Model Relasi

Eri Prasetyo

http://Pusatstudi.gunadarma.ac.id/pscitra

http://eri.staffsite.gunadarma.ac.id

Sources :

-Database Systems: Design, Implementation & management, 5th, edition, Rob & Coronel

- Database System Concept, Silberschatz, Korth and Sudarshan

Mode Relasi

- Data dan hubungan antar data direpresentasikan dengan kumpulan tabel yang mempunyai kolom dengan nama yang unik.
- Kolom merepresentasikan nama field/Atribut
- Baris merepresentasikan rekord / tuple

Relasi tabel

Batch no	No id product	Kode mode test
95 0001	1	01 001
95 0001	1	02 001
95 0002	3	03 001
95 0003	4	04 001
95 0003	4	05 001

Kode mode test	Nama mode test
01 001	P/S nilai tegangan
02 001	Motherboard test
03 001	Test RAM
04 001	Cek Dos boot
05 001	Cek windows

Basic Structure

- himpunan D_1, D_2, …. D_n a **relasi** *r* sebuah subset dari
  D_1 x D_2 x … x D_n
  sebuah relasi adalah sebuah himpunan dari n-tuples (a_1, a_2, …, a_n)
  dimana
  setiap $a_i \in D_i$
- contoh: jika

  customer-name = {Jones, Smith, Curry, Lindsay}
  customer-street = {Main, North, Park}
  customer-city = {Harrison, Rye, Pittsfield}

  maka *r* = { (Jones, Main, Harrison),
  (Smith, North, Rye),
  (Curry, North, Rye),
  (Lindsay, Park, Pittsfield)}
  adalah sebuah relasi pada *customer-name x customer-street x customer-city*

Skema Relasi

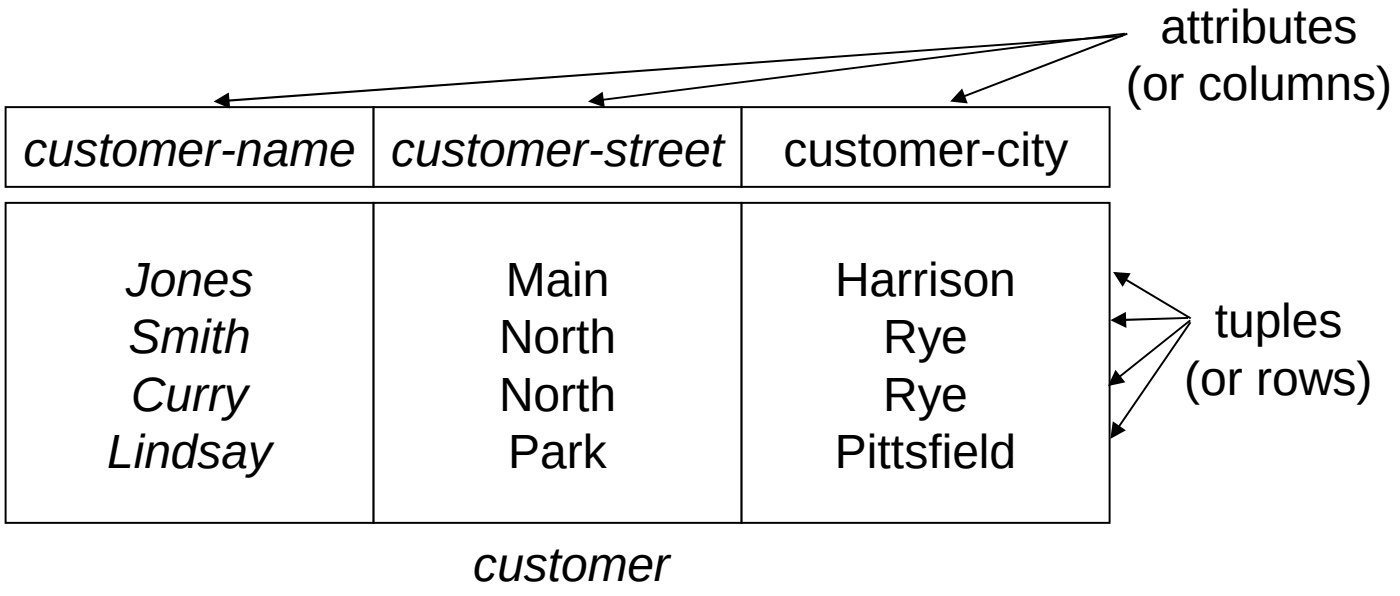
- A_1, A_2, …, A_n are *attributes*
- $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

