02강 데이터 조회하기

SQL의 개념

- DBMS는 기본적으로 질의요청과 결과응답으로 작동한다.
- 이때 질의를 쿼리(query)라고 하는데 이 쿼리를 적절한 문법에 맞게 사용해야한다.
- SQL은 이러한 쿼리를 문법적으로 정리한 것이다.

SQL의 개념

- 기본적인 SQL의 표준은 국제표준화 기구(ISO)이 정립했지만

각 데이터베이스 회사마다 자기 제품에 특화된 SQL을 사용한다.

  - oracle에서는 PL/SQL이라 부르고
  - ms-sql에서는 T-SQL
  - my-sql에서는 그냥 SQL이라고 부른다.

DD

- SQL은 크게 3가지로 분류한다.
  - DDL (데이터 정의어)
  - DML(데이터 조작어)
  - DCL (데이터 제어어)

DDL 종류

- DDL은 테이블이나 뷰등 데이터베이스 개체를 생성 변경 삭제하는 역할을 한다.
- 중요한 점은 트랜젝션을 발생시키지 않는 점이다.
  - 즉 복구가 불가능하다.

  - 생성 : create
  - 변경 : alter
  - 삭제 : drop

DML 종류

- DML은 실제 데이터를 조작하는 데 사용되는 언어이다.
  - 사용되는 대상은 데이터, 즉 테이블의 행이다.
  - DML은 트랜젝션을 발생 시킨다.
  - 그러므로 트랜젝션을 완전히 적용시키기 위해 commit을 해주어야 한다.

  - 생성 : insert
  - 변경 : update
  - 삭제 : delete
  - 조회 : select

DCL 종류

- DCL은 데이터베이스에서 데이터에 대한 액세스를 제어하기 위한 언어
- 주로 사용자 권한에 관련된 기능이 있다

  - 권한 부여 : grant
  - 권한 해제 : revoke

데이터 조회

- 본격적인 내용을 보기에 앞서 예제 코드를 저장하고 진행하도록 하자.

- 테이블 3개와 관련 예제 데이터를 입력후 사용한다

데이터 조회

- 사원 정보를 담은 테이블 정보 : EMPLOYEE

컬럼명	데이터 타입	크기	의미
ENO	number	4	사원번호
ENAME	varchar2	10	사원명
JOB	varchar2	9	업무명
MANAGER	number	4	해당 사원의 상사번호 (EMPLOYEE 테이블의 ENO와 연결됨)
HIREDATE	date		입사일
SALARY	number	7,2	급여
COMMISSION	number	7,2	커미션
DNO	number	2	부서번호 (DEPARTMENT 테이블의 DNO와 연결됨)

데이터 조회

- 부서 정보를 담은 테이블 정보 : DEPARTMENT

컬럼명	데이터 타입	크기	의미
DNO	number	2	부서번호
DNAME	varchar2	14	부서명
LOC	varchar2	13	지역명

데이터 조회

- 급여 정보를 담은 테이블 정보 : SALGRADE

컬럼명	데이터 타입	크기	의미
GRADE	number		급여 등급
LOSAL	number		급여 하한 값
HISAL	number		급여 상한 값

데이터 조회

- 테이블 구조를 살펴보기 위한 쿼리 : DESC
  - DESC 테이블명

```
DESC EMPLOYEE;
```

- 하나의 결과물을 출력하기 위한 가상의 테이블 : DUAL
  - DESC DUAL

```
DESC DUAL;
```

- 오늘 날짜 출력하기

```
SELECT sysdate FROM dual;
```

데이터 조회 - SELECT

- 데이터베이스의 테이블에서 원하는 정보를 추출하는 명령어이다.
- 가장 기본적이고 가장 많이 사용되는 구문이다.

