Oracle DBMS

<필요한 데이터 값만 읽어내기 - where>
전체적인 선택을 하여 읽을때는 SELECT * FROM EMP;와 같이 한다.
전체적인 데이터를 선택하여 읽으면 필요없는 데이터들까지 읽게 되기 때문에 필요한 데이터 값만을 효율적으로 읽어야 한다.

SELECT * FROM EMP WHERE DEPTNO = 30;와 같이 where를 사용하여 필요한 값을 읽어낸다.

<여러개의 조건식 사용 and, or, 논리 연산자>
and, or - 논리 연산자 : DB에서는 and와 or만 있음
and는 2가지 모두 맞아야 하고 or은 1가지만 맞아도 참값이 된다.
nand, nor, xand, xor이라는 논리 연산자도 있음.

SELECT * FROM EMP WHERE DEPTNO = 30 AND JOB = 'SALESMAN';
30과 SALESMAN의 모두 공통된 것만 나타냄.

SELECT * FROM EMP WHERE DEPTNO = 30 OR JOB = 'CLERK';
30이거나 CLERK이면 나타냄.

<연산자 종류와 활용법>
사칙연산, 대소비교
SELECT * FROM EMP WHERE SAL*12 = 36000;
<참고>
= 1개 대입 연산자로 쓰임(DB에서는 대입 연산자 쓰일 일이 없어서 같다는 의미로 쓰임)
= 2개 일반적인 프로그래밍(파이선 등)에서 같다는 의미로 쓰임
= 3개 node.js와 같은 언어에서 같다는 의미
전체를 선택하는 의미의 *와 곱셉을 의미하는 *임

SELECT * FROM EMP WHERE SAL >= 3000;
>= 크거나 같다의 의미

<문자 대소비교>
SELECT * FROM EMP WHERE ENAME >= 'F';
첫 글자가 F와 같거나 F 이후에 있는 것을 검색

<참고 - 문자와 숫자의 차이점>
1은 우리가 볼 때는 숫자로 보임 컴퓨터에서는 숫자로도 보지만 문자로도 본다.
그렇기에 따옴표를 써서 문자의 의미로 나타냄
ASCII CODE 영어로 기준은 하였다.

ASCII CODE 미국정보교환표준부호라는 의미로 미국에서 수와 문자열을 나타내기 위해서 표준으로 만든 것
★ 문자의 비교는 의미가 없다.

<등가 비교 연산자>
!=, <>, ^=로 같지 않다를 나타냄.
!= 다른 언어에서도 많이 쓰는 편

```
SELECT * FROM EMP WHERE SAL != 3000;
```

<논리 부정 연산자>
NOT 연산자
```
SELECT * FROM EMP WHERE NOT SAL = 3000;
```
앞에서 배운 SELECT * FROM EMP WHERE SAL != 3000;와 같은 의미.

<OR, IN 연산자>
```
SELECT * FROM EMP
WHERE JOB = 'MANAGER'
OR JOB = 'SALESMAN'
OR JOB = 'CLERK';
```

VS

```
SELECT * FROM EMP
WHERE JOB IN ('MANAGER','SALESMAN','CLERK');
```
같은 결과이지만 간소하게 하려면 IN연산자 사용
()안으로 표현하는 것 function(함수)라고 한다.

```
SELECT * FROM EMP
WHERE JOB != 'MANAGER'
AND JOB <> 'SALESMAN'
AND JOB ^= 'CLERK';
```
-도 아니고 -도 아니고 -도 아닌 것이다.라는 의미

VS

```
SELECT * FROM EMP
WHERE JOB NOT IN('MANAGER', 'SALESMAN','CLERK');
```
위의 것 간단하게 표현

<BETWEEN A AND B 연산자>

```
SELECT * FROM EMP
WHERE SAL >= 2000 AND SAL <= 3000;
```

VS

```
SELECT * FROM EMP
WHERE SAL BETWEEN 2000 AND 3000;
```

위의 것 간단하게 표현을 하였음.
사잇값이 아닌 것 표현 할 때

```
SELECT * FROM EMP
WHERE SAL NOT BETWEEN 2000 AND 3000;
```

사용

<LIKE 연산자>
같은 값, 유사한 값, 비슷한 값을 나타냄.

```
SELECT * FROM EMP
WHERE ENAME LIKE 'S%';
```

첫 글자가 S인 것을 찾아줌.

