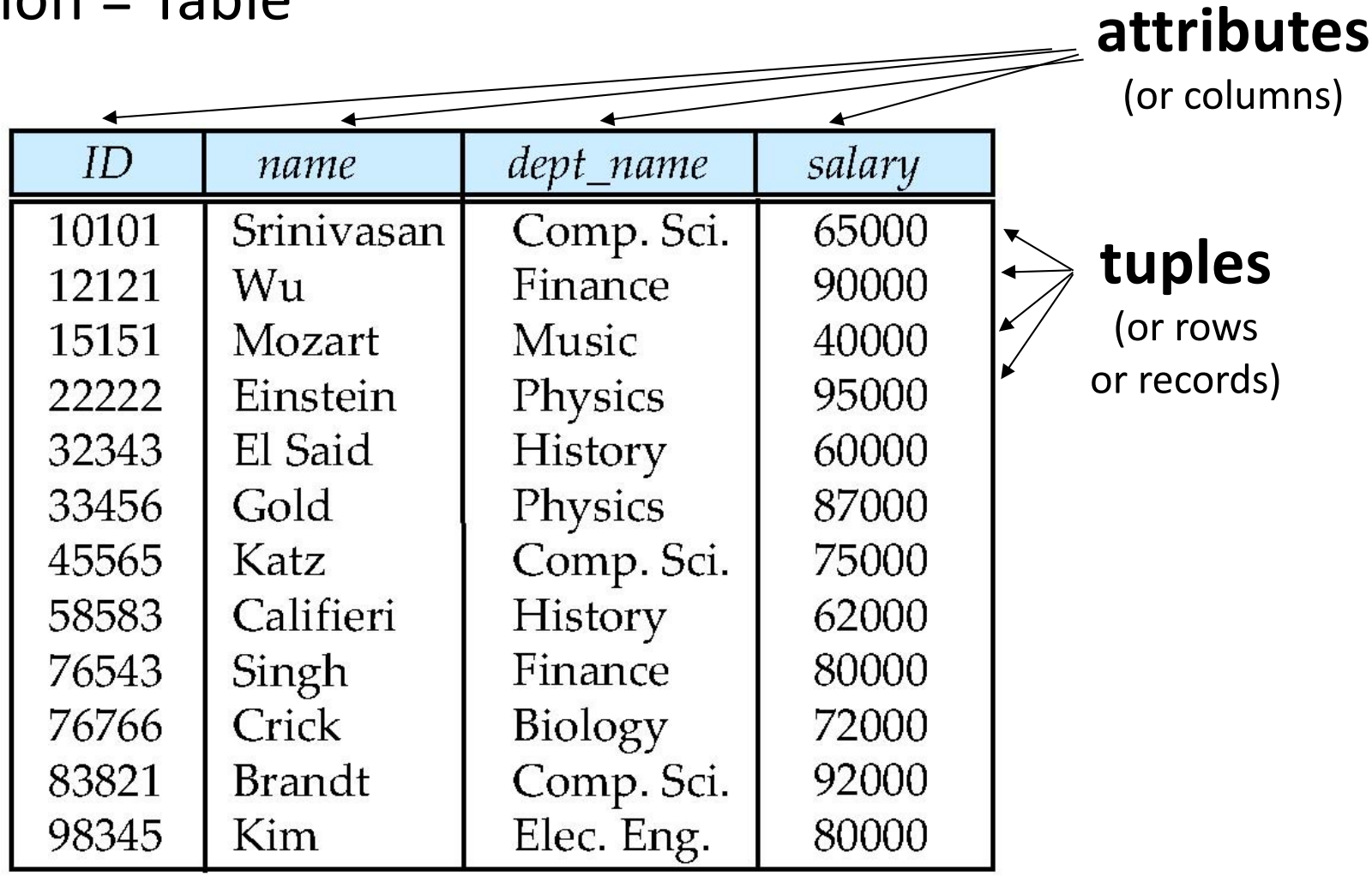
Database Systems
Lecture02 – Introduction to Relational Model

Beomseok Nam (남범석)
bnam@skku.edu

Relation

- Relation = Table



Domain: Attribute Types

- **Domain** is the set of allowed values for each attribute
  - Domain = data type + constraints
  - Eg. score domain: 0~100