  E.g. *Customer-schema* = (*customer-name, customer-street, customer-city*)
- *r*(*R*) is a *relation* on the *relation schema R*

  E.g. *customer (Customer-schema)*

Relation Instance

- The current values (*relation instance*) of a relation are specified by a table
- An element *t* of *r* is a *tuple*, represented by a *row* in a table



Bahasa Query

- Bahasa yang dapat dipakai pengguna untuk meminta atau melacak informasi dari suatu basis data
- Kategori bahasa Query
  - Procedural

  Pengguna harus memberikan diskripsi urutan operasi yang dilakukan pada basis data untuk mendapatkan informasi yang diinginkan dari basis data
  - non-procedural

  Pengguna hanya bertanggung jawab mendiskripsikan karakteristik informasi yang diinginkan tanpa memberikan urutan operasi

Aljabar Relasional

- Bahasa Procedural
- 6 Operasi
  - select
  - project
  - union
  - set difference
  - Cartesian product
  - rename
- The operators take two or more relations as inputs and give a new relation as a result.

Operasi Select

- Operasi ini digunakan untuk memilih atribut yang memenuhi suatu predikat tertentu.

- Operasi ini menggunakan notasi σ(sigma)

Select Operation

- Notation: $\sigma_p(r)$
- *p* is called the selection predicate
- Defined as:

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

Where *p* is a formula in propositional calculus consisting of terms connected by : $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
Each term is one of:

<attribute> *op* <attribute> or <constant>

where *op* is one of: =, $\neq$, >, $\geq$. <. $\leq$

- Example of selection:

$\sigma_{\textit{branch-name=“Perryridge”}}(\textit{account})$

Select Operation – Example

- Relation *r*

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

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

A	*B*	*C*	*D*
α	α	*1*	*7*
β	β	*23*	*10*

Operasi Project

- Digunakan untuk memilih kolom atau atribut tertentu dari tabel
- Operasi ini dinyatakan dengan notasi Π(phi)

Operasional Project

- Notation:

$$\Pi_{A1, A2, \ldots, Ak}\ (r)$$

where A_1, A_2 are attribute names and r is a relation name.

- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- E.g. To eliminate the *branch-name* attribute of *account*

$$\Pi_{account\text{-}number,\ balance}\ (account)$$

Project Operation – Example

- Relation *r*:

A	*B*	*C*
α	10	1
α	20	1
β	30	1
β	40	2

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

A	*C*
α	1
α	1
β	1
β	2

=

A	*C*
α	1
β	1
β	2

Operasi cartesian product

- Digunakan untuk melakukan kombinasi rekord dari satu tabel dengan tabel lain.
- Operasi ini memungkinkan asosiasi / relasi satu tabel dengan tabel lainnya yang saling berkaitan
- Dinyatakan dengan notasi X (kali)

Cartesian-Product Operation

- Notation r x s
- Defined as:

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

- Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \varnothing$).
- If attributes of *r(R)* and *s(S)* are not disjoint, then renaming must be used.

Cartesian-Product Operation-Example

Relations *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

Operasi union

- Digunakan untuk menggabungkan isi tiap atribut yang sama dari tabel yang berbeda
- Operasi ini dinyatakan dengan notasi U (union)

Union Operation

- Notation: $r \cup s$
- Defined as:

  $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$

- For $r \cup s$ to be valid.

