

DDL (Data Definition Language)

- 역할: 서버 / 프로젝트의 구조 및 변경 (Table, Index, View 등 구조 변경)
- 주요 명령어

- CREATE: 새로운 DB 객체 생성
- ALTER: 기존 객체의 구조 변경 (컬럼 추가/변경/삭제)
- DROP: 객체와 데이터를 완전히 삭제 (복구 불가)
- TRUNCATE: 테이블의 데이터를 삭제 (구조는 유지, 롤백 불가)
- RENAME: DB 객체 이름 변경
- COMMENT: 객체나 컬럼에 설명(주석) 추가 문서화 용도

· 주요 제약조건

- PRIMARY KEY: 유일 식별자. 중복 불가. NULL 불가
- UNIQUE: 중복된 값을 허용하지 않음.
- NOT NULL: 반드시 값이 있어야 함
- DEFAULT: 입력값이 없을 때 기본값 지정

· 테이블 생성

```
CREATE TABLE 테이블명 (  
    컬럼명1    데이터타입1    제약조건1,  
    ...  
    컬럼명n    데이터타입n    제약조건n  
);
```

· 컬럼 추가/수정/삭제

ALTER TABLE 테이블명

ADD COLUMN 컬럼명 데이터타입 제약조건; // 컬럼 추가

ALTER COLUMN 컬럼명 데이터타입 제약조건; // 컬럼 수정

DROP COLUMN 컬럼명; // 컬럼 삭제

DML (Data Manipulation Language)

- 역할: 행 단위로 데이터를 선택, 삽입, 수정, 삭제하는 데이터 조작
- 주요 명령어
 - SELECT: 테이블에서 원하는 데이터를 조회
 - INSERT: 테이블에 새로운 데이터 추가
 - UPDATE: 테이블의 기존 데이터를 수정
 - DELETE: 테이블에서 특정 데이터를 삭제
 - MERGE (Upsert): 조건에 따라 INSERT 또는 UPDATE를 수행
- 바인드 변수 (:bind parameter)
 - :(콜론)으로 시작
 - SQL 내에 직접 값을 쓰기 않고 외부에서 값을 전달하는 방식
 - 보안(SQL Injection 방지)과 성능 향상을 위해 사용됨
- 데이터 삽입 (INSERT)

```
INSERT INTO 테이블명 (컬럼1, 컬럼2, .....)  
VALUES ( ?, ?, .....); // Jobc Template 바인딩
```

```
INSERT INTO 테이블명 (컬럼1, 컬럼2, .....)  
VALUES (:컬럼1, :컬럼2, .....); // NamedParameter
```

- 데이터 수정 (UPDATE)

```
UPDATE 테이블명  
SET 컬럼1 = :컬럼1, 컬럼2 = :컬럼2, .....  
WHERE 조건;
```

- 데이터 삭제 (DELETE)

```
DELETE FROM 테이블명  
WHERE 조건;
```

- 데이터 조회 (SELECT)

```
SELECT 컬럼1, 컬럼2, .....  
FROM 테이블명  
WHERE 조건;
```

```
SELECT *  
FROM 테이블명  
WHERE 조건;
```

JDBC (Java DataBase Connectivity) 연동

- 정의: 자바에서 DB와 통신하기 위한 표준 API
- 드라이버: 각 DBMS (MySQL, PostgreSQL, Oracle 등) 회사가 제공
- 구성요소
 - DriverManager: 드라이버 등록, DB 연결/생성
 - Connection: DB 연결 세션
 - Statement / PreparedStatement: SQL 실행 객체
 - ResultSet: 조회 결과 집합
- 프로그래밍 기본 흐름
 1. 드라이버 로드
 2. Connection 취득
 3. Statement / PreparedStatement 생성
 4. SQL 실행 (executeQuery, executeUpdate)
 5. ResultSet 처리
 6. 리소스 종료 (close)

스프링의 선택

· Jdbc Template

- 역할: JDBC의 반복 코드 (Connection, Statement, ResultSet, Close 등)를 추상화하고 자동 처리
- 개발자 집중 영역: 개발자는 SQL과 결과 매핑 (RowMapper)만 신경 쓰면 됨
- RowMapper: 쿼리 결과 (ResultSet)의 한 행을 Java 객체로 변환
- queryForObject(): SQL 실행 결과에서 단일 객체 (1 row)를 매핑하여 반환

· NameParameterJdbcTemplate

- 특징: 파라미터를 이름 기반으로 전환해서 SQL 가독성이 높음

· 트랜잭션 관리

- @Transactional 어노테이션과 DataSourceTransactionManager

· ORM (JPA / Hibernate)

- JPA: JDBC 위에 추상화 계층을 얹어 객체 중심 개발 지원
- JPQL (Java Persistence Query Language)
 - 대상: 실제 DB 테이블이 아닌 엔티티 객체와 필드
 - 장점: DB 독립적, 객체지향적
 - 기능: @Query 사용, Pageable을 이용한 페이징 처리 자동 지원

· Native SQL

- 대상: 실제 DB 테이블과 컬럼
- 특징: nativeQuery = true 설정. DB 전용 기능 그대로 사용 가능