



DATABANKEN

GILLES CALLEBAUT

INLEIDING

EERSTE DEFINIETIE VAN DATABASE

Een logische coherente verzameling van gerelateerde gegevens (data), ontworpen voor een specifiek doel en opgeslagen op een “drager”.

File-niveau: adding & removing

Data-niveau: inserting, retrieving, updating deleting

STRATEGIE

- Modelleren:
inzicht krijgen in wat de gebruikers willen, wat resulteert in het **gegevensmodel** (Entiteit/Relatie model)
- Ontwerpen (van de relationele database):
Ontwerp die zo'n accurate mogelijke weergave is van het model
- Implementatie

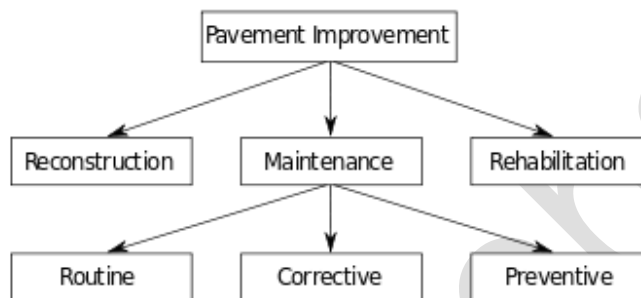
SOORTEN DATABANKEN

- Hiërarchische databank (1:N)

De hiërarchische database gaat ervan uit dat **elk record** in een database weer **kan verwijzen naar een n-aantal andere records**. Het is zo een boomstructuur, die steeds verder kan vertakken. Kenmerkend is wel dat ieder recordtype één en niet meer dan één eigenaar (owner) kent.

Het hiërarchische model kent maar één boom per database, **de takken hebben onderling geen samenhang** en de enige ingang van de boomstructuur is van bovenaf.

Hierarchical Model

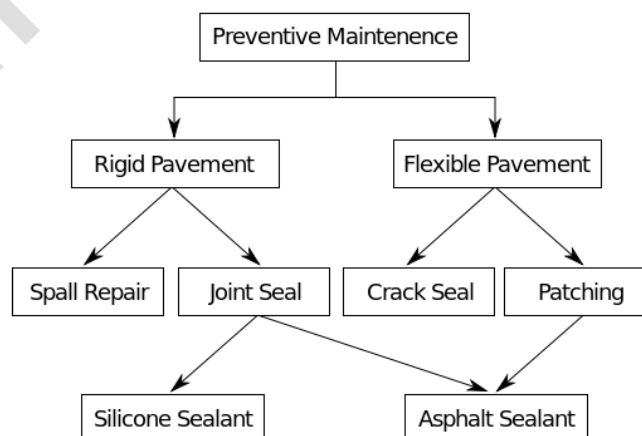


- Netwerk databanken (N:M)

Het breidt de mogelijkheden van gegevensopslag van de eerste generatie databases, de hiërarchische databases, uit met de **mogelijkheden van meerdere paden**, wat impliceert dat de parent-child-relatie tussen records niet meer 1:n is, maar **n:m**. Ieder child kan dus meerdere parents hebben.

De mogelijkheden van de netwerkdatabase zijn groter dan die van de hiërarchische database, maar blijven beperkt. 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. In dit model ontstaat er een **probleem** wanneer ergens in de **database pointers corrupt raken**, dan kunnen **hele delen** van de database opeens **niet meer bereikbaar** zijn. De vrijheid in de hoeveelheid parent-child-relaties maakt ook dat het ontwerp van een netwerkdatabase enorm **complex kan worden**.

Network Model

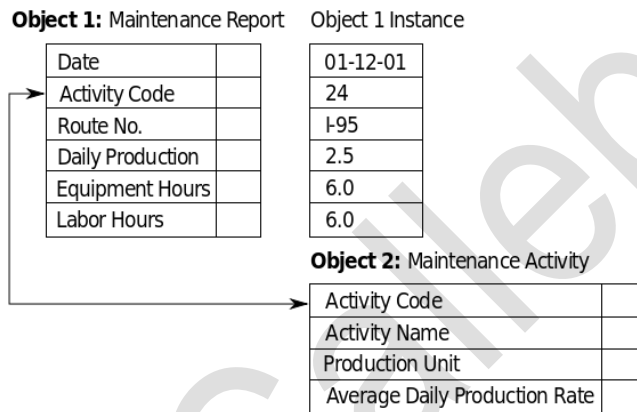


- Objectgeoriënteerde databank

Objectgeoriënteerde database model is een databasemodel waarin wordt gewerkt met objecten, net zoals in objectgeoriënteerde programmeertalen.

Het doel van zo'n database is het invoegen van dergelijke objecten in de database zo eenvoudig te maken. Zo wenst men de objecten die men gebruikt in een objectgeoriënteerde taal direct te kunnen opslaan in de database, zonder transitie naar tupels¹ zoals in een relationele database. Men kan **data sneller opvragen** doordat er **geen join operaties nodig zijn**, men kan de pointers rechtstreeks volgen. Dit is dan een voordeel ten opzichte van relationele databases.

Object-Oriented Model



DOELSTELLING

Gebruik maken van **SQL** om **toegang te krijgen** tot een databank en **bewerkingen uit te voeren** in een databank.

—Gilles Callebaut—

¹ Formeel is een tuple een element van een eindig Cartesisch product.

Een voorbeeld van een tuple is: ("Dorpsstraat", 123, "Jan Janssen", 38) met straatnaam (een string), huisnummer (een natuurlijk getal), naam (ook een string) en leeftijd (ook een natuurlijk getal). Dit is een ander tuple dan ("Dorpsstraat", 123, 38, "Jan Janssen") aangezien deze in een andere volgorde staan.

ALGEMENE BEGRIPPEN I.V.M. DATABASE-VERWERKING

(RELATIONELE) TABEL (FILE)

Een **entiteit** is iets waarover men **informatie** wil **opslaan**.

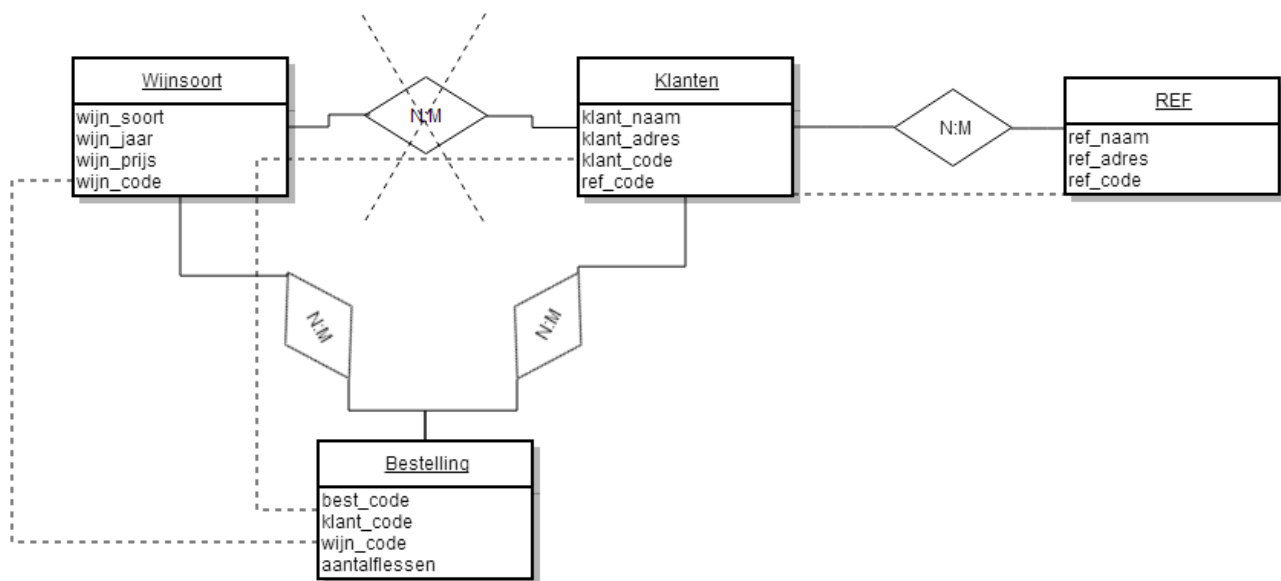
De gegevens hebben verwantschap met elkaar –vormen een **relatie**-, en worden in ene relationele databank opgeslagen in een **tabel**.

Een **rij** van zo'n **tabel** is een **record** of **tupel**.

Een **kolom** van zo'n **tabel** is een **veld** of **attribuut**.

Verschil tussen data en informatie:

Data is de waarden en informatie is de betekenis achter die waarden.



Een **veel-op-veel relatie** geeft aanleiding tot redundantie, dubbele gegevens, veel velden,... de oplossing hiervoor is een **tussentabel** te **creëren**.

Waarbij gebruik wordt gemaakt van sleutels.

CONSTRUCTIE DATABANK

- Definiëren databank
 - vastleggen recordstructuren van iedere file, tabel, datatypes en constraints (opgelegde beperkingen)
- Construeren databank (data opgeslagen)
- Manipuleren databank (quering, updating, rapporteren, generen)

MULTI-USER GEBRUIK

Inconsistent read:

Als er **tussen 2 opdrachten** de informatie al werd **aangepast**.



Phantom read:

Als aangepaste **gegevens niet werden bevestigd**.

Via een **rollback** moet men terug gaan naar de vorige staat.



Lost update problem:

Als er een **update** gebeurde **nadat de oude gegevens** al door een ander werden gelezen.

```
SELECT aantalVaten
FROM STOCK
WHERE item='Duvel'
Bereken 120 -10 =110
```

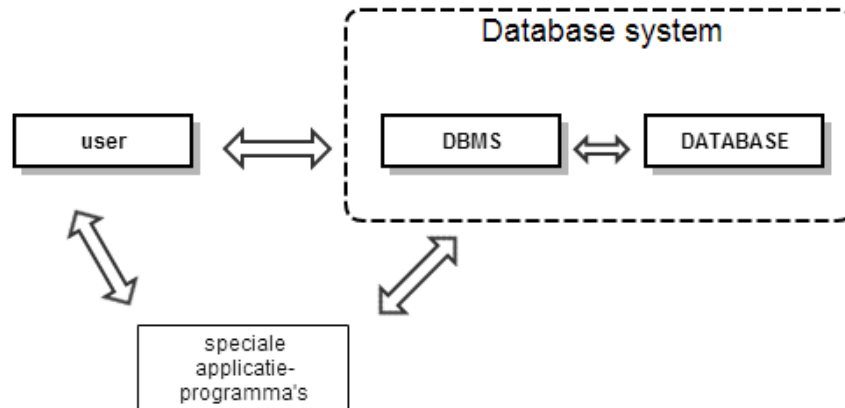
```
UPDATE STOCK
SET aantalVaten=110
WHERE item='Duvel'
```

```
SELECT aantalVaten
FROM STOCK
WHERE item='Duvel'
Bereken 120-30=90
```

```
UPDATE STOCK
SET aantalVaten=90
WHERE item='Duvel'
```

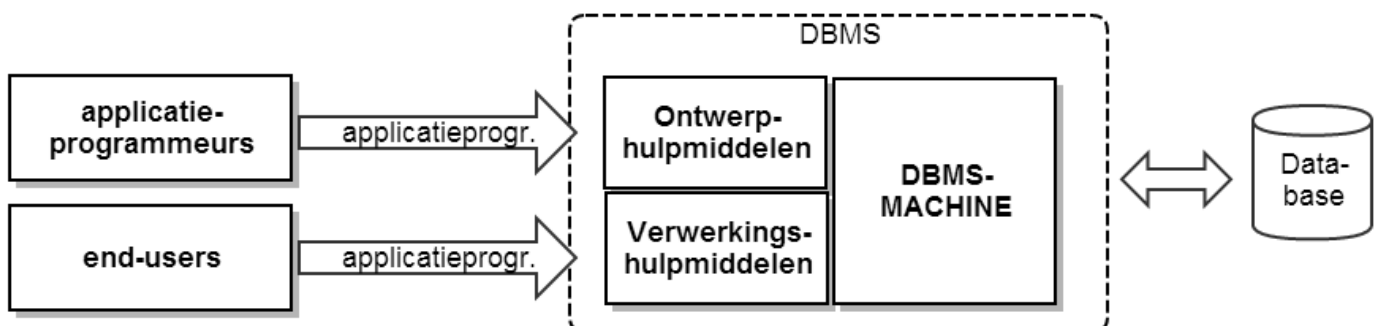
HET DATABASE SYSTEEM

De databank wordt **gecreëerd en gemanipuleerd via het DBMS** (=DataBase Management System).



