ISO
Het Relationele Database Model

Prof. dr. Paul De Bra

Gebaseerd op:
Database System Concepts, 5th Ed.

de praktijk: tabellen

- een database bestaat uit een aantal tabellen
  - elke tabel heeft een *naam* en een aantal *attributen*
  - elk attribuut heeft een *naam* en een *data type*
  - een tabel-*instantie* heeft een aantal rijen

  vb: een instantie van de *account* tabel:

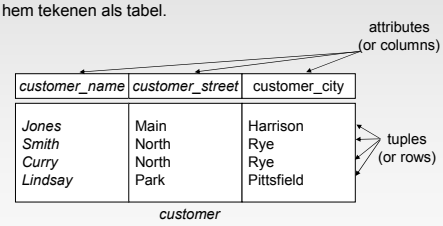
account_number	*branch_name*	*balance*
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

de onderliggende theorie: *relaties*

- Laat (attributen) D_1, D_2, …. D_n verzamelingen voorstellen; een **relatie** *r* is een deelverzameling van
  D_1 x D_2 x … x D_n
  een relation een verzameling *n*-tuples (a_1, a_2, …, a_n) waarbij elke $a_i \in D_i$
- Voorbeeld:
  customer_name = {Jones, Smith, Curry, Lindsay}
  customer_street = {Main, North, Park}
  customer_city = {Harrison, Rye, Pittsfield}
  r = { (Jones, Main, Harrison), (Smith, North, Rye), (Curry, North, Rye), (Lindsay, Park, Pittsfield) }
  is een relatie over
  customer_name x *customer_street* x *customer_city*

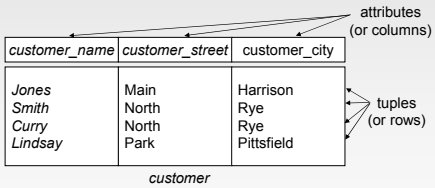
alternatieve voorstellingen

- Het maakt niet uit of we een tabel *customer* over de attributen *customer_name, customer_street* en *customer_city* schrijven als
  r = { (Jones, Main, Harrison), (Smith, North, Rye), (Curry, North, Rye), (Lindsay, Park, Pittsfield) } of dat we hem tekenen als tabel.



relatie schema en instantie

- schema = structuur: naam van de relatie, attribuunamen en waardenverzameling
- instantie = verzameling tupels of rijen in een tabel



rekenen met tabellen: relationele algebra

- procedurele taal: we bepalen *hoe* een vraag moet worden beantwoord
- 6 basisoperaties
  - selectie: σ
  - projectie: Π
  - vereniging: $\cup$
  - verschil: –
  - Cartesisch product: x
  - hernoeming: ρ
- de operaties maken van 1 of 2 tabellen weer een nieuwe tabel

de selectie van rijen

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

betekenis van de selectie

- notatie: $\sigma_p(r)$
- *p* is het **selectie predicaat**, *r* is een relatie (schema)
- de selectie betekent in de verzamelingenleer dat voor elke *instance* ***r*** :

$$\sigma_p(\boldsymbol{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

p is een logische formule met **termen** verbonden door: $\wedge$ (**en**), $\vee$ (**of**), $\neg$ (**niet**) en elke **term** is :

<attribute> *operatie* <attribute> of <constante>

waarbij *operatie* =, $\neq$, >, $\geq$. < of $\leq$ is

- vb: $\sigma_{branch\_name="Perryridge"}(account)$

de projectie (of selectie van kolommen)

- tabel *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

$\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

betekenis van de projectie

- notatie: $\Pi_{A_1,A_2,\ldots,A_k}(r)$

waarbij $A_1, \ldots, A_k$ attributen zijn en *r* een relatie

- voor elke instantie *r* geldt dat $\Pi_{A_1,A_2,\ldots,A_k}(r)$

$= \{ t \mid \exists\, s \in r, \forall\, i \in \{1,\ldots,k\}\ t(Ai) = s(Ai) \}$

- Vb: als *account* een relatie is met de attributen *account_number, branch_name, balance* dan verwijderen we het *branch_name attribute met* :

$$\Pi_{account\_number,\ balance}\ (account)$$

volgorde van bewerkingen is belangrijk

- wat is er mis met de volgende query om het rekeningnummer en saldo te verkrijgen van de rekeningen geopend in filiaal *Perryridge*?

