KU LEUVEN

Inleiding tot databanken

1. Introductie

Prof. dr. Paolo Pilozzi

Overzicht

1.0 Cursus (= Slides) & Examen
1.1 Databanken?
1.2 Concepten & Architectuur
1.3 Relationeel Model

1.1 Databanken?

Introductie op de inleiding tot databanken
Hoofdstuk 1

* Wat is een databank?
* DBMS: Databank Management Systeem
* Databank Systeem

Wat is een databank?

Databank = Collectie gerelateerde data
Data = Feiten met impliciete betekenis

Vb. Naam, Telefoonnr., E-mail, Adres van kenissen in adresboek, spreadsheet of contactenlijst

Maar! Dan zijn de woorden/zinnen in deze presentatie ook een databank ...

Wat is een databank?

Databank = Collectie gerelateerde data
Data = Feiten met impliciete betekenis

Databank heeft impliciete eigenschappen:

* Aspecten van de echte wereld = **mini-wereld**
  - Bij wijziging zal mini-wereld wijzigen
* Logisch coherrente collectie van data met **betekenis**
* Databank bevat data met een **doel**
  - Toepassing en gebruikers

Wat is een databank?

Databank = Collectie gerelateerde data
Data = Feiten met impliciete betekenis
Mini-wereld + Betekenis + Doel

M.a.w. *Databank vindt zijn oorsprong in de echte wereld vanwaar de data afgeleid wordt, kent interactie met de echte wereld om er een reflectie van te zijn, en heeft een publiek dat er actief interesse in toont!*

Wat is een databank?

Databank = Collectie gerelateerde data
Data = Feiten met impliciete betekenis
Mini-wereld + **Betekenis** + **Doel**

* Eender welke grootte en/of complexiteit
  Contactenlijst vs. Netflix vs. Google search
* Kan manueel of automatisch (algemeen & specifiek)
  Adresboek (Boek+Pen) vs. PostgreSQL (**DBMS**) vs. Google search engine

Vb. van Databank

STUDENT

Name	Student_NR	Class	Major
Smith	17	1	CS
Brown	8	2	CS

SECTION

Section_ID	Course_NR	Semester	Year	Instructor
85	MATH2410	Fall	23	King
92	CS1310	Fall	23	Anderson
102	CS3320	Spring	24	Knuth
112	MATH2410	Fall	24	Chang
119	CS1310	Fall	24	Anderson
135	CS3380	Fall	24	Stone

COURSE

Course_Name	Course_NR	Credits	Department
Intro CS	CS1310	4	CS
Data Struct	CS3320	4	CS
Discr Math	MATH2410	3	MATH
Intro DB	CS3380	3	CS

GRADES

Student_NR	Section_ID	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQ

Course_NR	Prereq_NR
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Vermijden van redundantie!
Duidelijke onveranderlijke naamgeving.

Vb. van Databank

STUDENT

Name	Student_NR	Class	Major
Smith	17	1	CS
Brown	8	2	CS

SECTION

Section_ID	Course_NR	Semester	Year	Instructor
85	MATH2410	Fall	23	King
92	CS1310	Fall	23	Anderson
102	CS3320	Spring	24	Knuth
112	MATH2410	Fall	24	Chang
119	CS1310	Fall	24	Anderson
135	CS3380	Fall	24	Stone

COURSE

Course_Name	Course_NR	Credits	Department
Intro CS	CS1310	4	CS
Data Struct	CS3320	4	CS
Discr Math	MATH2410	3	MATH
Intro DB	CS3380	3	CS

GRADES

Student_NR	Section_ID	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQ

Course_NR	Prereq_NR
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Hergebruik via referenties om redundatie te vermijden.

Vb. van Databank Meta-data

RELATIONS

Relation_Name	Nr_Of_Columns
STUDENT	4
COURSE	4
SECTION	5
GRADES	3
PREREQ	2

COLUMNS

Column_Name	Data_Type	Belongs_To
Name	Char (30)	STUDENT
Student_NR	Char (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_Type	STUDENT
Course_Name	Char (10)	COURSE
Course_NR	XXXXNNNN	COURSE
...	...	...
Prereq_NR	XXXXNNNN	PREREQ

*** Zelf-beschrijvend**
*** Data abstractie**
*** Afschermen van applicaties**
*** Opslagoptimalisatie**

Verborgen voor gebruikers:
Hoe records opslagen op datadrager.

Data_Item_Name	Starting_Pos_Record	Length_In_Bytes
Name	1	30
Student_NR	31	4
Class	35	1
Major	36	4

Vb. van Databank Views

TRANSCRIPTS

Student_Name	Student_Transcript				
	Course_NR	Grade	Semester	Year	Section_ID
Smith	CS1310	C	Fall	24	119
	MATH2410	B	Fall	24	112
Brown	MATH2410	A	Fall	23	85
	CS1310	A	Fall	23	92
	CS3320	B	Spring	24	102
	CS3380	A	Fall	24	135

COURSE_PREREQ

Course_Name	Course_NR	Prerequisites	Prereq_Name
Intro DB	CS3380	CS3320	Data Struct
		MATH2410	Discr Math
Data Struct	CS3320	CS1310	Intro CS

Ondersteuning van verschillende weergaves op basis van dezelfde data.

Bijkomend dienen databanken: Ondersteuning van simultane toegang!

Wat is een DBMS?

