

Inhoudsopgave: Datum: 28/09/2023

[1 Hoofdstuk 1: Databanken gekaderd 3](#_Toc155443959)

[1.1 Wie gebruikt databanktechnologie? 3](#_Toc155443960)

[1.2 Basisbegrippen 3](#_Toc155443961)

[1.3 Gegevensbeheer 3](#_Toc155443962)

[1.4 Delen van een databanksysteem 3](#_Toc155443963)

[1.5 Kenmerken van databanken en databankbeheer 5](#_Toc155443964)

[2 Hoofdstuk 2: Conceptueel model 9](#_Toc155443965)

[2.1 Fasen in databankontwerp 9](#_Toc155443966)

[2.1.1 Fase 1 uitgebreid: 9](#_Toc155443967)

[2.1.2 Fase 2 uitgebreid: 9](#_Toc155443968)

[2.1.3 Fase 3 uitgebreid 9](#_Toc155443969)

[2.1.4 Fase 4 uitgebreid: 9](#_Toc155443970)

[2.1.5 Fasen Samengevat: 9](#_Toc155443971)

[2.2 Entity Relationship Diagram 10](#_Toc155443972)

[3 Hoofdstuk 3 Zwakke entiteiten 13](#_Toc155443973)

[3.1 Zwakke entiteitstypes 13](#_Toc155443974)

[4 Hoofdstuk 4 enchanced ERD 14](#_Toc155443975)

[4.1 Inleiding 14](#_Toc155443976)

[4.2 Specialisatie 14](#_Toc155443977)

[4.3 Beperkingen van het ERD 15](#_Toc155443978)

[5 Hoofdstuk 5:Relationeel model 16](#_Toc155443979)

[5.1 Inleiding 16](#_Toc155443980)

[5.2 Mapping 18](#_Toc155443981)

[6 Hoofdstuk 6 Normalisatie 20](#_Toc155443982)

[6.1 Functionele afhankelijkheid 20](#_Toc155443983)

[6.2 normalisatiestappen 21](#_Toc155443984)

[7 Hoofdstuk 7: SQL 23](#_Toc155443985)

[7.1 Structured Query Language (SQL) 23](#_Toc155443986)

[7.2 SELECT statement 23](#_Toc155443987)

[8 Hoofdstuk 8 Werken met meerdere tabellen 26](#_Toc155443988)

[8.1 Group by en statistische functies 26](#_Toc155443989)

[8.2 Join 27](#_Toc155443990)

[8.3 Union: 28](#_Toc155443991)

[9 Hoofdstuk 9: SQL -- DML 29](#_Toc155443992)

[9.1 INSERT: 29](#_Toc155443993)

[9.2 Update 29](#_Toc155443994)

[9.3 Delete – Verwijderen van DATA 30](#_Toc155443995)

[10 H10 SQL – DDL 31](#_Toc155443996)

[10.1 Inleiding 31](#_Toc155443997)

[10.2 DDL - Databank 32](#_Toc155443998)

[10.3 DDL - Tabellen. 32](#_Toc155443999)

[10.4 Constraints 33](#_Toc155444000)

# Hoofdstuk 1: Databanken gekaderd

## Wie gebruikt databanktechnologie?

Wie gebruikt databanken?

Elke handelaar op aarde.

## Basisbegrippen

Wat is een databank?

* Een plek om gegevens bij te houden.
* **Een gedeelde verzameling van logisch met elkaar verbonden gegevens en hun beschrijving**, ontworpen om aan de informatienoden van een organisatie te voldoen.

Wat is een DBMS (Database Management System) ?

Software waar databanken op worden gemaakt.

## Gegevensbeheer

Delen van een databanksysteem. Wat is een voorbeeld van een databankmodel? NIET BELANGRIJK

* Kunstenaar (naam, geboorteplaats, geboortedatum)
* Kunstwerk (naam, museum, jaar)
* Museum (naam, stad)

Gedecentraliseerde aanpak:

* **Dubbele of redundante (overbodige)** opslag
* Risico op **inconsistente** data
* **Sterke koppeling** tussen applicaties en data
* Concurrente **toegang** **moeilijk** te realiseren
* Applicaties voor meerdere diensten/bedrijven **moeilijk** te realiseren

Gecentraliseerde aanpak:

* **Efficiënter**
* **Consistente** data
* **Eenvoudiger** te beheren
* **Losse koppeling** tussen applicaties en data
* DBMS biedt mogelijkheden bij het **concurrent ‘beheren’** van data

## Delen van een databanksysteem

Elementen van een databanksysteem.

* Databankmodel versus instances
* Data model
* 3-lagen architectuur
* Catalog
* Databankgebruikers

Wat is een instance (instantie)?

**De werkelijke data** (van het databankmodel + Toestand van een databank).

Kunstenaar (naam, geboorteplaats, geboortedatum) = Picasso, Wetteren, 1981.

Gegevensmodel (Onderstaande modellen zijn alle datamodellen):

Weergave van de gegevens met hun kenmerken en hun relaties.

Conceptueel gegevensmodel:

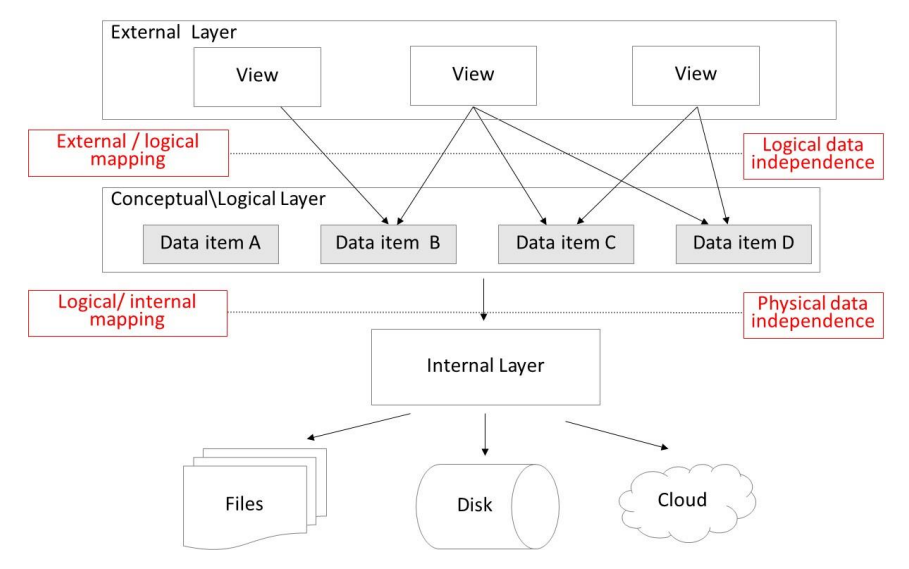
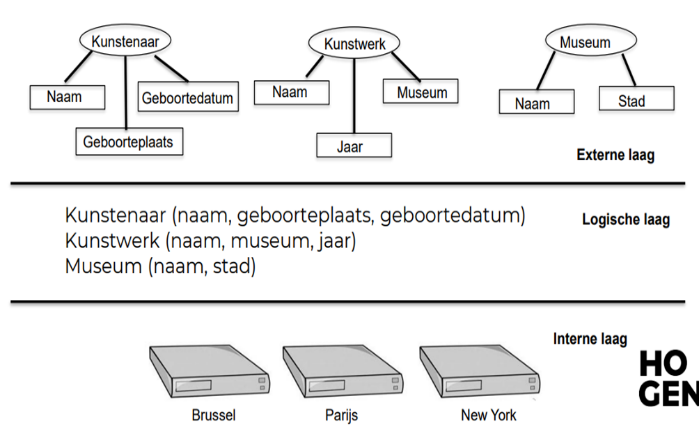
* **perfecte weergave** van de **gegevensvereisten** van de ‘business’ opgesteld met de business omgezet naar logisch en intern model
* **Algemene beschrijving gegevenselementen**, kenmerken en relaties
  + Gebruikt door ‘IT’ en ‘business’
  + Weergave hoe ‘de business’ de gegevens ziet
  + Voorstelling: **(E)ERD** diagram

