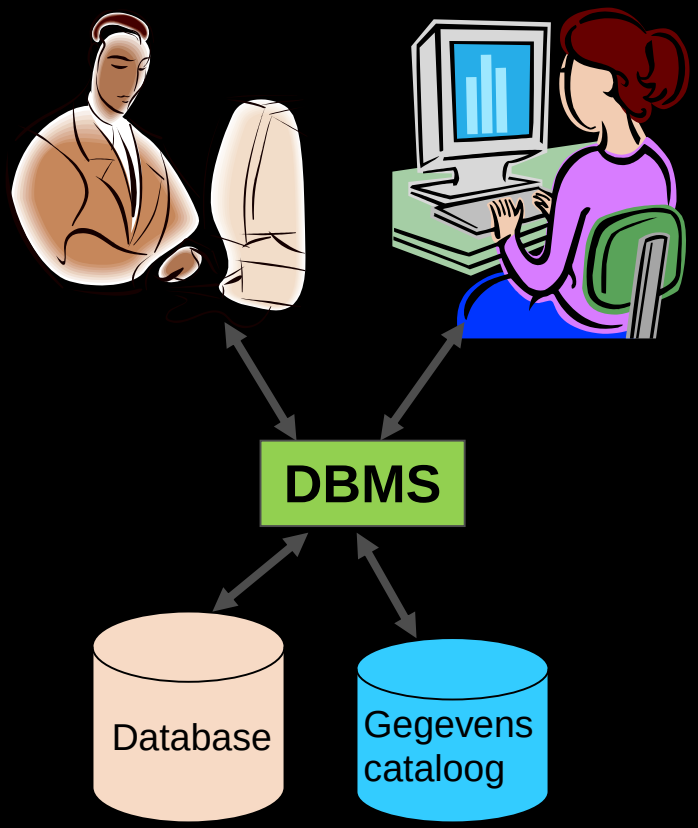




HO
GENT

DBMS: Onderdelen van een DBMS-systeem

DBMS



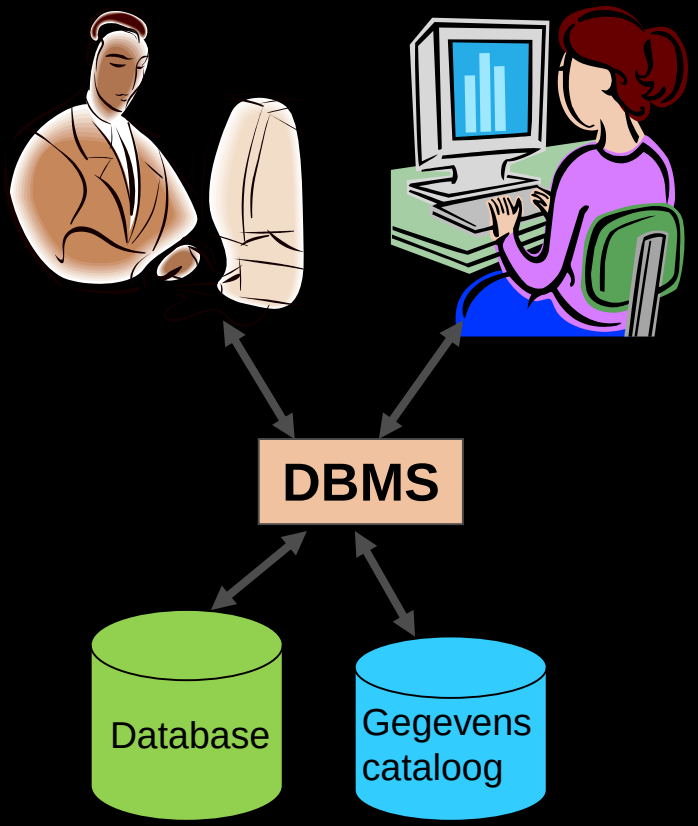
DataBase Management System biedt databasetalen en interfaces aan

- voor het definiëren van gegevens (definities uit de 3 modellen)
- om de gegevens te manipuleren
- voor het bewaken van de integriteit
- voor beveiliging, back-up en recovery
- beheerstools: statistieken omtrent efficiëntie en effectiviteit

HO GENT

DBMS: Onderdelen van een DBMS-systeem

Database



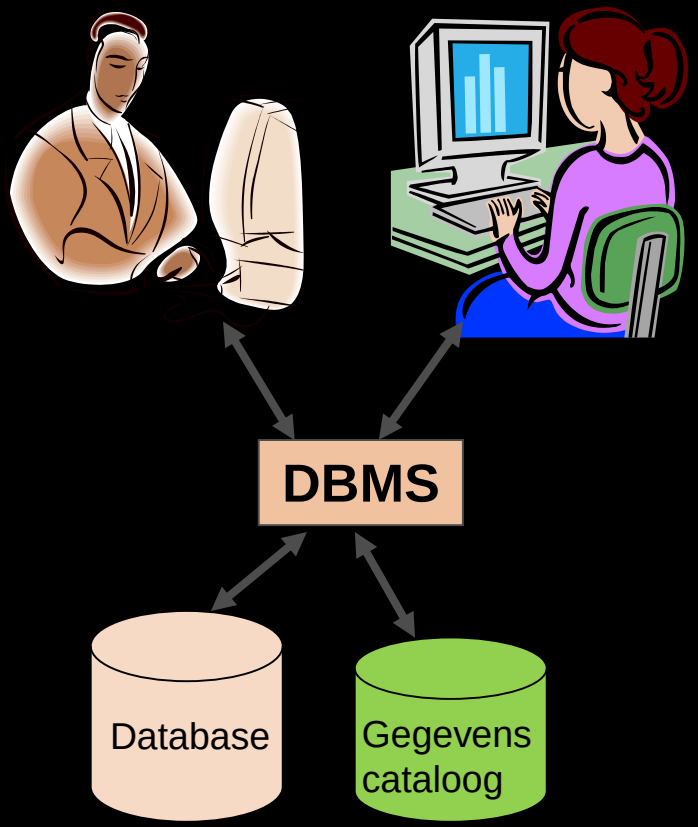
bevat de eigenlijke gegevens die kunnen opgevraagd worden via :

- queries
- DB-programma's

HO GENT

DBMS: Onderdelen van een DBMS-systeem

Gegevenscataloog



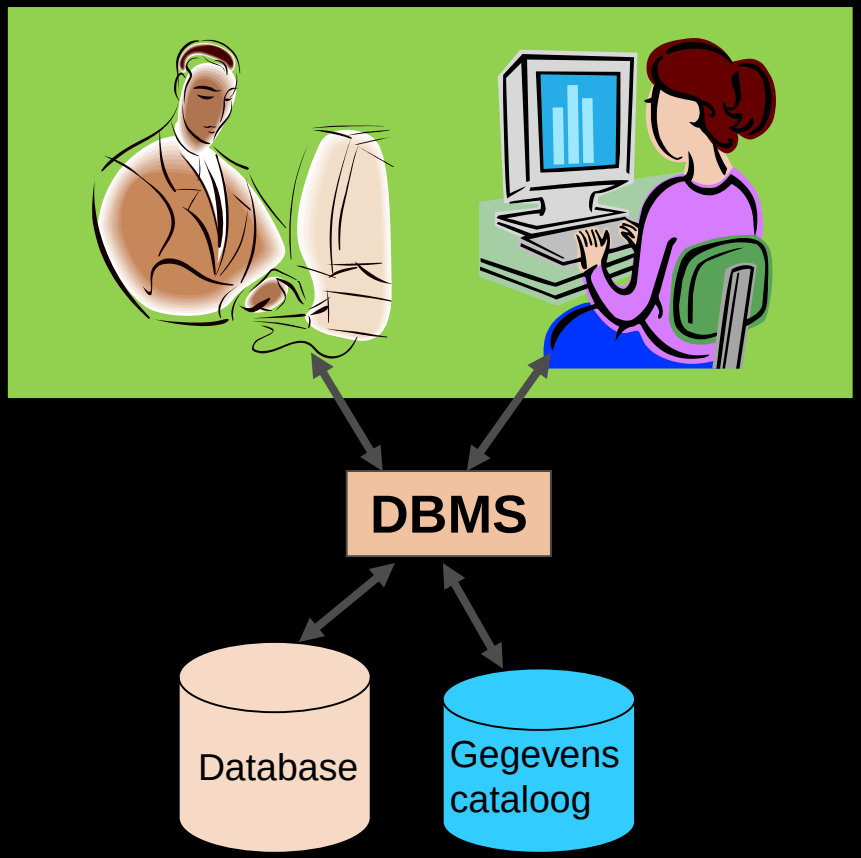
bevat <u>definities van</u>:

- **Logisch model**
  = datadefinities van alle gegevens.
- **Fysiek model**
  = hoe de gegevens fysiek zijn opgeslagen.
- **Externe modellen**
  = view van gebruikers op de gegevens.

HO GENT

DBMS: Onderdelen van een DBMS-systeem

Databaseprogramma



- Computerprogramma dat gegevens uitwisselt met de database.
- Gebruikers/programma's communiceren nooit rechtstreeks met de database.
- Gebruikers/programma's passeren steeds via het DBMS.

HO GENT

DBMS: Modellen

- Logisch model

  beschrijving van alle gegevens in de DB:
  - objecten
  - verbanden tussen de objecten
  - integriteitsregels

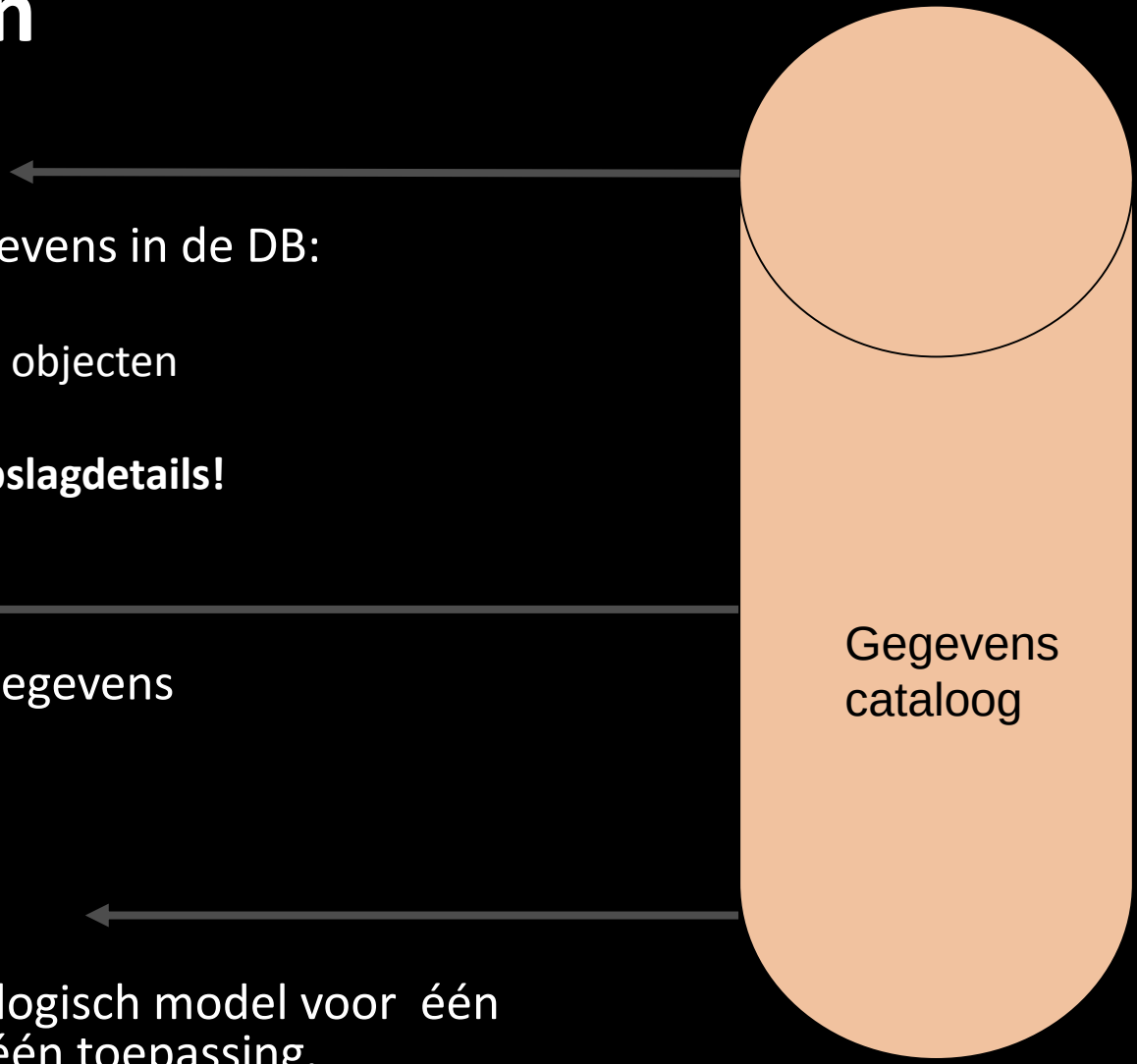
  Geen implementatie- of opslagdetails!