- Attribute values need to be **atomic**

Name_N_ID	*Department*
Alice 001	Comp. Sci.
Bob 002	Elec. Eng.
Charlie 003	NULL

Name	*ID*	*Department*
Alice	001	Comp. Sci.
Bob	002	Elec. Eng.
Charlie	003	NULL

- The special value ***null*** is a member of every domain

Relation Schema and Instance

- $A_1, A_2, \ldots, A_n$ are ***attributes***

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

  Example:

  instructor = (*ID*, *name*, *dept_name*, *salary*)

- a ***relation*** is an instance, i.e., a set of ***tuples***

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000

Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
  - i.e., a relation is a set.
- Example: instructor relation with unordered tuples

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Database

- A database consists of multiple relations
- e.g.) a university database consists of

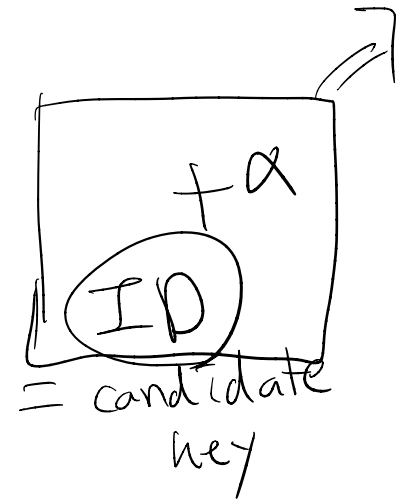
  instructor relation
  student relation
  advisor relation

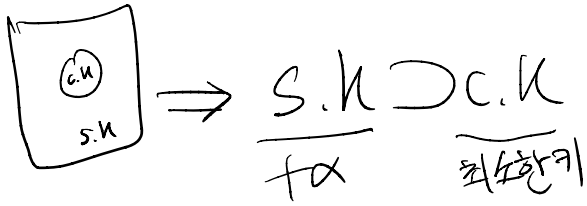
Key

- Q: What attribute plays the most significant role in the following relation?

Key attribute(s) uniquely identify a record in a table

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000
99999	Kim		





Definition of Keys

- Let $K \subseteq R$
- *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple in relation *R*.
  - Example: {*ID*} and {ID,name} are both superkeys of *instructor.*
- Minimal superkey is called **candidate key**

  Example: {*ID*} is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.

Relational Query Languages

- Two mathematical Query Languages
  - **Relational algebra**
    - More operational
    - Useful for representing execution plans
  - Relational calculus
    - More declarative, i.e., describes what they want rather than how to compute it.
    - Rather theoretical, so not covered in this class
    - Eg. Find students whose SSN is 10.

$$\{t \mid t \in STUDENT \wedge t[ssn] = 123\}$$

Relational Query Languages

- Relational operators
  - 5 fundamental relational operators
    - Union ($\cup$)
    - Selection (σ)
    - Projection ($\mathbf{\Pi}$)
    - Cartesian Product (x)
    - Set Difference (-)

5개로 표현가능

Union (∪)

- 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
α	2
β	1
β	3

Set difference (–)

- 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

Common tuple
= (α, 2)
이거만 빼줌

Selection (σ)

- Selection picks rows
- Relation r

- Select tuples with A=B and D > 5

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

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

A	B	C	D
α	α	1	7
β	β	23	10

Selection (σ) - Example

- Find ‘Comp. Sci.’. major student named “Kim”

Student

ID	name	dept_name
10	Kim	Comp. Sci.
20	Lee	Biology
30	Kim	Electrical Eng.

- $\sigma_{name=\text{‘Kim’} \wedge dept\_name = \text{‘Comp. Sci.’}}(Student)$

∧/and 둘중 하나 사용

Projection (Π)

- Projection picks columns and *removes duplicates*
- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- Select A and C
  - Projection
  - $\Pi_{A, C}(r)$