Logisch gegevensmodel:

* **Vertaling** **conceptueel gegevensmodel** naar het **type databankmodel**
  + Relationeel
  + hiërarchisch
  + oo
  + xml
  + nosql
* **Omzetten naar intern** (fysiek) gegevensmodel
  + Geeft informatie over fysieke opslag:
    - Waar worden welke gegevens opgeslagen
    - Onder welke vorm
    - Indexen die het ophalen versnellen
  + Zeer dbms afhankelijk
* Externe gegevensmodellen
  + Deelverzameling van het logisch model
  + Voor een specifieke doelgroep.

3-lagenmodel:

**Externe laag Logische laag Interne laag**



Catalog:

* **Schatkist** van DBMS
* **Definities en beschrijving** van de **elementen** in de DB (= metadata)
* **Definities logisch** gegevensmodel en **intern** gegevensmodel
* Zorgt voor **synchronisatie** en **consistentie** van de gegevensmodellen

Databank gebruikers:

* IT Architect ontwerpt **conceptueel gegevensmodel** in samenspraak met de ‘business’
* **Dbontwerper vertaalt** conceptueel model naar logisch en intern model
* DBA (databankbeheerder of database administrator) **implementeert** en **monitort** DB
* Applicatieontwikkelaar schrijft databankprogramma’s/**databankapplicaties**
* ‘Business’ gebruikt databankapplicaties en voert op die manier **databankacties** uit

## Kenmerken van databanken en databankbeheer

Gegevensonafhankelijkheid:

* Wijzigingen aan de gegevensbeschrijving hebben weinig tot geen impact op de applicaties
* Fysieke gegevensonafhankelijkheid: wijzigingen van de opslagspecificaties hebben geen invloed op het logisch model noch op de applicatie
  + Wordt opgevangen door het dbms
* Logische gegevensonafhankelijkheid: minimale aanpassingen aan de applicaties bij wijzigingen aan het logisch model

(On)gestructureerde gegevens.

* **Gestructureerde gegevens** 
  + Kunnen in een **logisch datamodel** voorgesteld worden
  + **Integriteitsregels** kunnen opgesteld en afgedwongen worden
  + Vereenvoudigen, opzoeken, verwerken en analyseren
  + Voorbeelden: naam, geboortejaar, geboorteplaats van een kunstenaar
* **Ongestructureerde gegevens** 
  + **kunnen niet** op een zinvolle manier **worden** **geïnterpreteerd** **door een applicatie**
  + Voorbeelden: gesprekken op social media, e-mails
  + Let op: er bestaat veel meer ongestructureerde data dan gestructureerde data

Semi-gestructureerde gegevens

* + De structuur van de gegevens is zeer onregelmatig of zeer wisselend.
  + Voorbeelden: webpagina's van individuele gebruikers op een social media platform, samenvattende human resources documenten

Integriteitsregels:

* **Integriteitsregels** worden gedefinieerd op basis van het **conceptueel model** en opgeslagen in de catalog
  + Worden afgedwongen door het DBMS
* Vastleggen **hoe gegevens** worden opgeslagen **(syntactische regel)**
  + Voorbeeld: geboortedatum, eenheidsprijs
* Vastleggen **wanneer gegevens** **correct** zijn **(semantische regel)**
  + Voorbeeld: Eenheidsprijs > 0
  + geboortedatum niet > vandaag

Redundante gegevens:

* Databank = centrale en unieke opslag gegevens
* Soms worden databanken gedupliceerd uit veiligheidsoverwegingen of omwille van performantie → **redundantie**
* DBMS is verantwoordelijk voor de synchronisatie en garandeert de juistheid van de gegevens

Indeling van DBMS-systemen:

* Op basis van het gegevensmodel
* Op basis van het gebruik

Indeling op basis van het gegevensmodel:

* **Hiërarchisch DBMS** 
  + Het **gegevensmodel** is een **omgekeerde boom**
  + DML is procedureel en gebaseerd op ‘**recordverwerking’**
  + **Geen query processor** (logisch en fysisch datamodel lopen door elkaar)
    - Voorbeeld: IMS (IBM)
* **Netwerk DBMS** 
  + Gebruiken een **netwerk gegevensmodel**
  + CODASYL DBMS
  + DML is procedureel en gebaseerd op ‘**recordverwerking’**
  + **geen query processor** (logisch en fysisch datamodel lopen door elkaar)
    - Voorbeeld: CA-IDMS (Computer Associates → Broadcom)
* **Relationeel DBMS** 
  + Maakt gebruik van het **relationeel gegevensmodel**
  + Momenteel het **meest frequent gebruikt** in de bedrijfswereld
  + **SQL** (beschrijvend en gebaseerd op resultset)
  + **Query processor**
  + Strikte **scheiding** tussen het logisch en het fysisch gegevensmodel
    - Voorbeelden: MySQL (open source, Oracle), Db2 (IBM), Oracle DBMS (Oracle), SQLServer (Microsoft)
* **Object-Oriented DBMS (OODBMS)** 
  + Gebaseerd op het **OO gegevensmodel**
  + Geen probleem om het gegevensmodel van de databank te **verenigen** met de **OO programmeertaal** 
    - Voorbeelden: db4o (open source, Versant), Caché (Intersystems), GemStone/S (GemTalk Systems)
  + Alleen doorgebroken in niche markten omwille van de **complexiteit** van het OODBMS
* **Object-Relationeel DBMS (ORDBMS)** 
  + Wordt ook **extended relationeel DBMS (ERDBMS)** genoemd
  + Gebruikt een **relationeel gegevens**model **uitgebreid** met **OO concepten**
  + **DML is SQL** (beschrijvend en gebaseerd op resultset)
    - Voorbeelden: Oracle DBMS (Oracle), DB2 (IBM), SQLServer (Microsoft)
* **XML DBMS** 
  + Maakt gebruik van het **XML gegevensmodel** om **gegevens op te slaan**
  + Native XML DBMS (voorbeelden: BaseX, eXist) zal de **boomstructuur** van een XML document projecteren op een **fysische opslagstructuur**
  + XML-enabled DBMS (voorbeelden: Oracle, IBM Db2) zijn bestaande DBMS-systemen die uitgebreid zijn met **faciliteiten** om **gegevens** uit een **XML gegevensmodel** **op te** **slaan**
* **NoSQL DBMS** 
  + Bedoeld om **massaal veel en ongestructureerde data op te slaan**
  + Kan ingedeeld worden in
    - **key-value stores** (sleutelwaardedatabanken)
    - **column-oriented databases** (gegevens worden per kolom gestockeerd ipv per rij)
    - **graph databases**
  + Focust op **schaalbaarheid** van de databank en op werken met onregelmatige en snel veranderende gegevensstructuren
    - Voorbeelden: Apache Hadoop, MongoDB, Neo4j