4 COMPONENTEN VAN DATABASE

- Data
 - Relatietabellen (gebruikersgegevens)
 - Meta-data (beschrijving van eigen structuur)
 - Indexen (toegankelijkheid en verwerkingssnelheid verhogen)
 - Applicatie meta-data (bv. Gegevens over de opmaak van formulieren)
- Hardware
- Software
 - DBMS is de software dat acces tot de databank afhandelt.
Hoofdfunctie: **user-interface** tot databank
= grens waardoor alles onzichtbaar is voor gebruiker.
Componenten DBMS:
 - Ontwerpgereedschappen (ontwerpen en construeren)
 - Run-time programmatuur (vb. query-verwerking)
 - DBMS-machine (verwerken, vb. aanvragen voor gegevens vertalen voor het lezen/schrijven op harde schijf)
- Users
 - End-users
 - Applicatieprogrammeurs
 - Database admin



HET DBMS

Collection of programs that enables users to create and maintain a database.

Vergemakkelijken van processen:

- **Defining** (structure, types, constraints)
- **Constructing** (storage)
- **Manipulating** (query)

DE EVOLUTIE VAN BESTANDSVERWERKINGSSYSTEEM NAAR DATABASEVERWERKINGSSYSTEEM

Bestandsverwerkingssysteem = File-processing

Groepen records in afzonderlijke bestanden opgeslagen.

Bij File-processing definieert en implementeert iedere gebruiker de bestanden, nodig voor een specifieke toepassing als deel van de toepassing.

Beperking:

- Gescheiden en geïsoleerde gegevens
- Gegevens worden vaak gedupliceerd
- Applicatieprogramma's afhankelijk van gebruikte file-structuur
- Incompatibele bestanden (geschreven in een andere taal)

Oplossing:

Database verwerkingssysteem

- Data-independence
- Definitie en beschrijving van database structuur en constraints opgeslagen in system catalog (=meta-data)

Voorbeeld bioloog vs zoöloog, verschillende externe niveaus, maar 1 concept niveau en 1 internal level.

DE 3 BELANGRIJKSTE KARAKTRISTIEKEN VAN EEN DATABANK

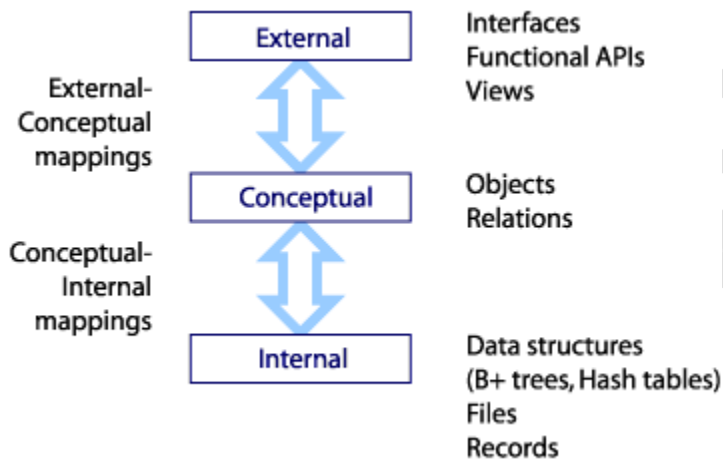
- **Isoleren** van data en programma's (fysische databank en de user applications)
- **Ondersteunen** van **verschillende user views** (zie vb. vissen in rivieren)
- Een **catalogoog** (meta-data) om de **database beschrijving** te stockeren. DB architectuur beïnvloeden.

DATABASE ARCHITECTUUR (ANSI/SPARC)

ANSI American National Standards Institute
SPARC Systems Planning & Requirements Committee

3-LAGEN STRUCTUUR

- **External level** (verschillende views)
external view; **hoe** wordt de **data** “gezien”.
- **Conceptual level** (1 view)
structuur van de gehele **databank**
Bijschrijving, entiteiten, datatypes, relaties, user operations, de constraints;
het opstellen van dit schema gebeurt bij het initieel ontwerp van de database.
- **Internal level**
Hoe data (fysisch) **opgeslagen** is.



Mapping

The processes of transforming requests and results between levels.

DDL:

data definition language

de definitie/ declaratie van data-objects

DML:

data manipulation language

manipulatie of processing van data-objects

De “**storage definition language**” voor de **internal level** wordt gewoonlijk apart gehouden en zit dus niet in SQL.

SQL is dus een high end level DML.

Er kunnen dus meerdere records tegelijk behandeld worden in 1 DML statement.

Taak van de database administrator:

- Definiëren van het conceptuele schema:
Welke info wordt opgeslagen.
gebruik van conceptuele DLL
- Bepalen van het “internal schema”:
Beschrijving van de fysische stockage.

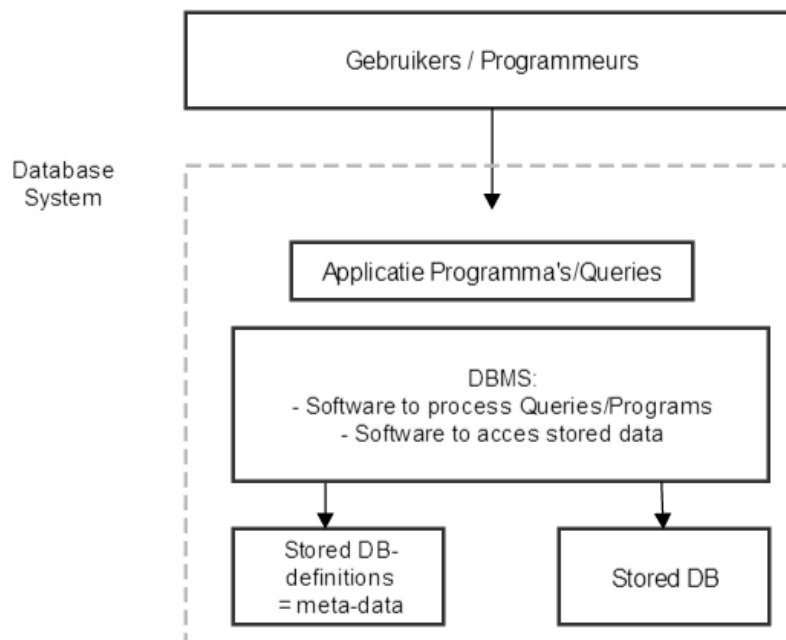
WERKING VAN EEN DBMS

Alle acces behandelt:

- De user verzoekt om acces
- Het DBMS intercepteert dat verzoek en analyseert het
- Het DBMS onderzoekt het “external schema” voor die user
- Het DBMS onderzoekt het “external/conceptual” mapping
- Het DBMS onderzoekt het “external schema”
- Het DBMS onderzoekt de “conceptual/internal” mapping
- Het DBMS onderzoekt het “storage-structure” definitie
- Het DBMS voert de nodige operaties uit op de databank

Functies van het DBMS:

- Het behandelen van de data-definities².
Het DBMS moet “language processor” componenten bevatten van de verschillende DLL-talen en de data-definities kunnen interpreteren en linken met de onderliggende niveaus.
- Data manipulatie (d.m.v. een DML-processor)
verzoek users kunnen behandelen
- Data security, gegevensintegriteit, data recovery
Data dictionary functie voorzien:
Metadata;
Alle schema's en mappings (data-definities dus) moeten fysisch opgeslagen worden in bron- en objectvorm in die bibliotheek, alsook andere informatie. (beschrijving databank)



NIEUWE DEFINITIE DATABANK

Een database is een zichzelf beschrijvende verzameling van geïntegreerde records.

Dus zowel gebruik gegevens MAAR OOK een beschrijving van zijn eigen structuur (data-dictionary).

Belang van data-dictionary:

- Data independence
Programma/gegevensonafhankelijkheid wordt vergroot
immuñiteit van applicaties voor veranderingen in de manier waarop de gegevens gestockeerd en ge-access worden
Externe documentatie zoals bestandsverwerkingssystemen is niet nog want de structuur kan uit de databank zelf gehaald worden
- Bij wijzigingen van de gegevensstructuur in de databank moeten die veranderingen enkel in de data-dictionary worden opgenomen.

Een databank is een model van een gebruikersmodel van de werkelijkheid.

