SQL 활용

KG아이티뱅크 부산학원 / 원종래

1. SQL

Structured Query Language, 구조화된 질의 언어

- DDL
  create, alter, drop, truncate
  Data Definition Lauguage (데이터 정의어)
  데이터베이스 객체의 형식을 정의한다

- DML
  insert, select, update, delete
  Data Manipulation Language (데이터 조작어)
  데이터베이스 객체에 데이터를 삽입/삭제/수정한다

- DCL
  grant, revoke, commit, rollback
  Data Control Language
  데이터베이스 객체의 권한 및 트랜잭션을 제어한다

2. DataType

Oracle DB에서 사용하는 주요 자료형

- CHAR(n) 고정길이문자열, 최대 2000byte, 기본값은 1byte
- VARCHAR2(n) 가변길이문자열, 최대 4000byte, 기본값은 1byte(★)
- LONG 최대 2GB크기의 가변길이 문자열
- CLOB 대용량 텍스트 데이터 타입 (최대 4GB)

- NUMBER 정수 및 실수 저장, 최대 22byte (★)
- DATE 날짜 및 시간, 초까지 입력 가능 (★)
- TIMESTAMP 날짜 및 시간, 밀리초까지 입력가능
- BLOB Binary Large OBject, 바이너리 데이터 저장용

3. 기본 DML 문형

select [컬럼] from [테이블];
-> 테이블에서 컬럼의 값을 모두 불러온다

insert into [테이블] (컬럼…) values (값…);
-> 테이블에 레코드를 추가한다. 컬럼과 값을 순서대로 작성한다

update [테이블] set [컬럼]=[값] where [조건];
-> 테이블에서 조건에 맞는 레코드를 찾아서, 컬럼의 값을 수정한다

delete from [테이블] where [조건];
-> 테이블에서 조건에 맞는 레코드를 삭제한다

4. where

select, update, delete 에서 조건을 명시할 때 사용한다

select * from [테이블] where age >= 20;
-> 테이블에서 age컬럼의 값이 20이상인 모든 레코드를 조회한다

update [테이블] set amount = amount - 1 where idx = 3;
-> 테이블에서 idx가 3번인 레코드의 amoun값을 1 감소시킨다

delete from [테이블] where age >= 20;
-> 테이블에서 age컬럼의 값이 20이상인 모든 레코드를 삭제한다

5. order by

select 에서 정렬 기준 및 순서를 지정한다

select * from [테이블] order by name;
-> 테이블에서 name 컬럼을 기준으로 정렬하여 불러온다 (기본순서는 오름차순)

select * from [테이블] order by age desc;
-> 테이블에서 age 컬럼을 기준으로 내림차순 정렬하여 불러온다

select * from [테이블] order by name, age desc;
-> 테이블에서 name 컬럼을 기준으로 오름차순 정렬하고,
이름이 같다면 나이를 기준으로 내림차순 정렬하여 불러온다

6. 집계 함수

테이블의 여러 레코드에 대하여 합계, 평균, 최대, 최소등을 구한다

sum() : 합계
avg() : 평균
min() : 최소값
max() : 최대값
count() : 개수

괄호 안에는 컬럼이름을 지정할 수 있으며, count의 경우 *을 주로 사용한다

select avg(score) from student;
-> student 테이블에서 score컬럼의 평균을 구한다

! where 절에서는 집계함수를 사용할 수 없다 => 서브 쿼리 사용

7. group by

집계함수를 사용할 때, 특정 그룹으로 결과를 묶어서 분리하여 조회한다

```
select avg(salary) from employees;
```
-> employees 테이블의 모든 레코드의 salary값 평균을 구한다

```
select department_id, avg(salary) from employees
    group by department_id
```
-> employees 테이블의 레코드를 department_id 별로 구분하여 salary값 평균을 구한다

```
select department_id, avg(salary) from employees
    group by department_id
    having department_id is not null;
```
-> employees 테이블의 레코드를 department_id 별로 구분하여 salary값 평균을 구한다 (department_id가 null인 레코드는 집계에서 제외한다)

8. 서브쿼리 (중첩 질의문)

쿼리문 내부에서 쿼리문을 사용한다

```
select * from employees
    where salary >= avg(salary);
```

-> employees 테이블에서 salar가 평균보다 높은 레코드를 조회한다
where 절에서는 집계함수를 사용할 수 없으므로 조회 실패 !!

```
select * from employees
    where salary >= (select avg(salary) from employees);
```

-> 괄호안의 select를 먼저 수행하여 salary의 평균을 구하고,
구해진 값을 이용하여 각 레코드의 salary와 비교를 수행한다

9. 조인 쿼리문

서로 다른 테이블에서 데이터를 함께 불러온다

```
select E.first_name, D.department_name
    from employees E, departments D;
```

-> employees 테이블에서 first_name, departments 테이블에서 department_name 을 불러온다. (각 테이블의 컬럼 개수를 곱하여 2600개 가량이 조회되며 중복이 많다)

```
select
    E.first_name, D.department_name
        from employees E
        join departments D
            on E.department_id = D.department_id;
```

-> employees 테이블에서 first_name, departments 테이블에서 department_name 을 불러온다. (외래키로 지정된 값이 일치하는 레코드 106개만 불러온다.)