- Fysiek model

  beschrijving van hoe de gegevens fysiek opgeslagen zijn.

- Externe modellen

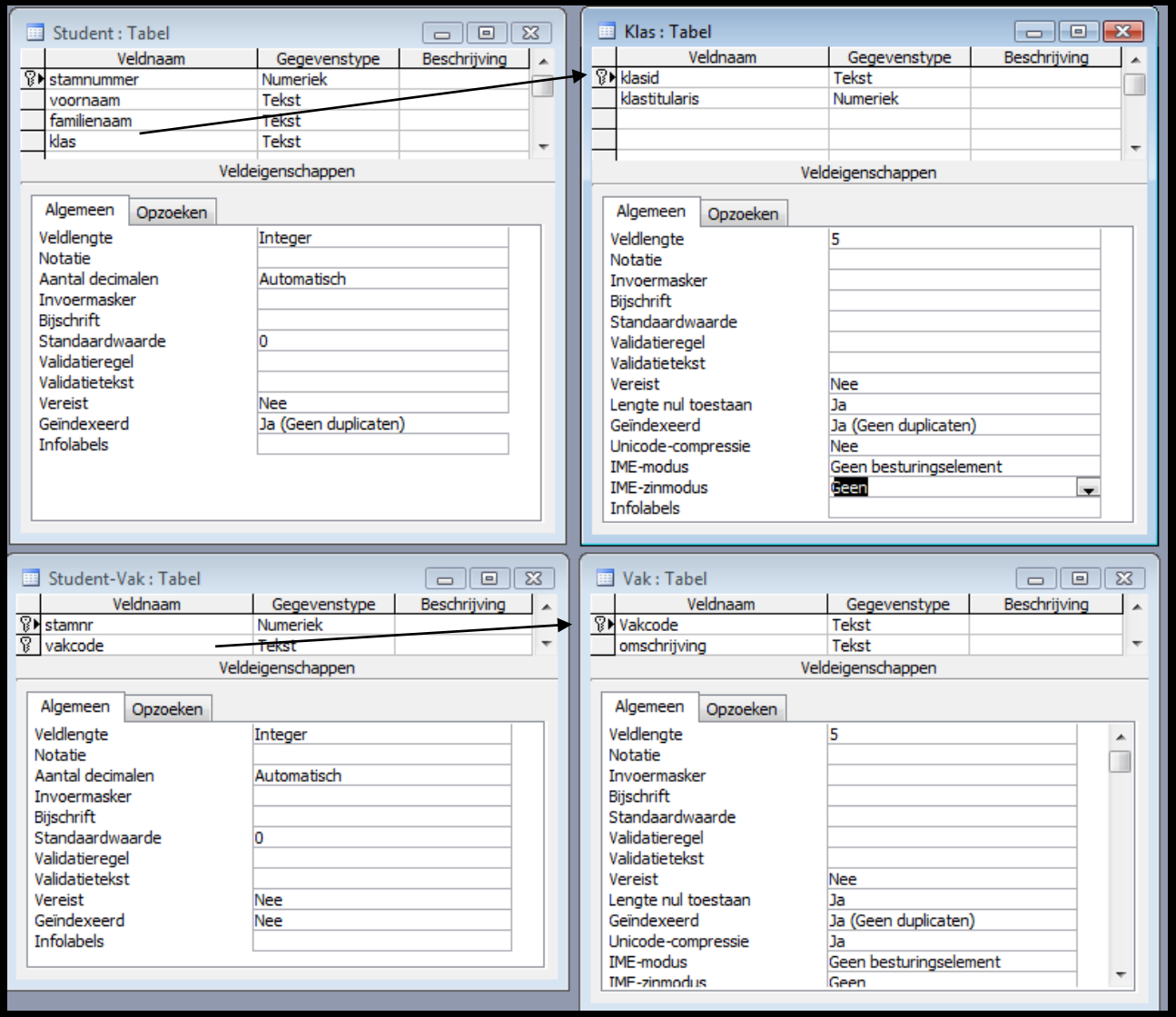
  deelverzameling van het logisch model voor één gebruikersgroep of voor één toepassing.

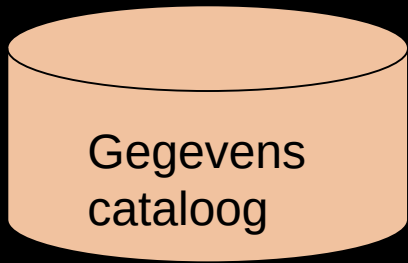


HO GENT

DBMS: Modellen

Logisch model





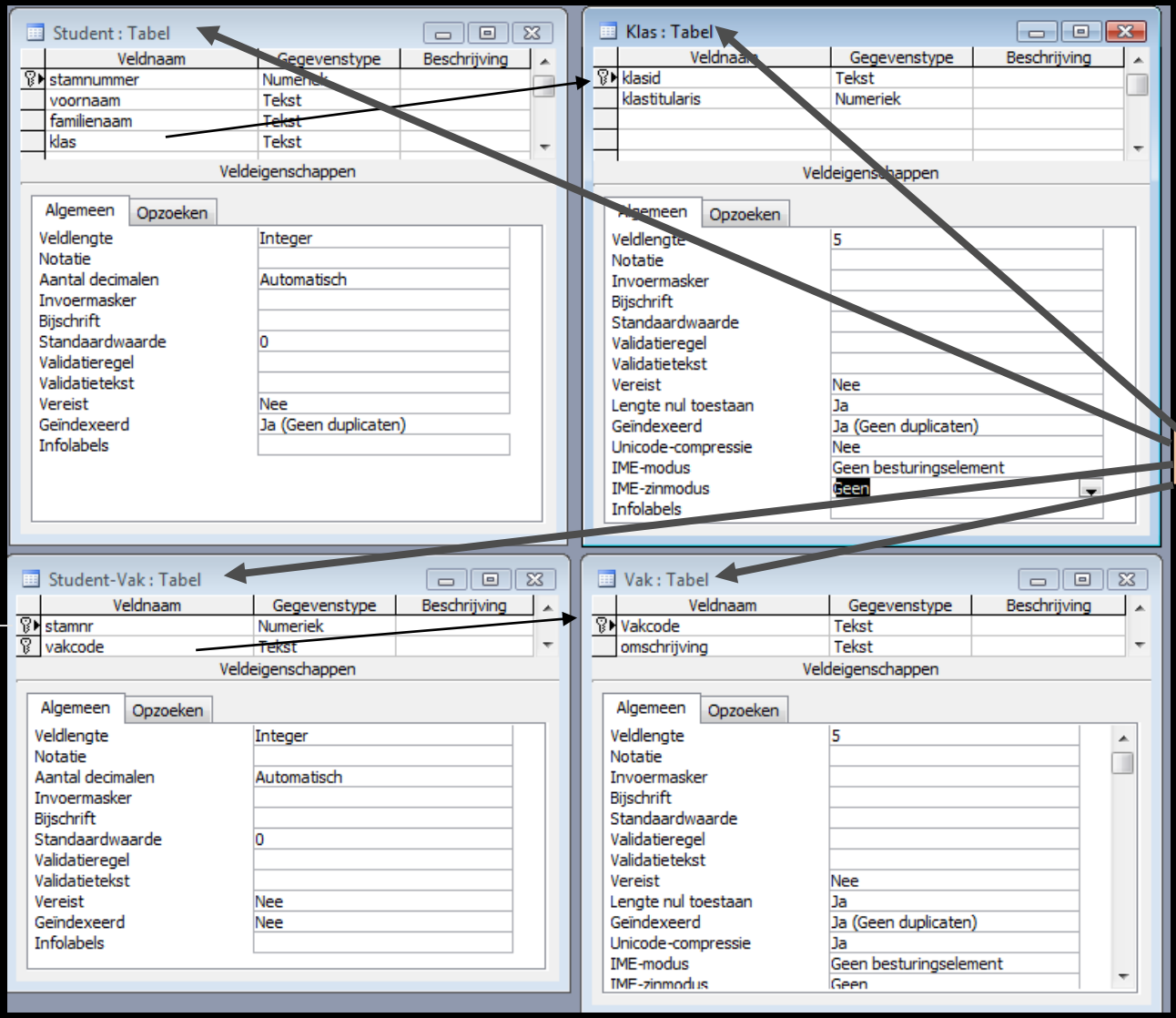
Beschrijving van alle gegevens in de DB:

- objecten
- verbanden
- integriteitsregels

HO GENT

DBMS: Modellen

Logisch model



Gegevens cataloog

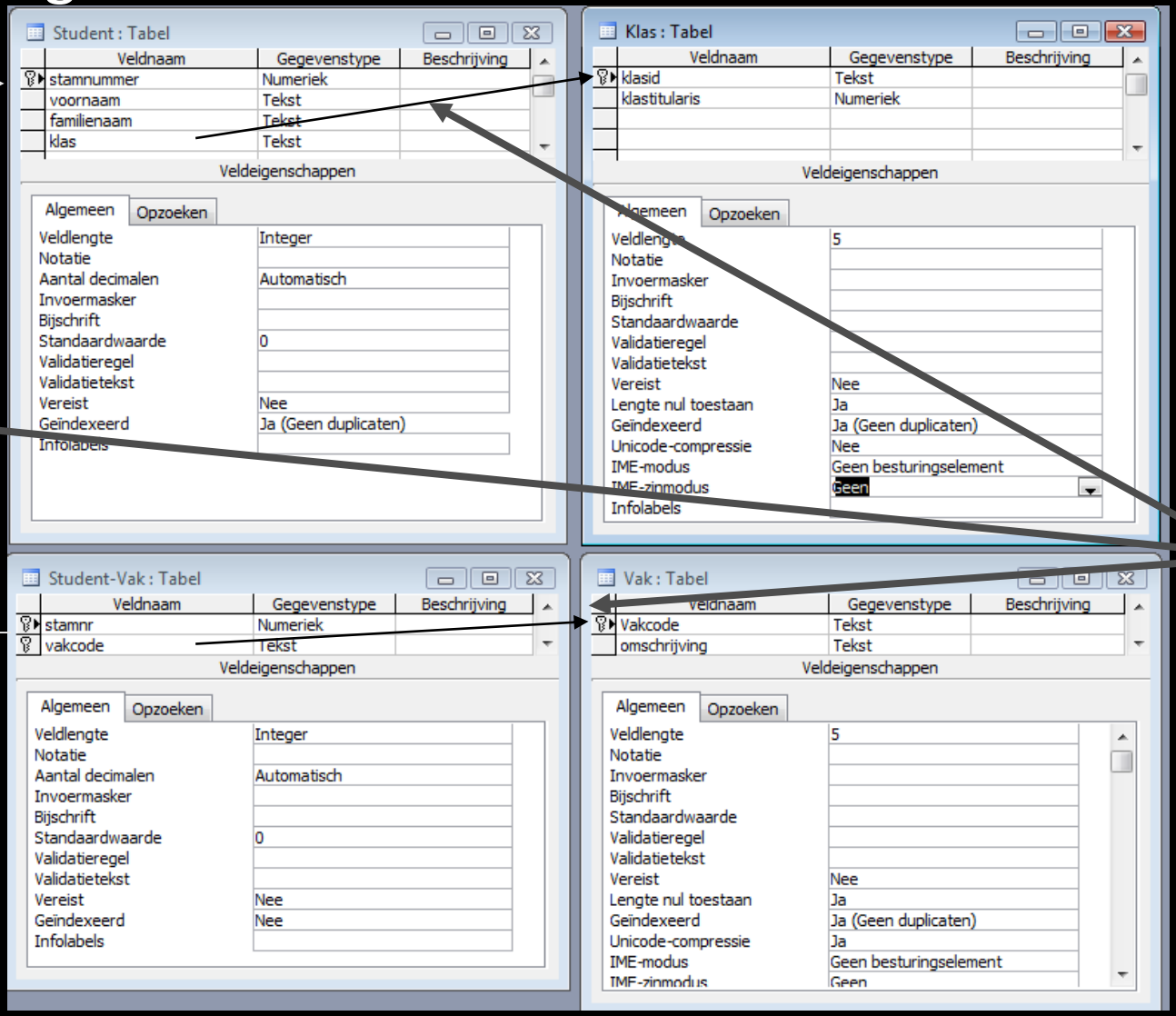
Beschrijving van alle gegevens in de DB:

