Database 101

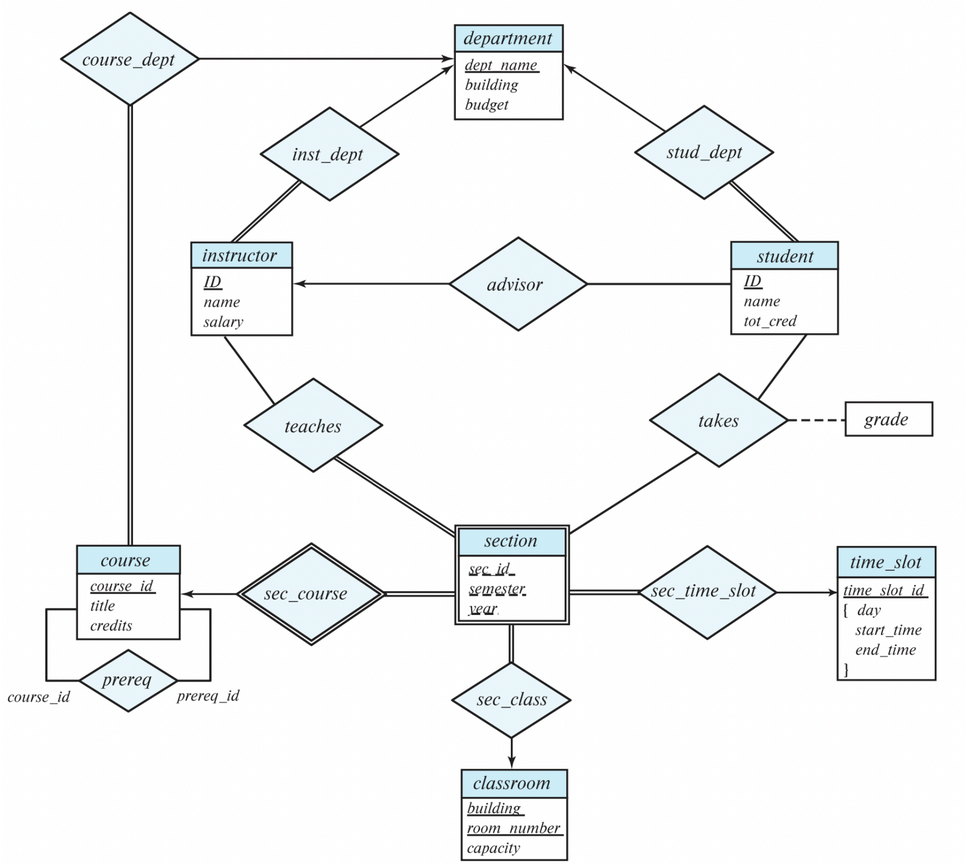
3. SQL Basic

김용담 강사

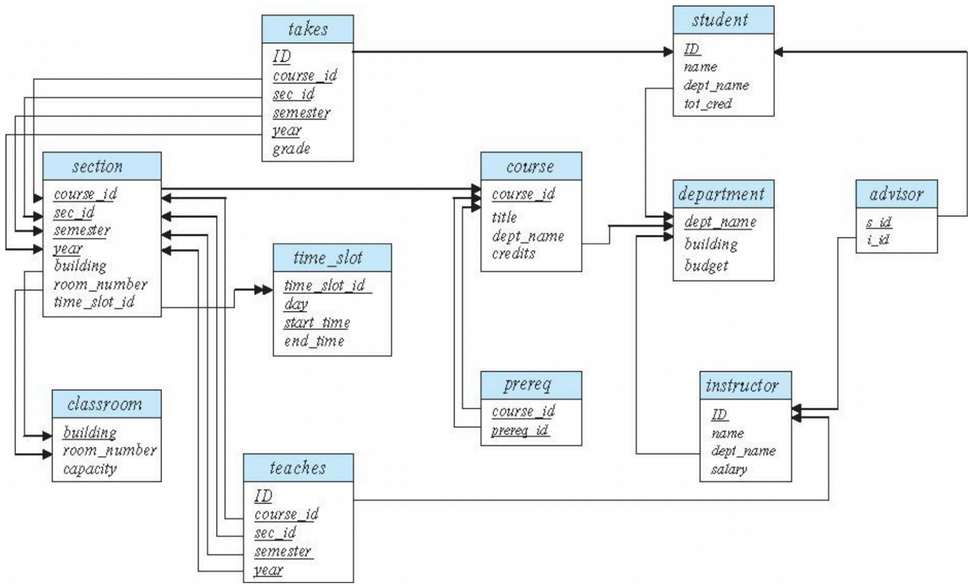
