3. SQL

Basic Query Structure

Generated by 이유림

참고 강의 : https://www.edwith.org/database/lecture/27854/

BASIC QUERY STRUCTURE

- SQL query의 기본적인 형태 (정보 검색을 위한 가장 기본적인 형태)

select	A1, A2, …, An	결과로 얻고 싶은 속성들
from	r1, r2, …, rm	쿼리에 사용되는 relation 이름들
where	P	검색조건. where 절이 없다면 모든 튜플이 조건을 만족하는 것으로 여겨, 모든 튜플을 반환함.

- 위와 같은 형태의 쿼리를 입력

  → DB: 1개 이상의 relation을 사용하여 1개의 relation(테이블 형태의 정보)을 반환

SELECT 절

: 쿼리 결과로 원하는 정보를 가진 속성들을 나열

- Selection 연산 + Projection 연산 수행 (Projection : 결과 relation을 가져올 때)
- SQL은 대소문자의 구별이 없음 예) Name = NAME = name
- select는 relation의 중복과 결과 query의 중복 값을 허용함.

  (중복을 제거하는 데 시간이 많이 들어서)

SELECT 절 – 기본 형태

예) *Instructor*의 이름(name)을 찾는 쿼리

select name — 가져오고 싶은 정보 : *instructor*의 이름(name) **[projection 연산 수행]**

from *instructor* — 사용되는 릴레이션의 이름 : *instructor*

where 절이 없음! → 모든 튜플이 where 절에 대하여 true임

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000

(*instructor* 릴레이션)

→

name
Srinivasan
Wu
Mozart
Einstein

(이거 반환)

SELECT 절 – DISTINCT 키워드

- **distinct** 키워드 : query 결과에서 중복을 제거

select distinct name

from *instructor*

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32713	Yurimi	Comp. Sci.	95000
45565	Katz	Comp. Sci.	75000

(*instructor* 릴레이션)

→

dept_name
Comp. Sci.
Music
Physics

(이거 반환)

SELECT 절 – ALL 키워드

- **all** 키워드 : 중복된 결과를 제거 **X** (default가 중복 제거, all 키워드 꼭 안 써도 됨)

select **all** name

from *instructor*

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32713	Yurimi	Comp. Sci.	95000
45565	Katz	Comp. Sci.	75000

(*instructor* 릴레이션)

→

dept_name
Comp. Sci.
Music
Physics
Comp. Sci.
Comp. Sci.

(이거 반환)

SELECT 절 – ASTERISK(*) 키워드

- **Asterisk(*)** 키워드 : 모든 attribute를 의미
- 속성의 이름을 나열하는 대신, *을 사용하면 모든 속성을 나타낼 수 있음

```
select    *
from      instructor
```

select * from *instructor*을 실행하면 *instructor* 릴레이션의 모든 속성들이 출력됨

- Select 문에서는 +, -, *, / 등의 사칙연산을 포함하는 산술 표현을 쓸 수 있음

```
select    ID, name, salary/12
from      instructor
```

쿼리의 결과로는 salary를 12로 나눈 값이 보여짐

기존의 *instructor* 릴레이션에는 영향 X

WHERE 절

: 쿼리의 결과로 나올 데이터의 조건

- from 절의 relation에서 조건에 만족하는 행 만을 추출하여 가져옴

  (관계대수 연산의 selection 연산과 같은 동작 수행 = relation에서 tuple을 select)

- SQL은 where 절에서 logical connectives(조건 충족 여부, and or not)를 사용함

  비교한 결과는 logical connectives를 사용하여 합칠 수 있음.

- SQL은 날짜, 시간과 같은 특수한 타입 뿐만 아니라 문자열이나 산술 표현을 비교하기 위한 비교연산자들도 사용 가능.