A	C
α	1
α	1
β	1
β	2

=

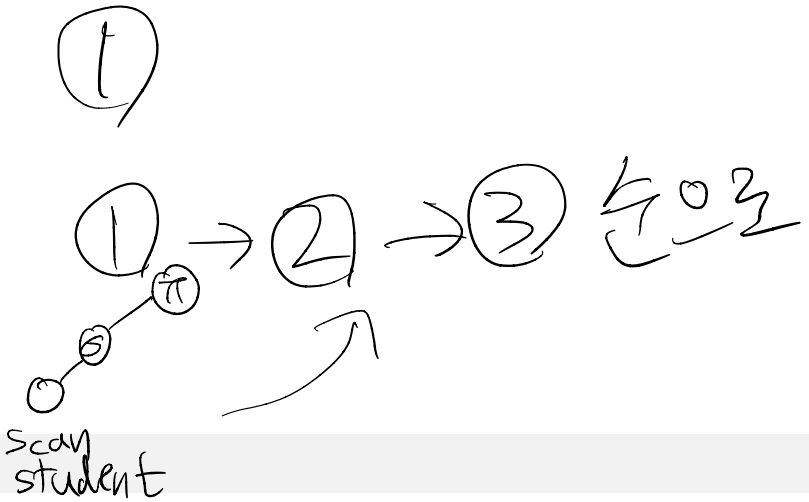
A	C
α	1
β	1
β	2

Projection (Π) - Example

- Find ID of 'Comp. Sci.'. major students.
  - Hint: relational operators can be cascaded

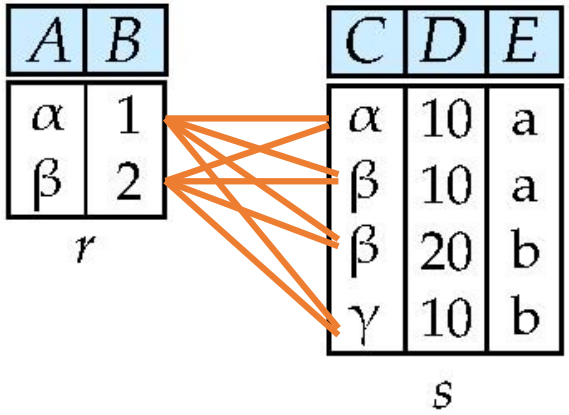
ID	name	dept_name
10	Kim	Comp. Sci.
20	Lee	Biology
30	Kim	Electrical Eng.

- $\Pi_{ID}(\sigma_{dept\_name = \text{'Comp. Sci.'}}(Student))$



Cartesian Product (x)

- Cartesian Product generates all possible pairs
- 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

works as in nested loops:

```
for each row i in r
    for each row j in s
        output i,j
```

Cartesian Product (x)

- Cartesian Product generates all possible pairs
- 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

works as in nested loops:

```
for each row i in r
    for each row j in s
        output i,j
```

Cartesian Product (x) - Example

- Find name of students who take course ‘swe3003’

Student

ID	name	dept_name
10	Kim	Comp. Sci.
20	Lee	Biology
…		

Take

ID	c_id	grade
10	swe2021	A
20	swe3003	B
30	sw4016	A

ID	name	dept_name	ID	c_id	grade
10	Kim	Comp. Sci.	10	swe2021	A
10	Kim	Comp. Sci.	20	swe3003	B
10	Kim	Comp. Sci.	30	sw4016	A
20	Lee	Biology	10	swe2021	A
20	Lee	Biology	20	swe3003	B
20	Lee	Biology	30	sw4016	A

Student x Take

Cartesian Product (x) - Example

- Find name of students who take course ‘swe3003’

Student

ID	name	dept_name
10	Kim	Comp. Sci.
20	Lee	Biology
…		

Take

ID	c_id	grade
10	swe2021	A
20	swe3003	B
30	sw4016	A

ID	name	dept_name	ID	c_id	grade
10	Kim	Comp. Sci.	10	swe2021	A
~~10~~	~~Kim~~	~~Comp. Sci.~~	~~20~~	~~swe3003~~	~~B~~
~~10~~	~~Kim~~	~~Comp. Sci.~~	~~30~~	~~sw4016~~	~~A~~
~~20~~	~~Lee~~	~~Biology~~	~~10~~	~~swe2021~~	~~A~~
20	Lee	Biology	20	swe3003	B
~~20~~	~~Lee~~	~~Biology~~	~~30~~	~~sw4016~~	~~A~~

$$\sigma_{\text{Student.ID=Take.ID}}(\text{Student} \times \text{Take})$$

Cartesian Product (x) - Example

- Find name of students who take course 'swe3003'

Student

ID	name	dept_name
10	Kim	Comp. Sci.
20	Lee	Biology
…		

Take

ID	c_id	grade
10	swe2021	A
20	swe3003	B
30	sw4016	A

Student ⋈ Take

ID	name	dept_name	ID	c_id	grade
10	Kim	Comp. Sci.	10	swe2021	A
10	Kim	Comp. Sci.	20	swe3003	B
10	Kim	Comp. Sci.	30	sw4016	A
20	Lee	Biology	10	swe2021	A
20	Lee	Biology	20	swe3003	B
20	Lee	Biology	30	sw4016	A