Database Management System

Collectie van programma's voor beheer van een databank.
= General purpose software met volgende processen:

- Defniëren: Data types, Structuren, Constraints
  = Meta-data deel van databank (Catalog/Dictionary)
- Constructie: Opslaan van data op datadrager (opslagmedium)
- Manipulatie: Queries voor ophalen, aanpassingen, rapportage
- Delen: Simultane toegang voor gebruikers en toepassingen

Wat is een DBMS?

Database Management System

Een applicatie/gebruiker stuurt **query's** naar een DBMS voor interactie met een databank.

Query = Vraag => Enkel data ophalen uit een databank?
Neen! Vraag tot ophalen, aanmaak, verwijderen, aanpassen, e.a.
Een **transactie** is een logisch geheel van operaties op een databank (~= meerdere query's die een werkgeheel vormen).

Wat is een DBMS?

Actoren (in grote databanken)

* **Administrator**
  - Toegang, Coordinatie, Monitoring, SW en HW
* **Ontwerpers**
  - Welke data? Welke structuren? Hoe opslaan? Hoe weergeven? Hoe gebruiken?
  - Coordinatie met gebruiker!
* **Eindgebruikers**
  - Casual (database query language)
  - Naive (Parametrisch) standaard query's (canned transactions)
  - Sophisticated (ingenieurs, ..., analysten) ter implementatie v. toepassingen
  - Standalone (DB onderhoud via user interfaces)

Wat is een DBMS?

Actoren (in grote databanken)

Achter de schermen

* System design
* System development
* Tool development
* Operations (Sysops)
* Maintainance

Wat is een DBMS?

DBMS algemeen vs. specifiek

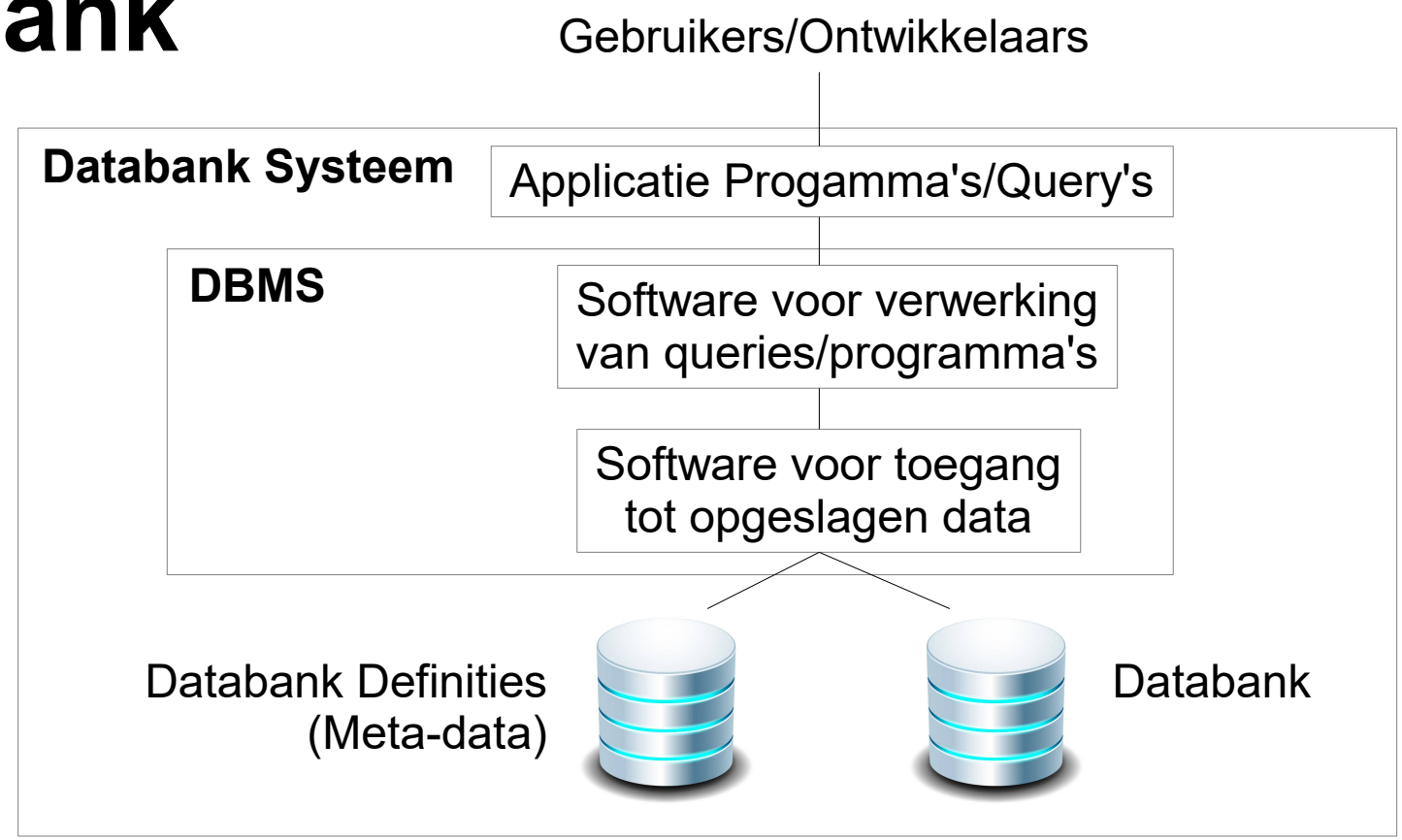
Bekende (R)DBMS systemen voor algemene doeleinden:
PostgreSQL, MySQL, MSSQL, OracleDB, MariaDB, ...

