DBMS

6일차

강사 : 김근형

트랜잭션

◆ 쪼갤 수 없는 업무 처리의 최소 단위

- 데이터베이스의 상태를 변화시키기 위해서 수행하는 작업의 단위
- 질의어(SQL)을 이용하여 데이터베이스를 접근 하는 것을 의미
- 작업의 단위는 SQL 한 문장이 아님
- 작업 단위는 많은 SQL 명령문들을 사람이 정하는 기준에 따라 정하는 것을 의미
- 예시) 게시판

  게시판 사용자는 게시글을 작성하고, 올리기 버튼을 누른다. 그 후에 다시 게시판에 돌아왔을 때, 게시판은 자신의 글이 포함된 게시판을 보게 된다.

  이 상황을 데이터베이스 작업으로 옮기면, 사용자가 올리기 버튼을 눌렀을 시, insert 문을 사용하여 사용자가 입력한 게시글의 데이터를 삽입한다. 그 후에 게시판을 구성할 데이터를 다시 select 하여 최신 정보로 유지. → 현재 작업 단위는 insert 문과 select 문을 합친 것 → 이러한 작업 단위를 하나의 트랜잭션이라 함

  트랜잭션 설계를 잘하는 것이 데이터를 다루는 것에 많은 이점

트랜잭션

◆ 특징

- 원자성
- 일관성
- 독립성
- 지속성

트랜잭션

◆ 원자성 Atomicity

- 트랜잭션이 데이터베이스에 모두 반영되던가, 아니면 전혀 반영되지 않아야 한다는 것
- 트랜잭션은 사람이 설계한 논리적인 작업 단위로서, 일 처리는 작업 단위 별로 이루어 져야 사람이 다루는 데 무리가 없다
- 만약 트랜잭션 단위로 데이터가 처리되지 않는다면, 설계한 사람은 데이터 처리 시스템을 이해하기 힘들 뿐만 아니라, 오작동 했을 시 원인을 찾기가 매우 힘들어질 것이다

트랜잭션

◆ 일관성 Consistency

- 트랜잭션의 작업 처리 결과가 항상 일관성이 있어야 한다는 것
- 트랜잭션이 진행되는 동안에 데이터베이스가 변경 되더라도 업데이트된 데이터베이스로 트랜잭션이 진행되는 것이 아니라
- 처음에 트랜잭션을 진행 하기 위해 참조한 데이터베이스로 진행
- 이렇게 해서 각 사용자가 일관성 있는 데이터를 볼 수 있는 것

트랜잭션

◆ 독립성 Isolation

- 둘 이상의 트랜잭션이 동시에 실행되고 있을 경우 어떤 하나의 트랜잭션이라도, 다른 트랜잭션의 연산에 끼어들 수 없다는 점

- 하나의 특정 트랜잭션이 완료될 때까지, 다른 트랜잭션이 특정 트랜잭션의 결과를 참조할 수 없다

트랜잭션

◆ 지속성(영속성) Durability

- 완료(Commit)된 트랜잭션은 데이터베이스에 영구적으로 보존되어야 함

트랜잭션

◆ 상태

- Active
- Partially Commited
- Commited
- Failed
- Aborted

트랜잭션

◆ 병행제어

- Locking
- Time Stamp

트랜잭션

◆ 연산

- Commit → 하나의 트랜잭션이 성공적으로 끝났고, 데이터베이스가 일관성 있는 상태에 있을 때, 하나의 트랜잭션이 끝났다는 것을 알려주기 위해 사용하는 연산

- Rollback → 하나의 트랜잭션 처리가 비정상적으로 종료되어 트랜잭션의 원자성이 깨진 경우, 트랜잭션을 처음부터 다시 시작하거나 트랜잭션의 부분적으로만 연산이 완료된 결과를 다시 취소 시키는 연산