Waarom databanken:

- Redundantie reduceren
- Data kan 'geshared' worden
- Gecentraliseerde controle van data
- Standaardisatie van de gegevens
- veiligheidsbeperkingen i.v.m. toegang

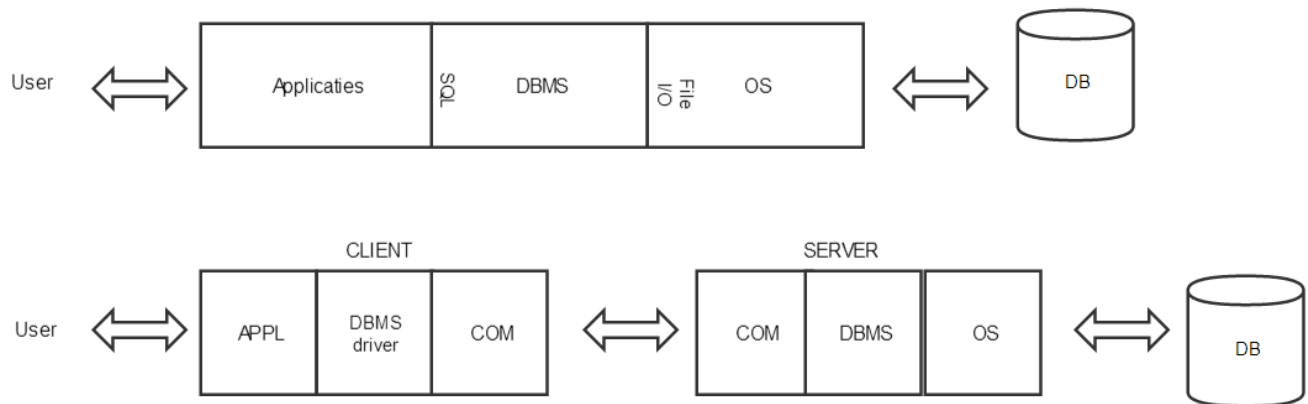
CLIENT/SERVER ARCHITECTUUR

Een client/server systeem bestaat uit een netwerk van computers:

- Client-computers
Deze bevatten applicatieprogramma's en voer applicaties uit.
Ze bevatten DBMS-driver die de aanvragen (in SQL) voor gegevensverwerking van de applicatie ontvangt en doorgeeft aan de DBMS.
Ze accepteren ook de antwoorden van het DBMS en zetten ze om naar de geschikte vorm voor de applicatie
- Server(s)
Deze bevat het DBMS en de programma's voor bestandsbeheer (OS), hier wordt ook de "concurrency controle" (controle op gelijktijdige toegang tot gegevens) afgehandeld.

Zowel client-computers als servers bevatten communicatiesoftware om de berichten tussen DBMS-driver en DBMS te kunnen verzenden en ontvangen.

We maken gebruik van Pc's die voorbereidend werk kunnen leveren (ontlasten van de servers).



HET ENTITEIT-RELATIEMODEL

INLEIDING

Ontwikkelen van databasetoepassingen

- Identificeren van de entiteiten
- Beschrijven van de structuur
- Onderlinge relaties van deze entiteiten

= creëren van gebruiksgegevensmodel

Twee strategieën

- Top-Down
Uitwerking vanuit het gezichtspunt van de hele organisatie
E/R is geschikter
- Bottum-Up
Ontwikkelen specifiek systeem, om daarna verder op te bouwen
Sneller en minder riskant

DEFINITIES

Entiteit

Zwakke entiteit

ID-afhankelijke entiteit

~tabel: onscheidbaar object uit de werkomgeving van de gebruiker

afhankelijk van een andere (sterke) entiteit

entiteit waarvan de identifier ook de identifier van een andere entiteit bevat. Bv een appartement en zijn gebouw

Attribuut

Domain

Identifier

Sleutel

~kolommen: beschrijven kenmerk van entiteit

Pool of legal values

Die een entiteitsinstantie identificeren

Een groep van een of meer attributen die een rij in een tabel **uniek** identificeren

Relatie

Een verband tussen entiteiten,

hoe hangen de tabellen aan elkaar.

Graad van de relatie

Hoeveel entiteiten in de relatie betrokken zijn,

Bij E/R-model bijna alleen binaire relaties

Subtype

De eigenschappen van het supertype worden "geërfd" door het subtype.

Een subtype beschrijft een IS-EEN relatie, nooit een heeft-een relatie!

Het subtype zal dezelfde sleutel hebben als het supertype.

Het supertype zal een algemeen beeld geven waarbij het subtype specifiek, specialiseert.

Bijvoorbeeld, werknemer (supertype) <-> verkoper (subtype).

E/R-DIAGRAMMEN

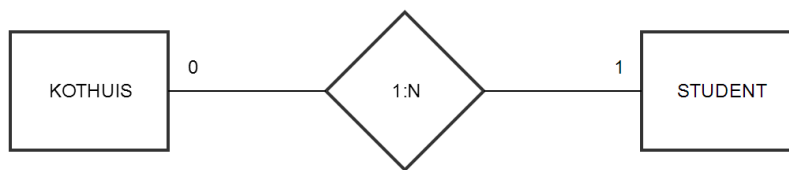
Maximale cardinaliteit

Het maximale aantal entiteiten dat met een andere entiteit kan verbonden worden.

Dit wordt weergegeven in de ruit.

Minimale cardinaliteit

Geeft het minimale verplichte aantal entiteiten weer, wat wordt weergegeven op de verbinding tussen de entiteiten.

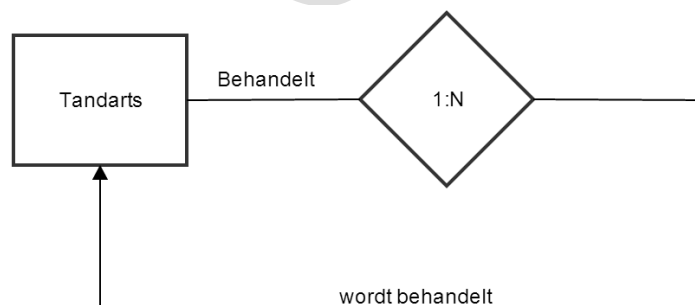


Een student moet geen kot bezitten, maar een kotbaas moet wel minstens 1 student hebben. In één huis kunnen meerdere studenten zitten.

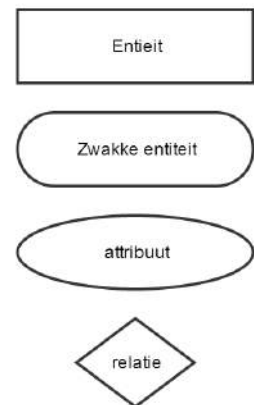
Recursieve relaties (heeft-een relatie)

Relaties tussen entiteiten van dezelfde entiteitsklasse kunnen ook voorkomen.

Een werknemer kan in dezelfde werkgroep zitten met andere werknemers.



Een tandarts behandelt meerder tandartsen, maar hij wordt maar door 1 tandarts behandeld.



ALGEMENE STAPPEN OM TE KOMEN TOT EEN GEGEVENSMODEL

1. Identifieer
Entiteiten, attributen, identificatoren, relaties, subtypen
2. Ontwerp
een set van formele objecten
(E/R model en E/R diagram)
3. Bedenk de formele integriteitsregels
4. Bedenk de nodige formele operatoren
(om stap 2 en 3 te implementeren)

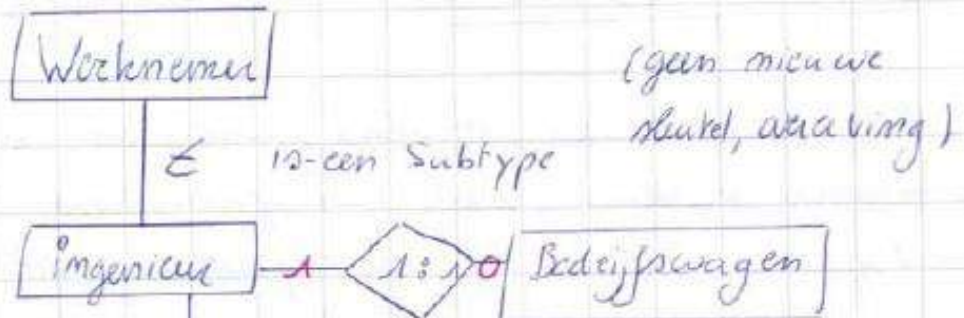
Ontwerp procedure

- Gebruik E/R model om via top-down methodologie de “grote” relaties te ontwerpen, gebruikmakend van entiteiten,...
- Maak van deze “grote” relaties “kleinere” relaties via normalisatie

Oefening:

Een ingenieursbureau heeft een aantal medewerkers. Enkele daarvan zijn ingenieurs. Een ingenieur kan een bedrijfswagen hebben, maar iedere bedrijfswagen is toegekend aan één ingenieur. Ingenieurs verlenen diensten aan klanten. Een ingenieur kan nul of meer diensten verlenen, maar iedere dienst wordt verleend door precies 1 ingenieur. Aan een klant kunnen verschillende diensten verleend worden en dezelfde dienst aan verschillende klanten. Bij iedere klant hoort minstens 1 dienst, maar niet iedere dienst hoeft een klant te hebben. Bij deze relatie wordt de betaling bijgehouden. Som verkrijgt men een klant op aanbeveling van een klant. Aanbeveling is door hoogstens 1 andere klant mogelijk.

Begeleiding



(geen nieuwe
sleutel, afdeling)

is een subtype

- ing kan Bedrijfsw hebben
- maar moet niet o
- wagen moet ingenieur hebben : 1



Dienst

We zullen 3^e tabel nodig hebben met 3 kolommen



Betaling

Klant

D	kl	Betaling

aankomst

Aankomst
aankomst
worden

Opmerking! is een relatie 1:N nooit 1:N / N:N
Heeft een relatie 1:1 / N:M / 1:1

NORMALISATIE

DEFINITIE NORMALISATIE

Normalisatie is een procedure, waardoor een relatie die bepaalde problemen bevat, kan geconverteerd worden naar 2 of meer relaties die deze problemen niet bevatten, en zo ook de werkelijkheid en correctheid van de relatie kan bevorderen.

HET RELATIONELE MODEL

Relatie:

Een 2-dimensionele tabel, waarvan de rijen de gegevens bevatten van een grootheid en de kolommen gegevens over een bepaalde eigenschap van die grootheid.

Een tabel is pas een relatie als

- Er geen dubbele rijen zijn
- Rijen ongeordend, van boven naar onder
- Kolommen ongeordend, van links naar rechts
- Alle kolomwaarden zijn atomic

Of nog uitgebreider

- Iedere cel van de relatie bevat één waarde
- Iedere kolom heeft een unieke naam
- De waarden van de kolommen zijn allempaal van hetzelfde domein
- De volgorde van kolommen heeft geen belang
- **Iedere rij is uniek, er zijn geen duplicaten**
- De volgorde van de rijen is niet van belang

FUNCTIONELE AFHANKELIJKHEID

Een verzameling van attributen Y is functioneel afhankelijk van een verzameling attributen X als en enkel als iedere X-waarde met juist één Y-waarde correspondeert.

Notatie: $X \rightarrow Y$

met X=determinant, dus als je X kent ken je Y maar niet omgekeerd (N:1 verband).

Determinant

Bepaald eenduidig het attribuut, maar daarom niet volledig de rij

Sleutel

Identificeert een rij uniek

Een sleutel is daarom altijd een determinant, maar niet omgekeerd.

Oefening 1

- Volgende tabel houdt bij welke studenten op een sportdag aan welke activiteit deelnemen en welke bijdrage ze daarvoor moeten betalen.

Bepoeking: een student kan maar 1 activiteit doen.

Bepaal determinanten, zoek de sleutel.

Stnr	St. naam	Activiteit	Bijdrage.
001	Jan	Schaken	2
002	Annie	voetbal	3
003	Fred	voetbal	3
004	Mathilde	Yennin	3

- Als je SNe kent, ken je dan activiteit? JA : $SNe \rightarrow Act$
- Bepaald Sme ook de bijdrage? JA : $Sme \rightarrow Bijdrage$
- Bepaald Activiteit de bijdrage? JA : $Act \rightarrow Bijdrage$
- Bepaald Activiteit Sme? Nee
- Bepaald Activiteit Snaam? Nee
- Bepaald Bijdrage Activiteit? Nee
- Bepaald SNe, SNaam? JA : $Sme \rightarrow SNaam$
Determinanten

? Is Sme sleutel? Is Activiteit sleutel?