Contents

1. University DB

대학교 CRM을 위한 데이터베이스



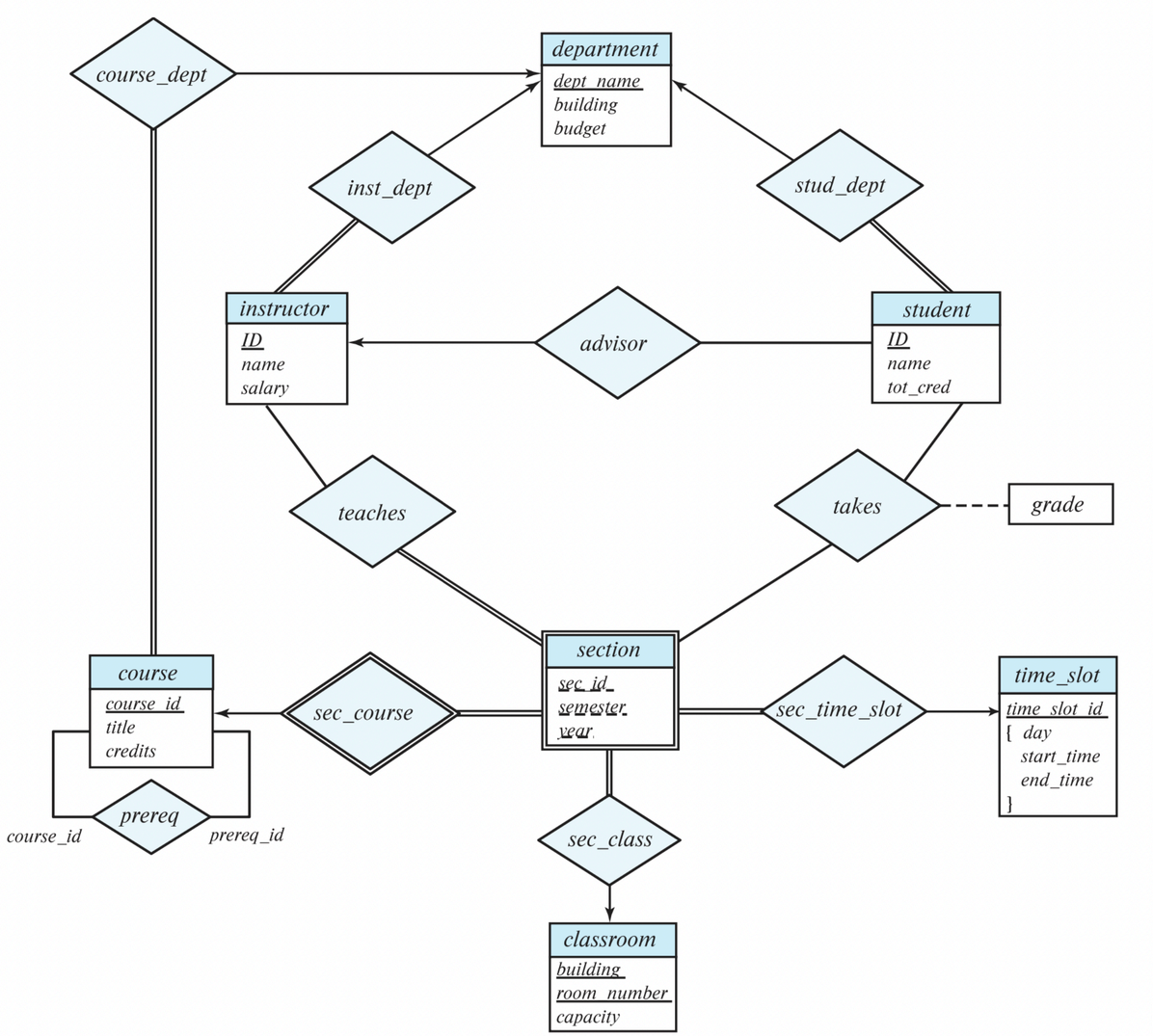
Conceptual Model(ERD)