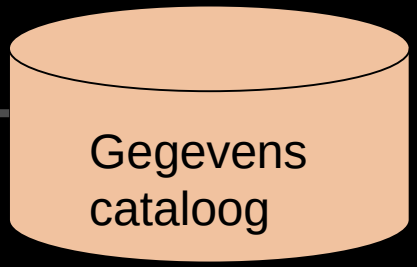
- objecten
- verbanden
- integriteitsregels

HO GENT

DBMS: Modellen

Logisch model





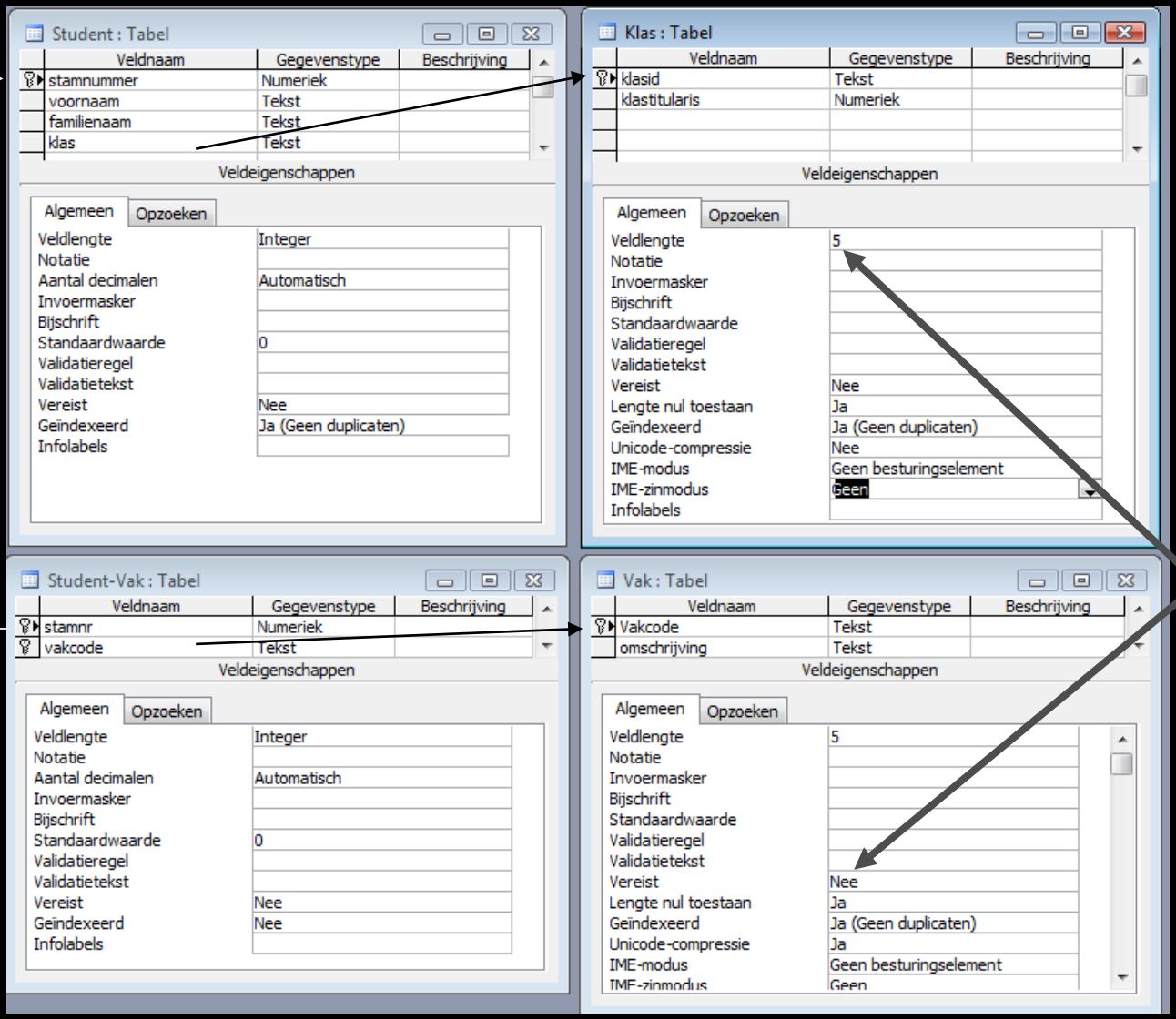
Beschrijving van alle gegevens in de DB:

- objecten
- verbanden
- integriteitsregels

HO GENT

DBMS: Modellen

Logisch model



Gegevens cataloog

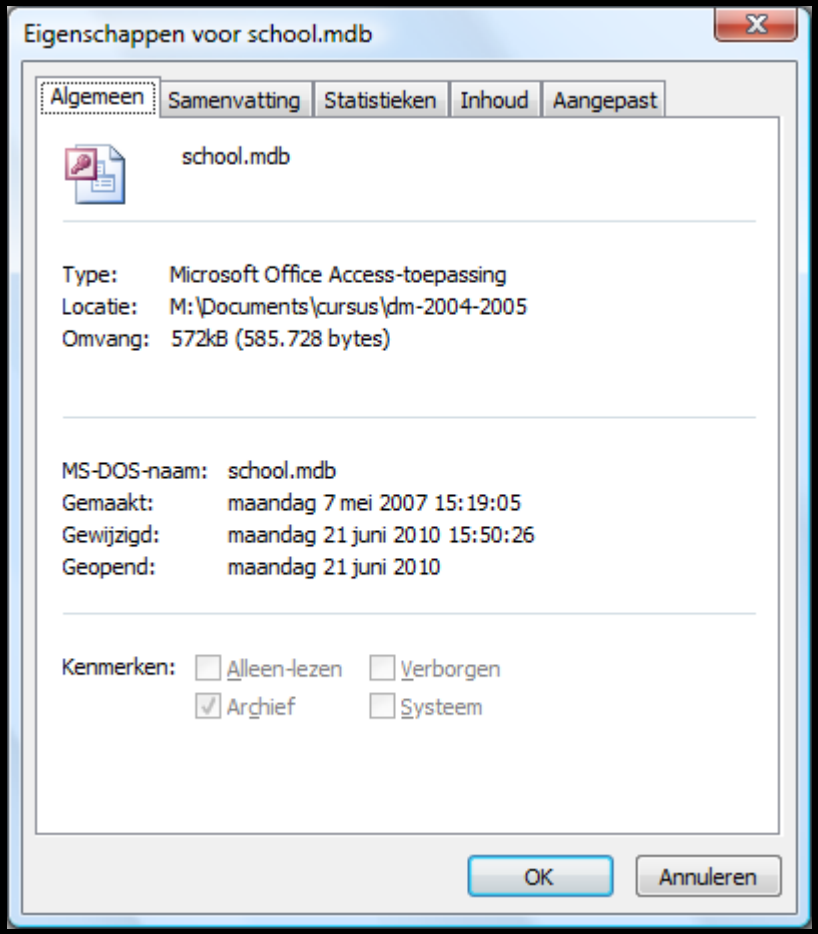
Beschrijving van alle gegevens in de DB:

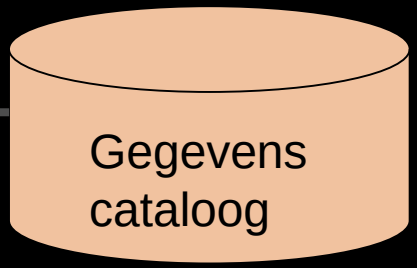
- objecten
- verbanden
- integriteitsregels

HO GENT

DBMS: Modellen

Fysiek model





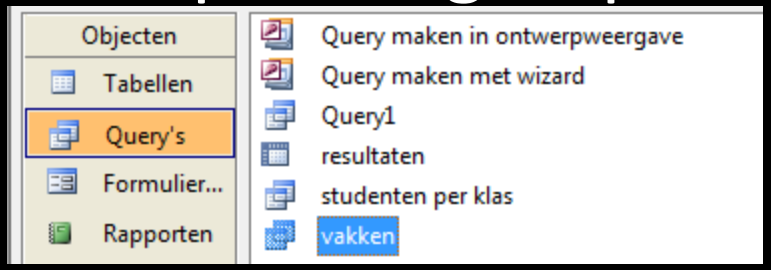
Beschrijving van hoe de gegevens fysiek opgeslagen zijn.

HO
GENT

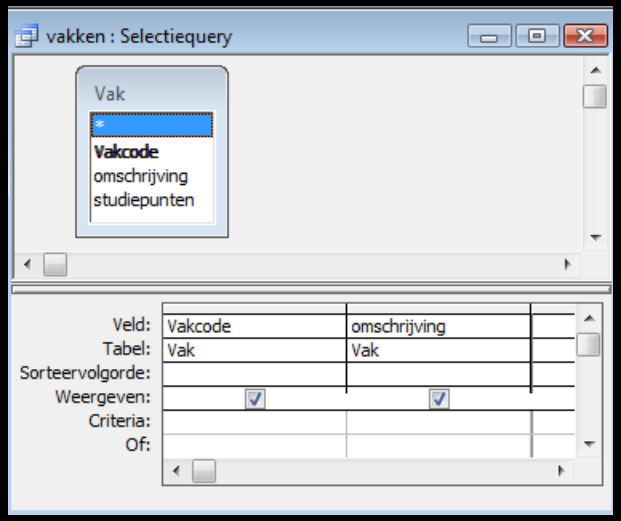
DBMS: Modellen

Externe modellen

Views van een welbepaalde groep van gebruikers



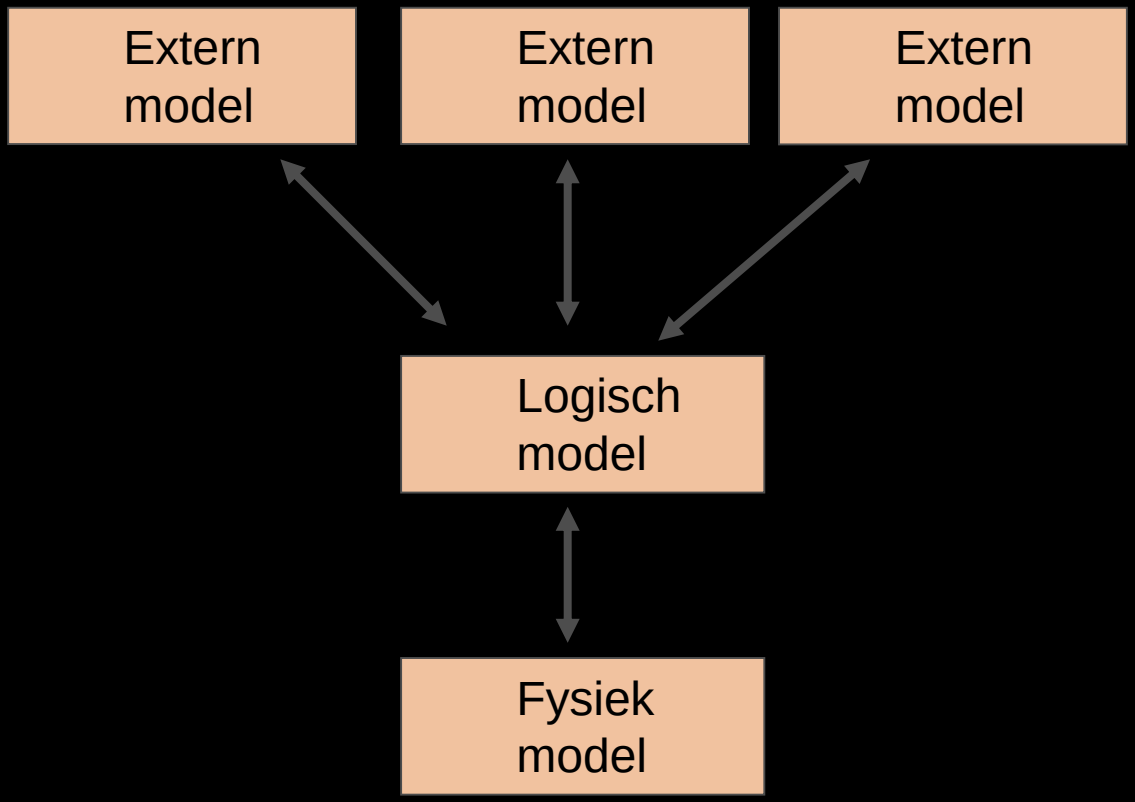
= deelverzameling van logisch model:



HO GENT

DBMS: Modellen

3 lagen



Een wijziging in één laag mag geen gevolgen hebben voor de andere lagen.