Indeling op basis van gebruik:

* **On-line transaction processing (OLTP)** 
  + Focust op het **beheren** en **verwerken** van **operationele en transactionele gegevens**
  + De databankserver moet heel veel **eenvoudige transacties** per tijdseenheid kunnen verwerken
  + Het DBMS moet goed ontworpen zijn om **heel veel korte eenvoudige query’s** uit te voeren
* **On-line analytical processing (OLAP)** 
  + Focust op het gebruiken van **operationele gegevens om strategische en tactische beslissingen te nemen**
  + Een beperkt aantal gebruikers zal **complexe queries** **uitvoeren**
  + Het DBMS moet **complexe queries efficiënt** kunnen verwerken.
* **Big Data & Analytics** 
  + **NoSQL databanken**
  + Focust op meer **flexibele databankstructuren** **en** **databanken zonder schema**
  + **Stockeert** **ongestructureerde data** zoals e-mails, Twitter tweets, Facebook posts, ...
* **Multimedia** 
  + Multimedia DBMS-systemen zorgen voor opslagruimte voor multimedia data zoals tekst, foto’s, audio, video, 3D games, ...
  + Moet ook de bijhorende query mogelijkheden (op basis van inhoud) aanbieden
* **Geometrische toepassingen** 
  + Een geometrisch DBMS laat toe om geometrische gegevens op te slaan en op te vragen (zowel 2D als 3D) Voorbeeld: Geografische Informatie Systemen (GIS)
* **Sensoren** 
  + Een sensor DBMS **beheert de gegevens van sensoren**
    - Voorbeelden: biomedische gegevens van kledij of medische apparatuur of gegevens uit telematica toepassingen
* **Mobiel** 
  + Een mobiele DBMS werkt op een **mobiel toestel** (smartphone, tablet , …)
  + **Moet altijd online z**ijn en kunnen werken in een **omgeving** **met** een beperkte **verwerkingskracht**, weinig opslagruimte **en beperkte** energievoorziening (**batterij**)
* **Open source** 
  + De code van een open source DBMS is voor iedereen toegankelijk en iedereen kan code toevoegen
  + Zie ook: www.sourceforge.net
  + Voorbeeld: MySQL (Oracle)

# Hoofdstuk 2: Conceptueel model

## Fasen in databankontwerp

Wat zijn de vier fasen van het conceptueel model?

* Fase 1 = Verzamelen en analyseren van de functionele / inhoudelijke vereisten (informatie)
* Fase 2 = Conceptueel ontwerp
* Fase 3 = Logisch ontwerp
* Fase 4 = Fysiek ontwerp

### Fase 1 uitgebreid:

Wat is het doel?

Data verzamelen van wat we nodig hebben/wat nemen we op in de databank? Dit gebeurt via interviews, gesprekken …

Vragen die moeten beantwoord worden

* Welke data moet in de databank worden opgenomen?
* Hoe zal de data worden verwerkt?
* Wat is de beoogde functionaliteit?
* Waarvoor zal de data gebruikt worden?

### Fase 2 uitgebreid:

Wat is het conceptueel model?

Een voorstelling van de gegevens van hoe de databank er eventueel zou uitzien.

Welke gegevens wil je hier opslaan?

We vullen hier al enkele personen in met specifieke/persoonlijke informatie.

### Fase 3 uitgebreid

Type databank is bekend (relationele databank, NoSQL databank, hiërarchische databank, …)

Het product zelf ligt nog niet vast:

* voor relationele databank Microsoft SQL Server of MySQL of DB2 of …
* voor NoSQL document databank MongoDB of CouchDB of …
* voor hiërarchische databank IMS of …

!! Bij het opstellen van het conceptueel model en bij de overgang van het conceptueel model naar het logisch model is er mogelijk verlies van specificaties. → In een apart document !!

### Fase 4 uitgebreid:

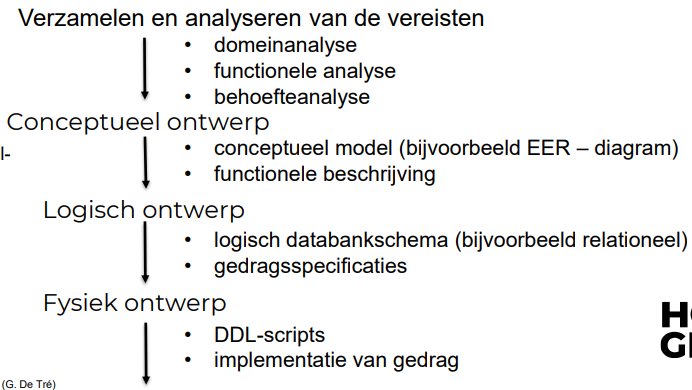
* Hier weet je exact wat je met alles gaat gebruiken.
* Technische details worden toegevoegd (datatypes van de attribuuttypes, …)

### Fasen Samengevat:

* Fase 1: informatie ophalen
* Fase 2: concept maken van alle velden
* Fase 3: concept omvormen in een echte databank met enkele de tabellen, kolommen …
* Fase 4: scripts aanmaken om types te geven aan onze tabellen/velden

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijvingFasen in databank ontwerp:



## Afbeelding met tekst, Lettertype, schermopname, nummer Automatisch gegenereerde beschrijvingEntity Relationship Diagram

Wat is een entiteitstype?

De naam van een table

Wat is een attribuuttype?

Eigenschap van het entiteitstype / beschrijft het entiteitstype.

Wat is het verschil tussen samengestelde en afhankelijke attributen?

* Samengesteld attribuuttype: het attribuuttype kan nog opgesplitst worden. Bijvoorbeeld het attribuuttype ‘**adres’** kan **samengesteld** zijn uit een ‘**straat’, een ‘nummer’, een ‘postcode’ en een 'woonplaats'.**
* Afhankelijk attribuut: **Afhankelijk** van de **context** zullen attribuuttypes soms verder opgesplitst worden of niet. Bijvoorbeeld als het **niet belangrijk** is dat **'straat' of 'woonplaats'** **afzonderlijk moet gekend** zijn, dan wordt **'adres'** een enkelvoudig attribuuttype. In dat geval kan niet met de afzonderlijke delen (straat, stad, …) gewerkt worden.

Wat is het verschil tussen enkelwaardig en meerwaardige attribuuttypes?

* Enkelwaardig attribuuttype: het attribuuttype heeft **één waarde**. Bijvoorbeeld het attribuuttype ‘**titel’ van ‘Film’** en de attribuuttypes ‘geboortejaar’ en ‘jaar overlijden’ van ‘Acteur’.
* Meerwaardig attribuuttype: het attribuuttype kan meerdere waarden bevatten. Bijvoorbeeld een **'Film'** kan **meerdere genres** hebben. In dat geval is 'genre' een meerwaardig attribuuttype.