Sleutel determiniert alle andere attributen/kolommen

→ Sme is een sleutel

→ Activiteit is geen sleutel, want sport kan meerdere keren gekozen worden en Act. determiniert niet Sme, Snaam!

- Sleutel kan maar 1 keer voorkomen oor of oor of...
- Determinant kan meerdere keren voorkomen vb voetbal

- De tabel is geen goede tabel, want als we vb een student schrappen voot dan zijn we alle info over schaken ook kwijt.

- Als SNaam uniek is kan dat ook een sleutel zijn!

Toelichting bij "voor alle mogelijke geldige waarden in de relatie":

Naar volgende tabel zou je tot een verkeerd besluit komen:

Leverancier	Stad	Product	Hoeveelheid
L ₁	Londen	P ₁	100
L ₁	Londen	P ₂	100
L ₂	Parijs	P ₁	200
L ₂	Parijs	P ₂	200
L ₃	Parijs	P ₂	300

Leverancier → Stad

Leverancier ~~X~~ Product

Leverancier ~~X~~ Hoeveelheid → waarde afhankelijkheid, kan twee zijn!

Dus voor alle mogelijke geldige waarden in de relatie: Leverancier is geen sleutel, want bepaald niet alle kolommen.

Leverancier & product bepalen wel de Stad en de Hoeveelheid!

(Lever, product) → Stad

→ Hoeveelheid

Dus (Lever, Product) is de sleutel.

Opm! Er is altijd een sleutel in een relatie van een relationele tabel.

SLEUTELS

Sleutel	Per definitie heeft iedere relatie minstens één sleutel, dit is een set van één of meer attributen waardoor een rij in een tabel uniek geïdentificeerd wordt.
Supersleutel	De sleutel moet hierbij minimaal zijn.
Kandidaat sleutel	Als een relatie meerder sleutels heeft, dan zijn al die sleutels kandidaat sleutels en wordt er één primaire sleutel gekozen, en de anderen zijn dan de secundaire sleutels.
Vreemde sleutel	De primaire sleutel die wordt gebruikt in een andere tabel is daar de vreemde sleutel.

NORMALISATIE

Modificatie anomalieën

- Verwijderanomalie
Wanneer bij het verwijderen van info over ene bepaald feit er ook info over een ander feit verloren gaat.
Voorbeeld: Karel voetbalt niet meer, alle info over voetbal kwijt want hij was de enigste die voetbalde in de tabel.
- Invoeganomalie
Als er gegevens over een bepaald feit pas kunnen ingebracht worden als er gegevens zijn over een ander feit.
Voorbeeld: Men kan rugby niet invoegen omdat het sleutelveld niet is ingevuld.

NORMALISATIE

EERSTE NORMAALVORM (1NF)

- Alle velden uniek zijn
- Data in een kolom van dezelfde soort
- Nooit 2 identieke rijen (dus zeker een sleutel)
- Volgorde rijen en kolommen maakt niet uit

TWEEDE NORMAALVORM (2NF)

Een relatie is in 2NF als alle niet-sleutel attributen functioneel afhankelijk zijn van **de volledige sleutel**, dus niet partieel afhankelijk.

Beleving 1

	SNR	Activiteit	Bijdrage
1	100	netbal	200
2	100	Schaken	30
3	110	Schaken	30
4	120	Judo	75
5	130	Rugby	100
6	140	Zwemmen	300
7	150	judo	75

↳ Rij nr is mooie sleutel, want altijd uniek.

Bepaal de determinanten en sleutel(s)?
Welk attribuut is partieel afh van de sleutel?
Zijn er nog anomalieën mogelijk?

funct. afh : $Act \rightarrow Bijcl$

$(Act, SNR) \rightarrow Bijcl \rightarrow$ sleutel, dus 1 NF

$(Act, Bijcl) \not\rightarrow SNR$

$Act \not\rightarrow SNR$

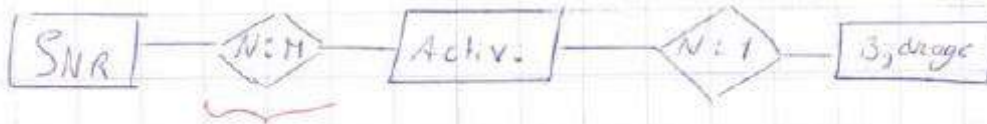
\vdots

\rightarrow partiële afh. want bijclage hangt af van een deel van de sleutel, nl. Activiteit.

Dus Bijcl. is niet afh vd volledige sleutel.

\Rightarrow geen 2NF

goede tabel ?



- kan leiden tot een slechte tabel

- geen goede tabel want tabel heeft anamorfieën

Deze tabel staat niet in 2NF

Hoe kunnen we goede tabel maken?

(Sme, Act)

en

(Act, Bijdrage)

Sme	Act
100	V
100	Z
110	Z
120	V
130	S

(Sme, Act) sleutel

Act	Bjdr.
V	200
Z	100
S	50
B	50

sleutel

FK

PK

- We hebben de tabel gesplitst in 2 tabellen, waarom de FK vd eerste, gebruikt wordt als PK in de tweede tabel!

Oefening 2

SNR	Act	Bijdrage
2000	v	200
2100	z	100
2200	v	200
2300	v	200
2400	v	200

Beperking : Student kan maar 1 activiteit doen.

Functionele afhankelijkheid : $S \rightarrow A$
 $S \rightarrow B$ } sleutel is hier
 $A \rightarrow B$ } SNR!



1NF? JA, want er is een sleutel

2NF? JA, want er is geen part. afh. elke niet-sleutel is afh van de volledige sleutel.

Toch geen goede tabel, want staat niet in 3NF!

Oefening 3

SNR	Huis	Huur
1	BGL10	230
2	BGL10	230
3	CTSR	210
4	WSTR	250
5	WSTR	250

SNR → Huis

Huis → ~~X~~ SNR

Huis → Huur

Huur → ~~X~~ Huis (hier wel, maar vals info) !

SNR → Huur

SNR is de sleutel, dus INF

2NF = ? JA, want sleutel bestaat maar uit 1 attribuut.

3NF = ? SNR → Huis → Huur, wel transitiviteit
dus geen 3NF

DERDE NORMAALVORM (3NF)

Een relatie is in de 3NF als ze in de 2NF is en GEEN transitieve afhankelijkheden bevat.

Transitieve afhankelijkheid: Als $A \rightarrow B$ en $B \rightarrow C$ dan $A \rightarrow C$

Voorbeeld:

Een student huurt een kot in een huis. Iedereen in dat huis betaalt dezelfde huur. Als in 1 tabel bijgehouden wordt welke studenten in welk huis wonen en welke huur ze betalen, dan zondigt deze tabel tegen 3NF.

BOYCE-CODD-NORMAALVORM (BCNF)

Een relatie is in BCNF indien elke determinant een kandidaat sleutel is.

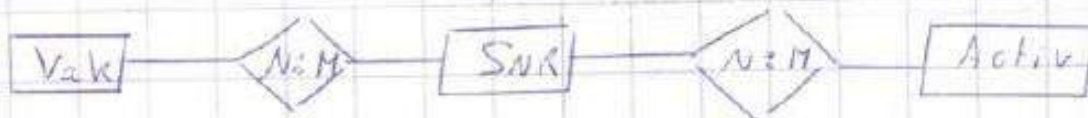
Bepijning 3

SNR	VAK	Activiteit
1	Wink	Zwemmen
1	ENG	Zwemmen
1	Wink	voetbal
1	ENG	voetbal
1	FR	Zwemmen
1	FR	voetbal
1	Wink	Schaken
1	ENG	Schaken
1	FR	Schaken

Met je Hoesken,
andew zou je denken
dat hij enkel wink
kan doen als hij

Is geen goede tabel,
veel te veel dubbele info

Wijzig anomalie! Schaken en geen ENG of FR.



1NF? Ja want we hebben een sleutel.
(SNR, Act, Vak)

2NF? Ja, want Allen zijn 1 van heel de sleutel
afh. (3 tabellen maken Fk)

3NF? Ja geen transitiviteit. $SNR \rightarrow Vak \rightarrow JAARACT$
X

BCNF? Ja want elke determinant is kandidaat sleutel.

4NF? NEE

Blossing kan zijn door ze te splitsen.

(SNR, Vak) (SNR, Act)
Sleutel(S,V) Sleutel(S,A)

↳ Ze hebben nu met elkaar te maken

VIERDE NORMAALVORM (4NF)

Een relatie is aldus in 4NF als de relatie in BCNF is en geen meerwaardige afhankelijkheid bevat.

Meerwaardige afhankelijkheid

In een relatie $R(A,B,C)$ is er een meerwaardige afhankelijkheid als bij een waarde van A meerdere waarden van B en meerder waarden van C kunnen voorkomen, waarbij de waarden van B en C onderling onafhankelijk zijn.

Oplossing voor de anomalieën

2 relaties maken die elk data van één van de twee meerwaardige attributen bevatten.

Oefening:

Studenten (enkel hun Snr wordt in de tabel bijgehouden) leggen examens af van vakken en daarvan worden de punten bijgehouden. Van de student houdt men ook bij wat zijn/haar school was van het vorige schooljaar. In welke NF staat deze tabel.

Snr	vak	School	quotering
1	W	ST-AL	14
1	E	ST-AL	9
1	F	ST-AL	16
2	W	H-F	15
2	E	H-F	10

1NF?

$SNR \xrightarrow{A} school$
 $(SNR, vak) \xrightarrow{C} quotering$
 $(SNR, vak) \xrightarrow{A, B} school$
 $A \rightarrow C$
 $A, B \rightarrow C$

Dus $(SNR, vak) \rightarrow$ Sleutel

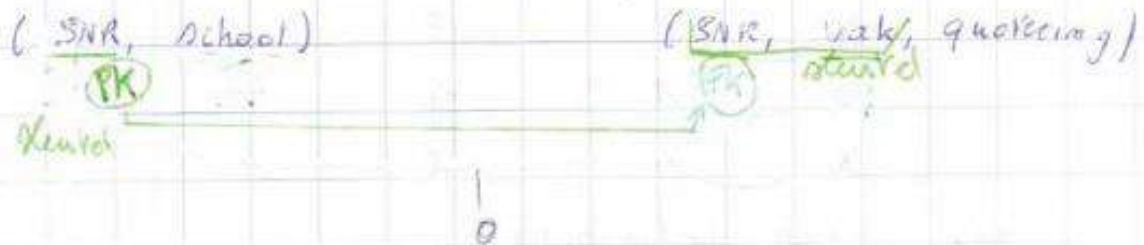
Dus JA 1NF

2NF?

Nee want school is slechts afhankelijk van een deel van de sleutel nl. SNR.

oplossen door te splitsen in 2 tabellen!