HO GENT

DBMS: Rollen binnen een DBMS

Databaseontwerpers

Databasebeheerders

Eindgebruikers

HO GENT

DBMS: Rollen binnen een DBMS

Wij !

Ontwerpen de database:

- Ontwikkelen van de modellen
- Bepalen constraints
- Bepalen van relaties
- ….

HO GENT

DBMS: Rollen binnen een DBMS

Databasebeheerders

Beheren de database:

optimaliseren van de DB

HO
GENT

DBMS: Rollen binnen een DBMS

Eindgebruikers

Gebruiken gegevens uit de database:

toevoegen,

opvragen,

wijzigen,

verwijderen

HO GENT

DBMS: Voordelen & Nadelen

HO
GENT

DBMS: Voordelen

Beheersen van data-redundantie

Stand-alone bestandsstructuren

zelfde informatie in verschillende bestanden omdat verschillende gebruikers verschillende data-behoeften hebben

(Gedeelde) Databank

views (externe modellen) voor elke groep van gebruikers ➔

geen dubbele gegevens

HO GENT

DBMS: Voordelen

Beheersen van data-redundantie

Opmerking!

Soms doelbewuste replicatie:

- Verhogen van performantie
- Backups
- Gedistribueerde systemen

HO GENT

DBMS: Voordelen

Consistente data

Doordat gegevens slechts éénmaal opgeslagen worden, moet een update van de data slechts één keer gebeuren

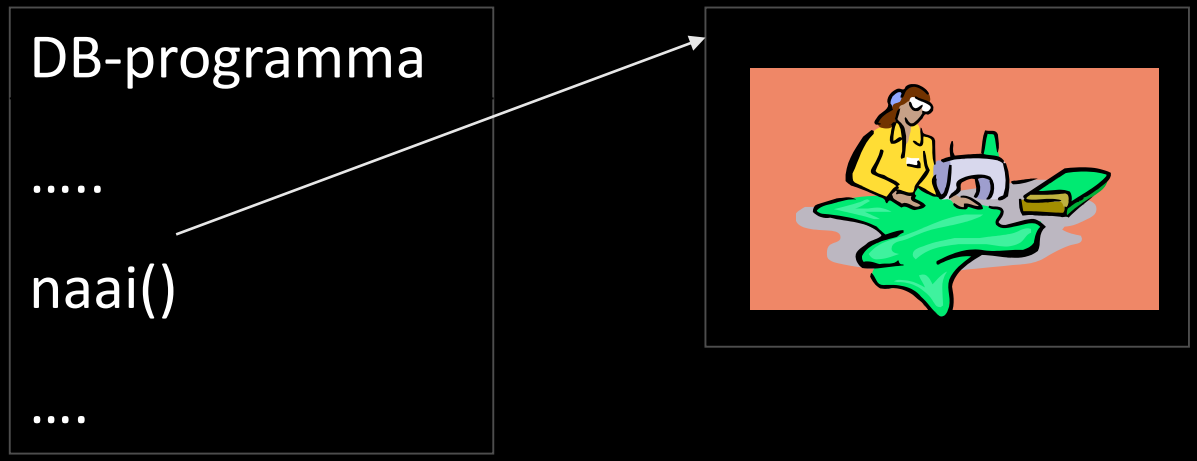
➔ nieuwe waarde is onmiddellijk beschikbaar voor alle gebruikers

HO GENT

DBMS: Voordelen

Opdrachtonafhankelijkheid

Programma's moeten niet gewijzigd worden als de implementatie van een opdracht wordt veranderd



HO GENT

DBMS: Voordelen

Flexibiliteit

Verschillende views (externe modellen) op dezelfde data zijn mogelijk

Zij

Hij

HO GENT

DBMS: Voordelen
Performantie

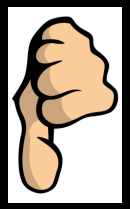
- beste toegangspad bepalen
- intelligent verdelen van de gegevens over de opslagmedia
- invoeren van indexen (heeft een index ook nadelen?)
- taak van database-administrator

HO
GENT

DBMS: Nadelen

Niks is perfect, ook een DBMS niet …

- Complex
- Kosten (hardware / software)
- Grote gevolgen bij defect (failsave systeem nodig)



HO GENT

Historiek.

Eerste generatie DBMSen

- Hiërarchische databanken
- Codasyl

Tweede generatie DBMSen

- Relationele databanken

Derde generatie DBMSen

- Objectgeörienteerde databanken
- Objectrelationele databanken
- NoSQL databanken

HO
GENT

Historiek.

Eerste generatie DBMSen

- Hiërarchisch model:
  - NASA Apollo maan project (1964) heeft geleid tot de ontwikkeling van GUAM (Generalized update Access Method) door IBM
  - Opzet: Kleinere componenten samenvoegen als onderdeel van een groter geheel
  - Resultaat: Omgekeerde boom ➔ hiërarchische structuur
  - In 1966 commercialisatie: **IMS** (Information Management System) door IBM
  - Wordt nog gebruikt bij mainframes

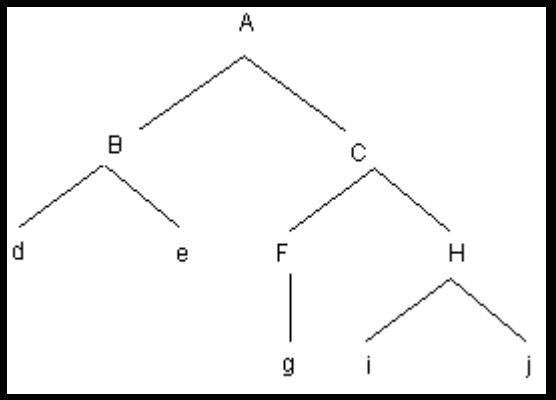
One small step for men, one giant leap for mankind

HO GENT

Historiek.

Eerste generatie DBMSen

- Hiërarchisch model:
  - elk record in een databank kan verwijzen naar een n-aantal andere records (children);
  - ieder recordtype heeft één en niet meer dan één eigenaar (owner);
  - het hiërarchische model kent maar één boom per databank;
  - de takken hebben onderling geen samenhang;
  - de enige ingang van de boomstructuur is van bovenaf.



HO GENT

Historiek.

Eerste generatie DBMSen

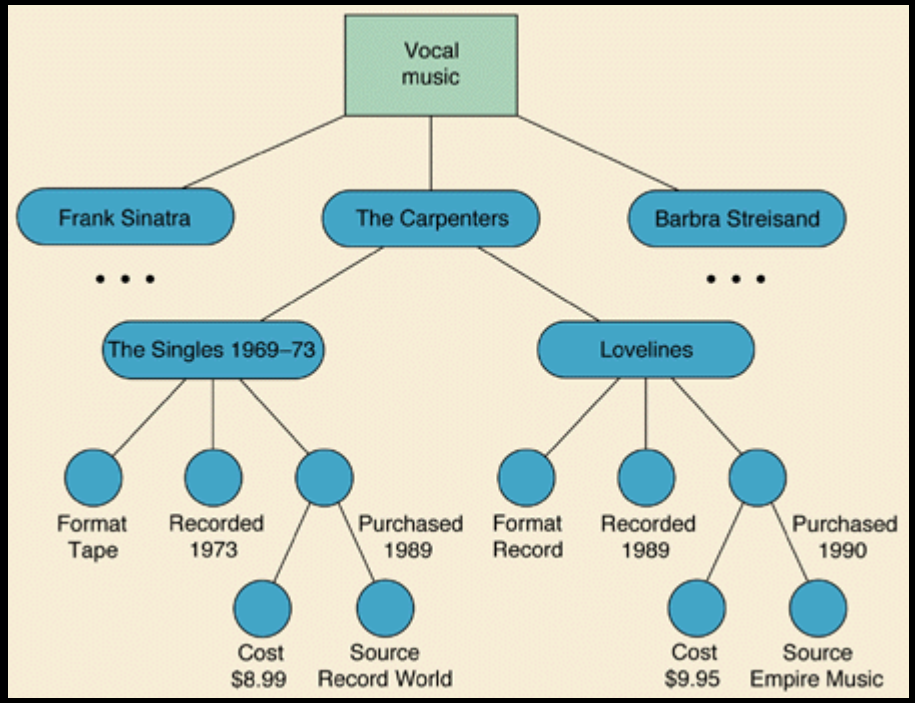
- Hiërarchisch model:

  Voordelen:

  - performantie
  - eenvoudig te begrijpen

  Veel nadelen:

  - enkel 1 op veel verbanden
  - procedurele toegang



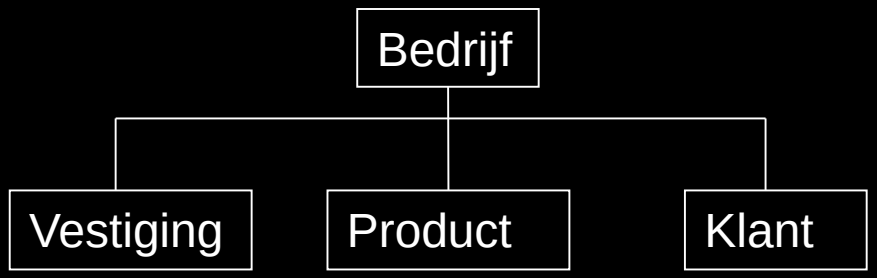
HO GENT

Historiek.

Eerste generatie DBMSen

- **Hiërarchisch model:** voorbeeld 1

  De databank “Bedrijf" heeft als hoofdtakken “Vestiging", “Product“ en “Klant".



➔ Geen relatie tussen vestiging en product, product A kan in elke vestiging gemaakt worden.

➔ Wanneer product A alleen in vestiging X gemaakt wordt, kan dat in dit hiërarchische model niet vastgelegd worden.

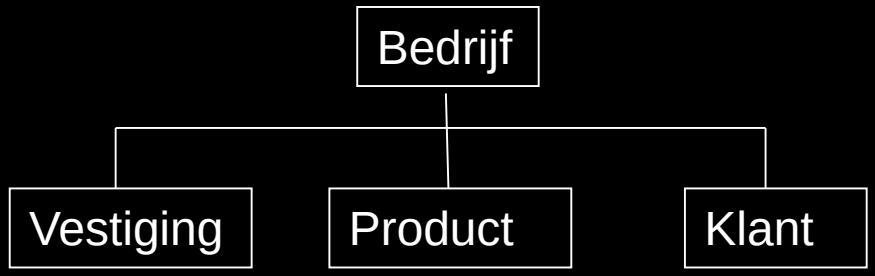
HO GENT

Historiek.

Eerste generatie DBMSen

- **Hiërarchisch model:** voorbeeld 1

  De databank “Bedrijf" heeft als hoofdtakken “Vestiging", “Product“ en “Klant".



Orders moeten bijgehouden worden in de databank:

➔ Onder Product?

➔ Onder Klant?

➔ Beide?

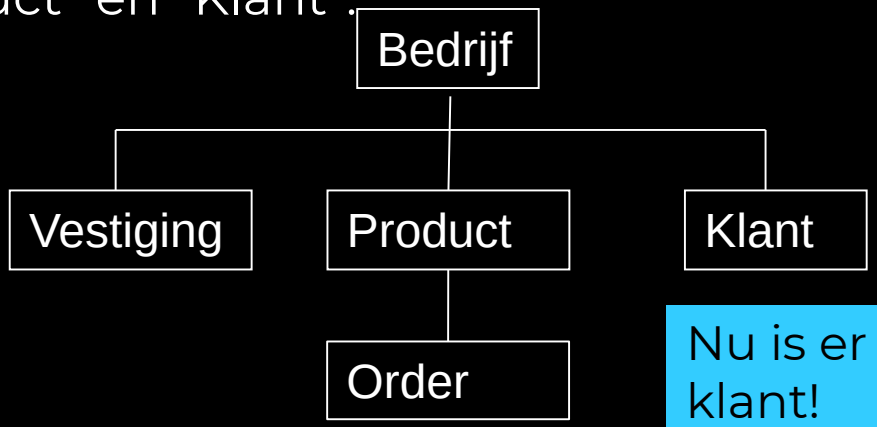
HO GENT

Historiek.