트랜잭션

◆ TCL

- 트랜잭션을 제어하는 명령
- COMMIT과 ROLLBACK이 있다

트랜잭션

◆ TCL - COMMIT

- 트랜잭션 처리가 정상적으로 종료되어 트랜잭션이 수행한 변경 내용을 데이터베이스에 반영하는 연산
- 내용을 변경한 트랜잭션이 완료되면 그 트랜잭션에 의해 데이터베이스 새롭게 일관된 상태로 변경되며, 이 상태는 시스템 오류가 발생하더라도 취소되지 않는다
- COMMIT 명령어 같은 경우는 데이터베이스에서 자동으로 실행해준다
- 그 예시로 COMMIT 명령어를 따로 입력해주지 않아도 데이터베이스에 값을 저장하면 그대로 반영된다

◆ TCL – ROLLBACK

- 하나의 트랜잭션 처리가 비정상적으로 종료되어 데이터베이스의 일관성이 깨졌을 때, 트랜잭션이 행한 모든 변경 작업을 취소하고 이전 상태로 되돌리는 연산

- ROLLBACK 연산 시 해당 트랜잭션은 받았던 자원과 잠금을 모두 반환하고, 재시작 또는 폐기가 된다

- ROLLBACK 연산 시 마지막 COMMIT 이후의 작업으로 돌아가는데 이 때, 만약 COMMIT을 하지 않았다면 모든 작업이 사라진다.

- 문법

```
ROLLBACK TO 'SAVEPOINT_NAME';
```

트랜잭션

◆ TCL – SAVEPOINT

- 현재의 트랜잭션을 작게 분할하는 명령어
- 저장된 SAVEPOINT는 ROLLBACK TO SAVEPOINT 문을 사용하여 지정한 곳까지 ROLLBACK 할 수 있다
- 여러 개의 SQL문을 수반하는 트랜잭션의 경우, 사용자가 트랜잭션 중간 단계에서 SAVEPOINT를 지정할 수 있다
- SAVEPOINT는 ROLLBACK과 함께 사용하며, 현재 트랜잭션 내의 특정 SAVEPOINT까지 ROLLBACK 할 수 있다
- 문법

```
SAVEPOINT 포인터명;
```

◆ TCL – SAVEPOINT

- 주의 사항
  - SAVEPOINT 후 COMMIT 연산을 하게 된다면 COMMIT 연산 이전에 만든 SAVEPOINT들은 모두 사라진다
  - SAVEPOINT는 여러 개 생성할 수 있다

◆ TCL – START TRANSACTION

- 트랜잭션을 시작할 때 사용하는 명령
- 이 명령어 외에 BEGIN이나 BEGIN WORK를 입력해도 된다

인덱스

◆ 테이블 검색 속도 향상을 위한 자료 구조

- 장점
  - 조회 속도 증가 → 성능 향상
  - 전반적인 시스템의 부하 감소 → 오버헤드 감소
- 단점
  - 인덱스 관리용 저장 공간 필요 → DB의 10%
  - 관리를 위해 추가 작업 필요
  - 잘못 사용시 성능 저하 → 역효과 발생 가능성 존재

키의 종류

◆ 슈퍼키(Super Key)

- 유일성만 만족하는 키

◆ 후보키(Candidate Key)

- 유일성과 최소성을 만족하는 키(기본키 후보)

◆ 대체키(Alternate Key)

- 후보키 중 기본키가 되지 못한 키

◆ 기본키(Primary Key)

◆ 외래키(Foreign Key)

함수적 종속성(Functional Dependency)

◆ 어떤 릴레이션 R이 있을 때 X와 Y를 각각 속성의 부분집합이라고 가정했을 때

- X 값을 알면 Y 값을 바로 식별할 수 있고, X 값에 의해 Y 값이 달라질 때 Y는 X의 함수적 종속이라고 한다.
- 이 경우 X를 결정자, Y를 종속자라고 한다.
- 기호로 표현 : X → Y
- 종속 관계 종류
  - 완전 함수적 종속
  - 부분 함수적 종속
  - 이행 함수적 종속

