SQL 학습 노트

테이블 간 관계와 조인 기본 개념

- 조인은 테이블이 정규화되어 나뉘었을 때 관련 데이터를 함께 조회하는 방법
- 일반적으로 테이블 간 기본키(Primary Key)와 외래키(Foreign Key) 관계를 통해 조인
- 실무에서는 단일 테이블만으로 필요한 모든 정보를 가져올 수 없어 조인이 필수적

셀 그레이드(Salary Grade) 테이블과 비동등 조인

- 셀 그레이드 테이블은 급여 범위별 등급을 정의(예: 700-1200 사이는 1등급)
- 다른 테이블과 직접적인 관계(PK-FK)는 없지만 기준값을 저장하는 테이블
- 비동등 조인(Non-EQ Join 또는 THETA Join)을 통해 사원의 급여가 어느 등급에 해당하는지 조회 가능
- 예시: `SELECT e.ename, e.salary, s.grade FROM emp e, salgrade s WHERE e.salary BETWEEN s.min_salary AND s.max_salary`

외부 조인(OUTER JOIN)

- 내부 조인(INNER JOIN)은 조인 조건을 만족하는 행만 반환
- 외부 조인은 조건을 만족하지 않는 행도 결과에 포함시키는 방법
- 유형:
  - LEFT OUTER JOIN: 왼쪽 테이블의 모든 행 포함(부서가 없는 사원도 표시)
  - RIGHT OUTER JOIN: 오른쪽 테이블의 모든 행 포함(사원이 없는 부서도 표시)
  - FULL OUTER JOIN: 양쪽 테이블의 모든 행 포함(MySQL은 지원하지 않아 UNION 사용)
- MySQL은 Oracle 스타일의 (+) 문법을 지원하지 않음

셀프 조인(SELF JOIN)

- 한 테이블이 자기 자신을 참조할 때 사용하는 조인 방식

- 개념적으로 같은 테이블을 두 개로 나누어 생각하면 이해가 쉬움
- 예: 사원-관리자 관계 조회 시 사원 테이블과 관리자 테이블로 구분하여 조인
- 사원 테이블의 관리자 번호(MGR)와 관리자 테이블의 사원 번호(EMPNO)를 연결
- 댓글-대댓글 같은 계층 구조 데이터 조회에도 활용 가능

조인 사용 시 주의사항

- 테이블 간 관계를 명확히 이해하고 적절한 조인 조건 설정 필요
- 필요에 따라 알맞은 조인 유형(내부/외부/셀프) 선택
- 복잡한 비즈니스 로직에서는 여러 테이블 조인이 필요한 경우가 많음
- 조인은 실무에서 가장 많이 사용되는 SQL 기능 중 하나

학습 접근법

- 기초 개념을 지속적으로 학습하는 단계에 있으므로 어려운 부분이 있더라도 멈추지 말고 계속 진행해야 함
- 이전 내용을 완전히 이해하지 못했더라도 현재 수업에 집중하는 것이 중요
- 녹화본에 의존하지 말고 수업 시간에 집중하는 것이 효과적인 학습 방법
- 네이버 부스트코스와 같은 무료 강의 자료도 보충 학습에 활용 가능

서브쿼리 개념

- 서브쿼리: 쿼리문 안에 또 다른 쿼리문이 들어가는 형태
- 하나의 쿼리로 결과를 얻을 수 없을 때 활용 (예: 특정 사원이 속한 부서의 평균 급여 찾기)
- 서브쿼리 결과에 따라 두 가지 유형으로 구분
  - 싱글 로우 서브쿼리: 결과가 한 줄만 반환되는 경우
  - 멀티 로우 서브쿼리: 결과가 여러 줄 반환되는 경우

싱글 로우 서브쿼리

- 비교 연산자(=, <, >) 사용 가능
- 예시:

    - 평균 급여보다 적은 급여를 받는 사원 찾기
    - 가장 먼저 입사한 사원 찾기
    - 부서 이름이 'Sales'인 사원 정보 찾기

멀티 로우 서브쿼리

- 비교 연산자(=, <, >) 직접 사용 불가
- IN, ANY, ALL 같은 특수 연산자 사용 필요
- IN: 서브쿼리 결과 중 하나와 같은지 비교 (OR 조건의 결합)
- 예시: 각 부서별 최대 급여를 받는 사원 찾기

