Chapter 8. 데이터베이스 설계

34강. 데이터베이스 설계

1) 데이터베이스 설계

데이터베이스의 스키마 내에 테이블, 인덱스, 뷰 등의 데이터베이스 객체를 정의하는 것

- '스키마 설계'라 불리기도 함
- 주된 내용은 테이블의 이름이나 열, 자료형 결정

논리명과 물리명

- 물리명: 데이터데이서에서 사용될 이름으로, CREATE TABLE에 지정하는 이름
  - 데이터베이스 규칙에 따라 길이에 제한이 있거나 공백문자를 사용할 수 없는 제약
  - 일상에서 사용하는 단어로 이름 지정에 한계
  - 전통적으로 알파벳을 사용해 이름 지정
  - 변경이 어려움
- 논리명: 테이블의 '설계상의 이름'
  - 해당 테이블을 실제로 부를 때 사용하는 이름
  - 이름은 언제든 바꿀 수 있음
- 물리명과 논리명을 설계도나 정의서에 같이 작성하기도 함

자료형

- 테이블의 열에 자료형 지정
- 수치형→문자열 가능, 문자열 → 수치형 불가능

  → 열의 특성에 따라서 자료형을 신중히 지정
- 데이터 정합성 부분에 제약 기능을 적극적으로 사용
- 비고란에 값의 의미를 적어두는 것도 중요

고정길이와 가변길이

- 문자열 자료형을 지정할 때 저장할 데이터를 고려하여 고정/가변길이 결정
  - 제조번호처럼 자리수가 정해져 있는 경우 → 고정길이
  - 비고란처럼 변동폭이 클 경우 → 가변길이
- VARCHAR형으로 지정할 수 있는 최대 크기는 대략 수천 바이트
  - 큰 파일 저장 시, **LOB(Large Object)**
  - LOB은 인덱스를 지정할 수 없다는 제약

기본키

- 기본키로 지정할 열이 생각나지 않을 때? ⇒ 자동증가 열 사용
  - INSERT 할 경우 번호를 자동으로 증가시켜 저장해주는 기능
  - MySQL의 경우, *AUTO_INCREMENT*
    - INCREMENT로 지정한 열은 PRIMARY KEY/UNIQUE로 유일성 지정 필요

2) ER다이어그램

개체 간의 관계를 표현한 것

- E : Entity, 개체
  - 테이블 또는 뷰를 가르킴
  - ER 다이어그램에서 사각형으로 표현
  - 사각형 안에 개체의 속성(테이블의 열) 표기
  - 기본키부터 기술, 열 이름은 주로 논리명으로 표기
- R: Relationship, 연계
  - 개체와 개체 간의 연계
  - ER 다이어그램에서 선으로 이어 표현
  - **연계 표기 방법**(카디널리티/다중도)
    - 일대일(1:1)
    - 일대다(1:n)

        - 다대다(m:n)
- ER 다이어그램 연계는 데이터베이스에서는 외부참조제약으로 지정되는 경우가 있음

35강. 정규화

데이터베이스의 테이블을 규정된 올바른 형태로 개선해나가는 것

- 데이터베이스의 설계 단계에서 시행
    - 경우에 따라 기존 시스템 재검토 시에도 정규화
- 효율적인 데이터베이스를 설계

1) 정규화

정규화는 단계적으로 실시

2) 제1정규형

> 관계형 데이터베이스의 테이블에는 하나의 셀에 하나의 값만 저장 가능하다!

제1정규화에서는 도메인을 원자값으로 만든다!

- 하나의 셀에 하나의 값만 저장
- 반복되는 부분을 세로(행) 방향으로 늘려나감
- 중복을 제거하는 테이블의 분할도 이뤄짐

  ⇒ 제1정규화에서 테이블 분할과 기본키 지정이 이루어진다

3) 제2정규형

제2정규화에서는 부분 함수종속성을 찾아내어 분할한다!