Waarom specifieke DBMS ontwikkelen? Want heel veel werk.
Efficientie! Hoe algemener, hoe meer afwegingen, hoe trager!
Meestal enkel voor heel specifieke "eenvoudige" problemen die heel erg veel data betreffen, waarmee efficient gewerkt moet kunnen worden. Zoals een search engine. Dus applicatie afhankelijk!
Dus wanneer de nodige efficientie het werk waard is!

Wat is een Database Systeem?

DBMS + Databank

DBMS, wat nog?
Efficientie
Redundantie
Systeem beveiliging
Data beveiliging
Data integriteit & complexiteit
Onderhoud & Updates
Backups & Recovery
Distibutie & Beschikbaarheid
Archivering
...



1.2 Concepten en Architectuur

Evolutie van monolitisch, sterk geintegreerd, mainframe, gecentraliseerd naar modern, modulair, client/server, gedistribueerd.
Hoofdstuk 2

* Data modellen
* Schema's en instanties
* 3-schema DBMS architectuur
* Data-onafhankelijkheid
* DBMS talen en interfaces
* Databank systeem omgeving

Data modellen

Eén van de belangrijkste karakteristieken van de databank aanpak is:
Data abstractie = Gebruikers zien data met het gewenste detail.

Via **Data model** = concepten ter beschrijving structuur databank
Structuur = data types van, relaties tussen, beperkingen op de data
Data modellen kunnen ook

- *basis-operatoren op data beschrijven*
- *dynamische aspecten en gedrag beschrijven*

Categorieën v. Data modellen

Hoog-niveau: **Conceptuele modellen**
- Dichtbij gebruiker/ontwikkelaar
  - Vb. Entiteiten-relatie model (volgende les)

Midden-niveau: **Implementatiemodellen**
- Hoe data en relaties voorstellen
- Nog steeds abstract maar voldoende dicht bij gebruiker
  - Vb. Relationeel model

Laag-niveau: **Fysische modellen**
- Details van dataopslag, bestandsorganisatie, indexering (toegangspaden)

Naar implementatie toe is het nuttig om kennis te hebben van fysische modellen, en dit voor efficientie-redenen.

Schema's & Instanties

Databank Schema = Beschrijving van databank ≠ databank

Specificatie bij ontwerp en verandert zelden

Betreft eigenschappen databank: vb. entiteiten en hun attributen

Schema diagramma bestaat uit schema constructoren:

STUDENT

Name	Student_NR	Class	Major

COURSE

Course_Name	Course_NR	Credits	Department

SECTION

Section_ID	Course_NR	Semester	Year	Instructor

GRADES

Student_NR	Section_ID	Grade

PREREQ

Course_NR	Prereq_NR

Het schema diagramma bevat niet de data types, noch constraints.

Deze maken echter wel deel uit van een databank schema!

Een schema beschrijft een intentie, hoe data in de databank zit!

Schema's & Instanties

Instantie = *Een invulling van data volgens een schema construct*
Databank staat/toestand = *Data in een databank op een gegeven moment*
Een toestand is gelinkt aan een set van instanties voor elke constructor
Bij elke aanpassing van een databank is er een ***toestandsovergang***
Initiële toestand = *Een databank net na aanmaak*
Kan initiële data bevatten
Op elke moment heeft een databank een ***huidige toestand***

DBMS garandeert **valide toestanden** en verhindert overgangen indien validiteit niet kan gegarandeert worden (vb. Data type, structuren, beperkingen, etc.).
Schema evolutie: vb. Toevoegen van geboortedatum aan STUDENT

3-Schema DBMS architectuur

Scheiden van gebruiker/toepassingen en fysieke databank adhv. **niveaus**:

- Extern: hoe gegevens tonen aan gebruikers
- Conceptueel: implementatiemodel
- Intern: fysische opslag en toegangspaden

DBMS doet de **vertaling/mapping**

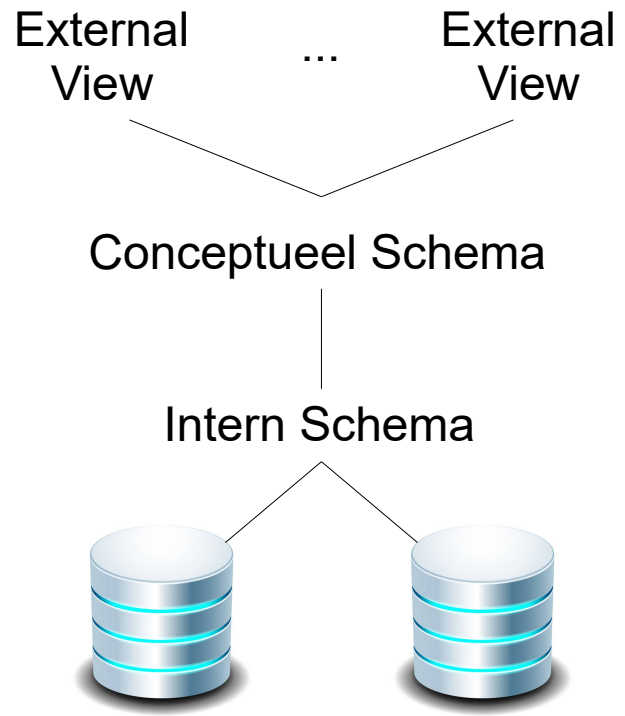
- Zelden perfecte scheiding

=> Logische data-onafhankelijkheid

vb. Constraints toevoegen

=> Fysiche data-onafhankelijkheid

vb. Nieuwe indexering



(ANSI/SPARC, 1975)

