8253jang@daum.net
장희정

Oracle과 MySQL의 데이터베이스 구조 비교

Oracle과 MySQL은 모두 RDBMS(Relational Database Management System)지만, 데이터베이스를 관리하는 방식이 다르다. 특히, **데이터베이스(Database) 개념과 사용자(User) 개념**에서 차이가 있다.

1. Oracle의 데이터 저장 구조

1) 개념적인 구조

Oracle 은 **계정(사용자, Schema)** 단위로 객체(테이블, 뷰 등)를 생성하는 구조

- Oracle 에서는 **데이터베이스(Database) = 하나의 인스턴스(Instance)** 로 본다.
- 하나의 Oracle 인스턴스 안에 여러 개의 **사용자(User)** 가 존재할 수 있다.
- 각각의 사용자는 **자신만의 테이블, 인덱스, 뷰 등을 가질 수 있다.** (이를 **Schema** 라고 한다.)
- 기본적으로 다른 사용자의 테이블은 접근할 수 없고 접근하려면 **권한(GRANT)** 을 부여해야 한다.

2) 계정과 데이터베이스의 관계

Oracle 에서는 계정을 생성하면, 그 계정이 곧 **스키마(Schema)** 가 된다.
즉, CREATE USER 계정명 IDENTIFIED BY 비밀번호; 를 실행하면 해당 계정 아래에 테이블을 만들 수 있다.

```
CREATE USER jang IDENTIFIED BY 1234; -- 계정생성
GRANT CONNECT, RESOURCE TO jang; -- 권한부여

-- 계정을 만든 후, 해당 계정으로 접속하면 바로 테이블을 생성
CREATE TABLE employees (

    id NUMBER PRIMARY KEY,

     name VARCHAR2(100)

);
```

이 테이블은 jang 계정(스키마) 안에 저장 된다.
다른 계정이 jang.employees 테이블을 조회하려면 권한을 받아야 한다.

8253jang@daum.net
장희정

2. MySQL의 데이터 저장 구조

1) 개념적인 구조

MySQL 은 **데이터베이스(Database) 단위**로 객체(테이블, 뷰 등)를 관리한다.

- **하나의 MySQL 서버에 여러 개의 데이터베이스(Database)를 생성할 수 있다.**
- MySQL 에서는 **각 데이터베이스가 독립적인 저장 공간을 가진다.**
- 계정(User)은 여러 개의 데이터베이스를 사용할 수 있다.
- 하지만 **데이터베이스 간에 기본적으로 테이블을 공유하지 않는다.** (다른 데이터베이스의 테이블을 사용하려면 데이터베이스명.테이블명 으로 접근해야 한다.)

2) 계정과 데이터베이스의 관계

MySQL 에서는 계정을 만든 후, **데이터베이스를 직접 생성해야 한다.**

```
CREATE DATABASE kosta; -- 데이터베이스 생성

USE kosta; -- 데이터베이스 선택

CREATE TABLE employees (
    id INT PRIMARY KEY,
    name VARCHAR(100)
);
```

kosta 데이터베이스 안에 employees 테이블이 생성된다.
MySQL 에서는 **하나의 계정이 여러 개의 데이터베이스를 가질 수 있다.**
즉, root 계정이 여러 개의 데이터베이스를 생성하고 사용할 수 있다.

```
CREATE DATABASE devops;
USE devops;

CREATE TABLE projects (
    id INT PRIMARY KEY,
    title VARCHAR(200)
);
```

이렇게 하면 **root 계정 안에 kosta 데이터베이스와 devops 데이터베이스가 따로 존재** 한다.
각 데이터베이스는 독립적이므로, 기본적으로 서로의 테이블을 공유하지 않는다.

하지만 **다른 데이터베이스의 테이블을 사용하려면** 데이터베이스명.테이블명 방식으로 접근하면

된다.

SELECT * FROM kosta.employees;
SELECT * FROM devops.projects;

이렇게 하면 devops 데이터베이스를 사용 중이더라도 kosta 데이터베이스의 테이블을 조회할 수 있다.

3. Oracle과 MySQL의 주요 차이점

항목	Oracle	MySQL
데이터베이스 개념	하나의 인스턴스(Instance) = 하나의 데이터베이스	하나의 MySQL 서버에 여러 개의 데이터베이스 가능
계정(User) 개념	계정 = 스키마(Schema), 계정이 테이블을 소유	계정은 여러 개의 데이터베이스를 가질 수 있음
테이블 저장 방식	계정(Schema) 단위로 테이블 관리	데이터베이스 단위로 테이블 관리
다른 테이블 접근 방식	`계정명.테이블명` 형식으로 접근 (권한 필요)	`데이터베이스명.테이블명` 형식으로 접근 가능
기본적인 데이터 공유	계정 간 테이블 공유 불가 (권한 필요)	데이터베이스 간 테이블 공유 기본적으로 불가

결론 및 정리

1. **Oracle 에서는 계정이 곧 스키마(Schema)** 이므로, 계정만 있으면 테이블을 만들고 사용한다.
2. **MySQL 에서는 계정이 여러 개의 데이터베이스를 가질 수 있으며**, 데이터베이스 안에서 테이블을 만든다.
3. **MySQL 에서는 기본적으로 데이터베이스 간 테이블을 공유하지 않지만, 데이터베이스명.테이블명 형식으로 접근이 가능하다.**
4. **Oracle 에서는 다른 계정의 테이블을 사용하려면 명시적인 권한을 부여해야 한다.**

MySQL에는 여러 가지 스토리지 엔진(Storage Engine)이 있지만, 그중에서 가장 많이 사용되는 두 가지가 **InnoDB**와 **MyISAM**있다.
이 두 엔진은 **데이터를 저장하는 방식과 동작 방식이 다르므로**, 어떤 용도로 사용하느냐에 따라 선택해야한다.