- 부분 함수종속성: 주식별자가 2개 이상으로 구성된 복합 식별자의 경우, 일반 속성이 주식별자의 일부에만 종속성을 갖는 경우
- 제2정규형은 부분 함수종속성을 제거한 상태
- 주 식별자가 단일 식별자인 경우, 제 2정규화가 필요X

- 주 식별자가 종속적이지 않은 속성을 별도의 테이블로 분리

4) 제3정규형

제3정규화에서는 이행적 함수 종속을 제거한다!

- 이행 함수 종속성: 주 식별자가 아닌 일반 속성 간에 함수 종속성이 존재하는 경우
- 이행함수 종속성이 제거된 상태가 제 3정규형

5) 정규화의 목적

정규화로 데이터 구조를 개선하는 것은 하나의 데이터가 한 곳에 저장되도록 하기 위함이다!

- 기본키는 내부적인 데이터이므로 변경될 일이 거의 없음
- 정규화를 통해 테이블에 대한 인덱스의 재구축 억제 가능

36강. 트랜잭션

1) 트랜잭션

발주처리

if) 주문 발생 시 처리 과정

- 주문 번호 지정
  - 자동 증가 사용 가능, 그렇지 않은 경우에 번호 중 가장 큰 값을 SELECT명령으로 가져와 1을 더하는 처리가 필요 →'MAX + 1'
- 번호를 발행 받고, 해당 번호를 키 삼아 INSERT
  - 주문 테이블에 INSERT 한 번, 주문상품 테이블에는 주문된 상품만큼 INSERT

```
-- 예제 8-1. 발주처리
INSERT INTO 주문 VALUES(4,'2014-03-01',1);
INSERT INTO 주문상품 VALUES(4,'0003',1);
INSERT INTO 주문상품 VALUES(4,'0004',2);
```

- INSERT 명령이 특정 원인으로 인해 에러가 발생했다고 가정
    - 트랜잭션을 사용하지 않는다면 INSERT명령을 일일이 DELETE 해줘야함

2) 롤백과 커밋

몇 단계로 처리를 나누어 SQL을 실행하는 경우에 트랜잭션을 자주 사용

- **롤백(rollback)** : 데이터를 추가하다 에러 발생시, 롤백을 이용해 종료
    - 롤백 시, 트랜잭션 내에서 행해진 모든 변경사항이 없어짐
- **커밋(commit)** : 에러가 발생하지 않는다면 변경사항을 적용하고 트랜잭션 종료

자동커밋

- 트랜잭션을 사용해서 데이터를 추가할 때는 자동커밋을 꺼야함
- mysql클라이언트에서 명령 실행 시, 자동커밋이 켜져 있는 상태
- 자동 커밋을 끄기 위해 명시적으로 트랜잭션 시작을 선언

```
-- 트랜잭션 시작
START TRANSACTION
```

- 트랜잭션 종료 시, 두 가지 방식이 있음

```
-- 트랜잭션 내에서 실행한 명령을 적용한 후 종료
COMMIT
```

```
-- 트랜잭션 내에서 실행한 명령을 파기한 후 종료
ROLLBACK
```

- 예제 8-2. 트랜잭션 내에서의 발주처리

```
START TRANSACTION;
INSERT INTO 주문 VALUES(4,'2014-03-01',1);
INSERT INTO 주문상품 VALUES(4,'0003',1);
```

```
INSERT INTO 주문상품 VALUES(4,'0004',2);
COMMIT;
```

3) 트랜잭션 사용법

- 세트로 실행하고 싶은 SQL 명령을 트랜잭션에서 하나로 묶어 실행
- 데이터베이스 제품에 따라 다른 트랜잭션 시작 명령
  - MySQL : *START TRANSACTION*
  - SQL Server, PostgreSQL : *BEGIN TRANSACTION*
  - Oracle, DB2 : 트랜잭션 시작 명령이 없음
- 자동커밋은 클라이언트 툴의 기능
  - 미들웨어도 데이터베이스 접속 시 대개 자동커밋
  - 데이터베이스 서버에서는 언제나 트랜잭션을 걸 수 있는 상태로 SQL 명령 실행
- 트랜잭션 내에서 DELETE시, ROLLBACK을 통해 실행 취소 가능