



# Pengaksesan Basis Data

*Aljabar Relasional*  
Fak. Teknik Prodi Teknik Informatika  
Universitas Pasundan

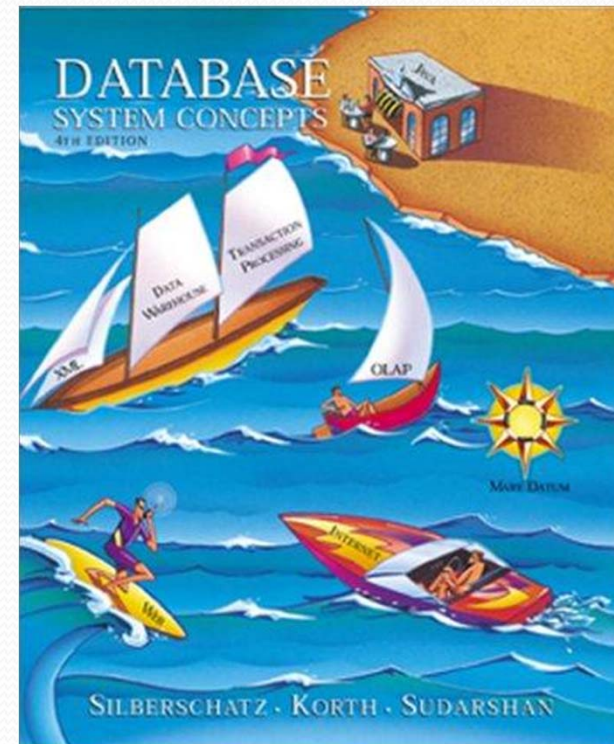
Caca E. Supriana, S.Si., MT.

[caca.e.supriana@unpas.ac.id](mailto:caca.e.supriana@unpas.ac.id)  
[caca-e-supriana.blogspot.com](http://caca-e-supriana.blogspot.com)

# Relational Algebra

**The relational algebra is a procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation as their result.**

Silberchatz/Korth/Sudarshan, 2002,  
*Database Systems Concepts*,  
4<sup>th</sup> Edition, McGrawHill



# Bahasa *Query*

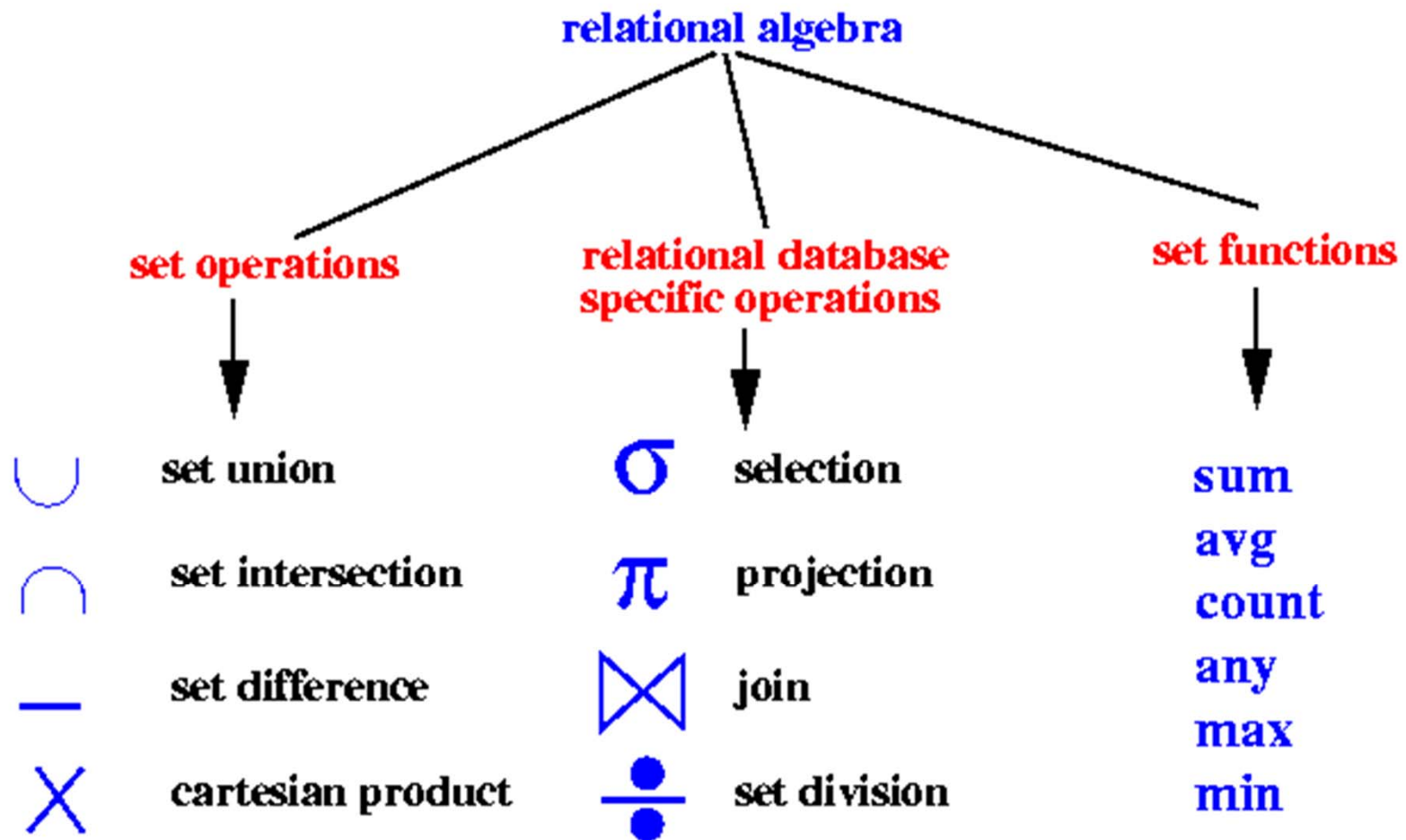
- Bahasa yang digunakan *user* untuk melakukan *request* informasi dari basis data
- Kategori Bahasa :
  - Prosedural
  - Non-prosedural
- Bahasa “Murni” (“*Pure*” languages) :
  - Aljabar Relasional
  - *Tuple Relational Calculus*
  - *Domain Relational Calculus*
- *Pure languages* membentuk dasar bahasa *query* yang digunakan *user*.