Logical Model(Schema)

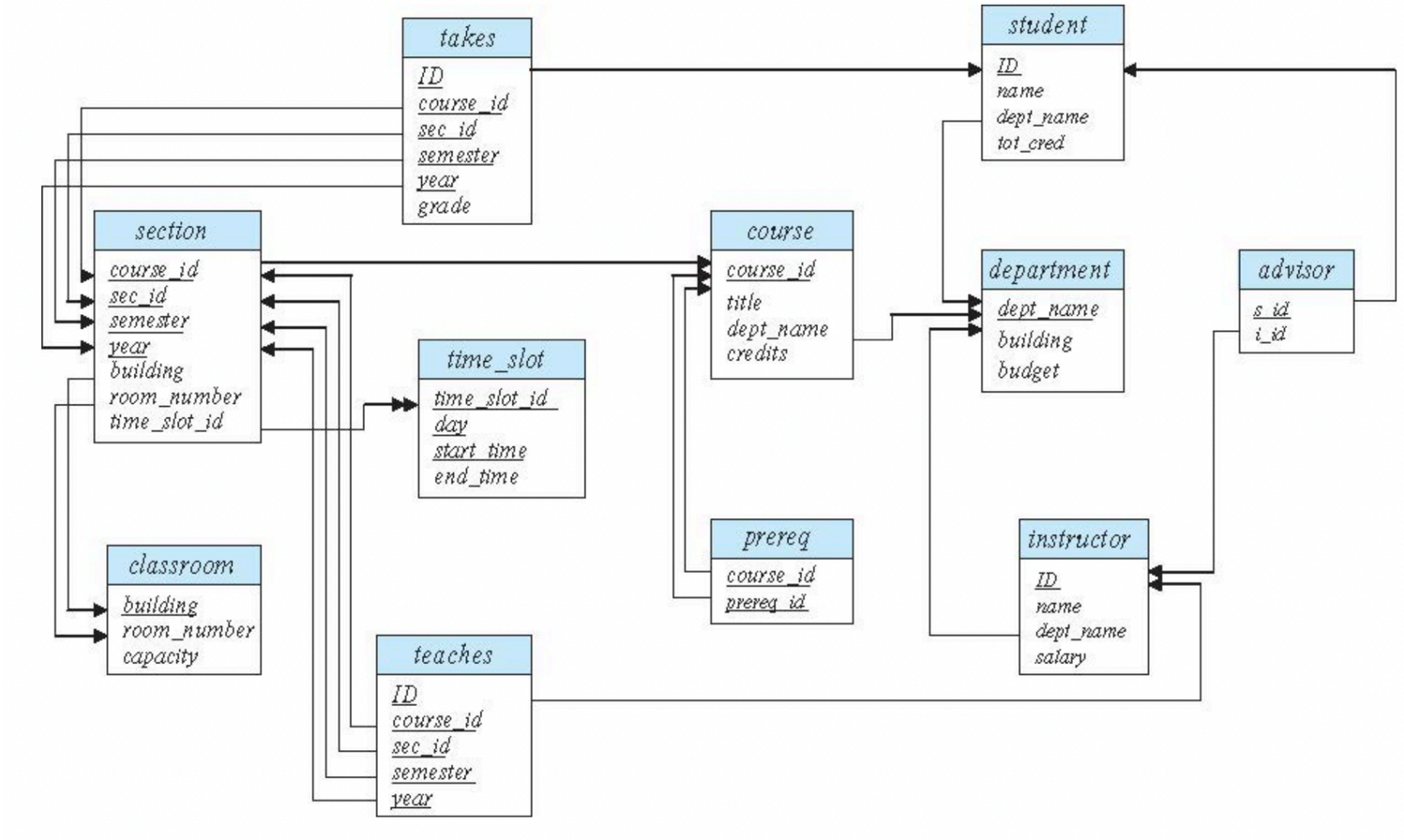
1. University DB

기능에 대한 이해 (ERD)