Wat is een afgeleide attribuuttype?

* Een typisch voorbeeld is ‘leeftijd’: de waarde kan **berekend** worden als het verschil van de huidige datum en de waarde van het attribuuttype ‘geboortedatum’
* Dit wordt **niet opgeslagen** in de databank.

Wat is een kandidaadssleutelattribuuttype?

* Een manier om een entiteit uniek voor te stellen met een **unieke waarde**.
* De attributen die deel uitmaken van een kandidaatsleutel noem je **kandidaatsleutelattributen**.
* Er kunnen **meerdere kandidaatsleutels** zijn. Later wordt uit de kandidaatsleutels één sleutel gekozen als primaire sleutel.

Hoe duiden we dit aan?

* 1 kandidaadssleutelattribuuttype:
  + de sleutel **onderlijnen**.
* Meerdere kandidaadssleutelattribuuttypes:
  + een cirkel buiten de klasse trekken met een u en lijntjes trekken naar de attribuuttypes.

Wat zijn relatietypes?

**Verbindingen tussen entiteitstypes**. Je houdt gegevens bij van een persoon, maar die heeft zit bv. in meerdere films of speelt als meerdere acteurs in films. Tussen deze entiteitstypes wordt dan een relatie gelegd.

Wat kunnen de entiteitstypes zijn?

* Één entiteittype: een acteur kan verwant zijn aan een andere acteur
* Twee entiteittypes: in een film spelen één of meerdere acteurs
* Drie entiteittypes: een arts schrijft een medicijn voor aan een patiënt.

Wat is een relatie-attribuut?

Ook **relatietypes** kunnen **eigenschappen** hebben: wanneer een kenmerk een eigenschap is van het relatietype en niet van één van de betrokken entiteittypes. We spreken van een relatie-attribuut.

Hoe gebeurt een relatie met zichzelf?

* Als er iets binnen de entiteit gebeurt binnen de relatie zelf.
* Bv. een kind heeft één stel ouders.

Hoe duiden we dit aan?

In het midden van de relatielijn tekenen we enkele stippellijntjes met aan het einde een kader met de attributen in.

Wat zijn cardinaliteiten?

* Elk relatietype heeft een minimum- en een maximumcardinaliteit.
* **Cardinaliteit betekent aantal en wordt uitgedrukt als een getal.**
* De cardinaliteiten moeten afgetoetst worden met de opdrachtgever! Deze zijn vaak afhankelijk van de bedrijfsregels. **Modelleer enkel wat je weet**. We veronderstellen niets!

Wat zijn maximumcardinaliteiten?

* Maximumcardinaliteit = het **maximum aantal** **entiteiten** van het entiteittype dat op een gegeven tijdstip kan deelnemen aan een relatie van het relatietype.
* Mogelijke waarden zijn **1 of N**.
  + 1: één entiteit kan in relatie staan met maximum 1 (andere) entiteit via dit relatietype
  + N: één entiteit kan in relatie staan met N (andere) entiteiten via dit relatietype. N is een willekeurig geheel getal groter dan 1.

Wat zijn minimumcardinaliteiten?

* Minimumcardinaliteit = het **minimum aantal entiteiten** van het entiteittype dat op elk tijdstip moet voorkomen in een relatie van het relatietype.
* Mogelijke waarden zijn **0 of 1**.
  + 0: sommige entiteiten nemen niet deel aan de relatie. De relatie is optioneel voor dat entiteittype.
  + 1: een entiteit moet altijd in relatie staan met minimum één andere entiteit
  + Bv. de steden van oefening Activiteiten, hoeveel minimumsteden moeten er zeker zijn.

Hoe worden de minimum- en maximumcardinaliteiten weergegeven?

* Links staan de minimumcardinlaiteiten.
* Rechts staan de maximumcardinaliteiten.

Wat zijn de mogelijke relaties met de cardinaliteiten?

* 1-op-1 relatie 🡪 1:1
* 1-op-veel-relatie → 1:N of N:1
* veel-op-veel-relatie → M:N of N:N

# Hoofdstuk 3 Zwakke entiteiten

## Zwakke entiteitstypes

Wat is een zwak entiteitstype?

Het is **bestaansafhankelijk** van 1 of meerdere andere entiteittypes en **heeft die entiteittypes nodig** om zich te kunnen identificeren.

Wat betekent bestaands afhankelijkheid?

Het kan niet bestaan zonder een andere entiteit.

Hoe duiden we een zwak entitystype aan + kandidaatssleutelattribuuttype?

Afbeelding met tekst, schermopname, diagram, Lettertype

Automatisch gegenereerde beschrijving

Hoe kunnen we zien of iets bestaands afhankelijk is?

* Kijken naar de minimmumcardinaliteiten.
* Het heeft geen kandidaatsatribuutsleutel

# Hoofdstuk 4 enchanced ERD

## Inleiding

Wat is EERD?

Enhanced Enitity Relationship Diagram.

## Specialisatie

Wat is specialisatie?

Een entiteittype is een verzameling van entiteiten met gemeenschappelijke karakteristieken.

Voorbeeld: CrewMember

* Een crewmember kan een acteur zijn, schrijver, regisseur … we houden algemeen gegevens bij van de crewmember maar specialiseren verder in andere entiteitstypes die de attribuuttypes overerven van het hoofdentiteitstype. Bij de extra entiteitstypes kunnen extra attributen worden toegevoegd bovenop de algemene attributen.
* Een subtype heeft GEEN kandidaatsleutelattribuuttype!

Afbeelding met tekst, schermopname, diagram, Lettertype

Automatisch gegenereerde beschrijving

Wat is particpate constraint?

**=> Optional**

**(het is optioneel wat er wordt voorgesteld + een auto hoeft geen onderhoud te krijgen.)**

**=> Mandatory**

**(Je moet dit of dat kiezen + je moet kiezen dat ofwel dit en dat of dit en dat kan zijn)**

Wat is disjoint constraint?

**=> AND**

**(Iets kan dit en dat zijn + een crewmember kan zowel acteur als regisseur zijn)**

**=> OR**

**(Het is of dit of dat + ofwel een klant ofwel een bedrijfsparticulier)**

Wat is een sub- en een supertype?

**Supertype**: de entiteit waar de specialisatie uit voorkwam.

**Subtype**: de 2 nieuwe entiteiten uit het supertype via specialisatie.

Wat is generalisatie en specialisatie?

* **Specialisatie** 
  + het creëren van **specifiekere subtypes** voor een gegeven entiteittype. Een specialisatie definieert een IS-EEN relatie.
* **Generalisatie** 
  + het creëren van een **algemeen supertype** dat de gemeenschappelijke attribuuttypes en relatietypes van een aantal gegeven entiteittypes verenigt. Dit is het omgekeerde proces van specialisatie.

## Beperkingen van het ERD

Wat zijn de beperkingen van het ERD?

