4.1 SELECT 문

- SELECT 기능
  : 데이터베이스로부터 데이터를 검색하는 기능을 갖는다.

- selection : 질의에 대해 테이블의 **행**을 선택하기 위해 사용.
- projection : 질의에 대해 테이블의 **열**을 선택하기 위해 사용.
- join : 여러 테이블이 공통적으로 가진 컬럼을 이용해서,다른 테이블에 저장되어 있는 데이터를 가져오기 위해 사용.

- 기본적인 SELECT 문법

```
SELECT [DISTINCT] { *, column [alias],... }
FROM    table;
```

4.2 SELECT 문

- 계정이 소유한 테이블 목록 보기

```
SQL> SELECT * FROM TAB;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
BONUS                          TABLE
DEPT                           TABLE
EMP                            TABLE
SALGRADE                       TABLE

SQL>
```

- 특정 테이블의 컬럼 구조 보기

```
SQL> DESC DEPT
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                    NOT NULL NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

4.3 SELECT 문

- 테이블내의 모든 데이터 보기

```
SQL> SELECT * FROM DEPT;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

- 테이블내의 특정 컬럼 데이터 보기
  - SELECT 뒤에 해당 컬럼을 차례대로 기술. 쉼표로 구분.

```
SQL> SELECT EMPNO, ENAME, JOB, HIREDATE FROM EMP;

     EMPNO ENAME      JOB       HIREDATE
---------- ---------- --------- --------
      7369 SMITH      CLERK     80/12/17
      7499 ALLEN      SALESMAN  81/02/20
      7521 WARD       SALESMAN  81/02/22
        ...
      7900 JAMES      CLERK     81/12/03
      7902 FORD       ANALYST   81/12/03
      7934 MILLER     CLERK     82/01/23
```

4.4 SQL 문 작성 지침

- SQL 문장은 대소문자를 구별하지 않는다.
- SQL 문장은 여러 줄에 걸쳐서 작성 가능하다.
- 키워드 (SELECT ,FROM 등)는 줄여 사용할 수 없고 , 여러 줄에 걸쳐서 작성할 수 없다.
- 일반적으로 키워드는 대문자로 작성한다.
- 모든 SQL 문장의 끝은 세미콜론 (;)으로 끝난다.

- 컬럼의 정렬 방식
  - SELECT 문장의 실행 결과는 각 컬럼의 데이터 타입에 따라 정렬형태가 달라진다.

```
SQL> SELECT EMPNO, ENAME, JOB, HIREDATE FROM EMP;

     EMPNO ENAME      JOB       HIREDATE
---------- ---------- --------- --------
      7369 SMITH      CLERK     80/12/17
      7499 ALLEN      SALESMAN  81/02/20
      7521 WARD       SALESMAN  81/02/22
        ...
      7900 JAMES      CLERK     81/12/03
      7902 FORD       ANALYST   81/12/03
      7934 MILLER     CLERK     82/01/23
```

4.5 산술연산을 이용한 SQL 문

- SQL 문장내의 숫자 및 날짜 타입에는 + , - , * , / 사용가능하다.
- 연산식이 컬럼명으로 표시된다.

```
SQL> SELECT EMPNO, ENAME, SAL * 1.1 FROM EMP;

     EMPNO ENAME          SAL*1.1
---------- ---------- ----------
      7369 SMITH             880
      7499 ALLEN            1760
      7521 WARD             1375
        ...
      7900 JAMES            1045
      7902 FORD             3300
      7934 MILLER           1430
```

4.6 컬럼에 별칭(alias) 사용.

```
SQL> SELECT EMPNO AS 사번, ENAME AS 성명, SAL AS 급여 FROM EMP;

      사번 성명             급여
---------- ---------- ----------
      7369 SMITH             800
      7499 ALLEN            1600
      7521 WARD             1250
        ...
      7900 JAMES             950
      7902 FORD             3000
      7934 MILLER           1300
```

4.7 널 (Null) 값의 정의 및 처리

- 이용 불가능한(Unavailable)
- 지정되지 않은(Unassigned)
- 알수 없는 (Unknown)
- 적용할 수 없는(Inapplicable) 값을 의미한다.
- Null이 산술연산에 사용되면 그 결과는 무조건 Null 이다.

```
SQL> SELECT EMPNO, ENAME, SAL, COMM FROM EMP;

     EMPNO ENAME           SAL       COMM
---------- ---------- ---------- ----------
      7369 SMITH             800
      7499 ALLEN            1600        300
      7521 WARD             1250        500
      7566 JONES            2975
      7654 MARTIN           1250       1400
        ...
      7902 FORD             3000
      7934 MILLER           1300
```

Null

```
SQL> SELECT EMPNO, ENAME, COMM, COMM + 100 FROM EMP;

     EMPNO ENAME           COMM   COMM+100
---------- ---------- ---------- ----------
      7369 SMITH
      7499 ALLEN            300        400
      7521 WARD             500        600
      7566 JONES
        ...
      7902 FORD
      7934 MILLER
```

4.8 연결 연산자

- 여러 개의 문자열을 연결하여 하나의 문자열로 생성.

```
SQL> SELECT ENAME || JOB AS "이름 직업" FROM EMP;

이름 직업
----------------------------------
SMITHCLERK
ALLENSALESMAN
WARDSALESMAN
JONESMANAGER
MARTINSALESMAN
BLAKEMANAGER
```

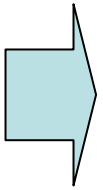
- 리터럴(Literal)
  - SELECT 문장에 포함된 컬럼명 또는 별칭 이외의 문자값, 숫자값, 날짜값이다.
  - 반드시 문자값 , 날짜값에는 ‘ ‘ 을 붙인다.

```
SQL> SELECT ENAME||'의 직급은 '||JOB||'이다' AS "사원별 직급" FROM EMP;

사원별 직급
------------------------------
SMITH의 직급은 CLERK이다
ALLEN의 직급은 SALESMAN이다
WARD의 직급은 SALESMAN이다
...
JAMES의 직급은 CLERK이다
FORD의 직급은 ANALYST이다
MILLER의 직급은 CLERK이다
```

4.9 중복행 제거

- DISTINCT 키워드 이용

```
SQL> SELECT JOB FROM EMP;

JOB
---------
CLERK
SALESMAN
SALESMAN
MANAGER
SALESMAN
MANAGER
MANAGER
ANALYST
PRESIDENT
SALESMAN
CLERK
CLERK
ANALYST
CLERK
```



```
SQL> SELECT DISTINCT JOB FROM EMP;

JOB
---------
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN
```