- 기본적인 내용부터 살을 붙여 나가면서 학습하도록 하자

데이터 조회 - SELECT

- 기본 형식

SELECT * FROM 테이블; (*: 모든 컬럼)

```
SELECT * FROM employee;
```

기본적으로 테이블을 작성할 때는 스키마.테이블 형식으로 사용해야 하지만 연결된 스키마와 같은 때는 생략해도 된다.

```
SELECT * FROM HR.employee;
```

데이터 조회 - SELECT

- 특정 컬럼만 조회

```
SELECT ENAME FROM EMPLOYEE;
```

- 여러 컬럼을 조회하고자 할 때는 각 컬럼을 ,로 구분한다.

```
SELECT ENO,ENAME FROM EMPLOYEE;
```

데이터 조회 - SELECT

- 조회 하고자 하는 컬럼이 숫자형 데이터인 경우 산술연산을 추가할 수 있다.(+,-,*,/)

```
SELECT eno,ename,salary, salary*12 FROM employee;
```

- 산술연산시 null인 경우(저장된 데이터가 없는 경우) 연산 결과가 잘못 나올 수 있다.

```
SELECT eno,ename,salary, salary*12, salary*12+commission FROM employee;
```

  - 이때 NULL인 데이터를 어떻게 연산할지 도와주는 함수가 있다(NVL)

```
SELECT eno,ename,salary, salary*12, salary*12+NVL(commission,0) FROM employee;
```

데이터 조회 - SELECT

- 연산을 이용 해당 컬럼명이 연산식으로 표시되는 것을 볼 수 있는데 이때 연산식이 무엇을 의미하는 가를 컬럼에 표시할 수 있다 이것을 별칭(alias)라고 한다.

```
SELECT eno,ename,salary, salary*12, salary*12+NVL(commission,0) AS 연봉 FROM employee;
```

- 다만 별칭을 지을 때 AS 키워드를 붙여야 하지만 생략도 가능하다.

```
SELECT eno,ename,salary, salary*12, salary*12+NVL(commission,0) 연봉 FROM employee;
```

- 별칭 내부에 공백이나 대소문자 또는 특수기호($,_,#등)를 포함해야 한다면 “”를 이용해서 별칭을 부여한다.

```
SELECT eno,ename,salary, salary*12, salary*12+NVL(commission,0) "$연_   _봉#" FROM employee;
```

데이터 조회 - SELECT

- 컬럼의 데이터가 중복되는 경우 중복 데이터를 제거하고 출력하기 위해 DISTINCT를 사용한다.

```
SELECT dno FROM employee;
```

```
SELECT DISTINCT dno FROM employee;
```

데이터 조회 - SELECT

- 데이터 조회 시 특정 조건에 해당하는 데이터만 필요할 수 있다.
- 이때 조건을 제시하는 구문을 WHERE 조건절이라고 한다.

SELECT 컬럼명 FROM 테이블 WHERE 조건;

```
SELECT * FROM EMPLOYEE WHERE salary >= 1500;
```

조건절에 나오는 데이터는 숫자,문자, 날짜등 다양한 데이터가 나올 수 있는데 숫자이외의 데이터는 ‘’로 묶어서 표시한다.

데이터 조회 - SELECT

- 비교연산

- 조건절에서 숫자,문자,날짜등의 크기나 순서를 비교할 때 사용한다.

연산자	의미	예제
=	같다	SELECT eno,ename,salary FROM employee WHERE salary=1500;
>	보다 크다	SELECT eno,ename,salary FROM employee WHERE salary>1500;
<	보다 작다	SELECT eno,ename,salary FROM employee WHERE salary<1500;
>=	보다 크거나 같다	SELECT eno,ename,salary FROM employee WHERE salary>=1500;
<=	보다 작거나 같다	SELECT eno,ename,salary FROM employee WHERE salary<=1500;
<>,!=,^=	다르다(같지 않다)	SELECT eno,ename,salary FROM employee WHERE salary<>1500;

데이터 조회 - SELECT

- 비교연산

- 문자나 날짜를 비교할 때는 ''로 묶어준다

```
SELECT eno,ename,salary FROM EMPLOYEE WHERE ename=SCOTT;

SELECT eno,ename,salary FROM EMPLOYEE WHERE ename='SCOTT';

SELECT eno,ename,salary FROM EMPLOYEE WHERE ename>'SCOTT';

SELECT * FROM EMPLOYEE WHERE hiredate<='1981/01/01';
```