* **Tijdelijke beperkingen (dit zijn beperkingen die gelden in een bepaald tijdsinterval) kunnen niet worden gemodelleerd.**
  + Bijvoorbeeld – een project moet binnen een maand toegewezen worden aan een specifiek departement
* **Het ERD kan geen consistentie garanderen tussen verschillende relatietypes**.
  + Bijvoorbeeld - – een werknemer moet werken in het departement waarvan hij de manager is.
* **In het ERD is het domein waartoe attributen behoren, niet bekend. Men kan niet de mogelijke verzameling van waarden voor een attribuut vastleggen.**
  + Bijvoorbeeld - – het aantal uren dat aan een project gewerkt wordt, moet groter of gelijk zijn dan 0.
* **In het ERD is het niet mogelijk de definitie van functies op te nemen.**
  + zoals leeftijd

Wanneer is het ERD correct?

Als het alle vragen naar informatie van de gebruiker kan beantwoorden.

Het model moet in staat zijn om op bepaalde vragen te antwoorden.

# Hoofdstuk 5:Relationeel model

## Inleiding

Het Relationeel **model** is het vervolg van het conceptueel model.

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

Wat zijn tupelverzamelingen?

Dit zijn de tabellen oftewel de relaties.

|  |  |
| --- | --- |
| Term | Definitie |
| Tupel | * Steeds uniek (geen 2 rijen met dezelfde waarden in zijn geheel) * Een geordende lijst met waarden van kenmerken die een object beschrijven) |
| Attribuut | Een kenmerk van een tupel (auto = kleur, aantalWielen, aantalDeuren …) |
| Attribuutwaarde | Inhoud van één cel |
| Domein | Een beperkte verzameling van mogelijke (toegelaten) waarden die voor de attributen in de tupels van een relatie kunnen worden gebruikt. (bv. getallen mogen geen negatieve getallen zijn / kleuren mogen enkel rood, groen, blauw zijn …) |
| Attribuuttype (datatype) | Elke attribuut heeft een type. Dit wordt afgeleid uit het domein.  Types kunnen zijn: numeriek, datum, password, String … |
| tupelverzamelingen | Een verzameling van alle tuppels – de volledige tabel |

Verschil met conceptueel model en relationeel model?

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

Enkele regels:

Elke attribuut kan maar één waard hebben (samengestelde attributen).

elk attribuut is atomair (kan niet meer opgesplitst worden) Specialisatie

Waar zorgen sleutels voor?

Dat elke tupel uniek is.

Het/de unieke attribuut(en) is de **sleutel**.

Welke sleutels zijn er?

* Kandidaatssleutels
* Primaire sleutels
* alternatieve sleutels

|  |  |
| --- | --- |
| Term | Definitie |
| Kandidaatssleutels | Een minimale verzameling van attributen waarvan de combinaitie van elke tupel uniek kan identificeren. Dit kan zowel enkelvoudig als samengesteld.   * Enkelvoudig: Studentnummer * Samengesteld: Vluchtnummer + dag. |
| Primaire sleutels | Deze wordt gekozen uit één van de kanditaatssleutels.  Het is altijd uniek en niet leeg. |
| alternatieve sleutels | Elke kandidaatsleutel die geen primaire sleutel geworden is, wordt een alternatieve sleutel genoemd. |
| Vreemde sleutel | De verbindende schakel tussen twee tupelverzamelingen. |

## Mapping

Hoe gaat mapping te werk?

* Elk entiteittype wordt een tupelverzameling of tabel (opgelet bij specialisatie!).
* Enkelvoudige attribuuttypes overnemen.
* Samengestelde attribuuttypes opsplitsen in enkelvoudige attribuuttypes.
* Meerwaardige attributen in een aparte, nieuwe verzameling plaatsen.
* Primaire sleutel bepalen (opgelet bij zwakke entiteiten!).
* Voor elke relatietype (verband) tussen entiteittypen de vreemde sleutel(s) bepalen.
* Integriteitregels bepalen van elke vreemde sleutel (zeggen wat deze sleutel is en naar welke primairy key hij wijst en of hij uniek is of niet.)

Wat zijn de regels voor het bepalen van de vreemde sleutels?

Je moet kijken welke soorten verbanden je hebt, er zijn 2 soorten verbanden:

**Binaire verbanden** (relatie met andere entiteitstypes):

* Een 1 – N verband: de vreemde sleutel moet aan de N-kant.
* Een 1 - 1 verband: Je mag kiezen waar de vreemde sleutel staat, MAAR, aan de kant met de minste lege (null) waarden.
* Een N – N verband: Je maakt een aparte tabel met 2 vreemde sleutels (beide de primairy keys).

**Unaire verbanden** (relatie met zichzelf):

* Een 1 – N verband: vreemde sleutel in dezelfde (naam van rol 1-zijde).
* Een 1 - 1 verband: vreemde sleutel met zichzelf.
* Een N – N verband: nieuwe tabel met 2 vreemde sleutels (rolnamen gebruiken)!

Zwakke entiteiten:

Bij zwakke entiteiten, neem je zowel de PK over van de entiteit waarmee het verbonden is en de en neem je de attribuuttypes over van de zwakke entiteit over naar het entiteitstype in kwestie.

Wat zijn integeriteitsregels?

* Regels die vermelden met welke primaire sleutel de vreemde sleutel is verbonden.
* Je bepaalt of de sleutel verplicht of optioneel is (dit doe je aan de hand van de miniumcardinaliteit 0/1)
* de sleutel is uniek bij relatie 1-1.

Hoe werkt mapping van specialisatie?

* {Mandetory, AND}:
  + Je maakt **één** tabel met de attributen van het supertype en de subtypes er in samen met 2 booleans van de suptypes op het einde.
* {Optional, AND}:
  + Je maakt **twee** tabellen: één voor het supertype en één voor de twee subtypes met booleans op het einde.
* {Mandetory, OR}:
  + Je maakt **twee** tabellen per subtype ZONDER het supertype met één primaire sleutel per tabel.
* {Optional, OR}:
  + Je maakt **drie** tabellen per soort type met een primaire sleutel.

Afbeelding met tekst, schermopname, Lettertype, algebra

Automatisch gegenereerde beschrijvingVoorbeeld mapping (zie ook excelbestanden):

Structuurbeperkingen:

* **Relationeel model bevat enkel 0 of 1 op veel verbanden:** 
  + Veel op veel verbanden: opsplitsen in twee 1 op veel verbanden
  + 1 op 1 verband: beperkingsregels toepassen: ‘uniek’
* **Minimum cardinaliteit 1:** 
  + bij max. cardinaliteit 1: beperkingsregel: mag niet null zijn
  + bij max. cardinaliteit veel: kan niet afgedwongen worden

# Hoofdstuk 6 Normalisatie

Doel normalisatie:

Meermaals dezelfde data weghalen (het is ruimtebesparend)

Abnormaliteiten weg te halen

Redundantie:

De tabel vertoont meermaals dezelfde data.

Redundantie heeft volgende nadelen:

* Een wijziging moet op verschillende plaatsen gebeuren
* Er kan inconsistentie optreden
* Er ismeer plaats nodig voor het stockeren
* Ook het bijwerken (updaten) van deze tabel geeft problemen, anomalieën genoemd

3 soorten anomalieën:

* Invoeg-anomalie
* Verwijder-anomalie
* Modificatie-anomalie

Oplossing anomalieën:

We kunnen hiervan een aparte tabel maken.