Eerste generatie DBMSen

- **Hiërarchisch model:** voorbeeld 1

De databank “Bedrijf" heeft als hoofdtakken “Vestiging", “Product“ en “Klant".



Nu is er nog geen verband met de klant!
Een child mag maar 1 ouder hebben!

Orders moeten bijgehouden worden in de databank:

- ➔ Onder Product?
- ➔ Onder Klant?
- ➔ Beide?

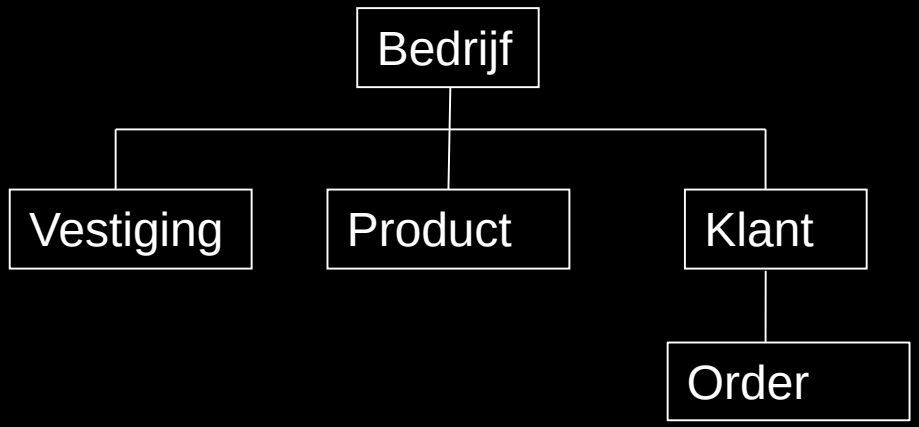
HO GENT

Historiek.

Eerste generatie DBMSen

- **Hiërarchisch model:** voorbeeld 1

  De databank “Bedrijf" heeft als hoofdtakken “Vestiging", “Product“ en “Klant".



Orders moeten bijgehouden worden in de databank:

- ➔ Onder Product
- ➔ Onder Klant?
- ➔ Beide?

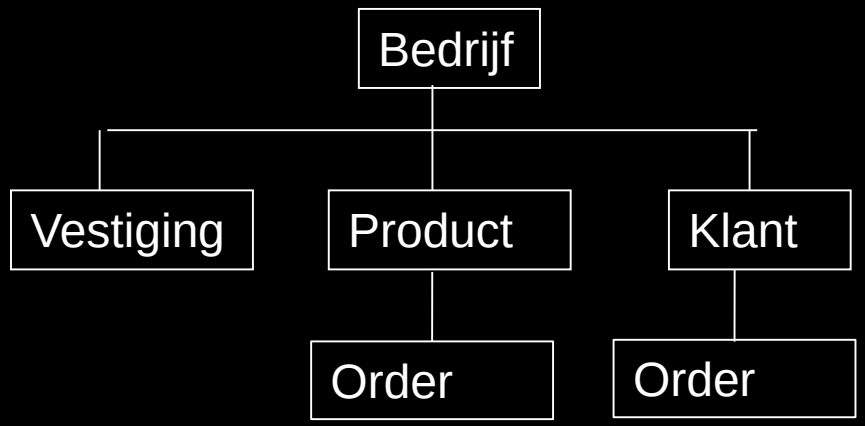
Nu is er geen verband met het product!
Een child mag maar 1 ouder hebben!

HO GENT

Historiek.
Eerste generatie DBMSen

- **Hiërarchisch model:** voorbeeld 1

  De databank “Bedrijf" heeft als hoofdtakken “Vestiging", “Product“ en “Klant".



Orders moeten bijgehouden worden in de databank:

- ➔ Onder Product?
- ➔ Onder Klant?
- ➔ Beide?

Hetzelfde order moet 2 keer opgeslagen worden!

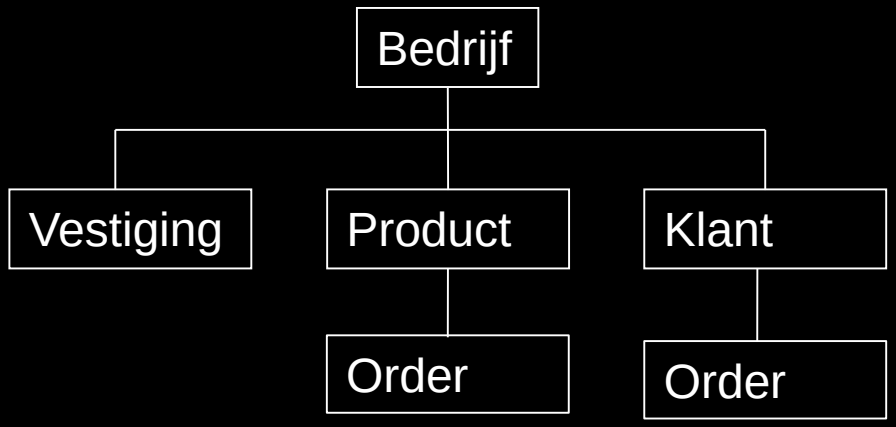
HO GENT

Historiek.

Eerste generatie DBMSen

- **Hiërarchisch model:** redundantie

  Om in deze structuur een order voor te stellen moet je het 2 keer opslaan! Want een order gaat over 1 product (child van product) maar ook over 1 klant (child van klant)



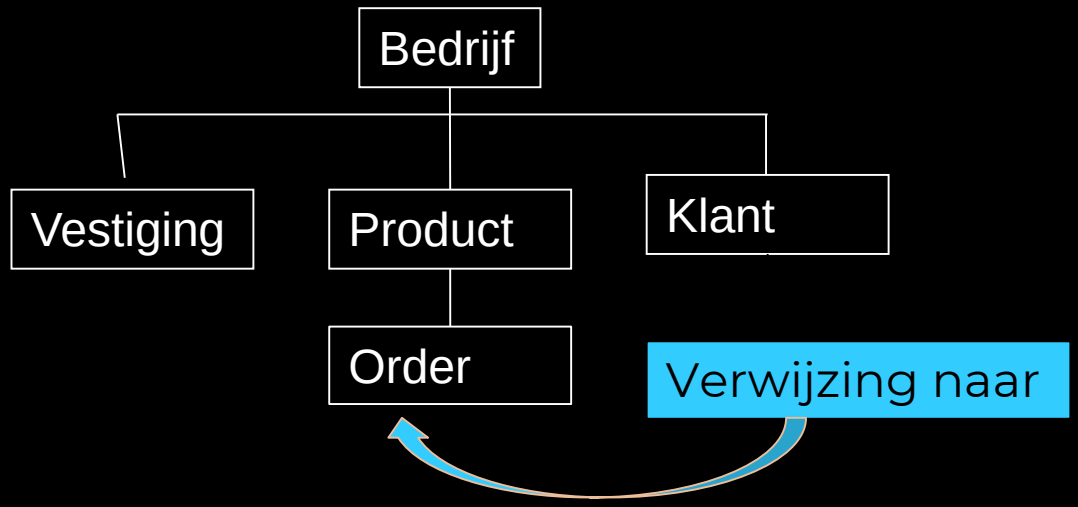
HO GENT

Historiek.

Eerste generatie DBMSen

- **Hiërarchisch model:** redundantie

  Een order zal vervangen worden door een verwijzing naar het eigenlijke order.



HO GENT

Historiek.

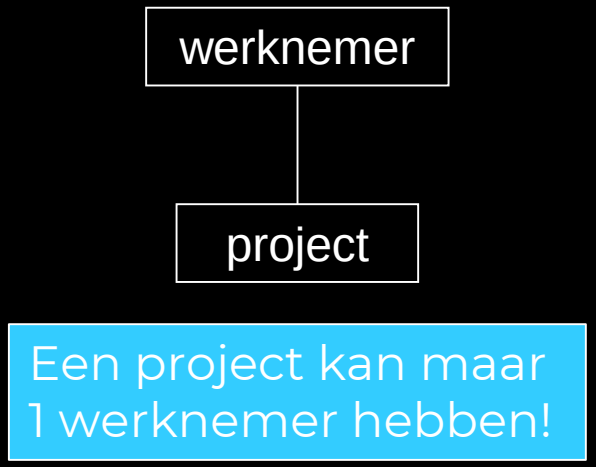
Eerste generatie DBMSen

- **Hiërarchisch model:** redundantie

  Ook een veel-op-veel relatie kan niet éénduidig voorgesteld worden in een hiërarchische structuur. Ook hier zal je data moeten herhalen.

  Voorbeeld 2: Aan een project werken meerdere werknemers, maar een werknemer kan ook aan meerdere projecten werken.

  Teken de hiërarchische structuur!

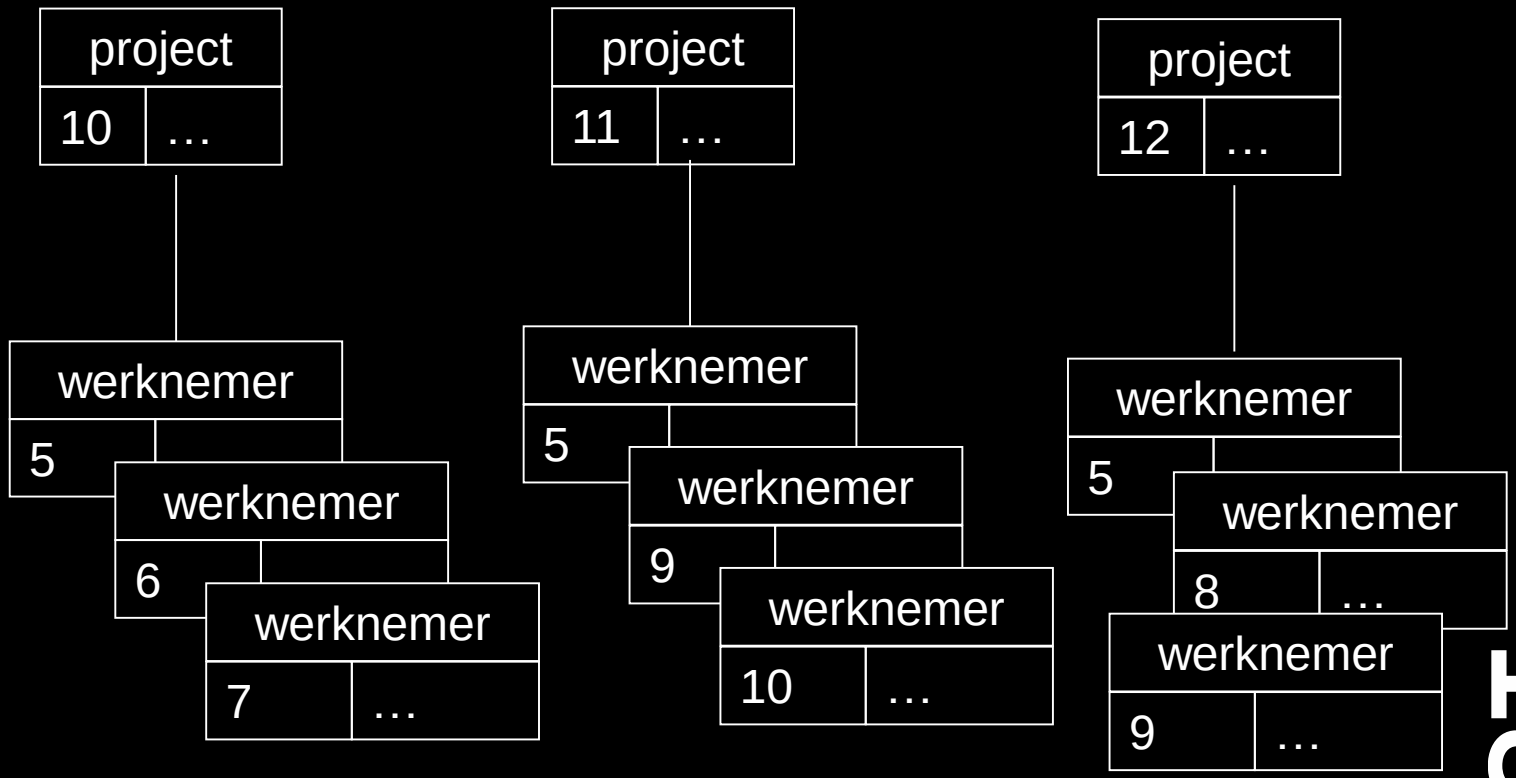


HO GENT

Historiek.