데이터 조회 - SELECT

- 논리연산

조건 여러 개를 조합해서 결과를 얻어야 하는 경우에 사용한 연산

연산자	의미와 예제
AND	두가지 조건 모두 만족해야만 조회할 수 있다. SELECT * FROM employee WHERE dno=10 AND job='MANAGER';
OR	두가지 조건 중 하나이상 만족하더라도 조회할 수 있다. SELECT * FROM employee WHERE dno=10 OR job='MANAGER';
NOT	조건에 만족하지 못하는 데이터만 조회한다. SELECT * FROM employee WHERE NOT dno=10

데이터 조회 - SELECT

- 논리연산
  - AND연산은 두 조건을 모두 만족할 때

조건1	조건2	AND
참	참	참
참	거짓	거짓
거짓	참	거짓
거짓	거짓	거짓

```
SELECT * FROM employee WHERE dno=10 AND job='MANAGER';
```

데이터 조회 - SELECT

- 논리연산
  - OR연산은 두 조건중 하나만 만족해도 참

조건1	조건2	OR
참	참	참
참	거짓	참
거짓	참	참
거짓	거짓	거짓

```
SELECT * FROM employee WHERE dno=10 OR job='MANAGER';
```

데이터 조회 - SELECT

- 논리연산
  - NOT연산은 반대가 되는 논리값을 구한다.

조건	NOT
참	거짓
거짓	참

```
SELECT * FROM employee WHERE NOT dno=10;
```

  - 비슷한 역할을 하는 연산자로 <>가 있다.

```
SELECT * FROM employee WHERE dno<>10;
```

데이터 조회 - SELECT

- 논리연산
  - 예제 : 급여가 1000에서 1500 사이인 사원을 조회하는 예

```
SELECT * FROM employee
WHERE salary>=1000 AND salary<=1500;
```

  - 예제 : 급여가 1000미만이거나 1500초과인 사원을 조회하는 예

```
SELECT * FROM employee
WHERE salary<1000 OR salary>1500;
```

  - 예제 : 커미션이 300이거나 500이거나 1400인 사원을 조회하는 예

```
SELECT * FROM employee
WHERE commission=300 OR commission=500 OR commission=1400;
```

데이터 조회 - SELECT

- BETWEEN~AND 연산
  - 특정 컬럼의 값이 하한값와 상한값 사이의 데이터를 조회하기 위한 연산

  - 예제: 급여가 1000이상 1500이하인 사원을 조회하기

```
SELECT * FROM employee
WHERE salary BETWEEN 1000 AND 1500;
```

  - 예제: 급여가 1000미만이거나 1500 초과인 사원을 조회해보자

  - 예제: 1982년에 입사한 사원을 조회해보자

데이터 조회 - SELECT

- IN 연산

  -BETWEEN 연산이 연속적인 범위를 조회하기 위한 연산이라면
  IN 연산은 불연속적인 값을 조회하기 위한 연산이다.

  -예제 : 커미션이 300이거나 500이거나 1400인 사원을 조회하는 예

```
SELECT * FROM employee
WHERE commission IN (300,500,1400);
```

  - 예제: 커미션이 300,500, 1400모두 아닌 사원을 조회하는 예

```
SELECT * FROM employee
WHERE commission NOT IN (300,500,1400);
```

데이터 조회 - SELECT

- LIKE 연산자와 와일드카드
  - 컬럼에 저장된 문자중 일부만 일치하더라도 조회가 가능하도록 하기 위해 사용되는 연산
  - LIKE연산은 필수적으로 패턴이 필요한데 패턴에는 두가지 형태의 와일드카드가 사용된다.