DBMS Talen en Interfaces

* 3-schema DBMS architectuur:
  - **SDL**: Storage Definition Language (intern)
  - **DDL**: Data Definition Language (conceptueel)
  - **VDL**: View Definition Language (extern)
* Als geen stricte scheiding DBMS architectuur
  - **DDL** voor conceptueel + intern
* Echter vaak: **SDL** via configuratie en **DDL** (DDL+VDL)
* DDL-compiler is deel van DBMS
  - Zet verklaringen in DDL om naar schema's en slaat ze op in catalogus

DBMS Talen en Interfaces

* Daarnaast gebruik van **DML**: Data Manipulation Language
  Vb. ophalen, invoegen, verwijderen, aanpassen van data
* 2 Types van DMLs:
  Hoog-niveau of non-proceduraal: complexe operaties (interactief of ingebed)
  Laag-niveau of proceduraal: 1 record per keer (sowieso ingebed, vb. lussen)
* Relational DBMS gebruikt **SQL** (= DDL+VDL+DML) = hoog-niveau
  - Bij interactief gebruik wordt het een **query-taal** genoemd.
  - Wanneer ingebed wordt het een **data-subtaal** genoemd.
  - De programmeertaal waar ingebed wordt de **hosttaal** genoemd.
* SQL is een declaratieve taal
  = **Zeg wat** er dient te gebeuren, **niet** de manier waarop (**hoe**).

DBMS Talen en Interfaces

* Verder zijn er **Interfaces** (al dan niet grafisch):
  - Menu-gebaseerd en Vraag-antwoord
  - Formulier-gebaseerd (canned transacties)
  - GUI (Grafische user interfaces) als Webapplicatie, OS-applicatie, etc.
  - Natuurlijke taal (gesproken/geschreven)
  - Parametrisch (leunt sterk aan bij grafische formulieren maar simplistischer)
  - Voor databank administratie (vb. Pgadmin)
  - LLMs (Large language models) zullen meer en meer gebruikt worden

Databank Systeem omgeving

Databank systemen enorm veel gebruikt!

Relationele databanken erg populair en perfect als Intro (RDBMS)

Volgt relationeel model
SQL als DDL+VDL+DML
Sterke theoretische basis

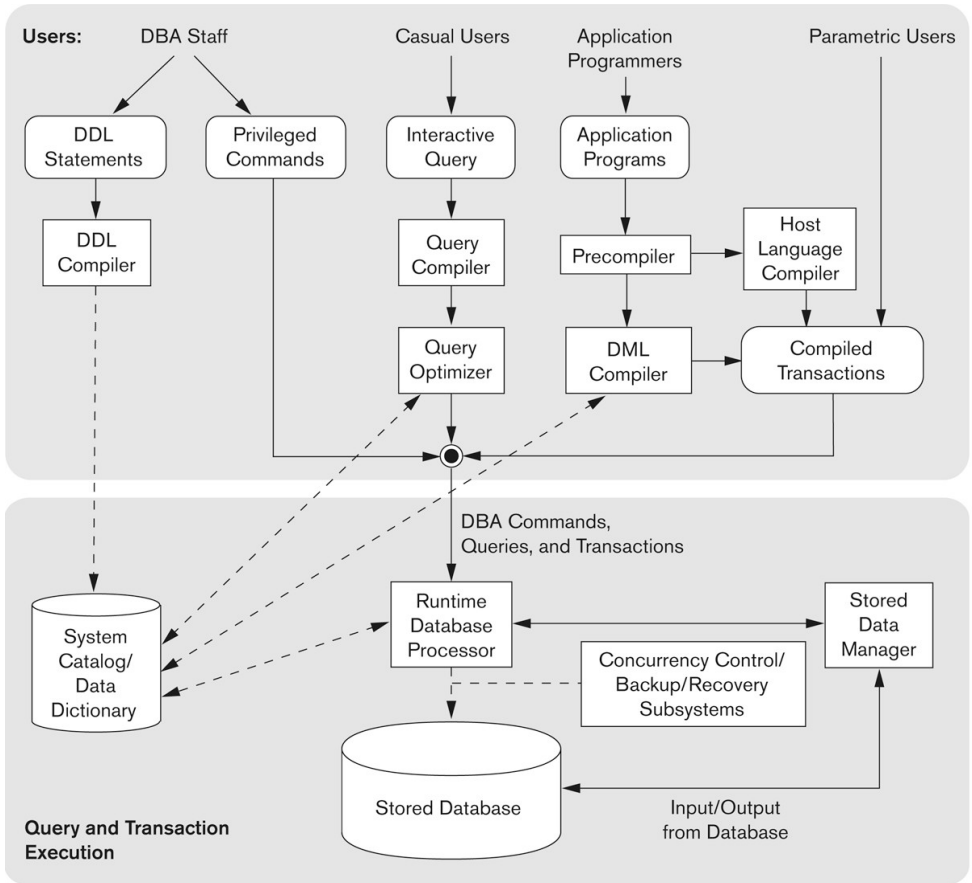


Figure 2.3
Component modules of a DBMS and their interactions.

