3. SQL

Null Value, Aggregate Functions

Generated by 이유림

참고 강의 : https://www.edwith.org/database/lecture/27857/

NULL VALUES (1)

- 튜플은 속성값으로 null 값을 가질 수 있음 (데이터 타입과 관계 X)
- **null** : 모르거나 존재하지 않는 값
- 산술, 비교, 집합 연산을 포함한 relational operation을 할 때 특별한 처리 필요
- null 값을 포함하는 연산의 결과는 항상 null

  예 1) 5 + *null* returns *null*

  예 2) 릴레이션 r에 A라는 속성이 있고, r.A가 *null*일 때

  r.A + 5 returns *null*

NULL VALUES (2)

- **Where 절**의 조건에 **is null**이 있으면, null 값을 가지는지 체크함
- **is null** : 제약조건을 위반하는지 알기 위해 유용하게 사용

  예) *instructor*에서 salary 값이 null인 *instructor* 내의 모든 튜플 찾기

```
select   name
from     instructor
where    salary is null      // salary가 null인지 아닌지 판별
                             // ( salary가 null이면 true )
```

UNKNOWN

- 논리연산의 결과는 true나 false로 표시되지만, null이 포함될 경우 비교의 결과는 unknown으로 처리함. (true인지 false인지 알 수 없기 때문)

  예) 5 < *null* = unknown

  null <> *null* = unknown

  null = *null* = unknown

- **where** 절의 결과가 unknown이면 결과를 false로 처리함

  (where 절 결과가 false나 unknown이면 그 튜플은 결과에 포함 **X**)

VALUED LOGIC

- 3가지 부울 연산 (OR, AND, NOT)

1. OR : (unknown **or true**) = **true**
   (unknown **or** false) = unknown
   (unknown **or** unknown) = unknown
2. AND: (unknown **and** true) = unknown
   (unknown **and false**) = **false**
   (unknown **and** unknown) = unknown
3. NOT: (**not unknown**) = **unknown**
4. "*P* **is unknown**" : precidate P가 unknown인지 아닌지 판단 (unknown이면 true)

AGGREGATE FUNCTIONS

- **Aggregate functions** (집계함수) – 많이 사용됨

  : 입력 = column들의 멀티셋 / 리턴 = 연산의 결과값 (1개)

- 집계 함수 종류

  - **avg** : 멀티셋 내 속성 값들의 평균 (숫자 입력만 가능)
  - **min** : 멀티셋 내 속성 값들 중 최솟값
  - **max** : 멀티셋 내 속성 값들 중 최댓값
  - **sum** : 멀티셋 내 속성 값들의 합계 (숫자 입력만 가능)
  - **count** : 멀티셋 내 속성 값들의 개수

- where 절 사용 불가 / select, having 절 사용 가능

AGGREGATE FUNCTIONS – 예시 1

예) 컴퓨터공학과 교수님들의 평균 급여 구하기

```
select   avg (salary)  as avg_salary      // avg의 결과는 1개의 릴레이션 (1속성, 1튜플)
from     instructor
where    dept_name='Comp. Sci.' ;
```

평균값을 구할 때는 중복을 제거하지 않음!

AGGREGATE FUNCTIONS – 예시 2, 3

예) 2010년 봄학기 수업을 하는 교수님의 수 구하기

```
select  count (distinct ID)        // distinct 사용 : 중복 제거 (한 교수님이 여러 과목 개설)
from    teaches
where   semester='Spring' and year=2010 ;
```

예) *course* 릴레이션 튜플들의 개수 구하기

```
select  count (*)          // count (*) : 릴레이션에 몇 개의 튜플이 있는지 확인
from    course ;
```

(뒷장에 계속)

AGGREGATE FUNCTIONS – 예시 2, 3

- 예시 2, 3의 **count** 함수 차이

  예시 2의 **count** = 한 속성에 대한 속성값의 개수 세기

  vs

  예시 3의 **count** = 릴레이션의 튜플 개수 세기

- **count (*)** : 튜플들의 개수, distinct 키워드와 함께 사용X
- **max, min**에서는 distinct 사용O, 근데 결과에 변화 거의 없음.

