3. SQL

With 절, Scalar Subquery, Database의 수정

Generated by 이유림

참고 강의 : https://www.edwith.org/database/lecture/27859/

WITH 절

- 뷰(view) : 다른 릴레이션을 기반으로 만들어지는 가상의 릴레이션

  (실제로 저장 X, 논리적 존재)

- WITH 절 : 임시 뷰 정의 (임시 뷰는 WITH절이 있는 쿼리에서만 유효함)

- WITH 절은 복잡한 쿼리 작성에 유용함
- 이해하기 쉽고 명확한 표현 가능
- 대부분의 DB 시스템에서 지원함

WITH 절 - 예시

예) 가장 많은 예산을 갖는 학과 구하기

```
WITH    max_budget (value) AS ( SELECT MAX (budget) // max_budget 릴레이션 정의
                                FROM   department )
SELECT  dept_name
FROM    department, max_budget
WHERE   department.budget= max_budget.value ; // 최대 예산값과 예산값을 비교해서 찾기
```

1. WITH 절을 통해 최대 예산을 속성값으로 갖는 *max_budget* 릴레이션을 정의함
2. WITH 절에서 만들어진 *max_budget* 릴레이션은 임시뷰로, 위의 쿼리 내에서만 유효함

WITH 절 – 복잡한 예시

예) 가장 많은 예산을 갖는 학과 구하기

```
WITH   dept_total (dept_name, value) AS (SELECT dept_name, SUM (salalry)
                                         FROM instructor
                                         GROUP BY dept_name ),
       dept_total_avg (value) AS (SELECT AVG (value)
                                  FROM dept_total )
SELECT dept_name
FROM   dept_total, dept_total_avg
WHERE  dept_total.value > dept_total_avg.value
```

임시 릴레이션 : *dept_name* (학과별 **salary** 합), *dept_total_avg* (학과별 **salary** 합의 평균)

SCALAR SUBQUERY

- **Scalar Subquery** : 1개의 값이 필요한 곳에 사용되는 서브쿼리

- 1개의 속성을 갖는 1개의 튜플 반환
- **Scalar subquery**가 하나 이상의 튜플을 반환하면 **runtime error**
- **Scalar subquery**는 딱 1개의 값을 갖지만, 쿼리 타입은 언제나 릴레이션임

SCALAR SUBQUERY – 예시 1

예) 모든 학과와 그 학과 교수들의 수 나열하기

```
SELECT  dept_name, (SELECT COUNT (*)
                    FROM   instructor
                    WHERE  department.dept_name = instructor.dept_name )
                  AS num_instructors // 서브쿼리결과=해당 학과 교수들의 수 (1개의 값)
FROM    department ;
```

SCALAR SUBQUERY – 예시 2

예) 소속된 학과의 예산이 자신의 salary 10배보다 큰 교수들의 명단 가져오기

```
SELECT  name
FROM    instructor
WHERE   salary*10 > (SELECT budget        // 서브쿼리결과=해당 교수의 소속 학과 예산
                     FROM   department
                     WHERE  department.dept_name = instructor.dept_name)
```

DATABASE 수정

- Deletion : 튜플 삭제
- Insertion : 튜플 삽입
- Updates : 튜플의 값 수정

DELETION

- 주어진 릴레이션으로부터 튜플 삭제하기
- 사용법)

```
DELETE FROM 릴레이션   // 릴레이션 = 어떤 릴레이션으로부터 삭제할지
WHERE 조건   // 조건 = 어떤 튜플을 삭제할지
```

- 튜플 전체 삭제 : O, 특정 속성의 값만 삭제: X
- 한 릴레이션에만 작동 (보통은 여러 릴레이션 삭제 시 DELETE를 일일이 사용해야 함)
- **DELETE**의 **WHERE** 절에 **subquery** 사용 가능
- **WHERE 절**이 없으면 릴레이션의 모든 튜플 삭제

DELETION - 예시

예) 교수님 모두 삭제하기

```
DELETE FROM instructor // 릴레이션의 모든 튜플 삭제
```

예) Finance department의 모든 교수님 삭제하기

```
DELETE FROM instructor
  WHERE dept_name='Finance' ;
```

예) Watson 건물에 있는 학과의 모든 교수님 삭제하기

```
DELETE FROM instructor // 메인쿼리=해당 학과의 instructor 튜플 모두 삭제
  WHERE dept_name IN (SELECT dept_name // 서브쿼리=Watson 건물의 학과
                      FROM department
                      WHERE building='Watson' ) ;
```

DELETION - 예시

예) 대학의 평균 급여보다 급여가 낮은 교수님의 레코드 모두 삭제

```
DELETE FROM instructor // 메인쿼리결과=평균급여보다 낮은 급여의 교수님 삭제
  WHERE salary < (SELECT AVG (salary)    // 서브쿼리결과=평균 급여
                  FROM instructor ) ;
```