함수적 종속성

◆ 완전 함수적 종속

- 종속자가 기본키에만 종속
- 기본키가 여러 속성으로 구성되어 있을 경우 기본키를 구성하는 모든 속성이 포함된 기본키의 부분집합에 종속된 경우

함수적 종속성

◆ 부분 함수적 종속

- 테이블(릴레이션)에서 종속자가 기본키가 아니고 기본키를 구성하는 속성 중 일부에 종속
- 기본키가 여러 속성으로 구성되어 있을 경우, 기본키를 구성하는 속성 중 일부만 종속되는 경우

함수적 종속성

◆ 이행 함수적 종속

- 테이블(릴레이션)에서 X, Y, Z라는 3개의 속성이 있을 때, X → Y, Y → Z 란 종속 관계가 있을 경우, X → Z가 성립될 때 이행 함수적 종속

- 즉, X를 알면 Y를 알고 그를 통해 Z를 알 수 있는 경우

정규화

◆ 목표 → 테이블 간의 중복된 데이터를 허용하지 않는 것

- 중복된 데이터를 허용하지 않음으로써 무결성을 유지할 수 있으며
- DB의 저장 용량 역시 줄일 수 있다
- 정규화에는 단계가 있음
    - 제 1 정규화
    - 제 2 정규화
    - 제 3 정규화
    - BCNF 정규화

- 두부이걸다줘?

정규화

◆ 제1 정규화

- 도메인(테이블의 컬럼)이 원자값(하나의 값)을 갖도록 테이블을 분해하는 것
- 아래 테이블(릴레이션)에서 추신수, 박세리는 여러 개의 취미를 가지고 있기 때문에 제 1정규형을 만족하지 못하고 있다
- 제 1정규화하여 분해

이름	취미들
김연아	인터넷
추신수	영화, 음악
박세리	음악, 쇼핑
장미란	음악
박지성	게임

이름	취미
김연아	인터넷
추신수	영화
추신수	음악
박세리	음악
박세리	쇼핑
장미란	음악
박지성	게임

정규화

◆ 제2 정규화

- 부분 함수적 종속 제거
- 제 1정규화가 완료된 테이블에 대해 완전 함수 종속을 만족하도록 테이블을 분해하는 것
- 기본키의 부분집합이 결정자가 되면 안된다는 것

- 기본키가 한 가지 속성으로 구성이 되어있다면, 2 정규화 생략

정규화

◆ 제3 정규화

- 이행 함수적 종속을 없애도록 테이블을 분해하는 것
- A → B / B → C 일 때, A → C가 성립되는 것을 이행 함수적 종속이라 한다

정규화

◆ BCNF 정규화

- 모든 결정자가 후보키가 되도록 테이블을 분해하는 것
- 결정자가 기본키가 아닌 것을 제거 → 테이블 분해

- 4 정규화
  - 다치 종속 제거
- 5 정규화
  - 조인 종속 제거

반정규화

◆ 성능 향상을 위해서 데이터 중복을 허용하고 조인을 줄이는 방법

◆ 조회(select) 속도 향상, 모델의 유연성이 낮아짐

- 이유
  - 정규화 수행 후 성능이 낮아진 경우(속도가 느려진 경우)
  - 다량의 범위를 자주 처리해야 하는 경우
  - 특정 범위 데이터만 자주 처리되는 경우
  - 요약/집계 정보가 자주 요구되는 경우

반정규화

◆ 절차

- 대상 조사 및 검토 : 반정규화 대상을 조사
- 다른 방법 검토 : 클러스터링, 뷰, 인덱스 튜닝, 파티션 등 검토
- 반정규화 수행 : 수행

◆ 기법

- 계산된 속성 추가
- 테이블 수직 분할 : 속성을 기준으로 분할
- 테이블 수평 분할 : 튜플을 기준으로 분할

반정규화

◆ 파티션(Partition)

- 논리적으로 하나의 테이블이지만, 여러 개의 파일에 분산 저장

- Range Partition : 범위 기준
- List Partition : 값 기준
- Hash Partition : 해시 함수 기준
- Composite Partition : 범위 + 해시