Steven naar BCNF (elke determinant, wordt kandidaat sleutel)



Appendix : Bependingen i.v.m. normaliseren

- Het normaliseren kan soms te veel werk veroorzaken (vb. bij queries) en dan kan denormaliseren aangewezen zijn.

Voorbeeld: Veranderet een 1:1 verband tussen gemeente en postcode.

KLANT (kl-naam, ..., Postcode, gemeente, ...)

Stad (gemeente, ^{1:1} postcode)

⇒ Is niet veel info dat eruit gaat, dus niet nodig, zal systeem wel vragen

⇒ Is niet maken geen 2^e tabel maken, is niet nodig!

- Er zijn soms meerdere oplossingen mogelijk, waarbij moet afgewogen worden wat de meest optimale oplossing is voor dat probleem.

Voorbeeld: Aan de universiteit in een land kan maar 1 rector zijn, maar oph. van de universiteit tot maximaal 3 vice-rectoren

Iedere oph. heeft zijn machts

opls Univ (U-naam, rector, vice-rector)

X	A	P
X	A	Q
X	A	R

⇒ Dubbele informatie!

opl 2 :

$\left. \begin{array}{l} \text{Naam} \\ \text{sector} \end{array} \right\} \text{ geen sleutel, bepaald niet eenduidig aan eig}$

2 kdommen maken

Univ

Uname	sector
X	A

↓
PK

Univ2

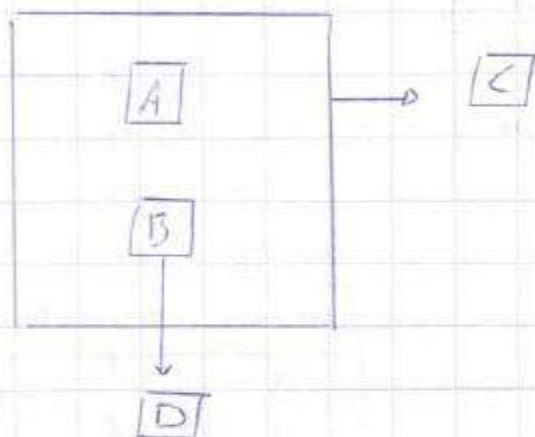
Uname	vr2sector
X	P
X	Q
X	R

↓
FK

⇒ vertaagd het systeem.

opl 3 : Univ (Uname, sector, vr1, vr2, vr3)

Examen Oefening 1



$$\begin{cases} (A, B) \rightarrow C \\ B \rightarrow D \end{cases}$$

A	B	C	D

* Sleutel : (A, B)

augmentation $(A, B) \rightarrow D$

* 1NF ? JA, we hebben een sleutel

* 2NF ? Nee want D is gedeeltelijk afh van de sleutel
namelijk alleen van B.

* oplossing ? deel BCNF, dus iedere ^{kandidaat} determinante / sleutel
 $B \rightarrow D$ hoort niet in de tabel.

Tabel Splitten!

* Stel tabel (A, B, C): 1NF? ja sleutel (A, B)

2NF? ja

3NF? ja

BCNF? ja

* Stel (B, D)

1NF? sleutel B

2NF? ja

3NF? ja

n.a. n.a.

ONTWERPBESLISSINGEN

RELATIESYNTHESE

- Éen-op-één relatie

Afhankelijkheden: $A \rightarrow B$ of $B \rightarrow A$

Sleutel: A of B

Voorbeeld 1:1

610 Bedrijfswagen

001	VW	2008	006
002	VW	2009	000
003	P	2002	012
PK			FK

410 Kadeelid

100	A	--	002
200	B	--	001
212	C	--	003
PK			FK

- PK van K wordt FK van B of omgekeerd.

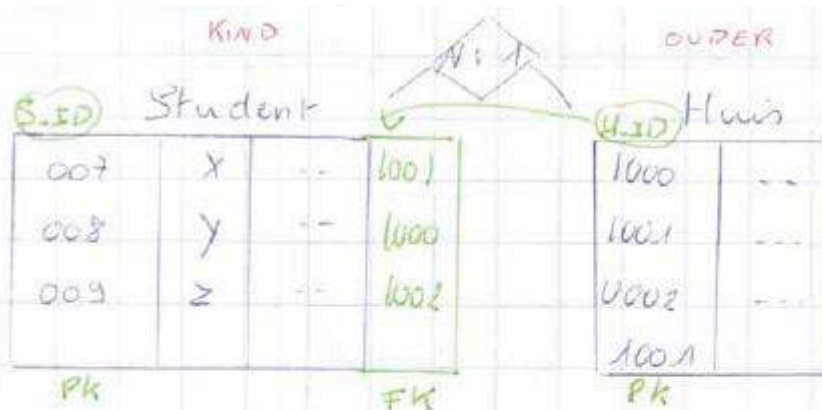
Hangt af van wat je het meest gebruikt dat als PK!

- Veel-op-één relatie

Afhankelijkheden: $A \rightarrow B$

Sleutel: A

Attribuut C mag slechts toegevoegd worden als $A \rightarrow C$



- PK van ouder moet als FK gebruikt worden bij kind, moest je andersom werken zal er dubbele info zijn.

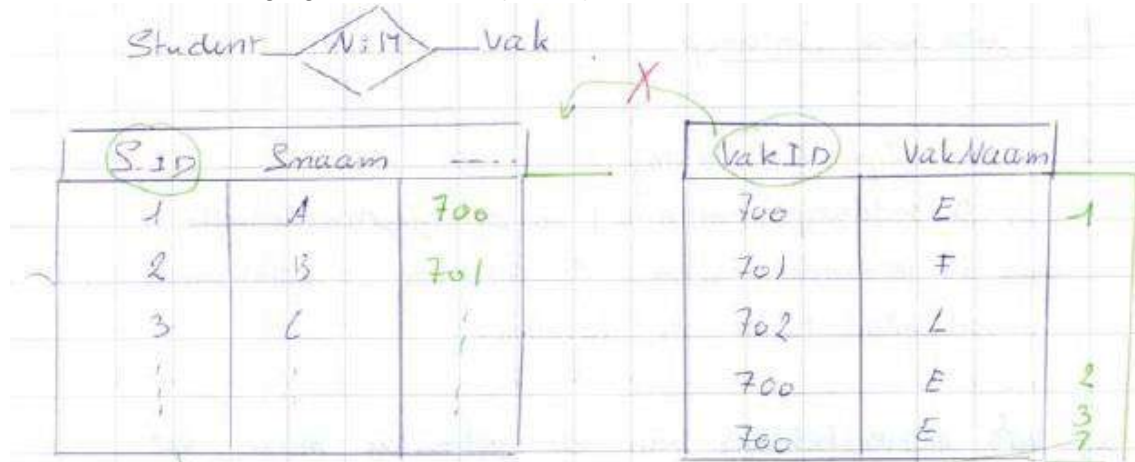
- Veel-op-veel relatie

Afhankelijkheden: noch $A \rightarrow B$, noch $B \rightarrow A$

Sleutels: (A,B), A én B dus

Attribuut C mag slechts toegevoegd worden als $(A,B) \rightarrow C$

Dus als door toevoeging C, de sleutel (A,B,C) wordt dan hoort C niet thuis in die tabel.



⇒ Doorsnedetabel maken

- geen dubbele info enkel nu dubbel, niet alle geg over die persoon of dat vak
- Score mag, want heeft betrekking op zowel Student als vak.

S.ID	V.ID	Score
1	700	14
1	702	16
2	701	8
3	703	20

Mogen we het toevoegen?

Nee, want heeft niets te maken met vak.
Indien we hem toevoegen is het geen goede tabel, want part. afh.

Mogen we toevoegen hoeveel keer student, vak gevolgd heeft? Ja, want heeft betrekking op zowel Student als vak (student, vak) → Hoeveel

Mogen we bijhouden in welk lokaal het vak gevolgd wordt? Nee, want dan is de sleutel (vak, student, lokaal)

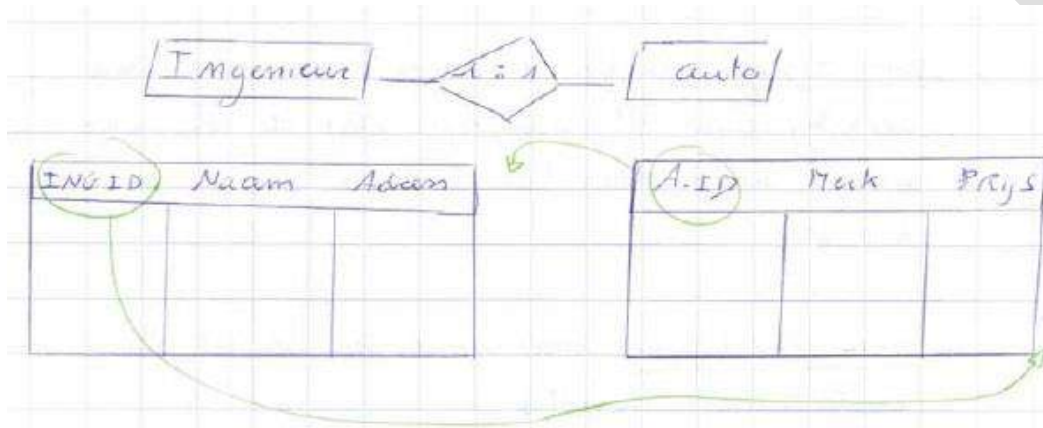
DATABASE ONTWERP

De binaire HEEFT-EEN relatie

1:1

Creëer voor iedere entiteit een tabel en neem de sleutel van één van beide tabellen op in de andere tabel.

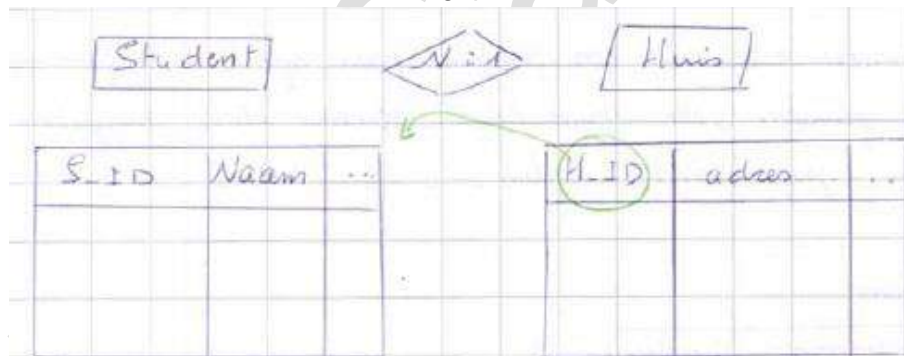
Deze sleutel wordt vreemde/ foreign key genoemd.



1:N

De "ouder" staat aan de 1-kant en het "kind" is de entiteit aan de N-kant.