1. University DB

구현에 대한 이해(Schema)



2. Make DB
CREATE TABLE

```
create table instructor (
        ID              char(5),
        name            varchar(20) not null,
        dept_name       varchar(20),
        salary          numeric(8,2),
        primary key (ID),
        foreign key (dept_name) references department);
```

```
create table takes (
        ID              varchar(5),
        course_id       varchar(8),
        sec_id          varchar(8),
        semester        varchar(6),
        year            numeric(4,0),
        grade           varchar(2),
        primary key (ID, course_id, sec_id, semester, year) ,
        foreign key (ID) references  student,
        foreign key (course_id, sec_id, semester, year) references section);
```

- **CREATE TABLE**을 사용하면 table을 정의할 수 있다.
- schema에 정의된 entity를 그대로 table로 구현하면 된다.
- attribute를 모두 정의한 뒤에 PK와 FK를 정의한다.
- 각 attribute는 적절한 data type(domain)으로 정의한다.

2. Make DB

INSERT INTO

```
insert into classroom values('Polya', 808, 28);
insert into classroom values('Gates', 707, 65);
insert into classroom values('Gates', 314, 10);
insert into classroom values('Main', 45, 30);
insert into classroom values('Taylor', 183, 71);
insert into classroom values('Power', 972, 10);
insert into classroom values('Garfield', 119, 59);
insert into classroom values('Rathbone', 261, 60);
insert into classroom values('Stabler', 105, 113);
insert into classroom values('Power', 717, 12);
insert into classroom values('Main', 425, 22);
insert into classroom values('Lambeau', 348, 51);
insert into classroom values('Chandler', 804, 11);
insert into department values('Civil Eng.', 'Chandler', 255041.46);
insert into department values('Biology', 'Candlestick', 647610.55);
insert into department values('History', 'Taylor', 699140.86);
insert into department values('Physics', 'Wrigley', 942162.76);
insert into department values('Marketing', 'Lambeau', 210627.58);
insert into department values('Pol. Sci.', 'Whitman', 573745.09);
insert into department values('English', 'Palmer', 611042.66);
```

- **INSERT INTO**을 사용하면 table에 tuple을 추가할 수 있다.
- 정의한 attribute 순서에 맞게 입력한다.
- 일부 attribute에만 데이터를 추가할 수도 있다.
- not null 조건이 있는 attribute라면 무조건 INSERT에 포함되어 있어야 한다. e.g. PK

3. Basic Query Structure

SELECT * FROM table;

SELECT	찾을 attribute
FROM	검색할 table
WHERE	검색 대상의 조건

3. Basic Query Structure

Examples

query : "instructor table의 모든 row들을 표시하라."

SELECT

FROM

WHERE

3. Basic Query Structure

Examples

query : "instructor table에서 ID와 name, 연봉의 12분의 1을 출력해라."

SELECT

FROM

WHERE

3. Basic Query Structure

Examples

query : "Computer Science 학부에 있는 모든 instructor를 찾아라"

SELECT

FROM

WHERE

3. Basic Query Structure

Examples

query : "Computer Science 학부에 있는 모든 instructor중에 연봉이 $80000 이상인 사람들을 찾아라"

SELECT

FROM

WHERE

3. Basic Query Structure

Examples

query : "미술 학부의 강사들 중에서 과거에 과목을 가르친 경험이 있는 모든 강사의 이름과 해당 과목의 ID를 출력하라"

SELECT

FROM

WHERE

3. Basic Query Structure

Examples

query : "컴퓨터과학부의 강사보다 높은 연봉을 가지는 모든 강사의 이름을 출력하라"

SELECT

FROM

WHERE

3. Basic Query Structure

검색 결과를 정렬하여 보여주기 (ORDER BY)

SELECT	찾을 attribute
FROM	검색할 table
WHERE	검색 대상의 조건
ORDER BY	**정렬 기준, 정렬 조건**

3. Basic Query Structure

Examples

query : "물리학과 학부생들을 알파벳순으로 정렬하여 출력하라"