서브쿼리 테스트 방법론

- 복잡한 쿼리는 작은 단위로 나누어 접근
- 안쪽 서브쿼리부터 먼저 테스트하여 제대로 동작하는지 확인
- 테스트 후 전체 쿼리에 통합

액션 아이템

- [ ] 수업 시간에 집중하고, 집중이 어려울 경우 강사에게 DM으로 알리기
- [ ] 복잡한 쿼리를 작은 단위로 나누어 테스트하는 습관 들이기
- [ ] 부스트코스 등의 보충 자료를 활용하여 개념 이해 보완하기

서브쿼리 개념 (심화)

- 하나의 쿼리문으로 원하는 결과를 얻을 수 없을 때 사용
- 쿼리문 안에 또 다른 쿼리가 들어가는 형태
- 예: 스미스라는 사원이 속한 부서의 평균급여를 알고 싶을 때
    - 먼저 스미스의 부서 확인 필요
    - 그 결과값으로 원하는 쿼리 실행

단일행 vs 다중행 서브쿼리

- 단일행 서브쿼리: 결과가 하나의 행으로 나오는 경우

    - 사용 가능 연산자: =, <, >, >=, <=, !=
- 다중행 서브쿼리: 결과가 여러 행으로 나오는 경우
    - 사용 가능 연산자: IN, ANY, ALL

IN, ANY, ALL 연산자

- IN 연산자
    - 등호(=)와 OR의 결합
    - 예: DEPT_NUMBER IN (10, 20, 30) = DEPT_NUMBER = 10 OR DEPT_NUMBER = 20 OR DEPT_NUMBER = 30
- ANY 연산자
    - 모든 비교 연산자와 사용 가능(=, <, >, >=, <=, !=)
    - OR 조건으로 결합
    - 서브쿼리 결과 중 하나라도 조건을 만족하면 TRUE
    - 예: SALARY > ANY (서브쿼리) = 서브쿼리 결과 중 가장 작은 값보다 크면 TRUE
- ALL 연산자
    - 모든 비교 연산자와 사용 가능
    - AND 조건으로 결합
    - 서브쿼리 결과의 모든 값이 조건을 만족해야 TRUE
    - 예: SALARY > ALL (서브쿼리) = 서브쿼리 결과 중 가장 큰 값보다 크면 TRUE

상관 서브쿼리

- 외부 쿼리와 내부 쿼리가 서로 연관된 서브쿼리
- 내부 쿼리가 외부 쿼리의 컬럼을 참조
- 외부 쿼리의 각 행마다 서브쿼리가 실행됨
- 예: 자신이 속한 부서의 평균 급여보다 많이 받는 사원 찾기
    - 각 사원마다 해당 부서의 평균 급여를 계산하여 비교

FROM 절의 서브쿼리

- 서브쿼리를 테이블처럼 사용

- 서브쿼리 결과를 가상 테이블로 만들어 조인 등에 활용
- 예: 각 부서별 최대급여를 받는 사원 찾기
  - 부서별 최대급여 테이블을 서브쿼리로 생성
  - 해당 테이블과 EMP 테이블 조인

실행 방식 이해

- SQL은 레코드 하나씩 순차적으로 처리
- 각 레코드마다 조건을 검사하여 결과에 포함할지 결정
- 상관 서브쿼리는 각 레코드마다 다른 조건으로 서브쿼리 실행

쿼리 실행 이해하기

- SQL 쿼리는 한 번에 실행되는 것이 아니라 한 줄(로우)씩 처리됨
- 서브쿼리를 이해할 때 중요: 메인 쿼리의 각 행마다 서브쿼리가 실행됨
- 서브쿼리 결과가 메인 쿼리의 조건에 맞으면 결과에 포함되고, 아니면 제외됨

서브쿼리 종류와 활용

- **단일 컬럼 서브쿼리**: 하나의 컬럼 값을 반환하여 비교
- **다중 컬럼 서브쿼리**: 여러 컬럼을 동시에 비교 (예: 부서별 최고급여를 받는 사원 찾기)
- **EXISTS 연산자**: 서브쿼리 결과가 존재하는지만 확인 (TRUE/FALSE 반환)
  - `SELECT 1`은 빠른 존재 확인을 위해 사용하는 패턴
  - 예시: 부하직원이 있는 사원 찾기 - `WHERE EXISTS (SELECT 1 FROM emp WHERE mgr = e.empno)`
  - NOT EXISTS로 반대 조건 검색 가능