# Aljabar Relasional

- Merupakan bahasa prosedural
- Operator dasar :
  - *select*
  - *project*
  - *union*
  - *set difference*
  - *cartesian product*
  - *rename*
- Operator yang melibatkan dua atau lebih relasi sebagai *input* akan menghasilkan satu relasi baru sebagai *output*.

# Operator & Notasi



# Selection

Pemakaian :

$\sigma_{\langle \text{selection condition} \rangle} ( \langle \text{relation name} \rangle )$

Definisi *selection* :

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

Di mana  $p$  adalah sebuah formula dalam kalkulus proposisi yang terdiri atas term yang dihubungkan dengan and, or & not



# Selection (contoh 1)


$\sigma$  = select

Sample query:  $\sigma_{\text{Type} = \text{"savings"}} \text{Account}$

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

	Number	Owner	Balance	Type
	103	J. Smith	5000.00	savings



# Selection (contoh 2)

$\sigma$  Balance < 4000 Account

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	104	M. Jones	1000.00	checking



# Projection

Pemakaian :

$\Pi_{\langle \text{attribute list} \rangle} (\langle \text{relation name} \rangle)$

- Hasil operasi project didefinisikan sebagai relasi dengan kolom sebanyak k yang diperoleh dengan menghapus kolom yang tidak termasuk dalam kriteria.
- *Duplicate rows* dihilangkan dari relasi hasil, karena relasi adalah sebuah himpunan.

# Projection (contoh 1)

$\pi$  = project

Sample query:  $\pi_{\text{Number, Owner, Type}}$  **Account**

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

	Number	Owner	Type
	101	J. Smith	checking
	102	W. Wei	checking
	103	J. Smith	savings
	104	M. Jones	checking
	105	H. Martin	checking

# Projection (contoh 2)

$\pi_{\text{Owner}}$  Account

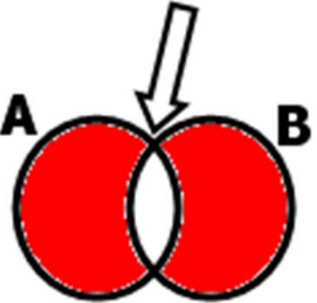
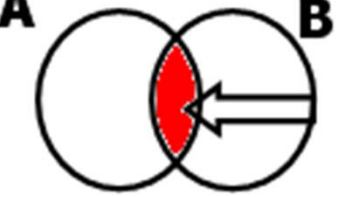
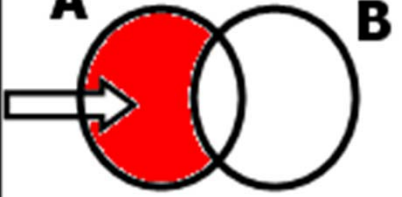
Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Owner
J. Smith
W. Wei
M. Jones
H. Martin