Oorzaak anomalieën:

* Zo goed als elk probleem is het gevolg van:
  + Partiële afhankelijkheden
  + Transitieve afhankelijkheden

## Functionele afhankelijkheid

Een attribuut in de tabel bepaalt altijd een ander attribuut (het is afhankelijk van).

Notatiewijze:

A 🡪 B

Attribuut A bepaalt attribuut B.

Determinant :

een attribuut dat meerdere attributen bepaalt.

Functionele afhankelijk:

Het attribuut dat afhankelijk is van dankzij de determinant.

Partiële functionele afhankelijkheid:

Iets is partieel afhankelijk als:



Het kan enkel samen bij samengestelde sleutels.



Afbeelding met tekst, Lettertype, wit, schermopname

Automatisch gegenereerde beschrijvingDuidelijk voorbeeld beide soorten afhankelijkheid (volledig en partieel)

Aantaluren is volledig afhankelijk van …

NaamPersLid is partieel afhankelijk van …

naamProject is partieel afhankelijk van …

Transitieve Functionele afhankelijkheid:

Iets is Transitieve Functionele afhankelijkheid als:



## normalisatiestappen

Het wordt toegepast op een verzameling met als doel elk attribuuttype te plaatsen in een nieuwe verzameling waar het volledig functioneel afhankelijk is van de gehele primaire sleutel. Waarbij we **partiële en transitieve afhankelijkheden wegwerken**.

Indien je je mapping goed hebt gedaan, moet je normaal niet meer normaliseren. maar de **2** methodes (**top-down en bottom-up**) zijn hier een goede controle van.

Normalisatie bottom up:

Identiferen van functionele afhankelijkheden aan de hand van:

* documenten, formulieren, rapporten …
* gesprek met gebruikers

Stappen normalisatie:

**Stap 1: samenstellen R0 (niet-genormaliseerde verzameling)**

* Welke gegevens komen **1 keer voor**?
* Welke **gegevens herhalen** zich?
* Wat zijn **procesgegevens** (iets wat kan berekent worden zoals **leeftijd of totaal factuur**) of **niet-relevante gegevens** (iets wat niet per se nuttig is om bij te houden)?
* Heeft het **document een identificatie**?

FA opstellen

R0 opstellen

**Stap 2: opstellen R1 a.d.h.v. R0 (R in de 1e normaalvorm)**

* wegwerken **procesgegevens** (kan reeds in voorgaande stap gebeuren – **data** **schrappen**)
* **opsplitsen samengestelde gegevens**
* wegwerken **herhalende gegevens** (HG)

uitwerking:

* herhalende gegevens in **nieuwe verzameling**(**tabel**)
* **sleutel** oorspronkelijke verzameling **mee** overnemen
* bepalen sleutel **nieuwe verzameling**

**Stap 3: opstellen R2 a.d.h.v. R1 (R in de 2e normaalvorm)**

* wegwerken **partiële afhankelijkheden** (enkel maar als er samengestelde sleutels zijn, is er iets partieel afhankelijk):

Uitwerking:

* zoek gegevens die **partieel afhankelijk** zijn
* **verwijder** **deze** uit oorspronkelijke tabel
* breng ze over naar **nieuwe tabel** met als **sleutel de determinant** waarvan ze FA zijn

**Stap 4: opstellen R3 a.d.h.v. R2 (R in de 3e normaalvorm)**

* wegwerken **transitieve afhankelijkheden** (je zoekt enkel in de attribuuttypes, niet in de primaire sleutels):

Uitwerking:

* zoek gegevens die **transitief afhankelijk** zijn
* **verwijder deze** uit oorspronkelijke tabel
* breng deze over in een **nieuwe tabel** met als **sleutel de determinant** waarvan ze FA zijn

**Stap 5: opstellen relationeel model (RM) a.d.h.v. R3**

* elke tabel een **beduidende naa**m geven
* **integriteitregels** toevoegen voor elke **vreemde sleutel**
* integreren in overkoepelend relationeel model (RM)

Samengevat:

* **R0: Herhalende gegevens, procesgegevens en niet-relevante gegevens zoeken**.
* **R1: Procesgegevens** **schrappen** / **samengestelde attributen opsplitsen**.
* Wegwerken **partiële (R2)** en **transitieve** **(R3)** gegevens.
* **R4:** Tabel een naam geven en **IR-regels** toevoegen.

# Hoofdstuk 7: SQL

## Structured Query Language (SQL)

Definitie SQL:

* relationele gegevenstaal voor relationele databanksystemen
* niet-procedurele taal

SQL bestaat uit 3 subtalen:

* **Data Definition Language (DDL)**
  + creatie van een database, en het definiëren van databaseobjecten (tabellen, stored procedures, views,…)
  + CREATE, ALTER, DROP
* **Data Manipulation Language (DML)** 
  + opvragen en manipuleren van de gegevens in een database
  + SELECT, INSERT, UPDATE, DELETE
* **Data Control Language (DCL)** 
  + gegevensbeveiliging en autorisatie
  + GRANT, REVOKE, DENY

DML: raadplegen van gegevens:

**Eén tabel raadplegen:**

* Basisvorm
* SELECT clausule
* WHERE clausule
* Formatteren van rijen
* Statistische functies
* Groeperen

**Meerdere tabellen raadplegen**

## **Afbeelding met tekst, schermopname, Lettertype, nummer Automatisch gegenereerde beschrijving**SELECT statement

Basisvorm SELECT statement:

* **SELECT \* of naam\_veld** **FROM naam\_table** selecteer alles of enkel het veld dat je nodig hebt.

SELECT … WHERE:

Specificatie van de **voorwaarden** waaraan de getoonde rijen moeten voldoen.

bv. WHERE categorieID = 1;

Vergelijkingsoperatoren:

* Wildcards
* Logische operatoren
* Een interval van specifieke waarden
* Een lijst van waarden
* Onbekende waarden
* Je kan haakjes gebruiken om de prioriteitsregels te doorbreken of het geheel leesbaarder te maken.

Vergelijkingsoperatoren:

* Afbeelding met tekst, Lettertype, schermopname, lijn

  Automatisch gegenereerde beschrijving=
* >
* >=
* <=
* <> (niet gelijk aan !=)

Afbeelding met tekst, Lettertype, schermopname, wit

Automatisch gegenereerde beschrijvingWHERE LIKE of NOT LIKE

**‘A%’ = De beginletter is, en de rest maakt niet uit wat volgt.**

Logische operatoren:

* Afbeelding met tekst, schermopname, Lettertype, lijn

  Automatisch gegenereerde beschrijvingOR (dit is de volgorde van prioriteit – haakjes tonen ook een prioriteit)
* AND
* NOT

Operatoren in een interval:

* Afbeelding met tekst, Lettertype, schermopname, wit

  Automatisch gegenereerde beschrijvingBETWEEN
* NOT BETWEEN

Lijst van waarden:

* Afbeelding met tekst, schermopname, Lettertype, lijn

  Automatisch gegenereerde beschrijvingIN
* NOT IN

Testen op onbekende waarden (niet-ingevulde waarden / niet gekend zoals een product die geen prijs heeft):

* Afbeelding met tekst, Lettertype, schermopname, lijn

  Automatisch gegenereerde beschrijvingIS NULL
* IS NOT NULL

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijvingSorteren van de data:

ORDER BY

ORDER BY … DESC (afgaand)

Afbeelding met tekst, Lettertype, schermopname, lijn

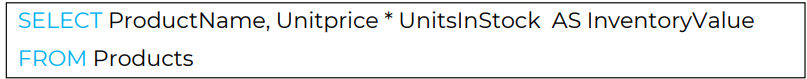
Automatisch gegenereerde beschrijvingUniciteit van de rijen (toont alle unieke waarden):

DISTINCT/ALL

Afbeelding met tekst, Lettertype, schermopname, lijn

Automatisch gegenereerde beschrijvingAliassen:

* Keyword: AS
* of je kan AS weglaten en gewoon een spatie zetten
* AS creeërt een nieuw fictief veld, dit is je uitkomst.