1.3 Relationeel Model

RDBMS als leidraad met het relationele model als grondslag
Hoofdstuk 5: basis principes

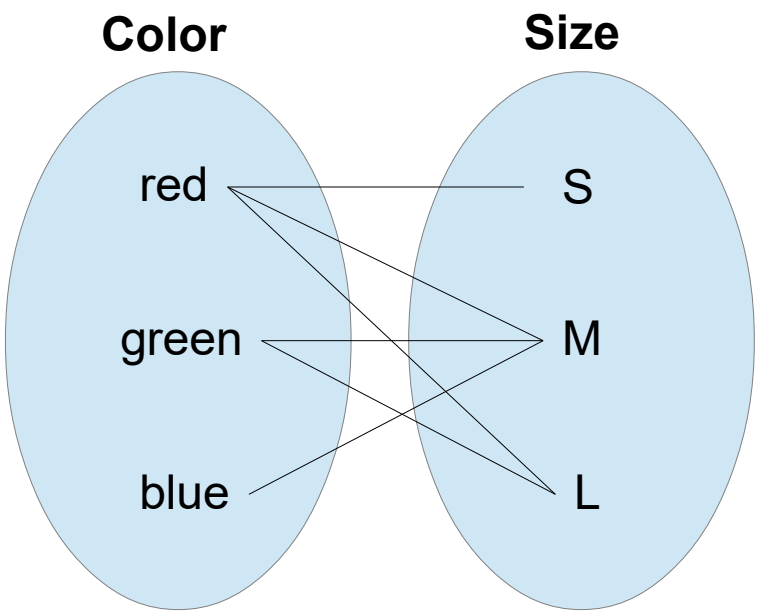
* Concepten
  - Domeinen, attributen, tupels, en relaties
  - Karakteristieken van relaties
  - Notaties
* Sleutels
* Integriteit
* Aanpassingen

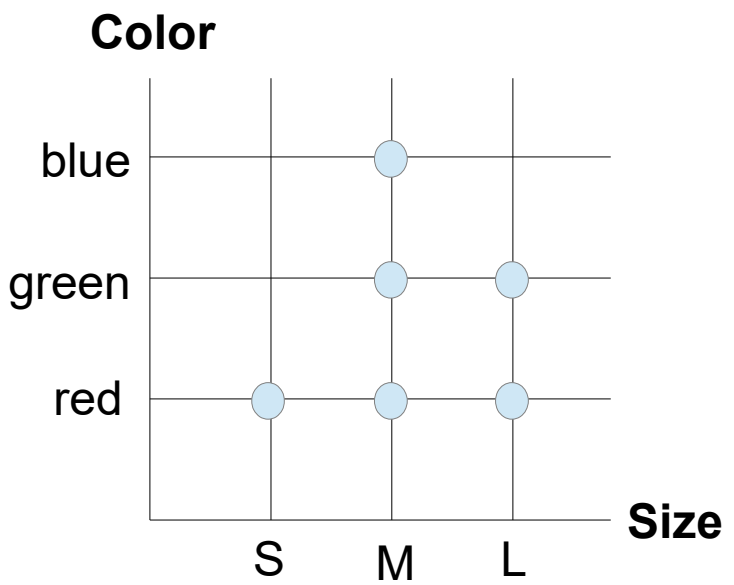
Definitie Relatie

Een **relatie** R op de verzamelingen $V_1,...,V_n$ is een deelverzameling van de productverzameling $V_1 \times ... \times V_n$: $\mathbf{R \subseteq V_1 \times ... \times V_n}$

TSHIRTS

Color	Size
red	S
red	M
red	L
green	M
green	L
blue	M

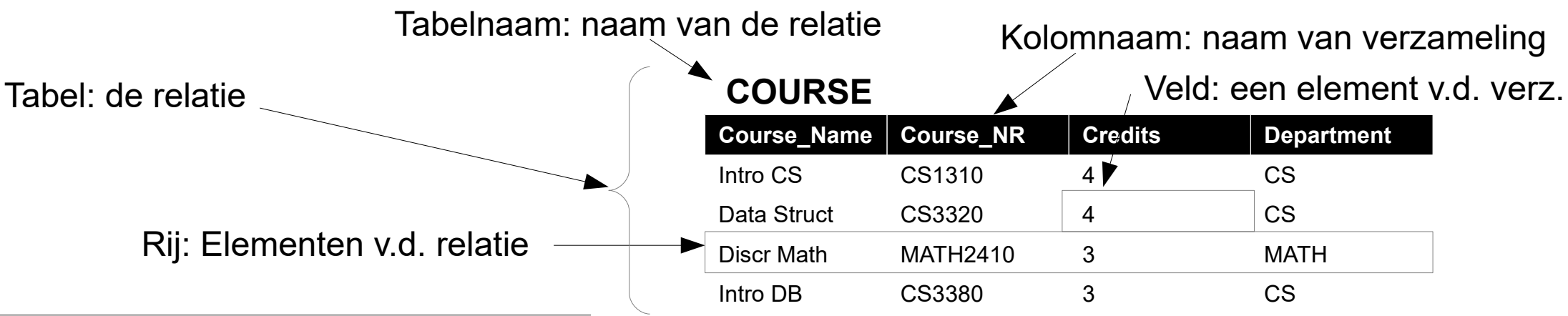




Definitie Tabel

Een relatie kan beschouwd worden als een tabel van waarden.

- Elke rij in de tabel (record) stelt gerelateerde data voor.
  Dit zijn de elementen van de relatie.
- De kolomnamen stellen de namen van de verzamelingen voor.
  Zodoende wordt de relatie eenduidig bepaald.
- De naam van de tabel komt overeen met de relatienaam.