*Duplicate rows  
dihilangkan*



# Operasi Himpunan

	Union $A \cup B$
	Intersection $A \cap B$
	Difference $A - B$

# Operasi Himpunan

- ***Union*** : menghasilkan relasi dimana termasuk semua tuple yang hanya muncul di  $R_1$  atau hanya di  $R_2$  atau di  $R_1 \& R_2$ , duplikasi tuple dihilangkan
- ***Intersection*** : menghasilkan relasi dimana termasuk semua tuple yang muncul, baik di  $R_1$  atau  $R_2$ ,  $R_1$  dan  $R_2$  harus *union compatible*
- ***Difference*** : menghasilkan relasi dimana termasuk semua tuple yang muncul di  $R_1$  tapi tidak ada di  $R_2$ ,  $R_1$  dan  $R_2$  harus *union compatible*

# Union

$\cup$  = union

Checking-account  $\cup$  Savings-account

Checking-account	c-num	c-owner	c-balance
	101	J. Smith	1000.00
	102	W. Wei	2000.00
	104	M. Jones	1000.00
	105	H. Martin	10,000.00

Savings-account	s-num	s-owner	s-balance
	103	J. Smith	5000.00

	c-num	c-owner	c-balance
	101	J. Smith	1000.00
	102	W. Wei	2000.00
	104	M. Jones	1000.00
	105	H. Martin	10,000.00
	103	J. Smith	5000.00



# Intersection

$(\pi_{c\text{-owner}} \text{Checking-account}) \cap (\pi_{s\text{-owner}} \text{Savings-account})$

Checking-account	c-num	c-owner	c-balance
	101	J. Smith	1000.00
	102	W. Wei	2000.00
	104	M. Jones	1000.00
	105	H. Martin	10,000.00

Savings-account	s-num	s-owner	s-balance
	103	J. Smith	5000.00

	c-owner
	J. Smith

# Difference

$$(\pi_{\text{c-owner}} \text{Checking-account}) - (\pi_{\text{s-owner}} \text{Savings-account})$$

Checking-account	c-num	c-owner	c-balance
	101	J. Smith	1000.00
	102	W. Wei	2000.00
	104	M. Jones	1000.00
	105	H. Martin	10,000.00

Savings-account	s-num	s-owner	s-balance
	103	J. Smith	5000.00

Checking-account	c-num	c-owner	c-balance
	102	W. Wei	2000.00
	104	M. Jones	1000.00
	105	H. Martin	10,000.00

# Cartesian Product

- *Cartesian Product* atau *Cross Product* (CP) berfungsi untuk mengkombinasikan tuple dari dua tabel
- Dalam contoh dibawah, CP akan menghasilkan setiap kemungkinan kombinasi dari Teacher X Course

X cross product

Teacher	t-num	t-name
	101	Smith
	105	Jones
	110	Fong

Teacher X Course

Course	c-num	c-name
	514	Intro to DB
	513	Intro to OS

	t-num	t-name	c-num	c-name
	101	Smith	514	Intro to DB
	105	Jones	514	Intro to DB
	110	Fong	514	Intro to DB
	101	Smith	513	Intro to OS
	105	Jones	513	Intro to OS
	110	Fong	513	Intro to OS



# Join

⋈ = join

Account ⋈<sub>Number=Account</sub> Deposit

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	Account	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/2/00	10,000.00

	Number	Owner	Balance	Type	Account	Transaction-id	Date	Amount
	102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
	102	W. Wei	2000.00	checking	102	2	10/29/00	200.00
	104	M. Jones	1000.00	checking	104	3	10/29/00	1000.00
	105	H. Martin	10,000.00	checking	105	4	11/2/00	10000.00



# Division

- Cari semua *account owners* yang mempunyai semua *account type*

÷ or / divide  $(\pi_{\text{Owner, Type}} \text{Account}) \div \text{Account-types}$

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Account-types	Type
	checking
	savings

	Owner
	J. Smith

# Operasi *Assignment*

- Operasi assignment berguna untuk mengekspresikan query kompleks. Menulis query sebagai sebuah program sekuensial yang terdiri atas :
  - # beberapa assignment
  - # diikuti oleh sebuah ekspresi yang nilainya ditampilkan sebagai hasil dari query tersebut.
- Assignment harus selalu dilakukan pada variabel relasi temporary, contoh :