WHERE 절 – 예시

예) 컴퓨터 학과에서 salary가 7만 이상인 교수님들 찾기

select name

from *instructor*

Logical connective

where dept_name = 'Comp. Sci.' **and** salary>=70000

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32713	Yurimi	Comp. Sci.	95000
45565	Katz	Comp. Sci.	75000

(*instructor* 릴레이션)

→

name
Yurimi
Katz

(이거 반환)

FROM 절

- 쿼리를 수행하기 위해 접근해야 하는 relation의 이름을 나열
- relation들은 콤마로 구분, 나열된 relation들로 Cartesian product 연산 수행
- 결과 relation은 from 절에 나열된 모든 relation의 속성들을 다 가짐

FROM 절 – 예시(1)

예) *Instructor* X *teaches* (*instructor*와 *teaches*에 대하여 Cartesian product 연산 수행)

select *

from *instructor, teaches*

결과=두 relation이 가진 tuple들의 페어들이 만들어짐

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000

(*instructor* 릴레이션)

ID	course_id	semester	year
10101	CS-101	Fall	2009
10101	CS-315	Spring	2010
15151	MU-199	Spring	2010
22222	PHY-101	Spring	2020

(*teaches* 릴레이션)

FROM 절 – 예시(2)

inst.ID	name	dept_name	salary	*teaches*.ID	course_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	Spring	2020
15151	Mozart	Music	40000	10101	CS-101	Fall	2009
15151	Mozart	Music	40000	10101	CS-315	Spring	2010
15151	Mozart	Music	40000	15151	MU-199	Spring	2010
15151	Mozart	Music	40000	22222	PHY-101	Spring	2020
22222	Einstein	Physics	95000	10101	CS-101	Fall	2009
22222	Einstein	Physics	95000	10101	CS-315	Spring	2010
22222	Einstein	Physics	95000	15151	MU-199	Spring	2010
22222	Einstein	Physics	95000	22222	PHY-101	Spring	2020

(*instructor* X *teaches*)

JOINS (1)

: 두 개 이상의 relation을 함께 사용해서 원하는 데이터를 얻는 방법 (in SQL)

(두 relation의 튜플을 서로 관계 있는 정보를 가진 튜플끼리 매칭시켜줌)

예) 컴퓨터학과에서 개설한 각 수업의 ID, semester, year, title을 찾는 쿼리

```
select    section.course_id, semester, year, title
from      section, course
where     section.course_id = course.course_id and
          dept_name = 'Comp. Sci.'
```

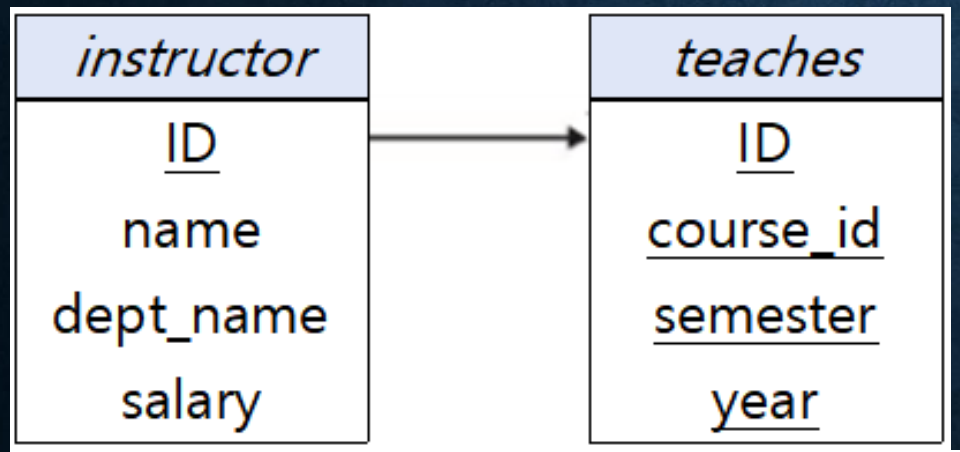
JOINS (2)

예) 대학에서 수업하는 모든 교수님의 이름과 과목 ID(course_id)를 찾는 쿼리

select name, course_id — 결과로 name과 course_id를 출력함

from *instructor, teaches*

where *instructor*.ID = *teaches*.ID — 모든 페어 중 의미 있는 튜플들을 고르려고



// ID 속성 = *instructor*에서는 primary key / *teaches*에서는 foreign key임