8253jang@daum.net
장희정

InnoDB 엔진

특징

- MySQL 의 **기본 스토리지 엔진** (MySQL 5.5 이상부터 기본).
- **트랜잭션(Transaction) 지원** (COMMIT, ROLLBACK, SAVEPOINT 가능).
- **외래 키(Foreign Key) 지원** (데이터 무결성을 보장).
- **클러스터형 인덱스(Clustered Index)** 사용 → **기본 키(PK)가 클러스터형 인덱스 역할을 함**.
- **데이터를 행 단위(Row-Level Locking)로 잠금 → 동시성(Concurrency)에 강함**.
-

장점

- 트랜잭션을 지원하여 **데이터 무결성이 강함**.
- 외래 키 지원으로 **관계형 데이터베이스(RDBMS)로 적합**.
- 행 단위 잠금으로 인해 **다중 사용자 환경에서도 성능이 우수**.
- 충돌 발생 시 자동 복구 기능이 있음.

단점

- MyISAM 보다 **디스크 공간을 더 많이 사용**.
- 트랜잭션을 관리해야 하므로 MyISAM 보다 **속도가 조금 느릴 수 있음** (특히 읽기 작업).
- 전체 테이블 스캔(Full Table Scan)이 많을 경우 성능이 떨어질 수 있음.

사용 사례

- 금융 시스템, 전자상거래(쇼핑몰), 은행, ERP 등 **데이터 무결성이 중요한 애플리케이션**.
- **동시 사용자가 많고, 자주 업데이트가 발생하는 시스템**.

```
CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    name VARCHAR(50),
    salary DECIMAL(10,2),
    dept_id INT,
    FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
) ENGINE=InnoDB;
```

MyISAM 엔진

특징

- MySQL 5.5 이전의 기본 스토리지 엔진.

8253jang@daum.net
장희정

- **트랜잭션(Transaction) 미지원** (COMMIT, ROLLBACK 불가능).
- **외래 키(Foreign Key) 미지원**.
- **비클러스터형 인덱스(Non-Clustered Index)** 사용 → 데이터와 인덱스가 분리 저장됨.
- **테이블 단위(Table-Level Locking)로 잠금** → 하나의 사용자가 데이터를 쓰면 다른 사용자는 대기해야 함.

장점

- 트랜잭션을 관리할 필요가 없어서 **InnoDB 보다 읽기 성능이 빠름**.
- **디스크 공간을 절약할 수 있음** (특히 작은 테이블일 경우).
- **단순한 읽기 중심(SELECT 중심) 애플리케이션에 적합**.

단점

- 트랜잭션을 지원하지 않으므로, 데이터 무결성을 보장할 수 없음.
- 테이블 단위 잠금으로 인해 **동시 사용자가 많을 경우 성능이 급격히 저하됨**.
- 충돌 발생 시 **데이터 복구 기능이 없음** → MyISAM 테이블이 깨지면 복구하기 어려움.

사용 사례

- **읽기(SELECT) 작업이 많은 환경 (예: 블로그, 뉴스 사이트, 로그 데이터 저장 등)**.
- **단순한 데이터베이스 설계** (외래 키 필요 없음, 트랜잭션 필요 없음).

```
CREATE TABLE logs (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    log_message TEXT,
    log_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
) ENGINE=MyISAM;
```

언제 InnoDB를 사용하고, 언제 MyISAM을 사용할까?

InnoDB 를 사용해야 하는 경우

- **트랜잭션이 필요한 경우** (쇼핑몰, 은행, 금융 서비스).
- **외래 키를 사용하여 데이터 무결성이 필요한 경우**.
- **다중 사용자 환경에서 동시성 처리가 필요한 경우**.

MyISAM 을 사용해야 하는 경우

- **트랜잭션이 필요 없고, 읽기 성능이 중요한 경우** (블로그, 뉴스, 검색 데이터 저장).
- **데이터 변경이 거의 없고 SELECT 쿼리가 많은 경우**.
- **데이터 무결성이 중요하지 않은 경우**.

8253jang@daum.net
장희정

MySQL 에서는 **테이블 단위로 엔진을 설정할 수 있으므로**, 같은 데이터베이스(DB) 내에서도 일부 테이블은 InnoDB, 일부 테이블은 MyISAM 으로 설정할 수 있다.

InnoDB 와 MyISAM 을 함께 사용할 때 고려할 점

(1) 외래 키(Foreign Key) 문제

- InnoDB 는 외래 키를 지원하지만, MyISAM 은 지원하지 않는다.
- **InnoDB 테이블이 MyISAM 테이블을 참조(FK)할 수 없다.**
- 즉, **외래 키가 필요한 경우 MyISAM 을 사용하면 안 된다.**

(2) 트랜잭션 문제

- InnoDB 는 COMMIT, ROLLBACK 을 지원하지만, MyISAM 은 지원하지 않는다.
- **트랜잭션이 필요한 경우, 모든 테이블을 InnoDB 로 설정하는 것이 좋다**.

(3) 성능과 데이터 무결성

- **읽기(SELECT)가 많은 테이블**은 MyISAM 이 더 빠를 수 있다.
- 하지만 **데이터 무결성이 중요한 테이블**은 InnoDB 를 사용해야 한다.