Operasi  $r \div s$  diterjemahkan menjadi :

$$\begin{aligned} \text{temp1} &\leftarrow \Pi R-S (r) \\ \text{temp2} &\leftarrow \Pi R-S ((\text{temp1} \times s) - \Pi R-S, S (r)) \\ \text{result} &= \text{temp1} - \text{temp2} \end{aligned}$$
- Hasil dari operasi yang dilakukan di ruas kanan di-assign ke variabel relasi di ruas kiri
- Variabel dapat digunakan dalam subsekuens operasi



# Contoh Skema Perbankan

- ***cabang*** (*nama-cabang, kota-cabang, aset*)
- ***pelanggan*** (*nama-pelanggan, alamat-pelanggan, status-pelanggan*)
- ***rekening*** (*nomor-rekening, jumlah-simpanan, nama-cabang*)
- ***pinjaman*** (*nomor-pinjaman, nama-cabang, jumlah-pinjaman*)
- ***depositor*** (*nama-pelanggan, nomor-rekening*)
- ***peminjam*** (*nama-pelanggan, nomor-pinjaman*)

# Contoh Query (1)

*pinjaman* (*nomor-pinjaman*, *nama-cabang*, *jumlah-pinjaman*)

- Cari semua *pinjaman* yang nilainya lebih dari \$1200

$\sigma \text{ jumlah-pinjaman} > 1200$  (*pinjaman*)

- Tampilkan nomor *pinjaman* untuk semua *pinjaman* yang nilainya lebih dari \$1200

$\Pi \text{ nomor-pinjaman } (\sigma \text{ jumlah-pinjaman} > 1200$   
(*pinjaman*))



## Contoh Query (2)

*depositor* (nama-pelanggan, nomor-rekening)

*peminjam* (nama-pelanggan, nomor-pinjaman)

- Cari nama semua pelanggan yang mempunyai pinjaman, rekening atau keduanya.

$\Pi$  nama-pelanggan (**peminjam**)  $\cup$   $\Pi$  nama-pelanggan (**depositor**)

- Cari nama semua pelanggan yang mempunyai pinjaman dan rekening.

$\Pi$  nama-pelanggan (**peminjam**)  $\cap$   $\Pi$  nama-pelanggan (**depositor**)



## Contoh Query (3)

**pinjaman** (nomor-pinjaman, nama-cabang, jumlah-pinjaman)

**depositor** (nama-pelanggan, nomor-rekening)

**peminjam** (nama-pelanggan, nomor-pinjaman)

- Cari nama semua nama pelanggan yang mempunyai pinjaman di cabang “Bandung”.

$\Pi$  nama-pelanggan ( $\sigma$  nama-cabang = “Bandung”  
( $\sigma$  peminjam.nomor-pinjaman = pinjaman.nomor-pinjaman(**peminjam** x **pinjaman**)))

- Cari nama semua nama-pelanggan yang mempunyai pinjaman di cabang “Jakarta”, tetapi tidak mempunyai rekening di cabang manapun.

$\Pi$  nama-pelanggan ( $\sigma$  nama-cabang = “Jakarta” ( $\sigma$  peminjam.nomor-pinjaman = pinjaman.nomor-pinjaman(**peminjam** x **pinjaman**))) –  
 $\Pi$  nama-pelanggan(**depositor**)

## Contoh Query (4)

*rekening* (nomor-rekening, jumlah-simpanan, nama-cabang)

- Cari nilai jumlah-simpanan yang paling besar

$\Pi \text{ jumlah-simpanan(rekening)} - \Pi \text{ rekening.jumlah-simpanan}(\sigma \text{ rekening.jumlah-simpanan} < d. \text{ jumlah-simpanan (rekening} \times d \text{ (rekening))})$



# Latihan

*pegawai* (nama-pegawai, alamat-jalan, alamat-kota)

*kerja* (nama-pegawai, nama-perusahaan, gaji)

*perusahaan* (nama-perusahaan, kota)

*manager* (nama-pegawai, nama-manager)

- a) Cari nama **pegawai** yang bekerja di PT Pasundan Jaya.
- b) Cari nama dan kota tempat tinggal semua **pegawai** yang bekerja di PT Pasundan Jaya.
- c) Cari nama, alamat dan kota tempat tinggal semua **pegawai** yang bekerja di PT Pasundan Jaya dan mempunyai gaji lebih dari 5 juta sebulan.
- d) Cari nama semua **pegawai** yang tinggal di kota yang sama dengan **perusahaan** dimana mereka bekerja.
- e) Cari nama semua **perusahaan** di kota Bandung yang menggaji pegawainya lebih dari 5 juta sebulan.