



# 3. Relational Model & Basic SQL

## ▼ Relational Query Languages

Relational algebra  
tuple relational calculus  
domain relational calculus

SQL(structured query languages)과 같은 표현식을 가지는 pure language

### ★ Relational Algebra ⇒ tuple 중복은 제거가 default

- **Selection** of tuples ( $\sigma$ ):  $\sigma_{A=B \text{ and } D>5}(r)$

tuple을 찾아주는 기능

$\sigma_{\text{조건}}(r)$   
relation

A	B	C	D
$\alpha$	$\alpha$	1	7
$\alpha$	$\beta$	5	7
$\beta$	$\beta$	12	3
$\beta$	$\beta$	23	10

⇒

A	B	C	D
$\alpha$	$\alpha$	1	7
$\beta$	$\beta$	23	10

- =, ≠, 부등호 사용 가능
- tuple 보여줌

- **Projection** of attributes :  $\Pi_{A,C}(r)$

→ attribute 선택.  
(중복 tuple 제거)

$\Pi(\text{조건})(\text{relation})$

A	B	C
$\alpha$	10	1
$\alpha$	20	1
$\beta$	30	1
$\beta$	40	2

⇒

A	C
$\alpha$	1
$\beta$	1
$\beta$	2

→ 한번만

- **Union** of two relations → 기본적으로 attribute가 같아야 함

→ 중복 tuple 제거

$r \cup s$

A	B
$\alpha$	1
$\alpha$	2
$\beta$	1

A	B
$\alpha$	2
$\beta$	3

A	B
$\alpha$	1
$\alpha$	2
$\beta$	1
$\beta$	3

→ 한번만

- 중복된 건 제거하고 올림

- **Set difference** of two relations (-)

- $r - s$

- → r에서 s와 공통으로 가지고 있는 tuple 제거한 r

A	B
$\alpha$	1
$\alpha$	2
$\beta$	1

A	B
$\alpha$	2
$\beta$	3

A	B
$\alpha$	1
$\beta$	1

- **Set Intersection** of two relations

$r \cap s$

A	B
$\alpha$	1
$\alpha$	2
$\beta$	1

A	B
$\alpha$	2
$\beta$	3

A	B
$\alpha$	2

- r과 s의 공통된 tuple만!

- joining two relations - **Cartesian Product(x)**

→ 성능저하 이슈

$r \times s$

A	B	C	D	E
$\alpha$	1	$\alpha$	10	a
$\alpha$	1	$\beta$	10	a
$\alpha$	1	$\beta$	20	b
$\alpha$	1	$\gamma$	10	b
$\beta$	2	$\alpha$	10	a
$\beta$	2	$\beta$	10	a
$\beta$	2	$\beta$	20	b
$\beta$	2	$\gamma$	10	b

→ 만들 수 있는 모든 pair, 두 relation에 들어 있는 모든 tuple

◦ 불필요한 연산, 데이터 ⇒ 손해

◦ 같은 이름을 갖는 attribute : *relation 이름, attribute 이름*

→ 서로 다른 pair라고 인식, 같은 attribute라도 다른 값으로 구분 ★

*Instructor*

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	Califeri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

*Teaches*

instructor.ID	name	dept_name	salary	teaches.ID	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2017
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2018
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2018
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
15151	Mozart	Music	40000	10101	CS-101	1	Fall	2017
15151	Mozart	Music	40000	10101	CS-315	1	Spring	2018
15151	Mozart	Music	40000	10101	CS-347	1	Fall	2017
15151	Mozart	Music	40000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
15151	Mozart	Music	40000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
22222	Einstein	Physics	95000	10101	CS-101	1	Fall	2017
22222	Einstein	Physics	95000	10101	CS-315	1	Spring	2018
22222	Einstein	Physics	95000	10101	CS-347	1	Fall	2017
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2018
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...

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

### • Joining two relations – Natural Join

$$r \bowtie s$$

*각각의 tuple 쌍*

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 이름을 갖는 attribute가 있는지 확인

→ 서로 같은 값을 가지는 tuple을 하나로 합쳐줌(모든 경우의 수에 대해 작성)

*RMS* → *if, 각각 tuple add* → *t : t가 R에 가지고 있음*

*t : t가 S에 가지고 있음*

- the rename operations → 많이 쓰이는 X

$$\rho_x(E)$$

- expression E를 x로 치환
- 복잡한 친구를 간단한 친구로 대체하는 operation

$$\rho(A_1, A_2, \dots, A_n)(E)$$

→ 표현식.  
↳ attribute

## ▼ Example Queries in Relational Algebra

- example 1

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

- Find all attributes of loans of over \$1200 (200를 넘은 loan을 가진 모든 attribute?.)

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

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

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

- example 2

depositor (customer-name, account-number)

borrower (customer-name, loan-number)

관련 → 영어로 읽기 가능!