$\sigma_{\text{Student.ID=Take.ID} \wedge \text{c\_id='swe3003'}}$ (Student x Take)

Cartesian Product (x) - Example

- Find name of students who take course ‘swe3003’

Student

ID	name	dept_name
10	Kim	Comp. Sci.
20	Lee	Biology
…		

Take

ID	c_id	grade
10	swe2021	A
20	swe3003	B
30	sw4016	A

ID	name	dept_name	ID	c_id	grade
10	Kim	Comp. Sci.	10	swe2021	A
10	Kim	Comp. Sci.	20	swe3003	B
10	Kim	Comp. Sci.	30	sw4016	A
20	Lee	Biology	10	swe2021	A
20	Lee	Biology	20	swe3003	B
20	Lee	Biology	30	sw4016	A

$$\Pi_{\text{name}} (\sigma_{\text{Student.ID=Take.ID} \wedge \text{c\_id='swe3003'}} (\text{Student} \times \text{Take}))$$

5 relational operators are enough! But,

- 5 fundamental relational operators are enough to write any query
  - Union ($\cup$)
  - Selection (σ)
  - Projection ($\mathbf{\Pi}$)
  - Cartesian Product (x)
  - Set Difference (-)

- But, derived operators, for convenience
  - Set intersection ($\cap$)
  - Join ($\bowtie$)
  - Rename (ρ)
  - Division ($\div$)
  - Group by (G)

Set Intersection of two relations (∩)

- Relation *r, s*:

A	B
α	1
α	2
β	1

r

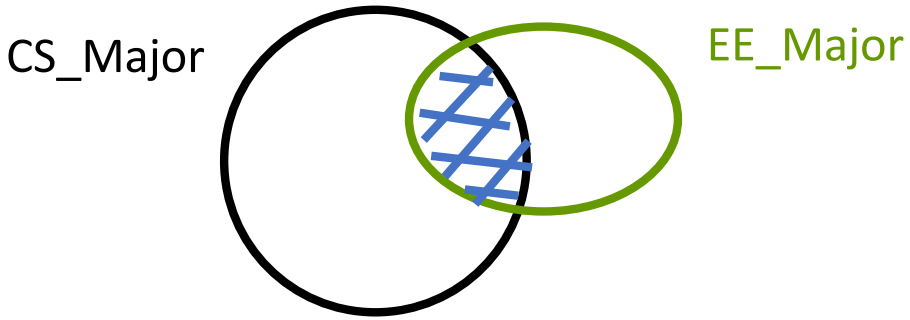
A	B
α	2
β	3

s

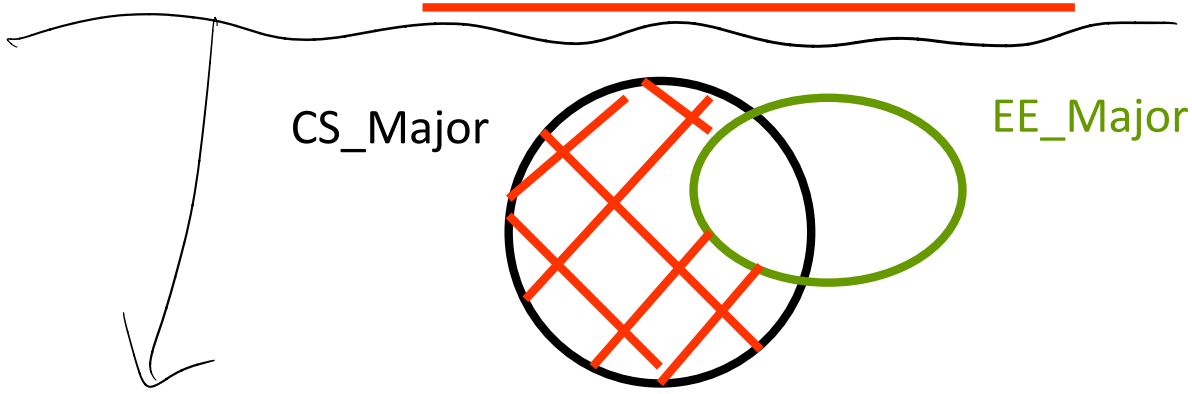
r ∩ *s*

A	B
α	2

Observation ∩를 5가지 연산으로 바꾸는방법



- CS_Major ∩ EE_Major =
  CS_Major – (CS_Major – EE_Major)



So, ∩ is not fundamental

Natural Join (⋈)

- Natural Join
  - $R \bowtie S = \sigma_{R.a=S.a \wedge R.b=S.b \wedge \ldots} (R \times S)$
  - For each pair of tuples t_R from R and t_S from S,
  - If t_R and t_S have the same values on common attributes, add a tuple t to the result, where
    - t has the same value as t_R
    - t has the same value as t_S
    - Drop duplicate attributes

Natural Join (⋈) - Example

- Find name of students who take course ‘swe3003’

Student

ID	name	dept_name
10	Kim	Comp. Sci.
20	Lee	Biology
30	Kim	Electrical Eng.