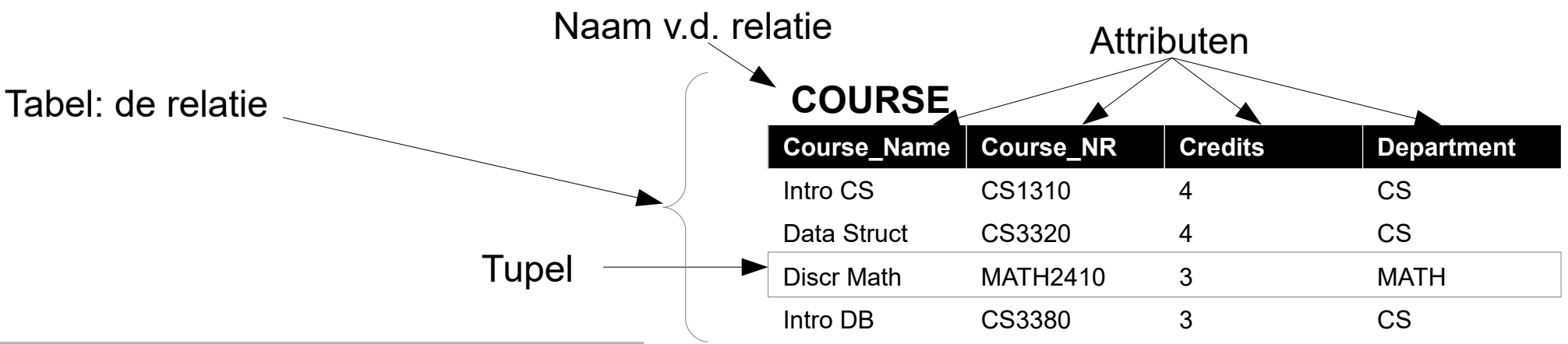
COURSE

Course_Name	Course_NR	Credits	Department
Intro CS	CS1310	4	CS
Data Struct	CS3320	4	CS
Discr Math	MATH2410	3	MATH
Intro DB	CS3380	3	CS

Definitie Tabel

In het relationele model gebruiken we volgende terminologie:

- Elke rij in de tabel is een **tupel**.
- De kolomnamen zijn **attributen**.
- De tabel is een **relatie**.
- Kolom data types: **domein** van mogelijke waarden voor attributen.



COURSE

Course_Name	Course_NR	Credits	Department
Intro CS	CS1310	4	CS
Data Struct	CS3320	4	CS
Discr Math	MATH2410	3	MATH
Intro DB	CS3380	3	CS

Domein, Attribuut, Tupel en Relatie

Domein D of **dom** is een verzameling van atomaire waarden.

- Atomair = niet opdeelbaar wat betreft het relationele model!
  - vb. Adressen (Celestijnenlaan 300C) zijn niet opdeelbaar in straatnamen
- Data types om een domein te definiëren: vb. pos. integer.
- Constraints op data types voor verfijning: vb. < 150.
- Geef zelf-beschrijvende namen: vb. Leeftijd = pos. integer < 150.
- Voorzie logische beschrijvingen van data types als uitgangspunt!

Een **relatieschema** Rel, geschreven als $R(A_1,A_2,...,A_n)$, bestaat uit een **relatienaam** R en een lijst van **attributen** $A_1,A_2,...,A_n$. Elk attribuut A_i stelt een **domein** van waarden voor dat deel uitmaakt van Rel. D is het domein van A_i, geschreven als $dom(A_i)$. De **ariteit** van Rel is het aantal attributen, n, in Rel.

Domein, Attribuut, Tupel en Relatie

Tupel over de **attributen** $X = \{A_1,A_2,\ldots,A_n\}$

$t = \{(A_1,w_1),(A_2,w_2),\ldots,(A_n,w_n)\}$ met

elke $w_i \in dom(A_i) \cup \{NULL\}$

Andere/Leesbare notatie: $<(A_1,A_2,\ldots,A_n),(w_1,w_2,\ldots,w_n)>$

Korte notatie als geen verwarring mogelijk: $<w_1,w_2,\ldots,w_n>$

Voorbeelden van hetzelfde tupel met drie verschillende notaties:
{(Course_Name,"Intro DB"),(Course_NR,"CS3380"),(Credits,3),(Department,"CS")}
<(Course_Name,Course_NR,Credits,Department),("Intro DB","CS3380",3,"CS")>
<"Intro DB","CS3380",3,"CS">

Merk op: Attributen zijn niet geordend! (wel in notatie/opslag: lijst)

Domein, Attribuut, Tupel en Relatie

Het i-de element van een tupel t, w_i, kan geschreven worden als t[i] en correspondeert zo tot het attribuut A_i.

Relatie r van het relatie schema $R(A_1,A_2,...,A_n)$, ook geschreven als r(Rel), is een verzameling tupels, $r = \{t_1,t_2,...,t_m\}$.

Merk op, ook tuples zijn niet geordend! (wel in notatie/opslag: lijst)
Merk op, een tuple kan maar één keer voorkomen in een relatie!

Karakteristieken van relaties

Relaties noch tupels kennen orde volgens hun formele definities. Echter praktisch is dit wel het geval: notatie en opslag. Vaak worden relaties en tupels daarom gezien als geordende lijsten ipv. verzameling.

Elke waarde in een tupel is atomisch. Samengestelde en meerwaardige attributen zijn niet toegelaten = "flat relational model" uitgaande van de "first normal form" assumptie.
Extensies hierop zijn mogelijk, zoals een genest relationeel model.

NULL waarden is een belangrijk concept in relationele modellen om waarden voor te stellen die of ongekend, of niet beschikbaar, of niet van toepassing zijn.

Notatie v. Relationele Modellen

Een relatie schema Rel van graad n wordt geschreven als $R(A_1,A_2,...,A_n)$.