Eerste generatie DBMSen

- **Hiërarchisch model:** voorbeeld 2
  Oplossing: Redundantie van werknemer toestaan?

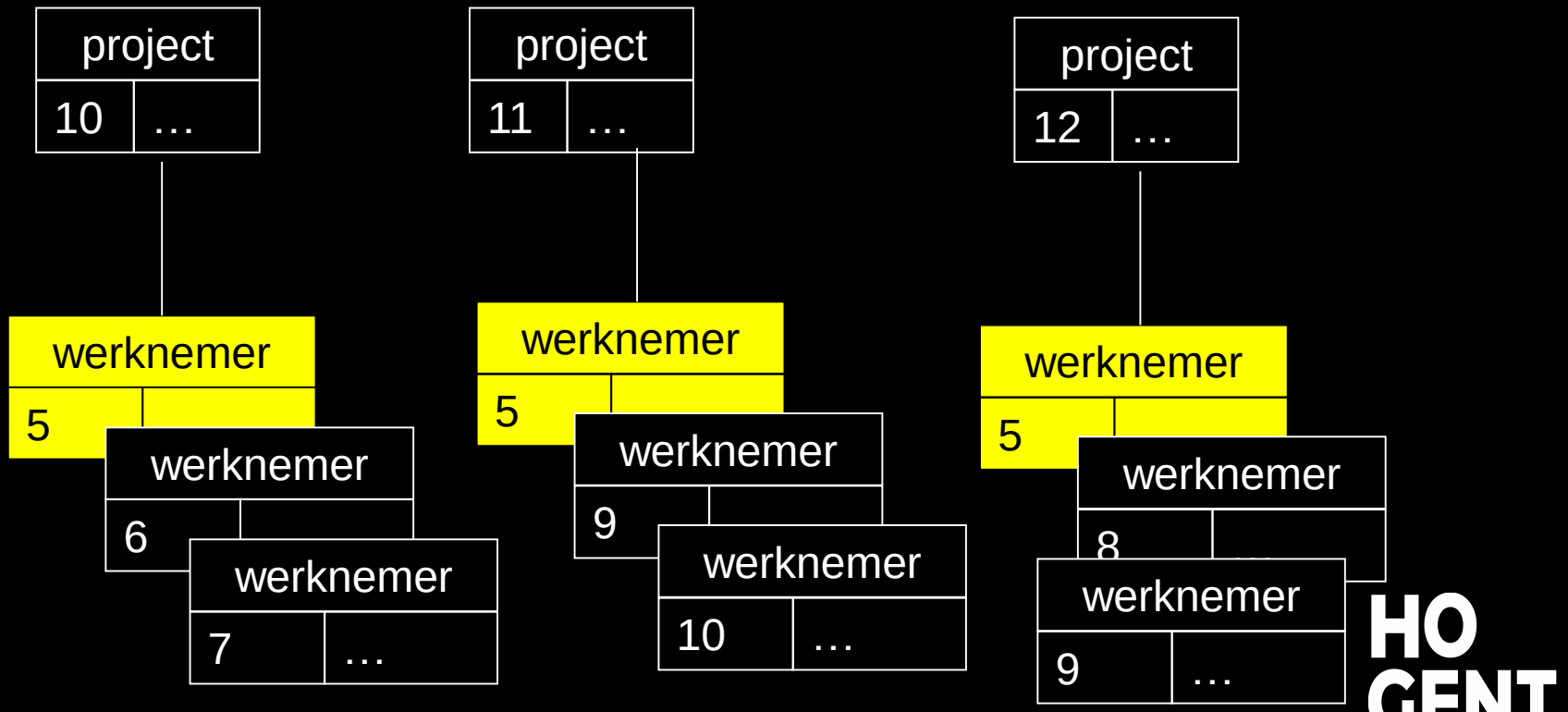


HO GENT

Historiek.

Eerste generatie DBMSen

- **Hiërarchisch model:** voorbeeld 2
  Oplossing: Redundantie van werknemer toestaan?



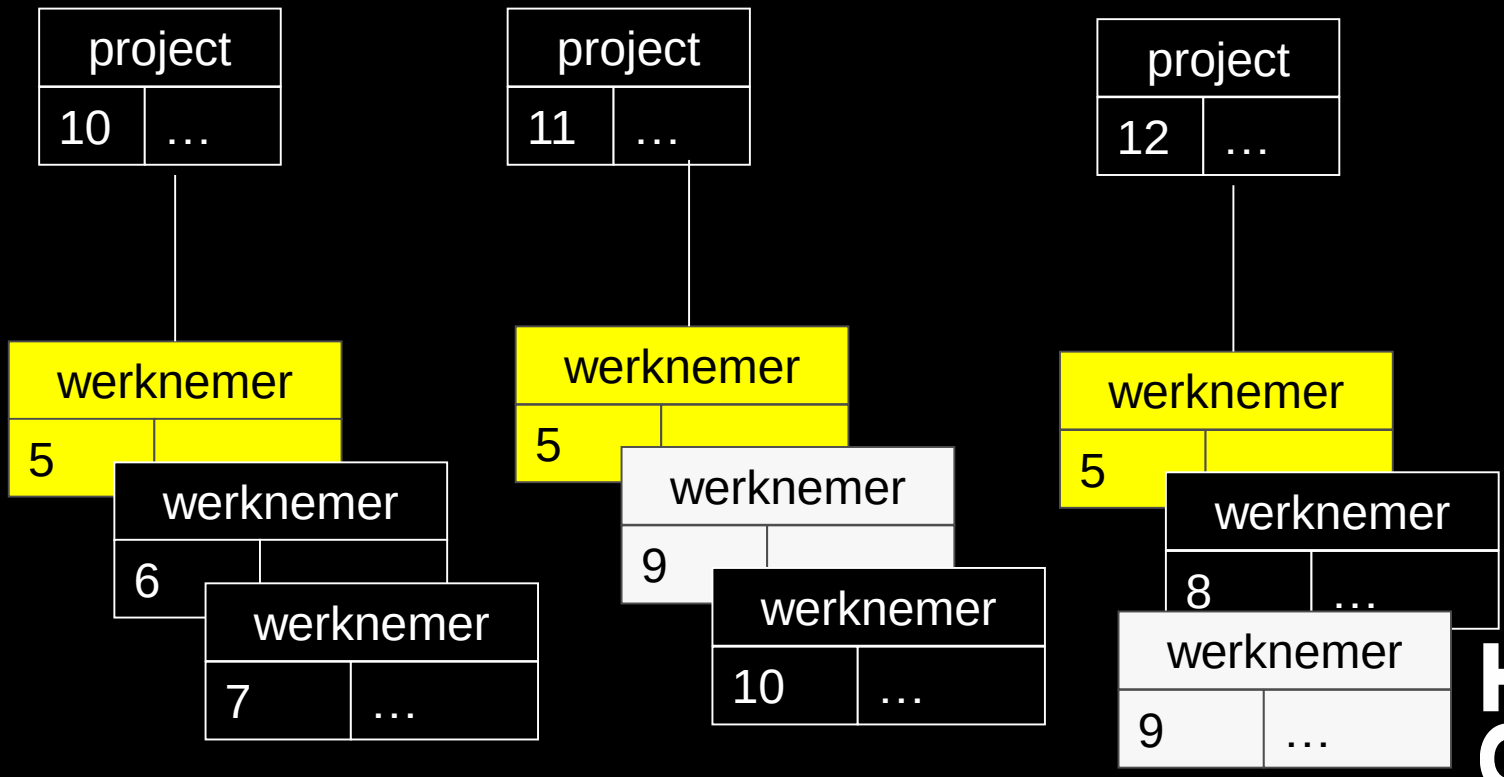
HO GENT

Historiek.

Eerste generatie DBMSen

- **Hiërarchisch model:** voorbeeld 2

  Oplossing: Redundantie van werknemer toestaan?



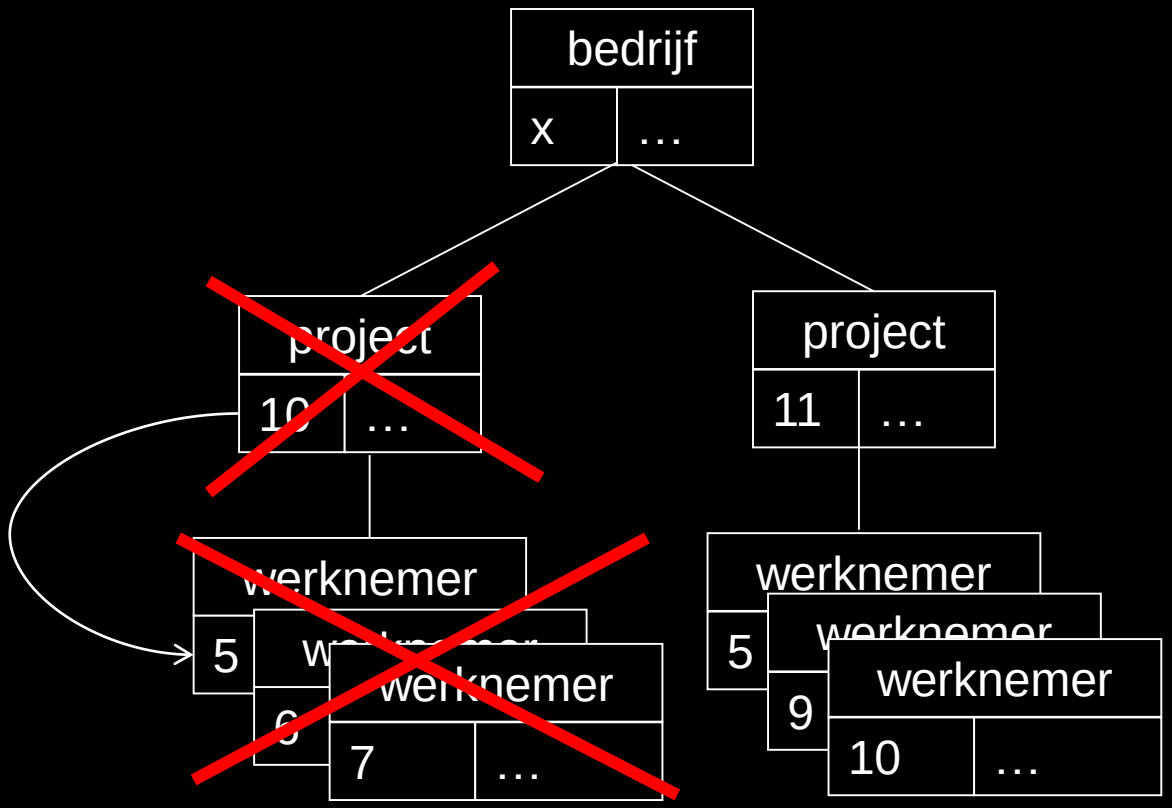
HO GENT

Historiek.

Eerste generatie DBMSen

- **Hiërarchisch model:** voorbeeld 2

  Probleem! Indien project 10 geschrapt wordt ➔ alle data omtrent werknemer 6 en 7 kwijt!