반정규화

◆ 테이블 병합

- 1:1 관계 테이블 병합
- 1:N 관계 테이블 병합(많은 양의 데이터 중복 발생)
- 슈퍼 타입과 서브 타입 관계 테이블 병합
  - OneToOne Type : 개별 테이블로 도출
  - Plus Type : 슈퍼 타입과 서브 타입 테이블로 도출
  - Single Type : 하나의 테이블로 도출

데이터 무결성

◆ 데이터의 정확성, 일관성, 유효성이 유지되는 것

- 정확성 : 중복이나 누락이 없는 상태
- 일관성 : 원인과 결과의 의미가 연속적으로 보장되어 변하지 않는 상태
- 유효성 : 데이터가 정확한 값만 입력되도록 하는 것
- 데이터 무결성 제약조건 종류
    - 개체 무결성
    - 참조 무결성
    - 도메인 무결성
    - Null 무결성
    - 고유 무결성
    - 키 무결성
    - 관계 무결성

데이터 무결성

◆ 개체 무결성

- 기본키에는 Null 값이 올 수 없음
- 기본키는 테이블 내에 오직 하나의 값만 존재해야함

◆ 참조 무결성

- 외래 키의 값은 Null 이거나 참조 릴레이션의 기본키의 값과 동일해야함
- 외래 키 속성은 참조할 수 없는 값을 지닐 수 없음
    - 즉, 외래 키 속성 값이 상위 테이블에 반드시 존재하거나 Null이어야 함

데이터 무결성

◆ 도메인 무결성

- 테이블에 존재하는 필드의 무결성을 보장하기 위한 것으로 필드의 타입, Null 값 허용 등에 대한 사항을 정의하고 올바른 데이터가 입력되었는지 확인하는 조건

- 예를 들어서 주민등록번호 필드에 문자가 입력되는 경우엔 도메인 무결성이 깨진 경우

◆ Null 무결성

- 테이블의 특정 속성 값이 Null이 될 수 없게 하는 조건

데이터 무결성

◆ 고유 무결성

- 테이블의 특정 속성에 대해 각 레코드들이 갖는 값들이 서로 달라야 하는 조건

◆ 키 무결성

- 하나의 테이블에는 적어도 하나의 키가 존재해야 하는 조건

◆ 관계 무결성

- 테이블의 어느 한 레코드의 삽입 가능 여부 또는 한 테이블의 레코드들 사이의 관계에 대한 적절성 여부를 지정한 조건

데이터 무결성

◆ 장단점

- 장점
  - 스키마를 정의할 때 일관성 조건을 오직 한 번만 명시
  - 데이터베이스가 갱신될 때 DBMS가 자동적으로 일관성 조건을 검사하므로 응용 프로그램들은 일관성 조건을 검사할 필요가 없음

- 단점
  - 프로그램 작업이 복잡해짐
  - 무결성 제약조건을 반복해서 구현해야 함
  - 무결성 제약조건들 간에 서로 충돌이 발생할 수 있음

SQL 검색 조건 심화

◆ LIKE 조건

- 컬럼에 저장된 문자열 중에서 LIKE 연산자에서 지정한 문자 패턴과 부분적으로 일치하면 참이되는 연산자

- % : 임의의 길이의 문자열% 모든 문자(글자 수 제한 X)
    - %DB : DB로 끝나는 문자열
    - DB% : DB로 시작하는 문자열
    - %DB% : 문자열 내부에 DB가 포함되기만 하면 됨

- _ : 글자 한 글자(글자 수 제한 O)
    - _DB : DB로 끝나는 문자(3글자)
    - DB_ : DB로 시작하는 문자(3글자)
    - _DB_ : DB가 포함된 문자(4글자)
    - 여러 개도 가능 EX) __DB

SQL 검색 조건 심화

◆ LIKE 조건 실습

- https://dev.mysql.com/doc/index-other.html 접속 후 world 다운로드