  1. *r, s* must have the *same arity* (same number of attributes)

  2. The attribute domains must be *compatible* (e.g., 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)

- E.g. to find all customers with either an account or a loan

  $\Pi_{customer\text{-}name}$ (*depositor*) $\cup$ $\Pi_{customer\text{-}name}$ (*borrower)*

Union Operation – Example

- Relations *r, s:*

A	*B*
α	1
α	2
β	1

r

A	*B*
α	2
β	3

s

r $\cup$ s:

A	*B*
α	1
α	2
β	1
β	3

Operasi Set Difference

- Dinyatakan dengan notasi – (minus)
- Digunakan untuk mendapatkan tuple yang berada pada satu relasi, tetapi tidak berada pada relasi yang lain

Set Difference Operation

- Notation $r - s$
- Defined as:

$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

- Set differences must be taken between *compatible* relations.
  - r and s must have the *same arity*
  - attribute domains of r and s must be compatible

Set Difference Operation – Example

- Relations *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

Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$
- $r \times s$

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

$\forall$ $\sigma_{A=C}(r \times s)$

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

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.

Example:

$$\rho_x(E)$$

returns the expression *E* under the name *X*

If a relational-algebra expression *E* has arity *n*, then

$$\rho_{x(A1, A2, \ldots, An)}(E)$$

returns the result of expression *E* under the name *X*, and with the

attributes renamed to *A1, A2, …., An*.

Banking Example

branch (branch-name, branch-city, assets)

customer (customer-name, customer-street, customer-only)

account (account-number, branch-name, balance)

loan (loan-number, branch-name, amount)

depositor (customer-name, account-number)

borrower (customer-name, loan-number)

Banking Example

branch (branch-name, branch-city, assets)

customer (customer-name, customer-street, customer-only)

account (account-number, branch-name, balance)

loan (loan-number, branch-name, amount)

depositor (customer-name, account-number)

borrower (customer-name, loan-number)

Example Queries

- Find all loans of over $1200

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

- Find the loan number for each loan of an amount greater than $1200

$$\Pi_{loan\text{-}number}(\sigma_{amount > 1200}(loan))$$

Example Queries

- Find the names of all customers who have a loan at the Perryridge branch.

$$\Pi_{customer\text{-}name}\ (\sigma_{branch\text{-}name=\text{“}Perryridge\text{”}}$$

$$(\sigma_{borrower.loan\text{-}number\ =\ loan.loan\text{-}number}(borrower\ x\ loan)))$$

- Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank.

$$\Pi_{customer\text{-}name}\ (\sigma_{branch\text{-}name\ =\ \text{“}Perryridge\text{”}}$$

$$(\sigma_{borrower.loan\text{-}number\ =\ loan.loan\text{-}number}(\text{borrower x loan}))\ -$$

$$\Pi_{customer\text{-}name}(\text{depositor})$$

Example Queries

- Find the names of all customers who have a loan at the Perryridge branch.
  - Query 1

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

  - Query 2

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

Example Queries

Find the largest account balance

- Rename *account* relation as *d*
- The query is:

$$\Pi_{balance}(account) - \Pi_{account.balance}$$

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

Formal Definition

- A basic expression in the relational algebra consists of either one of the following:
  - A relation in the database
  - A constant relation
- Let E_1 and E_2 be relational-algebra expressions; the following are all relational-algebra expressions:
  - $E_1 \cup E_2$
  - E_1 - E_2
  - E_1 x E_2

    $\sigma_p (E_1)$, P is a predicate on attributes in E_1

    $\prod_s (E_1)$, S is a list consisting of some of the attributes in E_1

    $\rho_x (E_1)$, x is the new name for the result of E_1

Additional Operations

We define additional operations that do not add any power to the

relational algebra, but that simplify common queries.