<와일드 카드 _와 %>

```
SELECT * FROM EMP
WHERE ENAME LIKE '_L%';
```

<IS NULL 연산자 - NULL값을 구하는 메서드>
NULL 값이 없는 경우를 표현

```
SELECT ENAME, SAL, SAL*12+COMM AS ANNSAL, COMM FROM EMP;
```

- 별칭 사용하여 데이터 읽어내기

```
SELECT ENAME, SAL, SAL*12+COMM AS ANNSAL, COMM FROM EMP
WHERE COMM = NULL;
```

결과 문이 없다.
NULL은 값이 없다는 말, 값과 비교를 하지 못하니 아무것도 안나옴.

```
SELECT ENAME, SAL, SAL*12+COMM AS ANNSAL, COMM FROM EMP
WHERE COMM IS NULL;
```

IS NULL로 표현을 해야 NULL값에 해당되는 값이 나온다.

```
SELECT * FROM EMP
WHERE SAL > NULL
AND COMM IS NULL;
```

NULL값은 비교할 수 있는 값이 아니니 답이 없다.(크다 작다를 논할 수 없다.)
NULL값은 =, >, < 등의 비교 자체가 필요 없음.

<집합 연산자>
UNION : 합집합

```
SELECT EMPNO, ENAME, SAL, DEPTNO FROM EMP
WHERE DEPTNO = 10
UNION
SELECT EMPNO, ENAME, SAL, DEPTNO FROM EMP
WHERE DEPTNO = 20;
```

UNION ALL : 모두
MINUS : 차집합
INTERSECT : 교집합

```
SELECT EMPNO, ENAME, SAL, DEPTNO
FROM EMP
WHERE DEPTNO = 10
UNION ALL
SELECT EMPNO, ENAME, SAL, DEPTNO
FROM EMP
WHERE DEPTNO = 10;
```

중복에 상관 없이 모두 나타낸다.

```
SELECT EMPNO, ENAME, SAL, DEPTNO
FROM EMP
MINUS
SELECT EMPNO, ENAME, SAL, DEPTNO
FROM EMP
WHERE DEPTNO = 10;
```

전체에서 값을 빼서 나타냄

```
SELECT EMPNO, ENAME, SAL, DEPTNO
FROM EMP
INTERSECT
SELECT EMPNO, ENAME, SAL, DEPTNO
FROM EMP
WHERE DEPTNO = 10;
```

공통된 부분의 값을 나타냄

<함수(function)>

수학적인 의미의 함수가 아니라 기능과 역할의 의미로 function이라고 함.
f(x)=x+1을 예로 들면 x에 1을 입력하면 결과값이 2가 나온다.
(입력, 출력)
함수 : 하나를 입력하면 결과 값이 하나가 나오는 역할을 한다.
사용자 함수 - 사용자가 만드는 함수
내장 함수 - 프로그램에서 만든 함수(프로그램에서 내장되어 있는 함수)
함수는 매개 변수와 결과값이 발생한다.

<UPPER, LOWER, INITCAP>
영어의 대문자, 소문자, 첫 글자만 대문자 변환

```
SELECT ENAME, UPPER(ENAME), LOWER(ENAME), INITCAP(ENAME)
FROM EMP;
```

메서드=펑션 어떤 기능을 하는 것, 입력을 하면 결과물을 만들어낸다.
이름() = 펑션
<LENGTH>
문자열의 길이

- DUAL은 임시테이블을 나타냄.

SUBSTR(Substract) 함수 - 필요한 글자만큼 빼내는 것
INSTR(Insubtract) 함수 - 특정 문자열 찾는 것

```
SELECT *
FROM EMP
WHERE INSTR(ENAME, 'S') > 0;
```

```
SELECT *
FROM EMP
WHERE ENAME LIKE '%S%';
```

: 같은 결과

<REPLACE 함수>

```
SELECT '010-1234-5678' AS REPLACE_BEFORE,
REPLACE('010-1234-5678' , '-', ' ')AS REPLACE_1,
REPLACE('010-1234-5678', '-')AS REPLACE_2
FROM DUAL;
```

<LPAD, RPAD>
왼쪽 패딩, 오른쪽 패딩값 지정

패딩 = 여백
마진과 패딩
패딩 내가 쓰고 있는 공간의 여백
마진 전체 공간 내에서의 여백

CONCAT(concatanate)함수
문자열 합치는 함수

TRIM 함수
TRIM : 가지친다라는 의미
문자 삭제 함수
|| - 프로그래밍에서는 or의 의미, 문자열 합치는 의미로도 쓰인다.

```
SELECT '[' || TRIM(' _ _Oracle_ _ ') || ']' AS TRIM,
       '[' || TRIM(LEADING FROM ' _ _Oracle_ _ ') || ']' AS TRIM_LEADING,
       '[' || TRIM(TRAILING FROM ' _ _Oracle_ _ ') || ']' AS TRIM_TRAILING,
       '[' || TRIM(BOTH FROM ' _ _Oracle_ _ ') || ']' AS TRIM_BOTH
  FROM DUAL;
```

Leading : 앞에 있는 것
Trailing: 뒤에 따라 오는 것
Both : 양쪽
공백 문자를 제거할 때 Trim을 이용해서 한다.

function은 기능
function의 사용으로 function의 기능을 추측할 수 있어야 한다.
매개변수를 반복해서 보아야 함.
외우지 마세요.