SELECT

FROM

WHERE

ORDER BY

3. Basic Query Structure

Examples

query : "물리학과 학부생들을 알파벳순으로 정렬하여 출력하라"

SELECT

FROM

WHERE

ORDER BY

4. Aggregate Functions

집계 함수

SUM	합계
AVG	평균
MIN	최소
MAX	최대
COUNT	빈도수

4. Aggregate Functions

Examples

query : "컴퓨터과학부 강사들의 평균 연봉을 출력하라"

SELECT

FROM

WHERE

4. Aggregate Functions

Examples

query : "2018년도 봄학기에 수강했던 학생들의 숫자를 출력하라."

SELECT

FROM

WHERE

4. Aggregate Functions

그룹별로 묶어서 결과를 보여주기 (GROUP BY)

SELECT	찾을 attribute
FROM	검색할 table
WHERE	검색 대상의 조건
GROUP BY	그룹을 묶는 기준

4. Aggregate Functions

Examples

query : "각 학부별 강사들의 평균 & 최대 연봉을 출력하라"

SELECT

FROM

GROUP BY

4. Aggregate Functions

Examples

query : "각 학부별 강사의 연봉이 $42000를 넘는 모든 학부를 출력하라"

SELECT

FROM

GROUP BY

HAVING

5. Practice Tips!

모든 것은 기본형으로 부터

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

(a) The *instructor* table

- query를 구성할 때는 SELECT, FROM, WHERE절을 기본으로 잡습니다.
- 가져올 데이터를 row / column 조건을 생각합니다.
- row 조건은 WHERE에
- column 조건은 SELECT에
- 가져오는 대상인 table은 FROM에

5. Practic Tips!

모든 것은 기본형으로 부터

- 어떤 column을 기준으로 같은 집단을 묶어야 한다면 GROUP BY를 사용합니다.
- 출력 결과에 정렬이 필요하다면 ORDER BY를 사용합니다.
- 조건에 부합하는 데이터를 찾은 뒤에, 추가 연산이 필요하다면 기능에 맞는 연산들을 추가해봅시다.
  e.g. SUM, AVG, COUNT, ...

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

(a) The *instructor* table

5. Practice Tips!

Apply

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

(a) The *instructor* table

- Q. "Kim"의 dept_name와 salary를 보여주세요.
- Q2. dept_name이 Physics인 사람의 name을 보여주세요.

5. Practice Tips!

Apply

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

(a) The *instructor* table

- Q3. salary가 65000을 넘는 모든 사람의 이름을 출력해주세요.

- Q4. Finance department에 속한 사람중에 salary가 70000이 넘는 사람의 이름을 출력해주세요.

5. Practice Tips!

Apply

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

(a) The *instructor* table

- Q5. 컴퓨터과학부에 속한 강사들의 평균, 최소, 최대 연봉을 출력해주세요.
- Q6. 가장 높은 연봉을 받는 강사가 속한 학부의 이름을 출력해주세요.