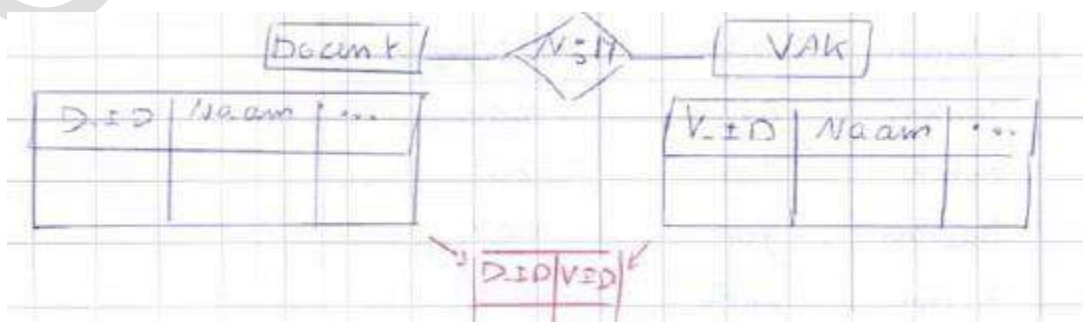
De sleutel van de ouder wordt in de kant -tabel geplaatst. Dit moet om redundantie te vermijden.



N:M

Om redundantie en anomalieën te voorkomen, moet er een doorsnede-tabel gecreëerd worden.

Splits daarom de oorspronkelijke N:M in twee 1:N relaties, waarbij de sleutel van de doorsnede-tabel gevormd wordt door de sleutels van de 2 ouders.



RECURSIEVE RELATIES

1:1 recursieve relatie

Een persoon kan sponsor zijn van 1 andere persoon, en men kan slechts door hoogstens 1 persoon gesponsord worden.

Person

P_ID	P_Naam	P_adres
001	Janssens	Bxl
002	Verbeke	Amr
003	MacKens	Gent
004	Pickens	Snrk



We hebben 1 tabel, maar we doen of ze 2x bestaat.

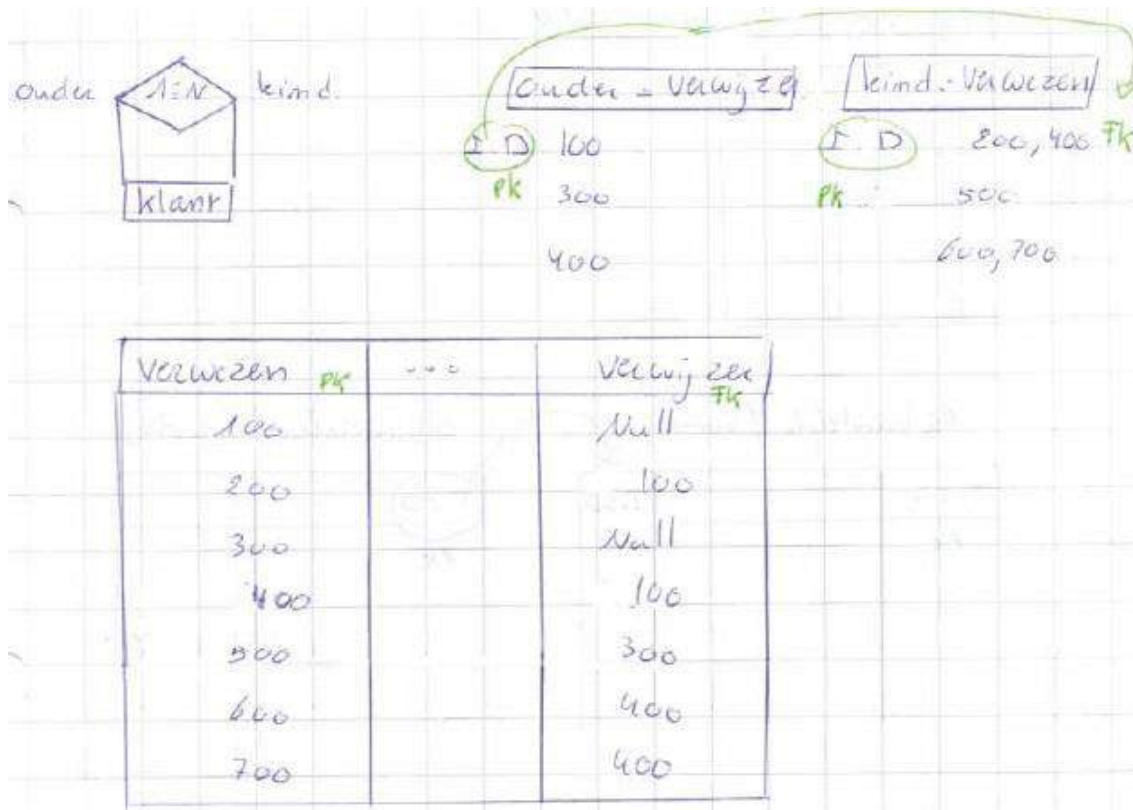
SPONSOR (P_ID, naam, ...)

GESPONSORD (ID, naam)

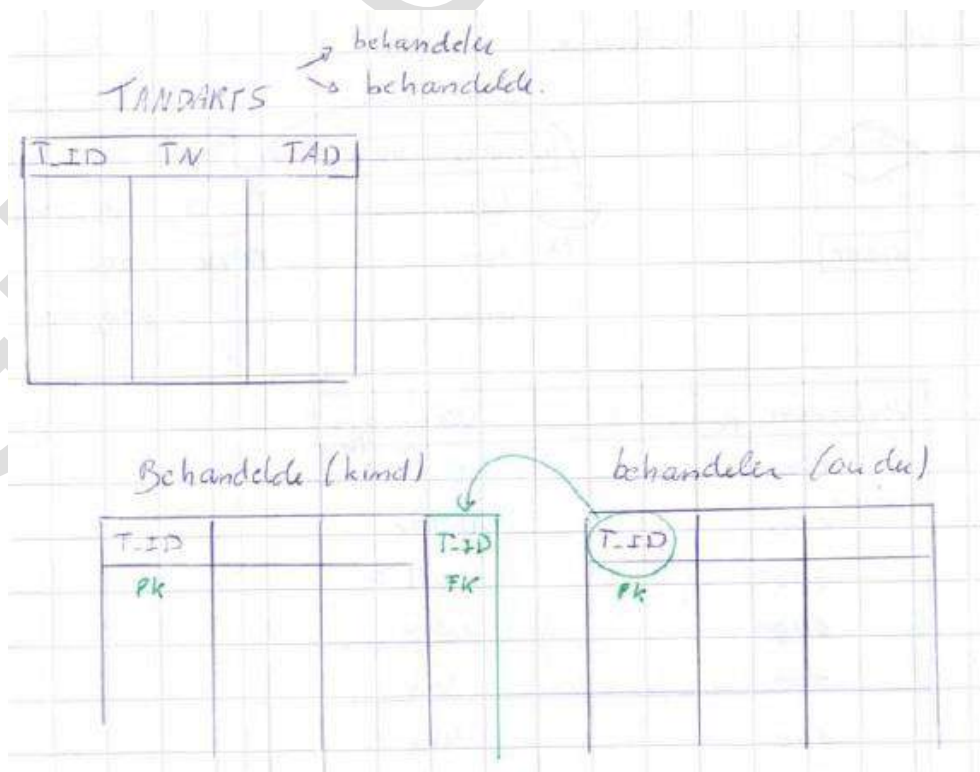
P_ID	P_Naam	P_adres	P_ID(gesponsord)
001	Janssens	Bxl	002
002	Verbeke	Amr	Null
003	MacKens	Gent	004
004	Pickens	Snrk	Null

1:N recursieve relatie

Een klant kan door meerde klanten aangeraden worden om in een winkel aankopen te doen, maar iedere klant wordt finaal maar door 1 klant echt overtuigd.