Take

ID	c_id	grade
10	swe2021	A
20	swe3003	B
30	sw4016	A

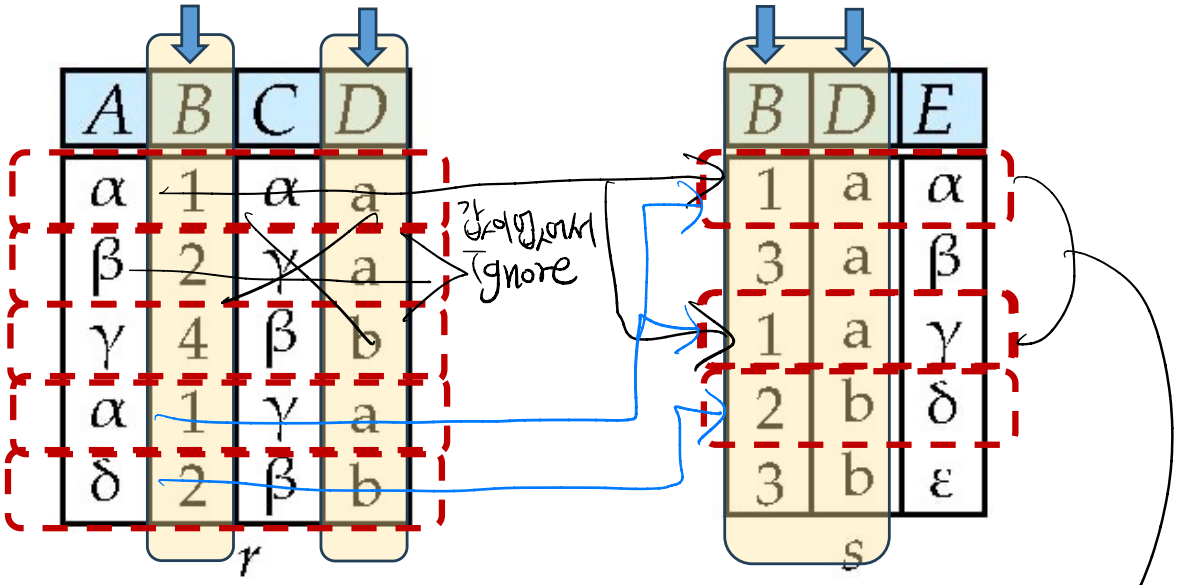
ID	name	dept_name	c_id	grade
10	Kim	Comp. Sci.	swe2021	A
20	Lee	Biology	swe3003	B
30	Kim	Electrical Eng.	swe4016	A

이 표가 Student ⋈ Take

$$\Pi_{name}(\sigma_{c\_id='swe3003'}(Student \bowtie Take))$$

Natural Join with multiple common attributes

- Relations r, s:



- Natural Join

$r \bowtie s$

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

같은 특징 & 같은 값만 합치기

rename (ρ)

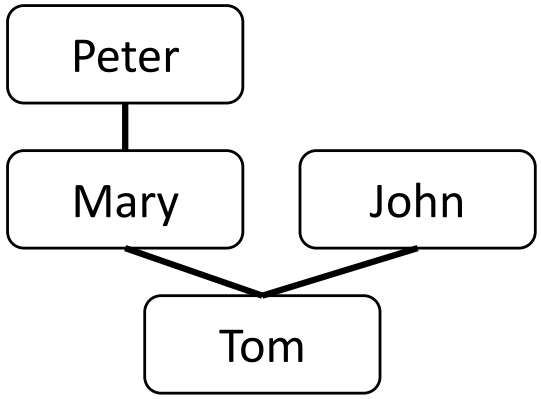
$$\rho_{AFTER}(BEFORE)$$

new before

- Q: why?
- A: The same table is used multiple times

rename (ρ)