* Q, R, S worden gebruikt voor relatiename.
* q, r, s worden gebruikt voor relatietoestanden (= verzameling tupels).
  - De naam van een relatie kan dit ook eenduidig bepalen.
* t, u, v worden gebruikt voor tupels.
* R.A wordt gebruikt voor attribuut A van relatieschema Rel aan te duiden.
* Een tupel t in een relatie r(Rel) wordt geschreven als $<w_1,w_2,...,w_n>$
  waarbij w_i de waarde is die overeenkomt met attribuut A_i.
* Componenten van een tupel kunnen bekomen worden via:
  - De notatie $t.A_i$ om de waarde w_i in t voor attribuut A_i te bekomen
  - De notatie $t[A_i,A_j,...,A_k]$ of $t.(A_i,A_j,...,A_k)$ met $A_i,A_j,...,A_k$ attributen van Rel om zo de subtupel $<w_i,w_j,...,w_k>$ te bekomen.

Sleutels

Relatie schema $R(A_1,A_2,...,A_n)$ met attributen $X=\{A_1,A_2,...,A_n\}$

* Super-sleutel $K \subseteq X$
  = Verzameling attributen die tupel van r(Rel) ondubbelzinnig bepalen
* Kandidaat-sleutel K = Super-sleutel K zonder overtollige attributen
  => Er bestaat geen super-sleutel $K' \subset K$ (of $K' \subseteq K$ met $K' \neq K$)

Soorten kandidaat sleutels:

* **Enkelvoudig**: Selchts 1 attribuut; en **Samengesteld**: Meerdere attributen
* **Primaire sleutel, PK,** (onderstreept, nooit NULL) en alternatieve sleutels

COURSE

Course_Name	Course_NR	Credits	Department
Intro CS	CS1310	4	CS
Data Struct	CS3320	4	CS
Discr Math	MATH2410	3	MATH
Intro DB	CS3380	3	CS

DEVICES

Vendor_ID	Device_ID	Device_Name
1	0001	Server
1	0002	Server
2	0001	Server
2	A001	Workstation

TELNRS

ID	Tel_NR
1	+32 444 11 11 11
2	+33 555 22 22 22
3	+32 666 33 33 33
4	+32 777 44 44 44

Sleutels

Een verzameling attributen FK (Foreign Key) van Rel_1, of $R_1(A_1,A_2,...,A_n)$,
is een **verwijssleutel** a.s.a.

* Attributen van FK hebben zelfde domein (of deel-domein)
  als primaire sleutel attributen PK (Primary Key) van Rel_2
* Elke waarde van FK in Rel_1 komt voor als waarde
  van een tupel in Rel_2 of is NULL

=> $\forall\ t_1 \in Rel_1$: t_1[FK] is NULL of $\exists\ t_2 \in Rel_2$: t_1[FK] = t_2[PK]

Verwijssleutel kan naar eigen relatie verwijzen = recursieve referentie
vb. Overste van werknemer in bedrijf is ook een werknemer

Vb. van verwijssleutel

STUDENT

Name	Student_NR	Class	Major
Smith	17	1	CS
Brown	8	2	CS

SECTION

Section_ID	Course_NR	Semester	Year	Instructor
85	MATH2410	Fall	23	King
92	CS1310	Fall	23	Anderson
102	CS3320	Spring	24	Knuth
112	MATH2410	Fall	24	Chang
119	CS1310	Fall	24	Anderson
135	CS3380	Fall	24	Stone

COURSE

Course_Name	Course_NR	Credits	Department
Intro CS	CS1310	4	CS
Data Struct	CS3320	4	CS
Discr Math	MATH2410	3	MATH
Intro DB	CS3380	3	CS

GRADES

Student_NR	Section_ID	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQ

Course_NR	Prereq_NR
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Verwijssleutels
om redundatie te vermijden.

Beperkingen

Domeinrestricties
- Beperkingen op de mogelijke waarden van een attribuut
- vb. Leeftijd van een mens als postieve integer < 150

Sleutelrestricties
- Primaire sleutels dienen uniek te zijn en mogen niet NULL zijn

Algemene integriteitsrestricties
= Juistheid en volledigheid van een databank
- Statische regels
  Slaan op databank toestand (elke mogelijke toestand)
- Dynamische regels
  Slaan op toestandsovergangen (elke mogelijke transitie)

Beperkingen: Statische Regels

Attribuutrestricties

Domein van attribuut is enkelvoudig = Eis van de 1e normaalvorm.

Entiteitrestricties

Tupel mag slechts 1 keer voorkomen (& Geen NULL waarden in PK).

Referentiële integriteit

Als tupel naar ander tupel verwijst moet ander tupel bestaan.

Sleutelrestricties

PK dient uniek te zijn en mag niet NULL zijn.

Domeinrestricties

vb. Werkuren >= 0 of Course_NR van de vorm XXXXNNNN
met X een letter in {'','A','B',...,'Z'} en N een nummer in [0,9]

Beperkingen: Dynamische Regels

Autorisatieregels
- Mag gebruiker actie uitvoeren?

Coördinatieregels
- Tupel mag slechts 1 keer voorkomen (& Geen NULL waarden in PK)

Pre -en postcondities
- vb. Max 600 EUR per dag afhalen.
- vb. Geld afhalen kan enkel als saldo groter dan 0 blijft.

Overgangsregels
- Welke volgorde van toestanden zijn toegestaan?
- vb. Enkel loonsverhogingen kunnen.
- vb. Loon werknemer kan niet hoger zijn dan dat van overste.

Beperkingen: Restricties

Intra-relationele restricties

Binnen eenzelfde relatie ("eenvoudig" te controleren).
vb. Maximaal één werknemer heeft geen overste.

Inter-relationele restricties

Verschillende relaties betrokken
vb. Loon werknemer kan niet hoger zijn dan dat van overste (lonen in andere tabel).