$\sigma_{branch\_name="Perryridge"}(\Pi_{account\_number,\ balance}\ (account))$

- de collecte uitwerking is:

$\Pi_{account\_number,\ balance}(\sigma_{branch\_name="Perryridge"}(account))$

- selectie en projectie mogen *soms* wel verwisseld worden:

$\sigma_{balance > 10000}(\Pi_{account\_number,\ balance}\ (account))$ of

$\Pi_{account\_number,\ balance}(\sigma_{balance > 10000}(account))$

de hernoeming operatie

- wordt gebruikt om attribuutnamen te veranderen (terwijl de inhoud van de tabel ongewijzigd blijft)
- laat ook toe om naar een hele tabel te verwijzen onder een andere naam.
- voorbeeld: $\rho_X(E)$ noemt het resultaat van E om tot X; $\rho_{X(A_1, \ldots, A_n)}(E)$ noemt het resultaat van een uitdrukking E met *n* attributen om tot X, met attributen $A_1, \ldots, A_n$.

de vereniging (unie)

- tabellen *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- r ∪ s:

A	B
α	1
α	2
β	1
β	3

betekenis van de vereniging

- notatie: $r \cup s$
- betekenis:
  $r \cup s = \{t \mid t \in r \text{ of } t \in s\}$
- voorwaarden om $r \cup s$ te mogen gebruiken:
  1. *r, s* moeten evenveel attributen hebben
  2. de overeenkomstige attributen moeten dezelfde waardenverzamelingen hebben.
  3. bij verschillende attribuutnamen: hernoeming
- vb: geef alle klanten met een rekening of lening:
  $\Pi_{customer\_name}(depositor) \cup \Pi_{customer\_name}(borrower)$

de verschil operatie

- tabellen *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- *r – s:*

A	B
α	1
β	1

betekenis van het verschil

- notatie: $r - s$
- betekenis:
  $r - s = \{t \mid t \in r \text{ en } t \notin s\}$
- voorwaarden om $r - s$ te mogen gebruiken:
  1. *r, s* moeten evenveel attributen hebben
  2. de overeenkomstige attributen moeten dezelfde waardenverzamelingen hebben.
  3. bij verschillende attribuutnamen: hernoeming
- vb: geef alle klanten die een rekening hebben maar geen lening:
  $\Pi_{customer\_name}(depositor) - \Pi_{customer\_name}(borrower)$

extraatje: de doorsnede

- tabellen *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$:

A	B
α	2

betekenis van de doorsnede

- notatie: $r \cap s$
- betekenis:
  $r \cap s = r - (r - s)$
- voorwaarden om $r \cap s$ te mogen gebruiken zijn uiteraard dezelfde als voor het verschil
- vb: geef alle klanten die een rekening en een lening hebben:
  $\Pi_{customer\_name}(depositor) \cap \Pi_{customer\_name}(borrower)$
- de doorsnede is niet nodig (wordt uitgedrukt door tweemaal een verschil) maar wel handig

het cartesisch (cartesiaans) product

- tabellen *r, s*:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

betekenis van het cartesisch product

- notatie: *r* x *s*
- betekenis:

  $r \times s = \{t\ q \mid t \in r \textbf{ en } q \in s\}$

- als *r* en *s* gelijknamige attributen hebben dan moeten we ze hernoemen
- we gebruiken vaak een *standaard* hernoeming met behulp van de tabelnaam: als *r* attributen *A, B* heeft en *s* attributen *A, C* dan gebruiken we vaak *r* x *s* als een tabel met attributen *r.A*, *r.B*, *s.A*, *s.C*.

het bank voorbeeld

branch (branch_name, branch_city, assets)

customer (customer_name, customer_street, customer_city)

account (account_number, branch_name, balance)

loan (loan_number, branch_name, amount)

depositor (customer_name, account_number)

borrower (customer_name, loan_number)

voorbeeld-vragen

- geef alle leningen met een bedrag van meer dan 1200

  $\sigma_{amount > 1200}(loan)$

- geef het leningnummer van alle leningen met een bedrag van meer dan 1200

  $\Pi_{loan\_number}(\sigma_{amount > 1200}(loan))$

voorbeelden

- geef de namen van de klanten die een rekening of lening hebben bij de bank (of allebei, want “of” heeft die betekenis)

  $\Pi_{customer\_name}(borrower) \cup \Pi_{customer\_name}(depositor)$

- geef de namen van de klanten die een rekening en een lening hebben bij de bank

  $\Pi_{customer\_name}(borrower) \cap \Pi_{customer\_name}(depositor)$

moeilijkere voorbeelden

- geef de namen van alle klanten die een lening hebben bij het filiaal “Perryridge”

  $\Pi_{customer\_name}(\sigma_{branch\_name=\text{“Perryridge”}}(\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan)))$

- geef de namen van alle klanten die een lening hebben bij het filiaal “Perryridge” maar die geen rekening hebben bij (eender welk filiaal van) de

