3. SQL

Natural Join, Rename Operation, String Operations

Generated by 이유림

참고 강의 : https://www.edwith.org/database/lecture/27855/

NATURAL JOIN

- **Where** 절을 사용하지 않고 릴레이션을 Join하는 방법
- 두 relation에 공통으로 있는 속성들의 값 비교 후, 공통 속성의 값이 같은 튜플끼리만 결합
- Input = 2개의 relation / Output = 1개의 relation
- 두 relation에서 같은 이름, 같은 도메인을 가진(호환 가능한) 공통 속성들만 비교함.

NATURAL JOIN – 예시 (1)

select *

from *instructor* **natural join** *teaches* ;

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)

NATURAL JOIN – 예시 (1)

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
15151	Mozart	Music	40000	15151	MU-199	Spring	2010
22222	Einstein	Physics	95000	22222	PHY-101	Spring	2020

(*instructor* **natural join** *teaches*)

공통으로 있는 속성(ID)에 대해서는 값이 같은지 확인 후, 같다면 딱 한 번만 써줌.

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

(*instructor* **natural join** *teaches*)

← 최종 결과

NATURAL JOIN – 예시 (2)

예) 교수님들의 이름과 그 교수님들이 가르치는 과목의 course_id를 찾는 예제

방법 1) **from**과 **where 절** 사용

```
select      name, course_id
from        instructor, teaches
where       instructor.ID = teaches.ID ;
```

방법 2) **natural join** 사용

```
select      name, course_id
from        instructor natural join teaches ;
```

NATURAL JOIN 유의사항 (1)

1. 속성의 이름은 같지만, 다른 의미로 사용되는 경우

   예) attribute의 이름 = name일 때, *student*에서는 학생 이름 *instructor*에서는 교수님 이름

2. 모든 공통 속성들을 비교함!

   A, B라는 두 개의 속성이 두 relation의 공통 속성인 경우, 한 속성만 비교하고 싶을 때도

   natural join은 2개의 공통 속성의 값을 모두 비교함.

NATURAL JOIN 유의사항 (2)

잘못된 예시

select name, title

from *instructor* **natural join** *teaches* **natural join** *course* ;

쿼리 수행 결과

1. *Instructor* **natural join** *teaches* → 하나의 relation 만듦
2. *(1의 결과 relation)* **natural join** *course* → course_id와 dept_name의 값을 비교

   (but, 2의 두 릴레이션에서 dept_name은 서로 다른 의미임.)

위와 같이 natural join을 하게 되면, 필요량보다 더 많은 조건 비교, 원래 원했던 결과와 다른 결과를 얻게 됨.

NATURAL JOIN 유의사항 (3)

올바른 예시 1

select name, title

from *instructor* **natural join** *teaches, course*

where *teaches.*course_id = *course*.course_id ;

원했던 결과를 얻기 위해서는

1. *Instructor* **natural join** *teaches* → *instructor*가 가르치는 과목에 대한 정보 얻음.
2. *(1의 결과 릴레이션)* X *course*
3. **where 절**에 원하는 튜플을 골라내기 위한 조건 적기

→ course_id가 같은 튜플만 매칭시켜 골라내면 원하는 결과를 얻을 수 있음.

NATURAL JOIN 유의사항 (4)

올바른 예시 2.

select name, title

from (*instructor* **natural join** *teaches*) **join** *course* **using**(course_id) ;

위에서 보았듯이,

1. **join using** 사용 → 튜플 연결 시 공통 속성 중 원하는 속성 선택 가능.
2. **join**할 때 사용하길 원하는 속성을 **using** 다음에 써 주기
3. **natural join**은 장점도 있지만, 예상과 다르게 수행되는 경우가 있으므로 **from, where** 절을 사용하는 것이 더 좋을 때가 있음.

RENAME OPERATION

: 릴레이션이나 속성의 이름을 **rename**할 수 있는 연산

- 기존의 이름 **as** 새 이름
- 릴레이션을 **rename** 한다 = 해당 릴레이션 복사 → 새로운 이름을 붙임
- **as** 라는 키워드는 선택적임. (*instructor* **as** *T* == *instructor T*)
  (**Oracle**에서는 **as** 절을 반드시 생략해야 함!)

RENAME OPERATION – ATTRIBUTE

1. 두 릴레이션이 동일한 이름의 속성을 가지는 경우

: 속성의 이름이 결과에 중복으로 나타남.

(→ 어떤 릴레이션의 속성 이름인지 알 수 없기 때문에 **as**절을 이용하여 이름을 바꿈)

2. 속성을 이용한 산술식을 사용하는 경우

select ID, name, salary/12 **as** monthly_salary

from *instructor*

: 산술식의 결과로 생긴 **attribute**는 따로 이름이 없어서 새 이름이 필요함.

RENAME OPERATION – RELATION

1. 릴레이션의 이름이 길 경우

   :편의를 위해 짧은 이름으로 바꾸어 사용.

2. 릴레이션의 튜플을 비교할 때, 튜플이 어떤 릴레이션의 것인지 모호한 경우

   예) 같은 릴레이션의 튜플 비교

   컴퓨터학과의 어느 교수님(비교 대상)보다 높은 **salary**를 갖는 모든 교수님의 이름 찾기

   select distinct *T.*name

   from *instructor* **as** *T, instructor* **as** *S*

   where *T.*salary > *S.*salary **and** *S.*dept_name = 'Comp. Sci.'

STRING OPERATIONS

: String-matching operator (문자열 비교를 위한 것)

- 사용
  1. 데이터베이스에서 찾고자 하는 데이터의 정확한 단어를 모를 때
  2. 특정 단어를 포함하는 단어를 찾고자 할 때
- **like** : 문자열 비교
- **%** : 한 개 이상의 문자를 의미함 (%가 들어간 부분에는 한 개 이상의 문자열이 들어갈 수 있음)
- _ : 한 개의 문자를 의미함 (_가 들어간 부분에는 하나의 문자가 들어갈 수 있음)

STRING OPERATION – 예시 (1)

예) “~~”를 포함한 문자열 찾기

name의 substring으로 “dar”을 갖는 모든 instructor를 찾는 예제

select name

from *instructor*

where name **like** ‘%dar%’ dar을 포함한 문자열

STRING OPERATION – 예시 (2)

예) 예약어(%, _문자와 같이 특정한 의미를 가진 문자)가 문장에 포함될 경우

"100%"라는 문자열을 찾는 예제

like '100\%' **escape** '\'

%를 문자 자체로 사용할 수 있음

추가 예시)

like 'ab\%cd%' **escape** '\'

ab%cd 라는 문자열 뒤에 문자열 더 들어옴

like 'ab\\cd%' **escape** '\'

ab₩cd 라는 문자열 뒤에 문자열 더 들어옴

참고) **not like** : 같지 않은 문자열

+ mysql을 사용한다면, 'https://dev.mysql.com/doc/refman/5.7/en/string-literals.html'에서 더 자세히 찾아볼 수 있음.

STRING OPERATIONS 2

- 패턴은 대소문자를 구분함.
- 예시

  1. 'Intro&' = "Intro"로 시작하는 문자열
  2. '%Comp%' = "Comp"가 포함된 문자열
  3. '_ _ _' (_3개 연속) = 3개의 문자로 이루어진 문자열
  4. '_ _ _%' = 3개 이상의 문자로 이루어진 문자열

- 기타 SQL이 제공하는 string operation

  : || (concatenation, 두 문자열을 결합하는 연산),

  대문자에서 소문자 변경, 문자열 길이, substring 추출 연산