Tendens

Steeds meer restricties laten controleren door DBMS.
Neemt verantwoordelijkheid bij ontwikkelaar weg.
Makkelijker in beheer tenminste gegeven communicatie naar ontwikkelaar.

Gegevens aanpassen: Toevoegen

Bij elke aanpassing moet integriteit gecontrolleerd worden!

Tupel toevoegen:
Nieuw vak 'Introductie DB', met nr. 'CS3381', 6 studiepunten, en prereq. 'CS3320' en 'MATH2410'

```
INSERT INTO PREREQ
VALUES ('CS3381','CS3320');
```

NOK!
Referentiële integriteit geschonden!
COURSE met Course_Nr 'CS3381' bestaat niet.

COURSE

Course_Name	Course_NR	Credits	Department
Intro CS	CS1310	4	CS
Data Struct	CS3320	4	CS
Discr Math	MATH2410	3	MATH
Intro DB	CS3380	3	CS

PREREQ

Course_NR	Prereq_NR
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Gegevens aanpassen: Toevoegen

Bij elke aanpassing moet integriteit gecontrolleerd worden!

Tupel toevoegen:
Nieuw vak 'Introductie DB', met nr. 'CS3381', 6 studiepunten, en prereq. 'CS3320' en 'MATH2410'

```
INSERT INTO COURSE
VALUES ('Introductie DB','CS3380',6,NULL);
```

NOK!
Uniciteit van sleutel geschonden, CS3380!
Department mag niet NULL zijn!

```
INSERT INTO COURSE
VALUES ('Introductie DB','CS3381',6,'CS');
```

OK!

COURSE

Course_Name	Course_NR	Credits	Department
Intro CS	CS1310	4	CS
Data Struct	CS3320	4	CS
Discr Math	MATH2410	3	MATH
Intro DB	CS3380	3	CS

COURSE

Course_Name	Course_NR	Credits	Department
Intro CS	CS1310	4	CS
Data Struct	CS3320	4	CS
Discr Math	MATH2410	3	MATH
Intro DB	CS3380	3	CS
Introductie DB	CS3381	6	CS

Gegevens aanpassen: Toevoegen

Bij elke aanpassing moet integriteit gecontrolleerd worden!

Tupel toevoegen:
Nieuw vak 'introductie DB', met nr. 'CS3381', 6 studiepunten, en prereq. 'CS3320' en 'MATH2410'

```
INSERT INTO PREREQ
VALUES ('CS3381','CS3320');
```

OK!

```
INSERT INTO PREREQ
VALUES ('CS3381','MATH2410');
```

OK!

PREREQ

Course_NR	Prereq_NR
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

PREREQ

Course_NR	Prereq_NR
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310
CS3381	CS3320
CS3381	MATH2410

PREREQ

Course_NR	Prereq_NR
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310
CS3381	CS3320

Gegevens aanpassen: Verwijderen

Bij elke aanpassing moet integriteit gecontrolleerd worden!

Tupel verwijderen:
Verwijder vak 'introductie DB', met nr. 'CS3381'

```
DELETE FROM COURSE
WHERE Course_NR = 'CS3381';
```

NOK!
Referentiële integriteit geschonden!
Er zij PREREQ tupels die naar dit vak verwijzen.

Opties DBMS (Default of via SQL meegeven):
* Cascade: Tupels met verwijzingen ook verwijderen
* Verwijzende waarden op NULL zetten (maar PK! NOK)

COURSE

Course_Name	Course_NR	Credits	Department
Intro CS	CS1310	4	CS
Data Struct	CS3320	4	CS
Discr Math	MATH2410	3	MATH
Intro DB	CS3380	3	CS
Introductie DB	CS3381	6	CS

PREREQ

Course_NR	Prereq_NR
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310
CS3381	CS3320
CS3381	MATH2410

Gegevens aanpassen: Wijzigen

Bij elke aanpassing moet integriteit gecontrolleerd worden!

Tupel wijzigen:
Wijzig Course_Name 'Intro DB'
naar 'Introductie DB'

```
UPDATE COURSE
SET Course_Name='Introductie DB'
WHERE Course_NR = 'CS3380';
```

OK! Warning:
Referentiële integriteit mogelijks geschonden.
Er zij PREREQ tupels die naar dit vak verwijzen.
Gezien het geen PK betreft, mogelijks geen probleem.

COURSE

Course_Name	Course_NR	Credits	Department
Intro CS	CS1310	4	CS
Data Struct	CS3320	4	CS
Discr Math	MATH2410	3	MATH
Intro DB	CS3380	3	CS

PREREQ

Course_NR	Prereq_NR
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Gegevens aanpassen: Wijzigen

Bij elke aanpassing moet integriteit gecontrolleerd worden!

Tupel wijzigen:
Wijzig Course_NR 'CS3380' naar 'CS3381'

```
UPDATE COURSE
SET COURSE_NR='CS3381'
WHERE Course_NR = 'CS3380';
```

COURSE

Course_Name	Course_NR	Credits	Department
Intro CS	CS1310	4	CS
Data Struct	CS3320	4	CS
Discr Math	MATH2410	3	MATH
Introductie DB	CS3380	3	CS

PREREQ

Course_NR	Prereq_NR
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

NOK!
Referentiële integriteit geschonden!
Er zij PREREQ tupels die naar dit vak verwijzen!
Bovendien betreft dit een wijziging PK. Altijd gevaarlijk!
Werk via verwijderen en dan toevoegen ipv. wijzigen of cascade opties voor wijzigingen!