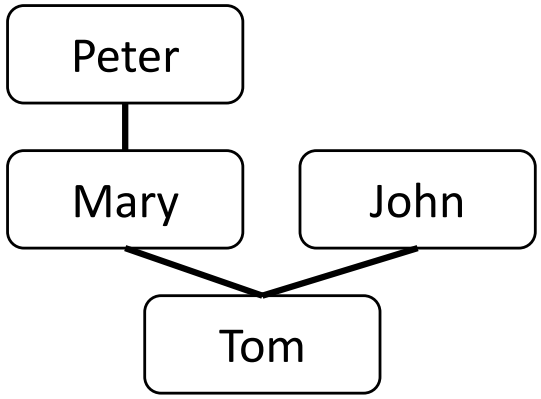
- find the grand-parents of 'Tom', given PC(parent-id, child-id)



parent	child
Mary	Tom
Peter	Mary
John	Tom

rename (ρ)

- find the grand-parents of 'Tom', given PC(parent-id, child-id)



copy!

parent	child
Mary	Tom
Peter	Mary
John	Tom

parent	child
Mary	Tom
Peter	Mary
John	Tom

- We need two different names for the same table - hence, the 'rename' op.

$$\pi_{PC1.parent}\left(\sigma_{PC1.child=PC.parent}\left(\rho_{PC1}(PC) \times PC\right)\right)$$

↑ PC.child = 'Tom'

Tom 조건을 만족하는 부사한 테이블의 이름을 변경함

Division (÷)

- Useful for expressing queries like:
  *Find students who have taken **all** CS courses.*

- Suppose *R* has 2 attributes *(x,y)* and *S* has (*y*).
- Then *R* ÷ S returns the set of *x's* that match all *y* values in *S*.
- Example: *R* = Friend(x,y). *S* = set of 10 students.
  - Then *R* ÷ *S* returns the set of all x's that are friends with all 10 students.

나누는 S테이블에 있는 모든 값이 있는 R테이블값 남김

Division (÷) - Example

- Find ID of students who take all 'Comp. Sci' courses

Take

ID	c_id
10	swe3003
10	ece2031
20	swe3003
20	swe4021
20	ece2031
30	swe3003
30	swe4021

dept_name	c_id
Comp. Sci.	swe3003
Comp. Sci	swe4021
Electrical Eng.	ece2031

Course

$\Pi_{c\_id}\ \sigma_{dept\_name=\text{'Comp. Sci.'}}\ (Course)$

c_id
swe3003
swe4021

$Take \div (\Pi_{c\_id}\ \sigma_{dept\_name=\text{'Comp. Sci.'}}\ (Course))$

ID
20
30

Division (÷) - Example

- Find ID of students who take all ‘Comp. Sci’ courses

R

ID	proj_no
10	p1
10	p2
10	p3
10	p4
20	p1
20	p2
30	p2
40	p2
40	p4

proj_no
p2

S_1

proj_no
p2
p4

S_2

proj_no
p1
p2
p4

S_3

ID
10
20
30
40

$R \div S_1$

ID
10
40

$R \div S_2$

ID
10

$R \div S_3$

Observations

- Division ($\div$) is reverse of cartesian product

- It can be derived from the 5 fundamental operators

$$r \div s = \pi_{(R-S)}(r) - \pi_{(R-S)}[(\pi_{(R-S)}(r) \times s) - r]$$

Let's practice using the following schema

- 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 in Relational Algebra

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

- Find all attributes of loans of over $1200

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

one column

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

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

Example Queries in Relational Algebra

depositor (customer-name, account-number)

borrower (customer-name, loan-number)

- Find the names of all customers who have a loan, an account, **or** both, from the bank

빌리거나 빌려준사람 모두

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

- Find the names of all customers who have a loan **and** an account at bank.

빌리고 빌려준사람

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

Example Queries in Relational Algebra

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

borrower (customer-name, loan-number)

depositor (customer-name, account-number)

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

$$\Pi_{\text{customer-name}} (\sigma_{\text{branch-name}=\text{“Suwon”}} (\text{borrower} \bowtie \text{loan})))$$

same expression

$\Pi_{\text{customer-name}}(\sigma_{\text{b-name}=\text{"suwon"}}(\text{loan}) \bowtie \text{borrower})$

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

$$\Pi_{\text{customer-name}} (\sigma_{\text{branch-name} = \text{“Suwon”}} (\text{borrower} \bowtie \text{loan})) - \Pi_{\text{customer-name}}(\text{depositor})$$

commutative associative Rule

$a*(b+c) = a*b+a*c$