GROUP BY

- 테이블에서 특정 속성의 값이 같은 튜플들을 모아 그룹을 만들고, 그룹 별 검색에 사용

- **group by ~** : group by 뒤에 그룹을 나눌 때 기준이 되는 속성을 적음

- 여러 튜플 집합에 대해 집계함수를 사용하고 싶을 때 사용

GROUP BY – 예시 1

예) 각 학과별로 교수님의 평균 salary 구하기 (→ 학과별로 avg 함수를 사용해야 함)

```
select   dept_name, avg (salary)        // 3. salary의 평균 구하기
from     instructor                     // 1
group by dept_name ;                    // 2. 같은 학과끼리 묶기
```

쿼리 결과)

1. 각 학과별로 묶어서 그룹 만듦 (**group by** dept_name)
2. 그룹(학과)별 salary의 평균값 계산

(group by 절이 없다면, 전체 릴레이션을 하나의 그룹으로 간주함)

GROUP BY – 예시 1 결과

- 원래 avg 연산을 하면 결과는 단 하나인데, group by를 함으로써 학과별로 평균값을 갖는 릴레이션을 결과로 얻음.

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
15151	Mozart	Music	40000
22222	Einstein	Music	95000
32713	Yurimi	Comp. Sci.	85000
45565	Katz	Comp. Sci.	75000
76766	Crick	Biology	72000

instructor 릴레이션

1

ID	name	dept_name	salary
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
32713	Yurimi	Comp. Sci.	85000
45565	Katz	Comp. Sci.	75000
15151	Mozart	Music	40000
22222	Einstein	Music	95000

group by dept_name

2

3

dept_name	avg_salary
Biology	72000
Comp. Sci.	75000
Music	67500

쿼리 결과

GROUP BY – 예시 2

잘못된 예)

select dept_name, ID, **avg** (salary) // group by 에 없는 속성(ID)이 나타남

from *instructor*

group by dept_name ;

- 그룹을 나누는 기준(group by ~)외의 속성은 select 절에 사용할 수 없음. (Group by 절에 존재하지 않는 속성들이 select 절에 나타난다면, 반드시 집계함수 안에 나타나야 함.)

그런데, 위의 예시에서는 ID가 select 절에 그대로 나타났으므로 잘못된 쿼리! (salary는 Ok)

오류 없애기 → ID를 group by 에 적어주거나 select절에서 ID 없애기

HAVING 절

: Group by로 그룹화된 것에 조건 적용　　유의) where 절 = 각 튜플에 조건 적용

예) 교수님의 평균 급여가 42,000을 넘는 학과 찾기

```
select   dept_name, avg (salary)          // 4. 학과들 중 2의 조건 충족하는 것 찾기
from     instructor                       // 1
group by dept_name                        // 2. 같은 과끼리 그룹화
having avg(salary) > 42,000               // 3. 그룹화된 상태에서 avg 사용
```

WHERE + HAVING 절 - 예시

예) 2009년에 개설된 각 수업 분반에 대하여 2명 이상인 분반에 등록한 학생들의 평균과 전체 학점 구하기

```
select   course_id, semester, year, sec_id, avg (tot cred)        // 5
from     takes natural join student                               // 1. natural join
where    year=2009                                                // 2
group by course_id, semester, year, sec_id                        // 3. 그룹화
having count (ID)>=2 ;                                            // 4
```

순서)

student **natural join** *takes* → where → group by → having 조건

내용 요약

- 집계함수 + **group by** + **having** 포함하는 쿼리

```
SELECT   (ALL|DISTINCT)  속성        // 6
FROM     릴레이션                    // 1. 릴레이션을 가져옴
(WHERE   절)                         // 2. 1의 결과 릴레이션에 조건 적용
(GROUP BY 속성)                      // 3. where절에 만족하는 튜플 그룹화
(HAVING 조건)                        // 4. 각각의 그룹에 대해 조건 적용
(ORDERED BY 속성 (ASC|DESC) );       // 5. 출력하는 순서 결정
```

( ) : 쿼리에서 선택적인 요소

NULL이 있는 집계함수

- Null 값은 집계함수의 처리를 복잡하게 만듦

- NULL이 포함된 집계함수

  - 일부 null : null 값을 무시하고 계산 (예외: **count (*)**)

  - 모두 null : 결과로 null 값 리턴 (예외: **count (*)**는 0 반환)