Wiskundige operatoren:

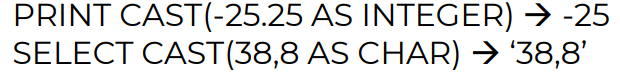
+, -, /, \*

Gebruik van functies:

* **String functies**: concat, left, right, length, substring, replace, ...
* **DateTime functies**: dateAdd, dateDiff, day, month, year, ...
  + **NOW():** retourneert huidige datum en tijd in DATETIME formaat: JJJJ-MM-DD UU:MM:SS
  + **CURDATE()** huidige datum in JJJJ-MM-DD
* **Rekenkundige fun**cties: round, floor, ceil, cos, sin, ...
* Aggregate functies: AVG, SUM, MIN, MAX …
* **IFNULL(kolom, x)** : vervangt gevonden NULL-waardes door x

Voorbeeld IFNULL:



Data conversie:

* CAST
* ****CONVERT

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijvingStrings:

* Stringoperator: concatenatie (CONCAT)
* Afbeelding met tekst, Lettertype, schermopname, lijn

  Automatisch gegenereerde beschrijvingGebruik maken van tekst (literals)

de CASE functie (SWITCH van java):

* SELECT OrderID, Freight,
  + **CASE** 
    - when Freight is null then 'Kost onbekend’ when Freight = 0 then 'Gratis verscheping'
    - when Freight < 20 then 'Lage verschepingskost'
    - when Freight < 100 then 'Gemiddelde verschepingskost'
    - else ‘Hoge verschepingskost'
  + **END** as ‘Verschepingskost’
* FROM orders;

# Hoofdstuk 8 Werken met meerdere tabellen

## Group by en statistische functies

Basisvorm SELECT statement:

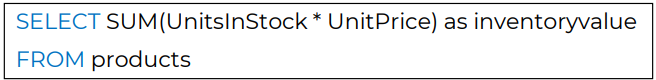
* **Afbeelding met tekst, schermopname, Lettertype, lijn

  Automatisch gegenereerde beschrijvingSELECT** clausule: specificeert de kolommen die je wenst te zien.
  + **DISTINCT** zorgt ervoor dat de getoonde rijen allen uniek zijn
* **FROM** clausule: geeft aan uit welke tabel de gegevens afkomstig zijn
* **WHERE** clausule: opgave van de voorwaarden waaraan de getoonde rijen moeten voldoen
* **ORDER** **BY** clausule: bepaalt de volgorde waarin de rijen getoond moeten worden
* **GROUP** **BY** en HAVING clausule: groeperen van de gegevens

Statistische functies:

* **SUM**(uitdrukking): som
* **AVG**(uitdrukking): gemiddelde
* **MIN**(uitdrukking): minimum
* **MAX**(uitdrukking): maximum
* **COUNT**(\*|[DISTINCT] kolomnaam): aantal

Deze functies geven één antwoord per kolom (of groep) en mogen dus niet in een WHERE clausule (= rijniveau) gebruikt worden.



Som en gemiddelde:

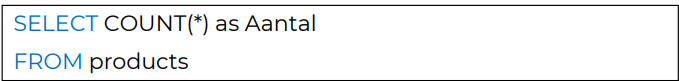
**SUM**

* Berekent de som van alle niet-NULL numerieke waarden in één kolom.

Afbeelding met tekst, Lettertype, schermopname, lijn

Automatisch gegenereerde beschrijving**AVG**

* Berekent het gemiddelde van alle niet-NULL numerieke waarden in één kolom

Aantal rijen tellen:

**COUNT(\*)**

* Telt het aantal rijen van de selectie

**Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijvingCOUNT(kolomnaam)**

* Telt het aantal niet-lege velden in een kolom

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijvingMinimum en maximum:

**MIN en MAX**

* retourneert de kleinste en de grootste waarde in een kolom
* zowel op numerieke als alfanumerieke argumenten

Afbeelding met tekst, Lettertype, schermopname, lijn

Automatisch gegenereerde beschrijvingGroeperen via GROUP BY:

**GROUP BY clausule :**

* Indeling van tabel in groepen van rijen met gemeenschappelijke kenmerken.
* Per groep wordt maar één rij teruggegeven!

SELECT clausule:

Alle waarden uit de SELECT-clause (result set) moeten van hetzelfde niveau zijn:

* Één waarde voor artcode en één waarde voor min(prijs) PER groep (per artcode)
* Meerdere waarden voor artcode en één waarde voor min(prijs) over de hele tabel

ZIE PP HIERTUSSEN

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijvingGroeperen verfijnen via HAVING:

**HAVING clausule**

* Selecteren van groepen op basis van bepaalde groepseigenschappen

WHERE != HAVING:

**Verschil tussen WHERE en HAVING**

* WHERE - heeft betrekking op enkele rijen
* HAVING - heeft betrekking op groepen

## Join

JOIN:

* Selecteren van kolommen uit meerdere tabellen
  + JOIN keyword: specifieert de tabellen die samengevoegd moeten wordem
    - Inner join
    - Outer join
    - Cross join
  + ON keyword: specificeert hoe de tabellen samengevoegd worden

Afbeelding met tekst, Lettertype, schermopname, lijn

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

LEFT en RIGHT en FULL JOIN:

LEFT geeft alles weer van de linkertabel ook al zijn de waarden in de rechtertabel leeg.

RIGHT geeft alles weer van de rechtertabel ook al zijn de waarden in de linkertable leeg.

FULL geeft alles weer van beide tabellen ook al zijn de waarden in beide tabellen leeg.

Inner JOIN:

* Afbeelding met tekst, schermopname, Lettertype

  Automatisch gegenereerde beschrijving= (equi-join)
* <
* >
* <>
* >=
* <=

Relatie met zichzelf:

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

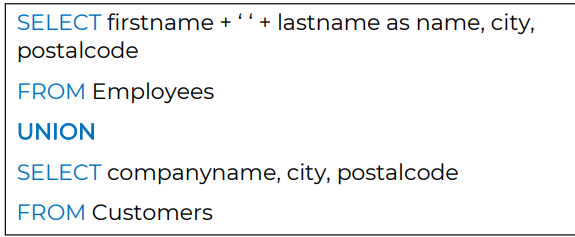
## Union:

Afbeelding met tekst, Lettertype, lijn, Rechthoek

Automatisch gegenereerde beschrijvingUnion:

Via een UNION combineer je het resultaat van 2 of meerdere query’s in 1 resultaattabel

Voorbeeld union:

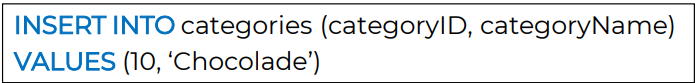
Voorbeeld: geef een overzicht van alle werknemers (naam en voornaam, stad en postcode) en alle klanten (naam, stad en postcode).