유의) 튜플의 삭제가 실행되기 전에 조건절의 테스트를 끝내야 함

: salary 값이 계속 바뀌면서 salary의 평균값이 바뀌면 삭제 범위가 달라질 수 있음

→ 해결과정 1) 평균 salary 계산

2) 삭제할 모든 튜플 찾기

3) 튜플 제거

INSERTION – 방법 1

방법 1. 삽입하려는 튜플 명시

- 삽입될 튜플의 속성은 해당 릴레이션 속성들의 도메인에 속해야 함
- **INTO**의 나열한 속성 이름과 **VALUES**의 속성값은 순서대로 일대일 대응해야 함
- 속성값의 타입이 문자, 날짜 타입인 경우, 작은 따옴표(‘ ’)로 표현
- 사용법)

INSERTION INTO *릴레이션* (속성 이름 나열)
VALUES (속성 값들의 리스트) ;

INSERTION – 방법 1 예시

예) *course* 릴레이션에 새로운 튜플 넣기

```
INSERTION INTO course ( course_id, title, dept_name, credits )
   VALUES ( 'CS-437', 'Database Systems', 'Comp. Sci.', 4 ) ;
```

```
INSERTION INTO course
   VALUES ( 'CS-437', 'Database Systems', 'Comp. Sci.', 4 ) ;
```

위의 두 개 쿼리는 동일함

예) *student* 릴레이션 tot_creds에 null 값을 갖는 새로운 튜플 삽입

```
INSERTION INTO student
   VALUES ( '3003', 'Green', 'Finance', null ) ; // 4번째 속성이 NOT NULL이 아닌 경우만
```

INSERTION – 방법 2

방법 2. 쿼리 작성 → 삽입될 튜플의 집합 생성

사용법) **INSERTION INTO** *릴레이션* (속성 이름 나열)
SELECT WHERE FROM 절 ; = 서브쿼리

INSERTION – 방법 2 예시

예) *student* 릴레이션의 모든 교수님 정보와 tot_creds 값이 0인 새 튜플 삽입

```
INSERTION INTO student
    SELECT ID, name, dept_name, 0
    FROM instructor
```

과정)

1. *instructor* 릴레이션에 있는 하나의 튜플을 가져오기
2. 그 튜플의 ID, name, dept_name은 그대로 쓰고
3. 마지막속성(tot_creds)에 0 값을 넣어서 *student* 릴레이션에 넣음

주의) **SELECT FROM WHERE 문에서 SELECT 문을 먼저 실행 후, 튜플을 삽입해야 함**

`INSERT INTO table1 SELECT * FROM table1` 에서 table1에 primary key 정의 X → 무한루프

UPDATES

- 튜플의 모든 값을 바꾸지 않고, 튜플 하나의 값을 바꿀 때 사용
- **SET 절**을 사용하여 속성값을 업데이트 함
- **INSERT, DELETE**처럼 쿼리를 사용하여 업데이트할 튜플 선택 가능
- 어떤 튜플을 업데이트할지 정한 후, 업데이트 시작

UPDATES – CASE 문

- 업데이트 순서와 관련된 문제 해결 가능
- 사용법)

```
CASE
        WHEN 조건1 THEN 결과1
        WHEN 조건2 THEN 결과2
        ...
        WHEN 조건n THEN 결과n
        ELSE default결과
END
```

UPDATES 예시

예) 연봉 10만 이상의 교수님은 연봉 3% 인상, 나머지는 5% 인상하기

```
UPDATE instructor
    SET salary = salary * 1.03
    WHERE salary > 100000 ;
UPDATE instructor
    SET salary = salary * 1.05
    WHERE salary <= 100000 ;
```

주의) 첫 번째 업데이트와 두 번째 업데이트의 순서가 바뀌면

어느 튜플이 두 번 업데이트 될 수 있음

UPDATES 예시 – CASE 문

예) 연봉 10만 이상의 교수님은 연봉 3% 인상, 나머지는 5% 인상하기

```
UPDATE instructor
    SET salary = CASE
                    WHEN salary <= 100000 THEN salary * 1.05
                    ELSE salary * 1.03
                 END
```

이전 예시의 업데이트 순서 문제 해결

UPDATES 예시 – SCALAR SUBQUERY

예) 각 학생 튜플의 tot_creds 속성에 해당 학생이 이수한 총 학점 넣기 – 성적: F, null 제외

```
UPDATE student S                    // 메인쿼리=student 릴레이션 업데이트
   SET tot_cred = ( SELECT SUM (credits) // 서브쿼리=총 이수 학점 (Scalar Subquery)
                    FROM takes NATURAL JOIN course
                    WHERE S.ID=takes.ID AND
                          takes.grade <> 'F' AND
                          takes.grade IS NOT NULL ) ;
```

문제) 학생이 수업을 하나도 이수하지 못한 경우, tot_creds이 null 값으로 설정됨

해결) 위의 쿼리에서 **SUM** (credits) 대신, **CASE문** 사용

```
SELECT CASE
            WHEN SUM (credits) IS NOT NULL THEN SUM (credits)
            ELSE 0
       END
```