  $\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Perryridge”}}(\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan))) - \Pi_{customer\_name}(depositor)$

verschillende correcte uitwerkingen

- geef de namen van alle klanten die een lening hebben bij het filiaal “Perryridge”
  - Query 1

    $\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Perryridge”}}($
    $\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan)))$
  - Query 2

    $\Pi_{customer\_name}(\sigma_{loan.loan\_number = borrower.loan\_number}($
    $(\sigma_{branch\_name = \text{“Perryridge”}}(loan)) \times borrower))$

moeilijke “truuk” vragen

- geef het hoogste saldo dat voorkomt in de bank.
  - strategie:
    - zoek eerst de saldi die niet het hoogste zijn (er is een hoger)
      - hernoem *account* zodat we paren van saldi met elkaar kunnen vergelijken
    - gebruik de verschil operatie om het saldo te vinden dat niet voorkomt in de vorige verzameling saldi.
  - de query wordt dan:

    $\Pi_{balance}(account) - \Pi_{account.balance}$
    $(\sigma_{account.balance < d.balance}(account \times \rho_{d}(account)))$

sleutels

- tupels (rijen) in een tabel zijn altijd uniek (een tabel instance is een verzameling).
- een database kan zo ontworpen worden dat ook sommige deelrijen (projectie op attributen) altijd uniek zijn:
  - supersleutel: deelrijen zijn uniek
  - kandidaat sleutel: deelrijen zijn uniek en nog kleinere deelrijen zijn dat niet
  - primaire sleutel: een door de ontwerper uitgekozen kandidaat sleutel

opgaven

- opgave 2.1 met meer vragen dan in het boek: de database bestaat uit volgende tabellen:

  employee (*person_name*, *street*, *city*)

  works (*person_name*, *company_name*, *salary*)

  company (*company_name*, *city*)

  manages (*person_name*, *manager_name*)

  de sleutels hebben hierbij belang!

opgaven

- Stel volgende vragen in de relationele algebra:
  1. geef de namen van alle bedienden die wonen in Eindhoven
  2. geef de namen van alle bedienden die niet in Eindhoven wonen
  3. geef de namen van alle bedienden die zichzelf als manager hebben
  4. geef de naam van de managers met een salaris van meer dan 100.000
  5. geef de naam van de bedienden met een manager met een salaris van meer dan 100.000

opgaven

- Stel volgende vragen in de relationele algebra:
  6. geef de namen van de bedienden die meer verdienen dan hun manager
  7. geef de naam van de bedrijven die gevestigd zijn in een stad waar nog een ander bedrijf gevestigd is
  8. geef de namen van alle bedienden die wonen in de stad waar ze werken
  9. geef de namen van alle bedienden die wonen in een andere stad dan hun manager
  10. geef de naam van de bedrijven die gevestigd zijn in een stad waar geen enkele bediende van dat bedrijf woont

opgaven

- Stel volgende vragen in de relationele algebra:

11. geef de naam van bedienden wiens manager voor een ander bedrijf werkt dan zij zelf
12. geef de namen van de bedrijven die werknemers hebben die in Eindhoven wonen
13. geef de namen van de bedrijven die geen werknemers hebben die in Eindhoven wonen
14. geef de naam van de bediende met het hoogste salaris
15. geef de naam van de manager met het hoogste salaris

opgaven

- Wat betekenen de volgende vragen?

1. $\Pi_{city}(company) - \Pi_{city}(employee)$
2. $\Pi_{company.city}(company) - \Pi_{company.city}($
   $\sigma_{employee.person\_name = works.person\_name \wedge works.company\_name = company.company\_name \wedge employee.city = company.city}(employee \times works \times company)$)
3. $\Pi_{m.person\_name}(\sigma_{works.salary < m.salary \wedge works.person\_name = manages.person\_name \wedge m.person\_name = manages.manager\_name}(works \times manages \times \rho_m(works)))$

extraatje: de *join* operatie

- Vaak voorkomende operatie: neem relaties r(a,b) en s(a,c) en stel de volgende vraag: Geef de b's van de r-en waarvoor er een s is met c = "blah". Dit wordt:
  $\Pi_b(\sigma_{r.a = s.a \wedge s.c = "blah"}(r \times s))$
- Er is een handige "afkorting": de *join* of $\triangleright\triangleleft$:
  $r \triangleright\triangleleft s = \Pi_{r.a, r.b, s.c}(\sigma_{r.a = s.a}(r \times s))$
- Dus de "blah" query wordt:
  $\Pi_b(\sigma_{s.c = "blah"}(r \triangleright\triangleleft s))$