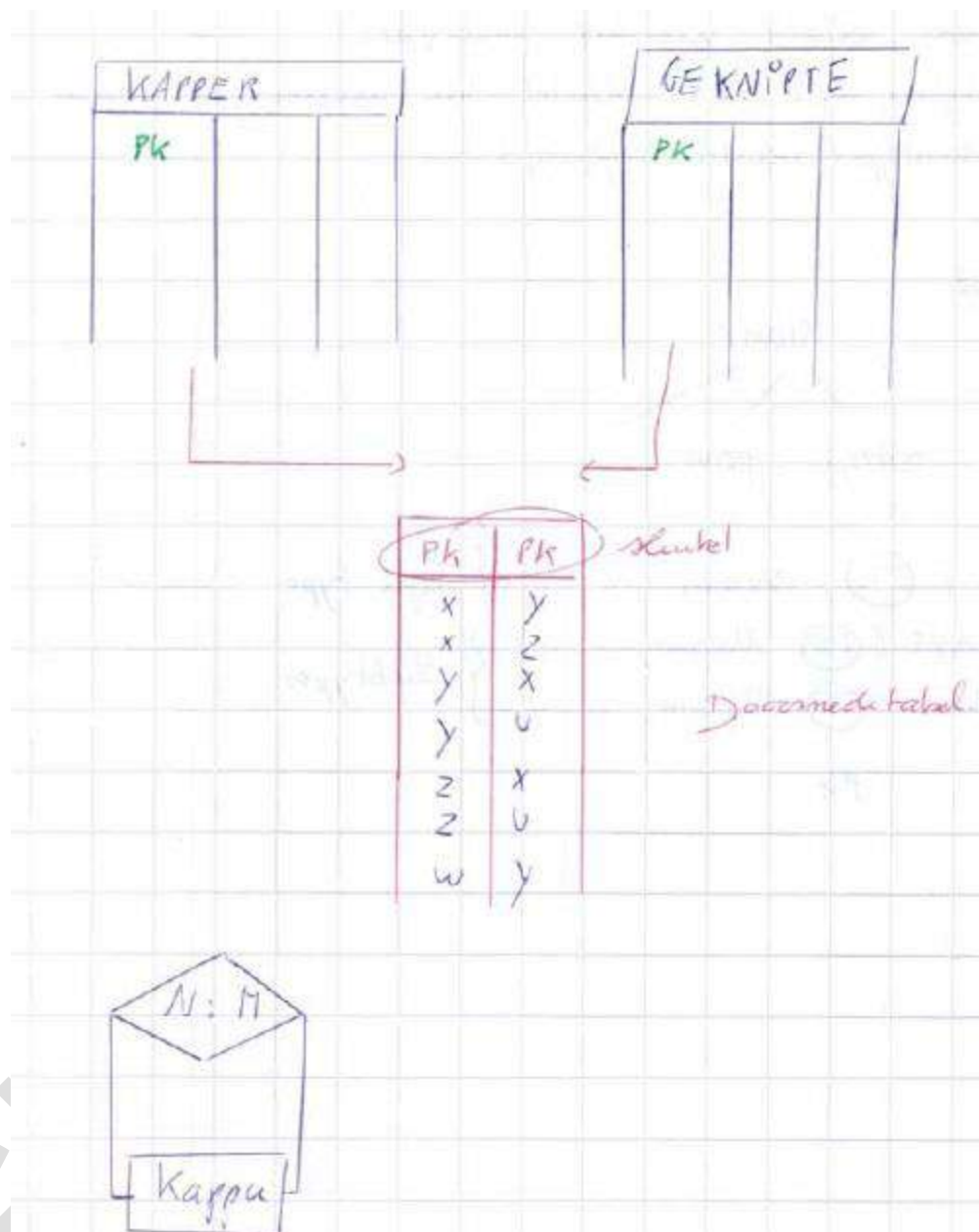


1 Tandarts kan meerdere tandartsen behandelen, maar hij wil maar door 1 tandarts behandeld worden.



N:M recursieve relatie

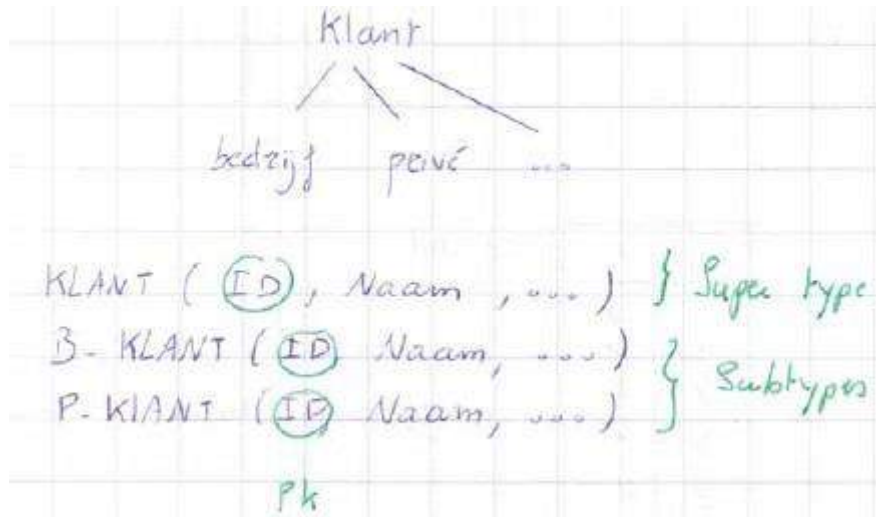
Een kapper kan het haar van meerdere kappers knippen, en laat zijn haar door meerder kappers knippen.



De binaire IS-EEN relatie

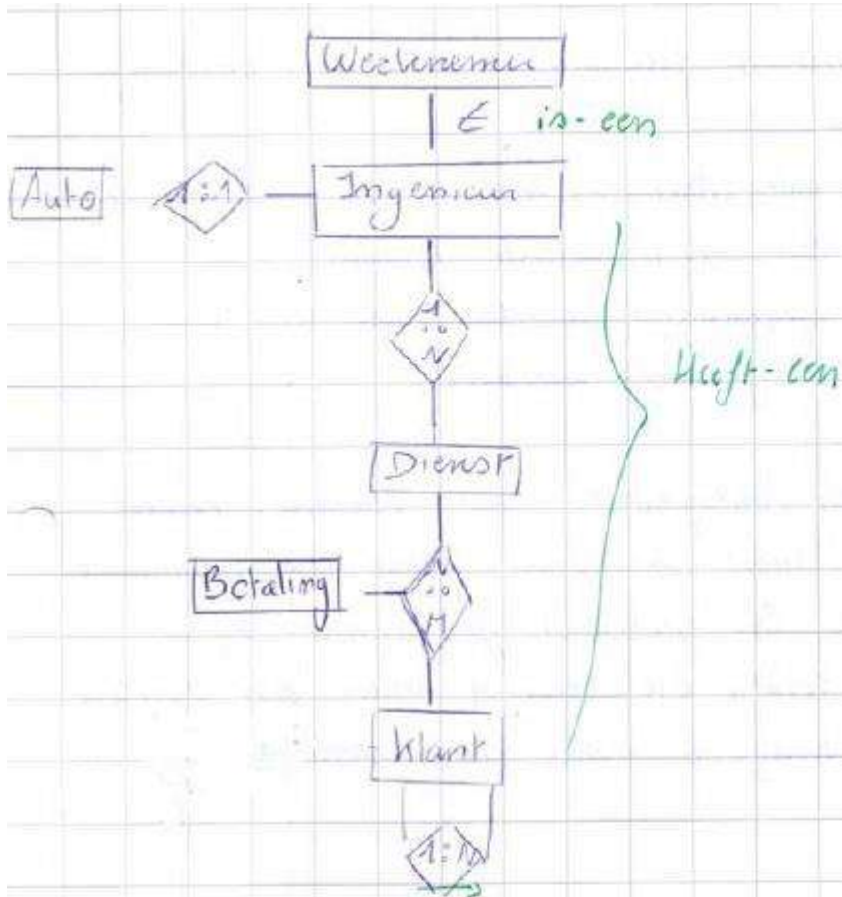
Deze relatie bestaat bij entiteiten die subtypen zijn binnen een bepaalde klasse.

Er wordt een relatie gecreëerd voor het supertype en een relatie voor het subtype.
Vervolgens wordt de sleutel van het supertype aan de subtype-sleutel toegevoegd.



Oefening:

Een ingenieursbureau heeft een aantal medewerkers. Enkele daarvan zijn ingenieurs. Een ingenieur kan een bedrijfswagen hebben, maar iedere bedrijfswagen is toegekend aan één ingenieur. Ingenieurs verlenen diensten aan klanten. Een ingenieur kan nul of meer diensten verlenen, maar iedere dienst wordt verleend door precies 1 ingenieur. Aan een klant kunnen verschillende diensten verleend worden en dezelfde dienst aan verschillende klanten. Bij iedere klant hoort minstens 1 dienst, maar niet iedere dienst hoeft een klant te hebben. Bij deze relatie wordt de betaling bijgehouden. Som verkrijgt men een klant op aanbeveling van een klant. Aanbeveling is door hoogstens 1 andere klant mogelijk.



Wekenreizen (W-ID^{PK}, Naam, ...)
 Ingenieur (W-ID^{PK}, Naam, ...)
 Auto (A-ID^{PK}, ... , W-ID^{FK})
 Dienst (D-ID^{PK}, ... , W-ID^{FK})
 Klant (KL-ID^{PK}, -kl-Naam, KL-ID^{FK})
 -kl-D (D-ID^{PK}, KL-ID^{FK}, Betaling^{FK})
 Dienst-aan-klant tabel.

aanzaden als FK
 in aangegeven tabel.

INTEGRITEIT

Het relationele model is frequent beschreven, inclusief 2 integriteitsregels:

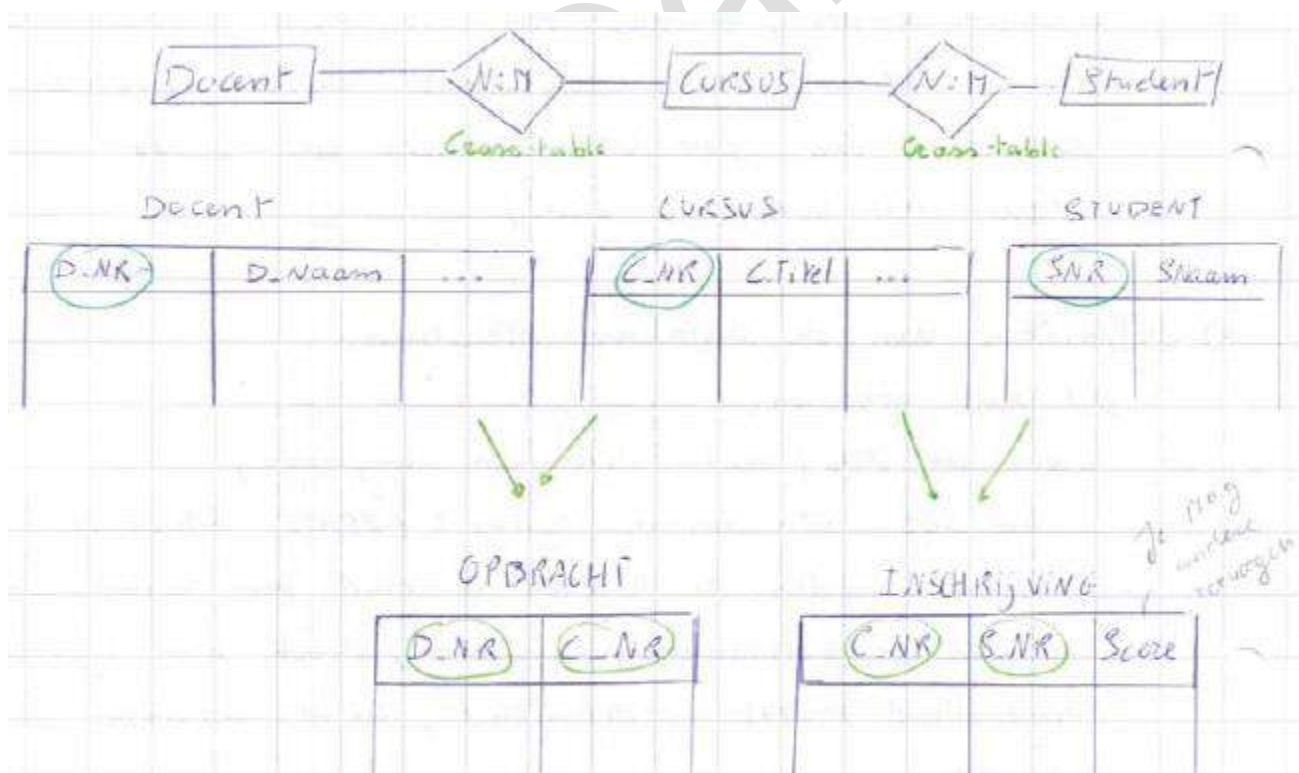
1. De entiteit integriteit (PK)
Geen enkele primaire sleutel kan een NULL-waarde hebben
(er kan geen info opgeslagen worden over iets dat niet kan geïdentificeerd worden)
2. De referentie integriteit (FK)
Een vreemde sleutel van een tabel moet verwijzen naar een primaire sleutel van de in verband gebrachte rij.
(een rij in één relatie die verwijst naar een andere relatie moet verwijzen naar een bestaande rij in de relatie)
3. De attribuut integriteit
Ieder attribuut moet voldoen aan de beperking dat zijn waarde komt uit een relevant domein.
(voorbeeld: *Geboortejaar kan niet bestaan uit 21bc*)

DATABASE IMPLEMENTATIE

Definiëren van de database-structuur

Voorbeeld:

Een docent geeft meerdere cursussen, die elk gevolgd worden door meerdere studenten.



CREATE TABLE DOCENT *moet je invullen, want is PK*
 (D_NR INTEGER NOT NULL PRIMARY KEY,
 D_NAAM VARCHAR(40) NOT NULL,
 D_ADRES VARCHAR(80)
)

CREATE TABLE CURSUS
 (C_NR Integer (ook: Serial) NOT NULL PRIMARY KEY,
 C_Titel VARCHAR(80) NOT NULL,
)

Naam mag je kiezen, maar laten wel koppeling maken.
 (CREATE TABLE OPDRACHT
 (Cursus_Nr Integer NOT NULL,
Docent_Nr Integer NOT NULL,
 CONSTRAINT Opdracht_PK PRIMARY KEY (Cursus_Nr,
 docent_Nr)
 CONSTRAINT Opdracht_FK1 FOREIGN KEY (docent_Nr),
 REFERENCES DOCENT (D_NR),
 CONSTRAINT Opdracht_FK2 FOREIGN KEY (Cursus_Nr),
 REFERENCES CURSUS (C_NR)
)

Basisbegrippen van relationele algebra

De bewerking op één of meer relaties is altijd opnieuw een relatie, daarom noemt men de relationele algebra: "gesloten".