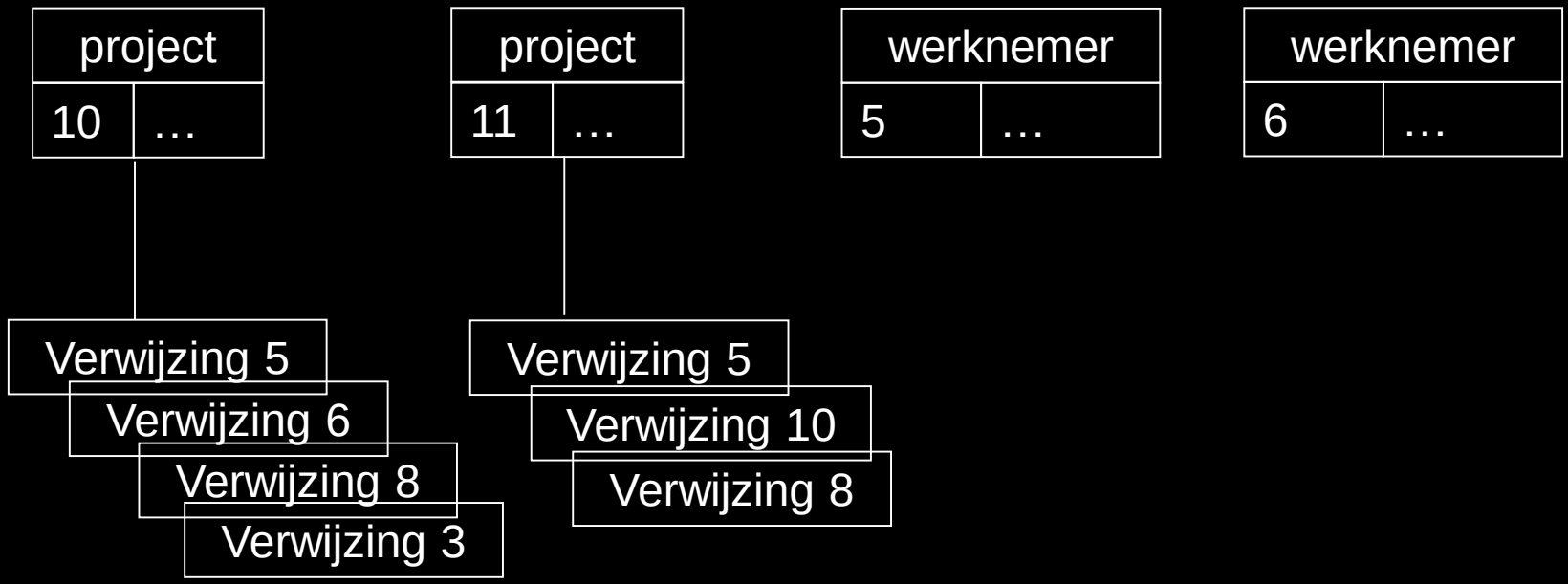


HO GENT

Historiek.

Eerste generatie DBMSen

- **Hiërarchisch model:** voorbeeld 2
  Oplossing: veelvuldige verwijzing naar de werknemer.



HO GENT

Historiek.

Eerste generatie DBMSen

- Codasyl model
  - ontworpen om nadelen van hiërarchisch systeem weg te werken
  - In 1967 IDS (Integrated Data Store) door General Electric
  - gebaseerd op het **netwerk datamodel** ➔ kan complexere data-verbanden voorstellen
  - In 1969: CODASYL Database Task Group Report legt standaarden vast voor netwerk-databanken

Charles Bachman

HO GENT

Historiek.

Eerste generatie DBMSen

Codasyl model: standaarden:

3 componenten:

1. **Netwerk schema**: organisatie van de DB gezien door de ogen van de DBA: DBnaam, recordtypes, componenten van elk recordtype
2. **Subschema**: het deel van de DB gezien door de ogen van de eindgebruiker of een DB-programma
3. **Data Management Language**: om de data te definiëren en te manipuleren

HO
GENT

Historiek.

Eerste generatie DBMSen

Codasyl model: standaarden:

3 talen:

1. **DDL**: Data Definition Language
2. **Subschema DDL**: Data Definition Language specifiek voor het subschema
3. **DML**: Data Manipulation Language

HO GENT

Historiek.

Eerste generatie DBMSen

Codasyl model: voor- en nadelen

Voordeel: kan ook veel-op-veel relaties voorstellen

Nadelen:

- Is nog steeds procedureel.
- Bij elk record moeten wijzers (pointers) opgeslagen worden, die de fysieke representatie van de parent-child-relaties zijn. De pointers verwijzen dus door naar andere records.
- Heel zwaar DBMS dat enorm veel resources gebruikt.

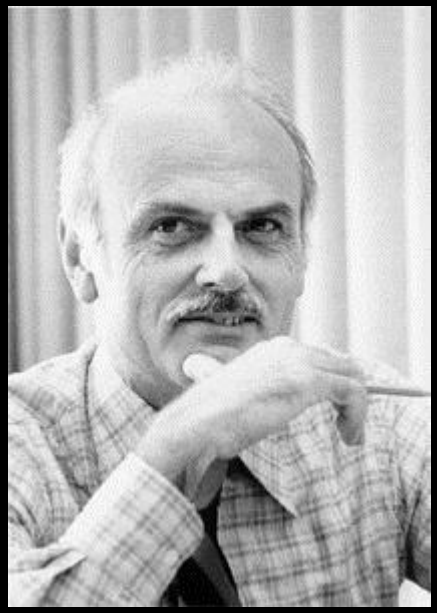
HO GENT

Historiek.

Tweede generatie DBMSen

Relationeel model

- 1970: Dr EF Codd, onderzoeker bij IBM, schrijft paper over het relationeel data model (‘A relational model of data for large shared data banks’)
- Ontwikkeling van verschillende relationele databanken
- Ontwikkeling van SQL (Structured Query Language) ➔ standaardtaal voor relationele systemen
- Tegenwoordig: veel commerciële relationele databasesystemen voor alle mogelijke platformen (PC, mainframe)
- Voorbeelden: SQLServer, MySQL, Ingres, Informix, Oracle, DB2



Dr. EF Codd

‘vader’ van relationele databanken

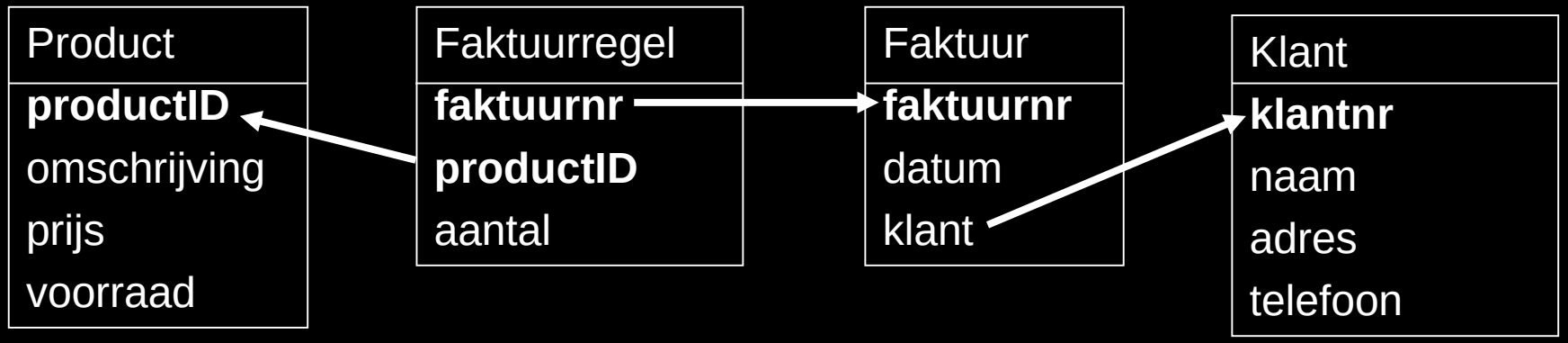
HO GENT

Historiek.

Tweede generatie DBMSen

Relationeel model

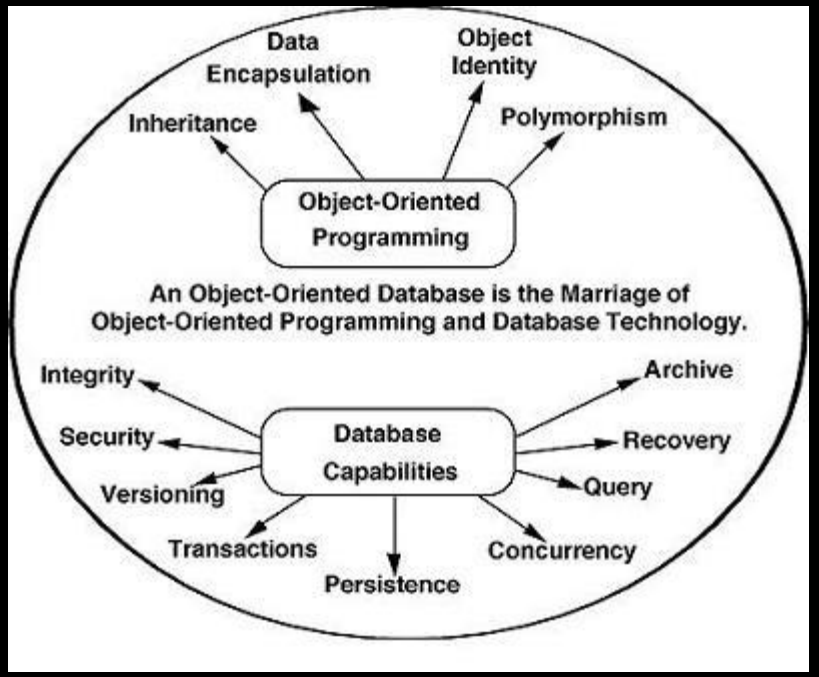
- Wiskundige basis
- Logische in plaats van fysieke verbanden tussen de gegevens ➔ vreemde sleutels



HO GENT

Historiek.
Derde generatie DBMSen

- Object Oriented DBMS



- Object relational DBMS

HO GENT

Architectuur.

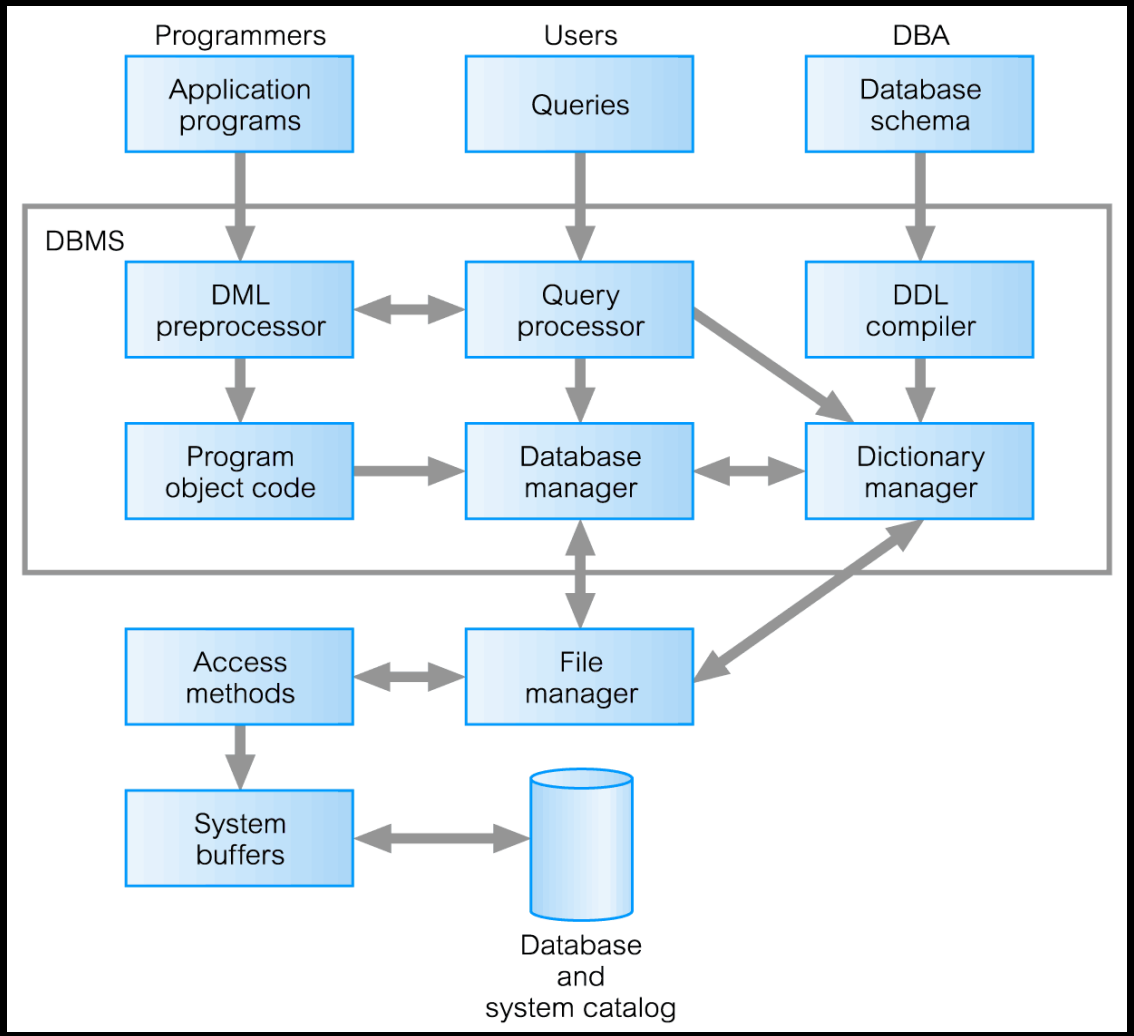
Interne architectuur

Systeem architectuur:

- Multi-user DBMS architectuur
- Webservices en SOA
- Gedistribueerde DBMSen

HO
GENT

Architectuur.