- Set intersection
- Natural join
- Division
- Assignment

Set-Intersection Operation

- Notation: $r \cap s$
- Defined as:
- $r \cap s = \{ t \mid t \in r$ **and** $t \in s \}$
- Assume:
  - r, s have the *same arity*
  - attributes of r and s are compatible
- Note: $r \cap s = r - (r - s)$

Set-Intersection Operation - Example

- Relation r, s:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- r ∩ s

A	B
α	2

Natural-Join Operation

- Notation: r $\bowtie$ s

- Let *r* and *s* be relations on schemas *R* and *S* respectively. Then, r $\bowtie$ s is a relation on schema $R \cup S$ obtained as follows:
  - Consider each pair of tuples t_r from *r* and t_s from *s*.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple *t* to the result, where
    - *t* has the same value as t_r on *r*
    - *t* has the same value as t_s on *s*

- Example:

  $R = (A, B, C, D)$

  $S = (E, B, D)$

  - Result schema = (A, B, C, D, E)
  - $r \bowtie s$ is defined as:

$$\Pi_{r.A,\ r.B,\ r.C,\ r.D,\ s.E}\ (\sigma_{r.B = s.B \wedge r.D = s.D}\ (r \times s))$$

Natural Join Operation – Example

- Relations r, s:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	$\in$

s

$r \bowtie s$

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

Division Operation

$$r \div s$$

- Suited to queries that include the phrase “for all”.
- Let *r* and *s* be relations on schemas R and S respectively where
  - $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - $S = (B_1, \ldots, B_n)$

  The result of r ÷ s is a relation on schema

  $R - S = (A_1, \ldots, A_m)$

$$r \div s = \{ t \mid t \in \prod_{R-S}(r) \wedge \forall u \in s (tu \in r) \}$$

Division Operation – Example

Relations *r*, *s*:

A	*B*
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
ϵ	6
ϵ	1
β	2

r

B
1
2

s

r ÷ *s*:

A
α
β

Another Division Example

Relations *r*, *s*:

A	*B*	*C*	*D*	*E*
α	a	α	a	*1*
α	a	γ	a	*1*
α	a	γ	b	*1*
β	a	γ	a	*1*
β	a	γ	b	*3*
γ	a	γ	a	*1*
γ	a	γ	b	*1*
γ	a	β	b	*1*

r

D	*E*
a	*1*
b	*1*

s

r ÷ *s*:

A	*B*	*C*
α	a	γ
γ	a	γ

Division Operation (Cont.)

- Property
  - Let $q - r \div s$
  - Then q is the largest relation satisfying $q \times s \subseteq r$
- Definition in terms of the basic algebra operation
  Let $r(R)$ and $s(S)$ be relations, and let $S \subseteq R$

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$$

To see why

$\Pi_{R-S,S}(r)$ simply reorders attributes of r

$\Pi_{R-S}(\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$ gives those tuples t in

$\Pi_{R-S}(r)$ such that for some tuple $u \in s$, $tu \notin r$.

Assignment Operation

- The assignment operation ($\leftarrow$) provides a convenient way to express complex queries.
  - Write query as a sequential program consisting of
    - a series of assignments
    - followed by an expression whose value is displayed as a result of the query.
  - Assignment must always be made to a temporary relation variable.
- Example: Write $r \div s$ as

$$temp1 \leftarrow \Pi_{R-S}(r)$$

$$temp2 \leftarrow \Pi_{R-S}((temp1 \times s) - \Pi_{R-S,S}(r))$$

$$result = temp1 - temp2$$

  - The result to the right of the $\leftarrow$ is assigned to the relation variable on the left of the $\leftarrow$.
  - May use variable in subsequent expressions.

Example Queries

- Find all customers who have an account from at least the “Downtown” and the Uptown” branches.

Query 1

$$\Pi_{CN}(\sigma_{BN=\text{“Downtown”}}(depositor \bowtie account)) \cap \Pi_{CN}(\sigma_{BN=\text{“Uptown”}}(depositor \bowtie account))$$

where *CN* denotes customer-name and *BN* denotes *branch-name*.

Query 2

$$\Pi_{customer\text{-}name,\ branch\text{-}name}(depositor \bowtie account) \div \rho_{temp(branch\text{-}name)}(\{(\text{“Downtown”}), (\text{“Uptown”})\})$$

Example Queries

- Find all customers who have an account at all branches located in Brooklyn city.

$$\Pi_{\textit{customer-name, branch-name}} (\textit{depositor} \bowtie \textit{account})$$

$$\div \Pi_{\textit{branch-name}} (\sigma_{\textit{branch-city}} = \text{“Brooklyn”} (\textit{branch}))$$