De unie

De rijen van de eerste relatie worden toegevoegd aan alle rijen van de tweede tabel toe te voegen.

1	x	+	3	z	→	1	x
2	y					2	y
						3	z

Het verschil

De derde tabel bevat die rijen van de eerste die niet in de tweede voorkomen.

1	x	-	1	x	→	2	y
2	y		3	z			

De doorsnede

De derde relatie bevat die rijen, die zowel in de eerste als in de tweede voorkomt.

1	x		1	x	→	1	x
2	y		3	y			

Het cartesisch product

Alle mogelijke combinaties maken met de rijen.

2	x	3
1	x	A
2	y	B
		C
		k
		L
		M

1x	AK	} = 6
1x	BL	
1x	CM	
2y	AK	
2y	BL	
2y	CM	

De selectie
De projectie

Maakt een horizontale deelverzameling uit een tabel
Selecteert één of meer kolommen uit een tabel
het maakt dus een verticale deelverzameling uit een tabel

SELECT SNR, Snaam
FROM STUDENT
WHERE SAGE < 20

projectie
Selectie

STUDENT				
SNR	Snaam	Sadres	...	SAGE
015	x	BXL		19
016	y	Amv		18

gewenste tabel

Join

Voegt rijen samen uit 2 relaties op grond van gespecificeerde attribuutwaarden

het is het resultaat van 3 bewerkingen:

1. $A * B$
2. Selectie volgens criterium
3. Projectie van de aangegeven kolommen

Hier wordt altijd een natural join bedoeld

de attributen in de voorwaarde moeten ook een gemeenschappelijk domein hebben

Noot o.v.m. de join:

- Equi-join of inner-join: alleen de ^(rij)records waarvan de waarde in het ^(kolom)gemeenschappelijk veld gelijk is, komen voor;

speciaal geval: natural join: in de resultaten tabel komt er maar 1 kolom voorzien voor de attributen die als selectiecriterium gebruikt werden.

Met als voorbeeld: LEDEN - Inschrijvingen - Sportclubs:

- left outer join: toont alle leden, ook diegenen zonder inschrijving (als dat geregeld is in het ontwerp) samen met de inschrijvingen waarnaar verwezen wordt vanuit leden
- right outer join: toont alle inschrijvingen (ook die waarvoor nog geen lid ingevuld is) samen met de leden die inschrijvingen hebben.

QUERIES UIT 1 TABEL

```
SELECT kolommen --projectie
FROM tabel
WHERE selectievoorwaarden --selectie
ORDER BY kolom ASC | DESC --sorteren
(meer in Hoofdstuk hierna)
```

QUERIES UIT MEERDERE TABBELEN

Met subqueries

```
SELECT kolom
FROM tabel1
WHERE selectievoorwaarden
IN (
    SELECT kolom
    FROM tabel2
    WHERE selectievoorwaarde
)
```

Deze werkwijze werkt maar goed zolang de attributen van het resultaat uit slechts 1 tabel komen, indien dit niet het geval is, moeten tabellen worden samengevoegd (join).

Alsook moet het resultaat van de tweede SELECT overeenkomen met attribuut dat in de WHERE werd gebruikt.

Voorbeeld:

Wat zijn de namen van de studenten die zich ingeschreven hebben voor Engels?

STUDENT		INS			CURSUS	
Snr	Snaam	Snr	Cursusnr	Score	Cnr	Cnaam
1	X	1	100	12	100	FR
2	Y	2	102	16	101	EN
3	Z	3	100	10	102	NL
		2	101	15		
		3	101	9		

VIA SUBQUERY

```
SELECT Snaam -- resultaat = Y en Z
FROM STUDENT
WHERE Snr
IN ( -- resultaat = 2 en 3
    SELECT Stnr
    FROM INS
    WHERE Cursusnr
    IN ( -- resultaat = 101
        SELECT Cnr
        FROM CURSUS
        WHERE Cnaam='EN'
    )
)
```

Zwakke punt van Subquery is dat het efficiënt is zolang dat de oplossing maar uit 1 tabel moet komen, dus redelijk beperkt.

VIA JOIN

Geef de naam van de studenten weer die Engels volgen met hun score

STUDENT		INS			CURSUS	
Snr	Snaam	Stnr	Cursusnr	Score	Cnr	Cnaam
2	Y	2	101	15	100	FR
3	Z	3	101	9	101	EN

Cartesisch product : $2 \times 2 \times 2$ (rijen) = 8 rijen

②	Y	②	101	15	100	FR
②	Y	②	①01	①5	①01	①EN
2	Y	3	101	9	100	FR
2	Y	3	101	9	101	EN
3	Z	2	101	15	100	FR
3	Z	2	101	15	101	EN
3	Z	3	101	9	100	FR
③	②	③	①01	①9	①01	①EN

Oplossing:

Y → 15
Z → 9

1. A*B
2. Selectie volgens criterium
3. Projectie

```
SELECT STUDENT.Snaam, INS.score --projectie
FROM STUDENT, INS, CURSUS -- cartesisch product
WHERE STUDENT.Snr = INS.St.nr
      AND INSCH.Cursusnr = CURSUS.Cnr
      AND CURSUS.Cnaam = 'EN' -- selectie (3 VW)
```

EXIST EN NOT EXIST (SUBQUERIES)

Zijn logische operatoren, hun waarde is dus TRUE als er rijen voorkomen die voldoen aan de voorwaarden die in hun kwalificaties zijn geformuleerd (omgekeerd FALSE).

EXIST

Voorbeeld:

Wat zijn de studentenummers van de studenten die op meer dan 1 vak zijn ingeschreven?

INS

Stnr	Cnr	G
1	100	15
1	102	9
2	100	12

A en B

A x B

A			B		
1	100	15	1	100	15
1	100	15	1	102	9
1	100	15	2	100	12
1	102	9	1	100	15
1	102	9	1	102	9
1	102	9	2	100	12
2	100	12	1	100	15
2	100	12	1	102	9
2	100	12	2	100	12

```
SELECT DISTINCT Stnr
FROM INS AS A -- enkel A deel wordt geselecteerd
WHERE EXISTS ( -- TRUE want subquery bevat rijen
  SELECT *
  FROM INS AS B
  WHERE A.stnr = B.stnr -- zelfde Stnr
  AND A.Cnr NOT = B.Cnr -- verschillend vak
)
```

NOT EXIST

Iedere rij van de FROM-tabel waarvoor het resultaat van de query leeg is wordt geselecteerd.

Voorbeeld:

Wat zijn de namen van de studenten die zich op ALLE cursussen hebben ingeschreven?

Vraag veranderen:

Zijn er studenten waarvoor NIET geldt dat er vakken zijn waarvoor ze zich NIET hebben ingeschreven.

STUDENT		INS		CURSUS
Snr	Snaam	Snr	Cursusnaam	Cnaam
1	X	1	C ₁	C ₁
2	Y	1	C ₂	C ₂
		2	C ₂	

```
SELECT Snaam
FROM STUDENTEN
WHERE NOT EXISTS ( -- studenten waarvoor niet geldt dat er vakken zijn die...
    SELECT *
    FROM CURSUS
    WHERE NOT EXISTS ( -- ...zich niet hebben ingeschreven
        SELECT *
        FROM INS
        WHERE INS.Snr = STUDEN.Snr
        AND Cursus.Cnaam= INS.Cursusnaam
    )
)
```

Het is niet echt een cartesisch product, maar we leggen het wel uit met een cart. product.

STUDENT

X
INS

CURSUS

1	X	1	C1	C1	←
1	X	1	C1	C2	
1	X	1	C2	C1	
1	X	1	C2	C2	←
1	X	2	C2	C1	
1	X	2	C2	C2	
2	Y	1	C1	C1	
2	Y	1	C1	C2	
2	Y	1	C2	C1	
2	Y	1	C2	C2	
2	Y	2	C2	C1	
2	Y	2	C2	C2	←

STUDENT

CURSUS

1 X C1

1 X C2

2 Y C1 ←

2 Y C2

We selecteren deze

lij want Student 2

is niet ingeschreven

voor C1

STUDENT

1 X ←

2 Y

opt.

We schrappen Stud

die zich niet voer

Alle vakken heeft ingeschreven

STRUCTURED QUERY LANGUAGE

SYNTAX

Extra uitleg

SELECT kolom_naam

Selecteren van data uit een database

FROM tafel_naam;

of

SELECT * **FROM** tafel_naam;

Selecteren van alle kolommen

SELECT DISTINCT kolom_naam

Enkel verschillende waarden selecteren

FROM tafel_naam;

SELECT kolom_naam

Selecteer enkel records die voldoen aan het criterium

FROM tafel_naam

WHERE kolom_naam operator waarde;

SELECT kolom_naam

FROM tafel_naam

WHERE kolom_naam operator waarde

AND (...OR...);

SELECT kolom_naam

FROM tafel_naam

ORDER BY kolom_naam, kolom_naam

Sorteren op stijgende of dalende volgorde (default=ASC)

ASC|DESC;

INSERT INTO tafel_naam

Voeg een nieuwe rij toe met deze waardes

(kolom1, kolom2,...)

VALUES (waarde1, waarde2,...);

UPDATE tafel_naam

Verander de waarden in kolom1 en kolom2

SET kolom1=waarde1, kolom2=waarde2,...

...

WHERE kolom=waarde;

op de plaats met kolom=waarde

DELETE FROM tafel_naam

Verwijder de rij waar kolom=waarde

WHERE kolom=waarde;

DELETE FROM * **FROM** tafel_naam;

Alle rijen verwijderen '*' moet er niet staan

SELECT COUNT(*)

Geeft het aantal geselecteerde rijen weer

FROM tabel

SELECT COUNT(**kolom**)
FROM table

Geeft het aantal rijen weer zonder NULL
waarde in de specifieke kolom

SELECT COUNT(**DISCTINCT** kolom)
FROM table

SELECT
AVG()
MAX()
MIN()
SUM()
FROM tabel

SELECT kolom, SUM()
FROM tafel
GROUP BY kolom

Die SUM() is een voorbeeld

Groeperen op die kolom

SELECT kolom, SUM()
FROM tafel
GROUP BY kolom
HAVING SUM(kolom) conditie

Kan ook een andere functie zijn bv. AVG()

Aliassen:
SELECT kolom **AS** alias_naam
FROM tafel;

SELECT kolom
FROM tafel **AS** alias_naam;

SELECT kolom
FROM tafel
WHERE kolom **IN** [...,...]

Meerdere waarden in een WHERE
selecteren

WHERE kolom **NOT IN** [...,...]