DDLcompiler

DDL: Data Definition Language

Zorgt voor het wegschrijven van de definities uit de 3 modellen naar de gegevenscataloog,

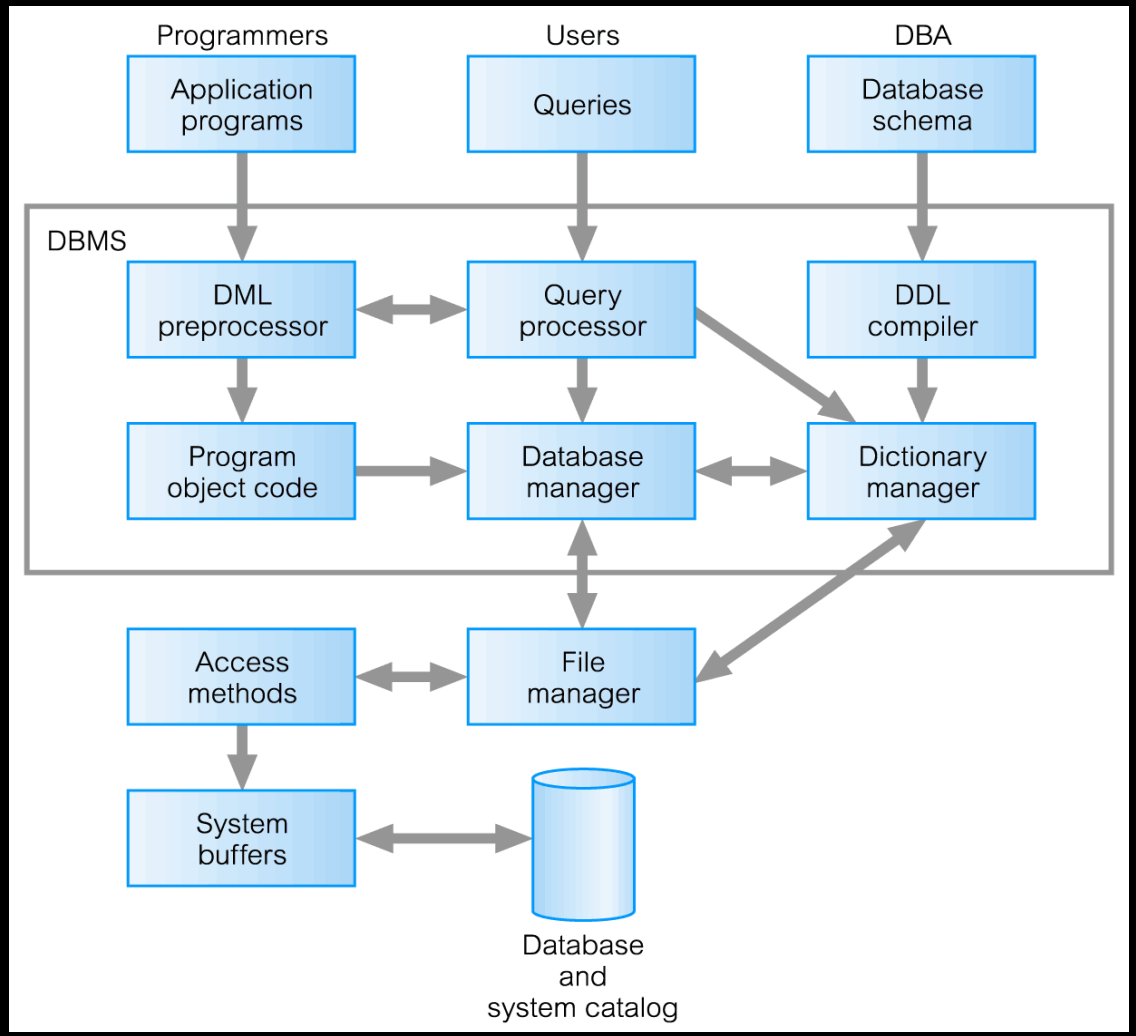
(tabellen met metadata)

Query processor

Vertaalt de queries naar een reeks van low-level instructies die door de **Database manager** dienen uitgevoerd te worden.

HO GENT

Architectuur.



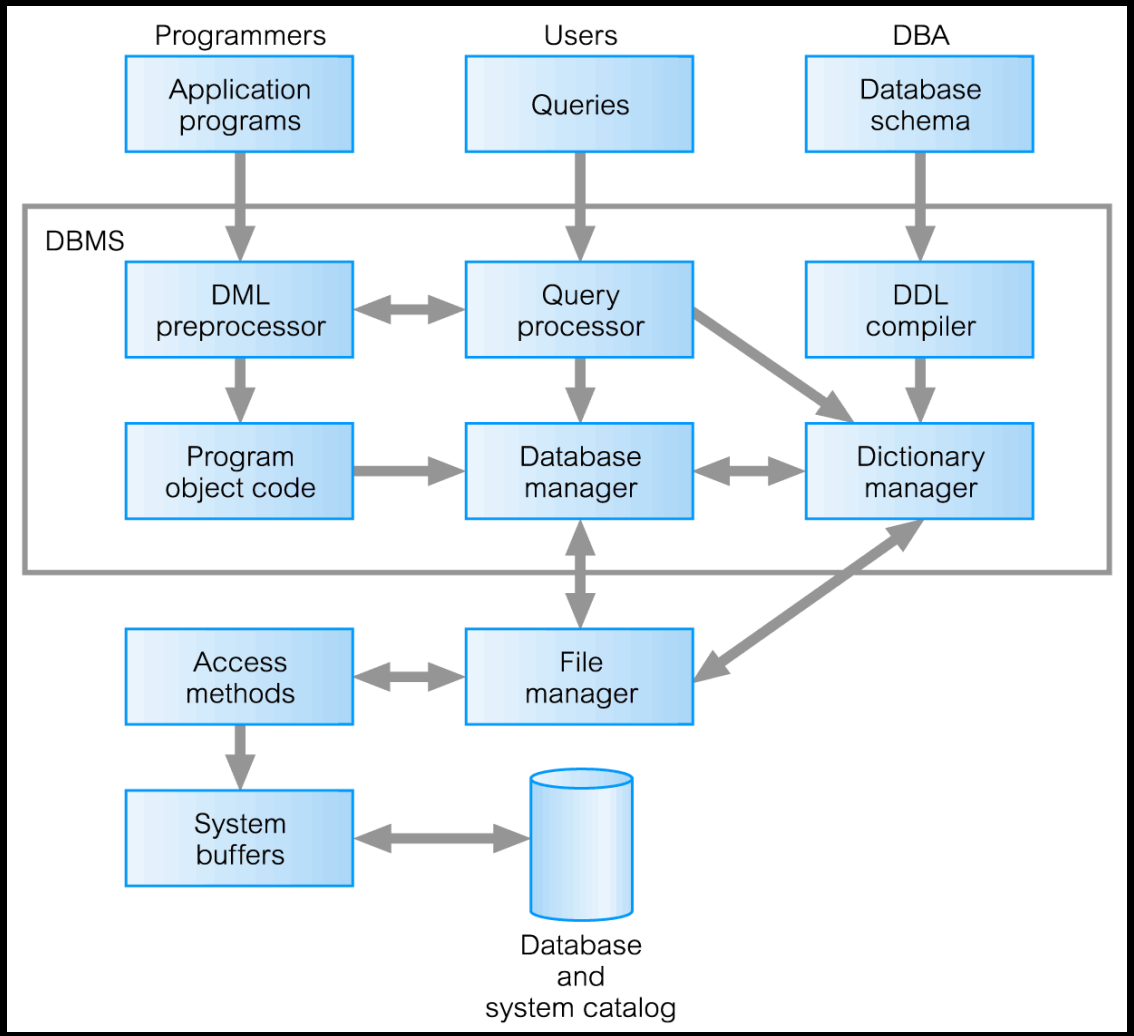
DML preprocessor

DML: Data Manipulation Language

- Zet DML-statements (ingebed in een applicatie) om naar standard function calls in de hostlanguage
- Werkt samen met **query processor** om de code te genereren

HO GENT

Architectuur.

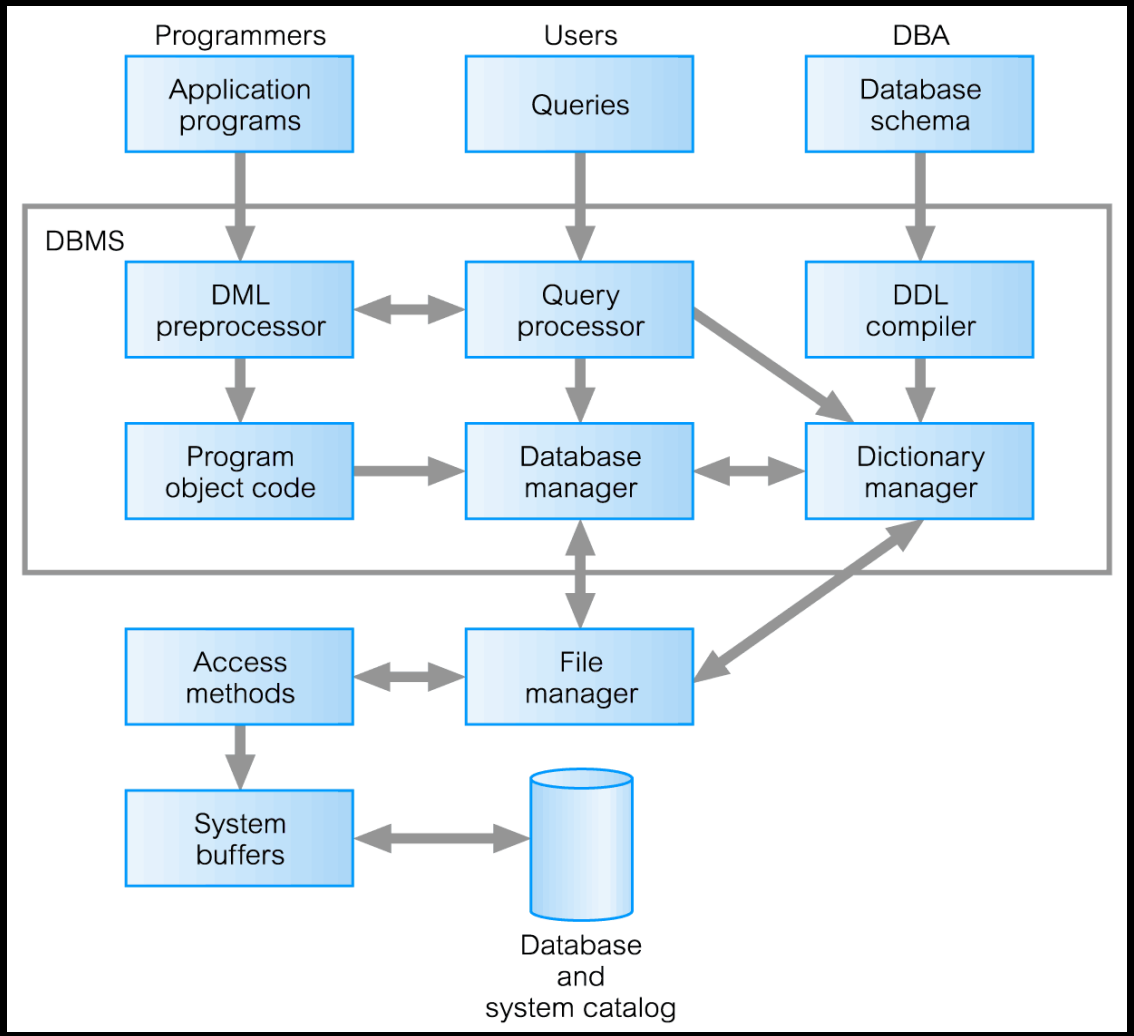


Database manager

- Ontvangt de queries.
- Onderzoekt externe schema's om te beslissen welke records nodig zijn.
- Plaatst een call naar **file manager** om de data op te halen.

HO GENT

Architectuur.



File manager

- Zorgt voor de allocatie van opslagruimte op schijf.
- Beheer van indexen.

Dictionary manager

- Zorgt voor de toegang naar de systeemcataloog (gegevenscataloog, meta data).
- Ook catalog manager genoemd.

HO GENT