5. Practice Tips!

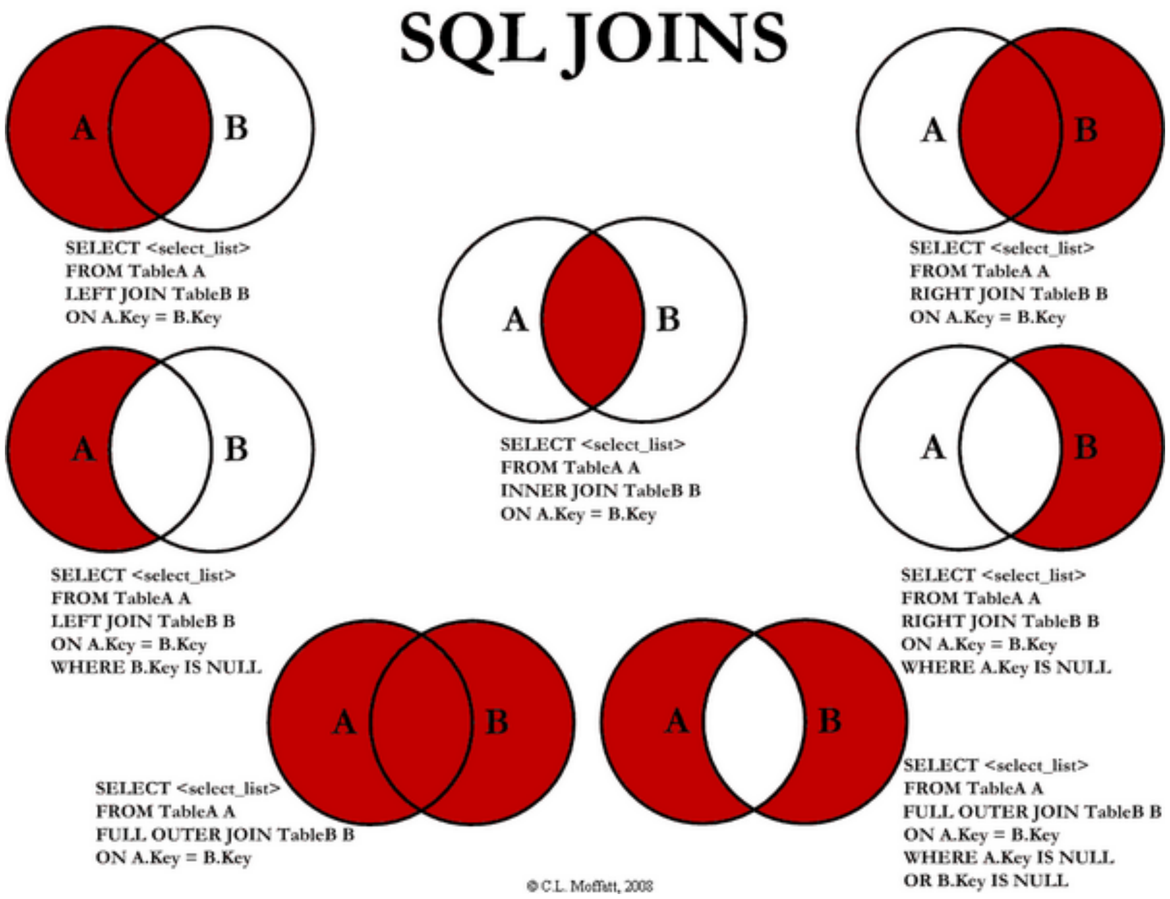
Summary

- SQL로 작성해야 하는 query는 기본 틀부터 잡고 시작합니다.
  SELECT 보여줘야 하는 column들
  FROM 테이블 이름
  WHERE 찾아야하는 row들

- 필요한 추가적인 연산이 있다면, ORDER BY / GROUP BY를 생각해봅니다.
- 굉장히 복잡한 조건이 섞여있는 query의 경우에는 subquery를 사용하는 것을 고려합니다.
  (다음 강의)
- 단계별로 출력되는 table의 결과를 미리 생각해보고 (또는 출력해보고) 연산을 하나하나 추가합니다.

