4. INTERMEDIATE SQL

Transaction, Integrity Constraints

Generated by 이유림

참고 강의 : https://www.edwith.org/database/lecture/27861/

TRANSACTIONS

- Transaction : 작업의 단위, 여러 개의 쿼리로 구성 O
- Atomic (원자성) : 더 이상 쪼갤 수 없음 = 모두 수행 or 모두 수행X
- Isolation (독립성) : 트랜잭션은 동시에 수행되는 다른 트랜잭션에 의해 영향 안 받음
- Consistency (일관성)
- Durability (지속성)

TRANSACTIONS

- SQL문이 시작할 때 “암묵적으로 " 트랜잭션이 시작됨

  But, 끝날 때는 “commit”이나 “rollback” 해야 함 (명령은 생략 가능)

- **Commit work :** DB의 레코드 업데이트 or 트랜잭션의 잠정적 변화 후 DB 저장하는 것

  (모든 사용자가 데이터의 변화 볼 수 O, 업데이트가 영구적으로 반영)

- **Rollback work :** 트랜잭션의 실행 실패 시 수행 취소 → DB를 수행 전의 상태로 복구
- 모든 작업 수행 → commit / 하나라도 수행 X → rollback (모두 수행 안 된 것처럼)
- 대부분 SQL 문은 하나의 트랜잭션으로 간주 (SQL 문 수행 → commit → DB에 반영)
- 여러 개의 SQL문인 경우, auto commit 해제 → 하나의 트랜잭션으로 묶기

INTEGRITY CONSTRAINTS

- **Integrity** (무결성) : 데이터에 결함이 없는 상태, 데이터가 정확히 유지되는 상태
- **Integrity Constraints** (무결성 제약조건) : DB의 수정 등으로부터 데이터의 무결성 일관성 보장, 실수로 인한 데이터의 손상으로부터 DB 보호
- 제약조건 예시

  예) 예금 계좌에 만 달러 이상의 잔고가 있어야 한다는 제약조건

  예) 은행 직원의 시급은 적어도 4달러가 되어야 한다 ~

  예) 고객은 반드시 전화번호를 가지고 있어야 한다 ~

INTEGRITY CONSTRAINTS

- 릴레이션에 대한 무결성 제약조건

1. **NOT NULL** : 모든 속성들은 null 값을 가질 수 있지만, 일부 적합하지 않은 경우 사용

    - NOT NULL 명시 → 해당 속성에는 null 값으로 저장 X, 수정 X

2. **PRIMARY KEY** : 어떤 속성들이 PRIMARY KEY를 이루고 있는지 선언할 때 사용

    - PRIMARY KEY 속성들은 한 릴레이션에서 해당 속성의 값 중복 X , 선언과 동시 NOT NULL 보장

3. **UNIQUE** : UNIQUE로 선언된 속성들이 릴레이션의 CANDIDATE KEY를 구성함

    - CANDIDATE KEY 속성들은 중복 튜플 존재 X, null 값 가질 수 O (명시적 NOT NULL 안하면)

4. **CHECK (P)** : 릴레이션의 모든 튜플이 만족해야 하는 조건 명시 (도메인 제한 가능)

    예) 성별이라는 속성은 여성, 남성 중 하나의 값을 가져야 한다는 제약조건

    CHECK (gender IN (male, female)) 과 같이 표현

REFERENTIAL INTEGRITY (1)

- Referential Integrity (참조 무결성)

  : 두 릴레이션 간 참조의 일관성 보장

  (= 참조하는 릴레이션의 foreign key 값은 참조되는 릴레이션의 값 중 있어야 함)

예) A라는 속성을 갖는 R, S 릴레이션이 있고 A가 S릴레이션의 primary key 일 때, A가 R 릴레이션의 foreign key라면 R에서 나타나는 속성 A의 값은 반드시 S 릴레이션에 존재해야 함

- 참조되는 릴레이션에 "REFERENCES" 키워드 사용하여 표현

REFERENTIAL INTEGRITY - 예시

```
CREATE TABLE department
        (dept_name      VARCHAR (20),
        building        VARCHAR (15),
        budget          NUMERIC (12, 2),
        PRIMARY KEY (dept_name) ) ;
```

```
CREATE TABLE instructor
        (ID             VARCHAR (5),
        name            VARCHAR (20) NOT NULL,
        dept_name       VARCHAR (20),
        salary          NUMERIC (8, 2),
        PRIMARY KEY (ID),
```

```
CREATE TABLE course
        (course_id      VARCHAR (7),
        title           VARCHAR (50),
        dept_name       VARCHAR (20),
        credits         NUMERIC (2, 0),
        PRIMARY KEY (course_id),
        FOREIGN KEY (dept_name) REFERENCES department ) ;
```

```
// 참조 무결성에 의해 department 릴레이션이 만들어진 후 course 릴레이션을 만들어야 함
// 참조되는 릴레이션의 primary key 속성과 참조하는 릴레이션의 foreign key 속성의 도메인은
   같아야 함. (이름은 같지 않아도 됨)
```

- 삭제 순서 : 참조하는 릴레이션의 튜플 삭제 → 참조되는 릴레이션의 튜플 삭제

REFERENTIAL INTEGRITY (2)

- **Cascading Action :** 참조된 릴레이션에 대한 삭제 또는 갱신이 제약조건을 위반하면, 삭제나 갱신 거부 대신 참조하는 릴레이션의 튜플을 변경하는 것

예) *course* 릴레이션의 **foreign key** 선언과 연관된 *department* 릴레이션 튜플을 삭제하면 참조 무결성 위반, 삭제 거부될 것임. → (삭제거부X) 이 때 **A** 릴레이션의 튜플 삭제 → *department* 릴레이션의 튜플 삭제

```
CREATE TABLE course (
        ...
        dept_name       VARCHAR(20),
        FOREIGN KEY (dept_name) REFERENCES department // department의 튜플 삭제하면
                ON DELETE CASCADE     // 삭제 거부 X, 참조하는 course의 튜플도 삭제함
                ON UPDATE CASCADE,    // 갱신 거부 X, course가 참조하는 속성값도 갱신함
        ...
)
```

REFERENTIAL INTEGRITY (3)

- Cascade의 대안 : ON DELETE +

1. SET NULL : 참조되는 값을 삭제하면, 해당 값을 참조하는 속성을 null로 설정
   (단, 대상 속성들이 null 값을 허용해야 함)
2. SET DEFAULT : 참조되는 값을 삭제하면, 삭제된 값 참조하는 속성값 default로 설정
   (명시적 default 값이 설정 안 된 경우, null이 default)