집합 연산자

- **UNION**: 두 쿼리 결과를 합치고 중복 제거
- **UNION ALL**: 두 쿼리 결과를 합치고 중복 유지
- **INTERSECT**: 두 쿼리 결과의 교집합 (MySQL에서는 직접 구현 필요)
- **MINUS**: 첫 번째 쿼리에서 두 번째 쿼리 결과 제외 (MySQL에서는 NOT IN 등으로 대체)

윈도우 함수

- **RANK()**: 순위 매기기 (동점자 있으면 다음 순위 건너뜀 - 1,2,3,3,5,6)
- **DENSE_RANK()**: 순위 매기기 (동점자 있어도 다음 순위 연속 - 1,2,3,3,4,5)
- **ROW_NUMBER()**: 각 행에 고유 번호 부여
- **PARTITION BY**: 특정 컬럼으로 그룹화하여 윈도우 함수 적용 (예: 부서별 급여 순위)

SQL 명령어 유형

- SQL은 용도에 따라 여러 카테고리로 나뉨
  - DDL(Data Definition Language): 데이터베이스 객체 구조 정의 (CREATE, ALTER, DROP)
  - DML(Data Manipulation Language): 데이터 조작 (INSERT, UPDATE, DELETE)
  - DCL(Data Control Language): 데이터베이스 접근 권한 제어
  - TCL(Transaction Control Language): 트랜잭션 제어

MySQL 데이터 타입

- 숫자 타입
  - INT: 정수 값 (가장 일반적으로 사용)
  - BIGINT: 더 큰 범위의 정수
  - DECIMAL: 소수점이 있는 숫자 (예: DECIMAL(8,2) - 총 8자리 중 소수점 2자리)
- 문자 타입
  - VARCHAR: 가변 길이 문자열 (최대 255자까지 가능)
  - CHAR: 고정 길이 문자열 (검색 속도가 빠르지만 항상 지정된 크기만큼 공간 차지)
  - TEXT: 매우 긴 텍스트를 저장할 때 사용
- 날짜/시간 타입
  - DATE: 연월일만 저장
  - DATETIME: 연월일시분초 저장
  - TIMESTAMP: 시간대(timezone) 정보를 반영하여 저장 (글로벌 서비스에 적합)

테이블 생성과 제약 조건

- 기본 테이블 생성 구문:

```
CREATE TABLE 테이블명 (
    컬럼명 데이터타입 [제약조건],
    ...
);
```

- 주요 제약 조건:
    - PRIMARY KEY: 기본키 설정 (중복 불가, NULL 불가)
    - NOT NULL: NULL 값 허용하지 않음
    - UNIQUE: 중복된 값 허용하지 않음
    - AUTO_INCREMENT: 자동으로 값이 증가하는 기능 (MySQL 특화 기능)
    - DEFAULT: 기본값 설정
    - CHECK: 입력되는 데이터의 조건 체크 (예: 나이는 1~120 사이만 허용)
- 예시 테이블 생성:

```
CREATE TABLE students (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    age INT CHECK (age BETWEEN 1 AND 120),
    email VARCHAR(100) UNIQUE,
    created_date DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

데이터 입력 (INSERT)

- 기본 INSERT 구문:

```
INSERT INTO 테이블명 (컬럼1, 컬럼2, ...) VALUES (값1, 값2, ...);
```

- 컬럼명 생략 시 모든 컬럼에 값을 순서대로 제공해야 함:

```
INSERT INTO 테이블명 VALUES (값1, 값2, ...);
```

- 여러 레코드 한 번에 삽입:

```
INSERT INTO 테이블명 (컬럼1, 컬럼2) VALUES
    (값1, 값2),
    (값3, 값4),
    ...;
```

주요 개념 및 팁

- 제약 조건은 데이터 무결성을 유지하는 중요한 요소
- AUTO_INCREMENT는 MySQL에서 제공하는 편리한 기능 (다른 DBMS에서는 별도 객체 필요)
- VARCHAR와 CHAR의 차이점을 이해하고 적절히 사용해야 함
- 테이블 설계 시 적절한 제약 조건을 설정하는 것이 중요
- DATETIME과 TIMESTAMP의 차이점은 시간대(timezone) 처리 방식에 있음