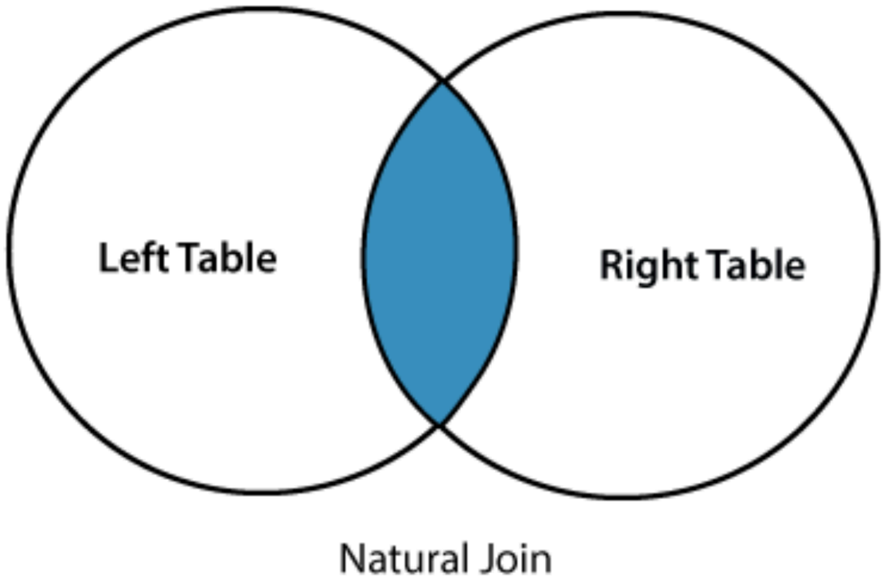
6. JOIN

서로 다른 table의 정보를 합칠 때 사용하는 연산



6. JOIN

natural join



- **NATURAL JOIN**을 사용하면 두 개의 table에서 서로 겹치는 모든 record들에 대해 합쳐준다.
- 합칠 때는 Cartesian Product 연산을 수행한다.

6. JOIN

Examples

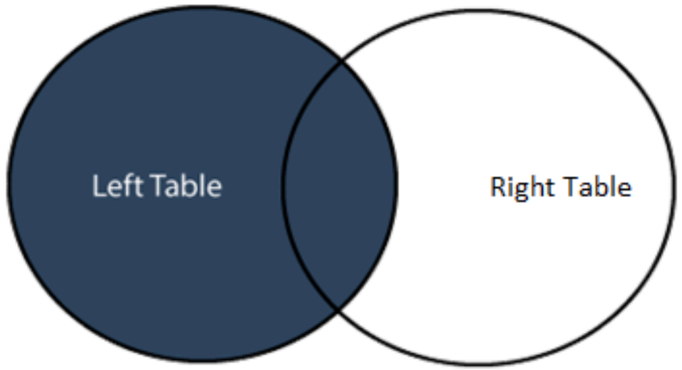
query : "모든 강사들의 이름과 그들이 가르쳤던 모든 과목 ID를 출력하라."

```
SELECT
FROM
WHERE
```

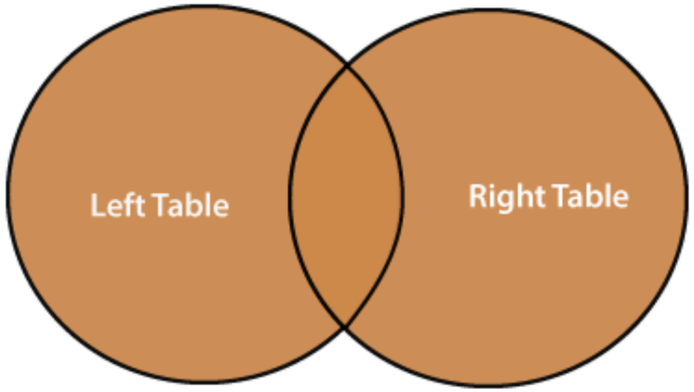
```
SELECT
FROM
NATURAL JOIN
```

6. JOIN

outer join



Left outer Join



Full outer join

- **OUTER JOIN**을 사용하면 두 개의 table에서 서로 겹치는 데이터를 포함하여 원본 record까지 합쳐준다.
- **LEFT, RIGHT, FULL** 옵션에 따라 합쳐지는 범위가 달라진다.

7. Subqueries

쿼리 안에 또 쿼리를 넣어서 복잡한 연산을 처리

```
SELECT

FROM   (SELECT ~
          FROM ... )

WHERE
```

- **subquery**를 사용하면 훨씬 더 많은 쿼리를 유연하게 처리가 가능하다.
- 주로 **FROM**과 함께 쓰이며, 검색할 대상을 연산으로 정의해야할 때 사용된다.
- subquery의 결과는 **table**이다.

7. Subqueries

Examples

query : "각 학부별 강사들의 연봉이 $42000가 넘는 학부와 그 때의 평균 연봉을 출력하라.

SELECT

FROM (SELECT

WHERE

7. Subqueries

Examples

query : "각 학부별로 가장 높은 연봉을 가지는 강사의 이름과 연봉을 출력하라."

```
SELECT

FROM ( SELECT



WHERE
```

연습하러 가볼까요?

https://leetcode.com/problemset/database/