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

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

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

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

## ▼ Introduction to SQL

SQL : data-definition to language → 사람에게 친숙한 형태의 DDL

- schema** : relation 표현 → 변형의 형태

- **domain** : attribute 범위

- type

- char(n) → 고정된 길이
- varchar(n) → 변수 사용
- int
- smallint (작은 int)
- numeric(p,d) → 유효숫자 [ precision ⇒ p digits  
decimal ⇒ d digits
- real ] floating point 표현 ⇒ byte 차이
- float(n)

- **integrity constraints**

- primary key (A1, ... , An)
- foreign key(Am, ... , An) references r ⇒ r의 primary key
- not null

### ▼ Create table construct

```
create table r (
    A1 D1, A2 D2, ..., An Dn,
    (integrity-constraint1),
    ...,
    (integrity-constraintk))
```

relation 이름 (relation rel schema)  
→ attribute 이름 / domain 이름.  
→ integrity constraint.

```
//example
create table student (
    ID varchar(5),
    name varchar(20) not null,
    dept_name varchar(20),
    tot_cred numeric(3,0),
    primary key (ID),
    foreign key (dept_name) references department);
```

→ null 값이 아님을 가늠 짓기  
Integrity constraint.

### ★ ▼ Updates to tables

- **Insert** : relation 안에 tuple value 추가
- **Delete** : 모든 tuple 다 삭제(table은 남아 있음)

- **Drop Table** : table 자체를 삭제(table도 사라짐)
- **Alter** : 수정하고 싶을 때 사용 ⇒ attribute 수정
  - `alter table r add A D` : attribute A를 domain D에 null 값으로 추가
  - `alter table r drop A` : attribute A를 삭제

① insert into (instructor) values ('10211', 'Smith', 'Biology', 6) → attribute 순서대로 작성  
 ② delete from (student) → 모든 tuple이 다 삭제됨. table은 남아있음. → tuple value 순서대로 attribute  
 ③ drop table r → table 자체까지 다 삭제  
 ④ alter [alter table r add A D → 추가 (null 값도)  
 alter table r drop A → attribute 삭제]

## ▼ Basic Query Structure

- **select** ⇒ 대부분의 query, 내가 찾고자 하는 tuple 찾아줌

SQL Statement

select A1, A2, ..., An → attribute  
 from r1, r2, ..., rm → relation  
 where P → predicate

//대소문자 구분 안 함

- **select distinct ...** : 중복된 건 하나만 출력 (중복문 remove)
- **select all ...** : 중복되어도 다 출력 (중복된 결과까지 다)
- **select \*** ... : 모든 tuple의 모든 attribute 출력
- **select (존재하지 않는 table 이름)** : 하나의 열, 행이 있는 table이 임시적으로 생김 → table 만들기.
- **select (") as foo** : foo의 attribute의 value가 "인 table이 임시적으로 생김
- **select 'A' ..** : tuple의 value가 'A'로 가득 찬 table 생성
- **where** : 조건문 (result가 반드시 만족해야함)

\* 컴퓨 모든 값 출력.

select name  
 from instructor //table 이름이 오는게 가장 심플하지만 표현이 올 수 있음.  
 where dept\_name = 'Comp. Sci.' //내가 원하는 조건에 대해 작성

\* 연방이 구한이 넘는 컴퓨의 모든 값을

select name  
 from instructor

```
where dept_name = 'Comp. Sci.' and salary > 70000
//여러 가지 조건에 대해 괄호로 구분 지어서 작성하면 good
```

- and, or, not 사용 가능
- 비교연산자, 부정연산자 사용 가능
- 결과에 arithmetic expression 적용할 수 있음

• from : query 안에 포함된 relation을 나타내

```
(relation algebra | Cartesian product 사용함)
select *
from instructor, teaches
//relation 여러 개 작성할 때 ,로 구분 => pair로 만들어줌.
```

- relational algebra의 cartesian product SQL ver.
- 가능한 모든 pair를 만들어 줌 → 두 relation (instructor, teaches)의 모든 attribute에 관한 pair
- 같은 attribute에 대해 → instructor.ID와 같이 설정됨
- where문과 함께 잘 쓰임. table attribute

• example

```
//1. 주어진 course를 가르친 모든 instructor의 이름과 course_id를 가져옴
select name, course_id
from instructor, teaches
where instructor.ID = teaches.ID //열린 과목의 ID = 교수의 ID
//비교하는 attribute가 겹치는 attribute가 아니라면 table 이름 명시(X)
```

```
//2. 주어진 course를 가르친 미대의 모든 instructor의 이름을 찾음
select name, course_id
from instructor, teaches
where instructor.ID = teaches.ID
and instructor.dept_name = 'Art'
```

→ 조건이 하나 더!

\* ordering 중요  
→ 연산?