와일드 카드	의미
%	문자가 없거나 하나 이상의 문자가 어떤 값이 와도 상관없음
_	하나의 문자가 어떤 값이 와도 상관없음

데이터 조회 - SELECT

- LIKE 연산자와 와일드카드
  - % 와일드 카드 사용하기

  - 예제 : 사원 이름이 F로 시작하는 모든 사원 조회하기

```
SELECT * FROM employee
WHERE ename LIKE 'F%';
```

  - 예제: 사원 이름에 M이 포함되어 있는 모든 사원 조회하기

```
SELECT * FROM employee
WHERE ename LIKE '%M%';
```

  - 예제: 사원 이름이 N으로 끝나는 모든 사원 조회하기

```
SELECT * FROM employee
WHERE ename LIKE '%N';
```

데이터 조회 - SELECT

- LIKE 연산자와 와일드카드
  - _ 와일드 카드 사용하기

  - 예제 : 이름의 두번째 글자가 A인 사원을 조회하기

```
SELECT * FROM employee
WHERE ename LIKE '_A%';
```

  - 예제: 이름의 세번째 글자가 A인 사원을 조회하기

```
SELECT * FROM employee
WHERE ename LIKE '__A%';
```

  - 예제: 이름에 A가 들어가지 않는 사원 조회하기

```
SELECT * FROM employee
WHERE ename NOT LIKE '%A%';
```

데이터 조회 - SELECT

- NULL을 위한 연산자 -is
  - null값은 미확정인, 알 수 없는 값이기 때문에 비교, 연산, 할당등이 불가능하다.

  - 커미션이 NULL값인 사원을 조회하기 --실패

```
SELECT * FROM employee
WHERE commission=null;
```

  - 컬럼의 값이 null인지 확인하기

```
SELECT * FROM employee
WHERE commission is null;
```

  - 컬럼의 값이 null이 아닌지 확인하기

```
SELECT * FROM employee
WHERE commission is not null;
```

데이터 조회 - SELECT

- 정렬하기 ORDER BY
  - 기존 SELECT문으로 데이터를 조회하는 쿼리를 작성한 후 결과를 보면 데이터가 입력된 순서대로 출력되는 것을 볼 수 있다.
  - 출력된 데이터를 알아보기 쉽게 하기 위해서 특정 컬럼을 기준으로 정렬을 할 수 있는데 이때 사용하는 것이 ORDER BY 절 이다.

  - 정렬 방법은 두가지가 있는데 오름차순과 내림차순이다.
  - 기준 컬럼 뒤에 ASC를 붙이면 오름차순
  - 기준 컬럼 뒤에 DESC를 붙이면 내림차순
  - 기준 컬럼 뒤에 정렬방식을 붙이지 않으면 기본값으로 오름차순 정렬이 된다.

데이터 조회 - SELECT

- 정렬하기 ORDER BY
  - 예제 : 급여를 기준으로 오름차순으로 정렬하기

```
SELECT * FROM employee
ORDER BY salary ASC;
```

```
SELECT * FROM employee
ORDER BY salary;
```

  - 예제 : 급여를 기준으로 내림차순으로 정렬하기

```
SELECT * FROM employee
ORDER BY salary DESC;
```

데이터 조회 - SELECT

- 정렬하기 ORDER BY
  - 숫자뿐 아니라 문자나 날짜도 같은 방법으로 정렬이 가능하다.

  - 사원 이름을 기준으로 오름차순 정렬

```
SELECT * FROM employee
ORDER BY ename ASC;
```

  - 각 사원의 입사할 기준으로 오름차순 정렬

```
SELECT * FROM employee
ORDER BY hiredate ASC;
```

데이터 조회 - SELECT

- 정렬하기 ORDER BY
  - 여러 개의 컬럼에 대해 정렬을 지정할 수 있다
  - 이 경우 먼저 나온 컬럼부터 정렬을 하고 먼저 나온 컬럼의 값이 같은 경우 다음 나온 컬럼의 정렬을 적용한다.

  - 예제 : 급여를 기준으로 내림차순 정렬하고 동일한 급여를 받는 사람은 이름을 기준으로 오름차순 정렬하세요.

```
SELECT * FROM employee
ORDER BY salary DESC, ename ASC;
```