# Hoofdstuk 9: SQL -- DML

## Afbeelding met tekst, schermopname, Lettertype Automatisch gegenereerde beschrijvingINSERT:

Toevoegen van rijen in een tabel gebeurt via het INSERT statement.

Toevoegen van enkele waarden:



Toevoegen van alle waarden:

Afbeelding met tekst, Lettertype, schermopname, lijn

Automatisch gegenereerde beschrijving

## Update

Alle waarden van een rij updaten:

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

Wijzigen van 1 rij of een groep van rijen:

Afbeelding met tekst, Lettertype, lijn, schermopname

Automatisch gegenereerde beschrijving

Update meerdere zaken van één product/rij:

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

Update … WHERE … ()

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

CASE – NULLIF

Afbeelding met tekst, Lettertype, lijn, nummer

Automatisch gegenereerde beschrijving

CASE kan hier vervangen worden door WHERE

## Delete – Verwijderen van DATA

Verwijderen van rij(en)

Afbeelding met tekst, Lettertype, schermopname, lijn

Automatisch gegenereerde beschrijving

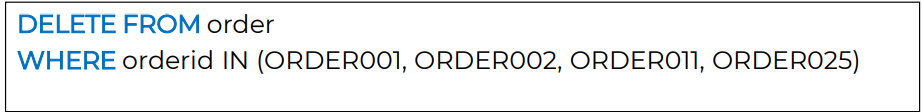
Verwijderen van alle rijen in een tabel



via TRUNCATE zal de het ID-nummer zichzelf weer goed ordenen. (1.2.4.5 🡪 1.2.3.4)



Verwijderen van data – DELETE:



# H10 SQL – DDL

## Inleiding

Volgorde van dit systeem vanbuiten kennen

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

Definitie fysiek model:

* Een logisch model wordt omgezet naar een fysiek model dat nadien wordt geïmplementeerd in een DBMS.
* Het fysiek model bestaat uit:
  + de definitie van de tabellen;
  + per tabel:
    - definitie van de primaire sleutel;
    - definitie van de vreemde sleutels;
    - definitie van de overige kolommen: not null-waarden, integriteitregels, …. (zie ook 2TI);
    - definitie van indexen (zie ook 2TI);
    - toewijzen aan tablespace (zie ook 2TI)

We geven aan onze attribuuttypes nu ook echt datatypes en veldlengtes:

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Datum in type:

yyyymmdd

Definitie van vreemde sleutels:

* Referentiële integriteitsregel:
  + Van een vreemde sleutel moet worden aangegeven of nullwaarden zijn toegestaan. Als uit het conceptueel model blijkt dat de minimale cardinaliteit (ten opzichte van de ‘parent’) gelijk is aan 1, zijn null-waarden niet toegestaan. Dan moet er een

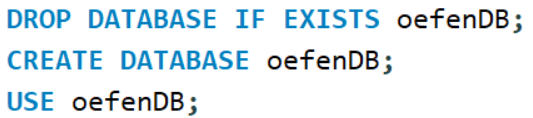
not-nulldeclaratie aan de vreemde sleutelkolom worden toegevoegd.

DDL:

* DDL wordt gebruikt voor
* het definiëren van databanken
* het definiëren van tabellen
* het vastleggen van datatypes
* het definiëren van constraints - data integriteit (2TI)
* het definiëren van indexen (2TI)

## DDL - Databank

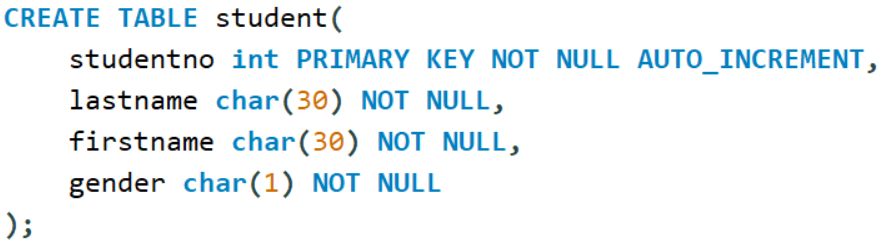
Creëren van een database:



## DDL - Tabellen.

Definiëren van tabellen:

naam veld – type attribuuttype – NULL of NOT NULL



ALTER TABLE:

Het toevoegen van een extra kolom en het aanpassen van de lengte van de datatypes:

Afbeelding met tekst, Lettertype, schermopname

Automatisch gegenereerde beschrijving

DROP TABLE:

DROP TABLE student

Soorten datatypes:

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

## Constraints

AUTO\_INCREMENT waarden:

* een AUTO\_INCREMENT kolom bevat
  + voor elke rij een unieke waarde
  + door het systeem gegenereerde (sequentiële) waarden
* slechts 1 AUTO\_INCREMENT kolom per tabel mogelijk
* maakt gebruik van een integer datatype
* een AUTO\_INCREMENT kolom kan geen NULL waarden bevatten
* een AUTO\_INCREMENT kolom kan je niet zelf aanpassen
  + via LAST\_INSERT\_ID() kan je de laatst gecreëerde waarde opvragen

Voorbeeld AUTO\_INCREMENT:

Afbeelding met tekst, Lettertype, schermopname

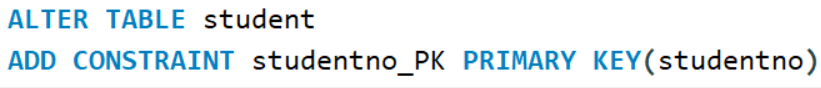
Automatisch gegenereerde beschrijving

Definitie primaire sleutel:

* 1 primary key constraint per tabel
* kan gedefinieerd worden op 1 of meerdere kolommen (samengestelde sleutel)
* waarde (of combinatie van waarden) moet uniek zijn
* NULL waarden zijn niet toegelaten
* DBMS creëert een unieke index op de kolommen

De tabel aanpassen de primaire sleutel zo toevoegen als een apart attribuuttype

(apart definiëren):



OF:

Afbeelding met tekst, Lettertype, schermopname, wit

Automatisch gegenereerde beschrijving

Definitie van vreemde sleutel:

* Gebruikt om verbanden tussen relaties uit te drukken
* 0, 1 of meerdere vreemde sleutels per tabel
* NULL waarden kunnen al dan niet toegelaten zijn
* de constraint waarborgt referentiële integriteit:
  + vreemde sleutels moeten verwijzen naar een primaire sleutel (in MySQL een geïndexeerde kolom) uit een tabel
  + de waarde van een NOT NULL vreemde sleutel moet voorkomen in de gerefereerde kolom
  + legt ook de trapsgewijze (cascading) referentiële integriteitsacties vast bij (2TI)
    - ON DELETE
    - ON UPDATE

Vreemde sleutel maken in een databank:

Afbeelding met tekst, Lettertype, schermopname, lijn

Automatisch gegenereerde beschrijving