

데이터베이스

관계형 데이터베이스

02. SQL의 기본

first
coding

- 오라클을 설치하면 제공되는 사용자인 SCOTT은 학습을 위해서 테이블들이 제공.
- SCOTT이 소유하고 있는 테이블을 살펴보기 위해서 다음과 같은 명령을 입력.

예

```
SELECT * FROM TAB;
```

- TAB은 TABLE의 약자로서 SCOTT 사용자가 소유하고 있는 테이블의 정보를 알려주는 데이터 디렉터리.

02 테이블 구조를 살펴보기 위한 DESC

- 테이블에서 데이터를 조회하기 위해서는 테이블의 구조를 알아야 합니다.
- 테이블의 구조를 확인하기 위한 명령어로는 DESCRIBE가 있다.
 - DESC 명령어는 테이블의 컬럼 이름, 데이터 형, 길이와 NULL 허용 유무 등과 같은 특정 테이블의 정보를 알려줌.

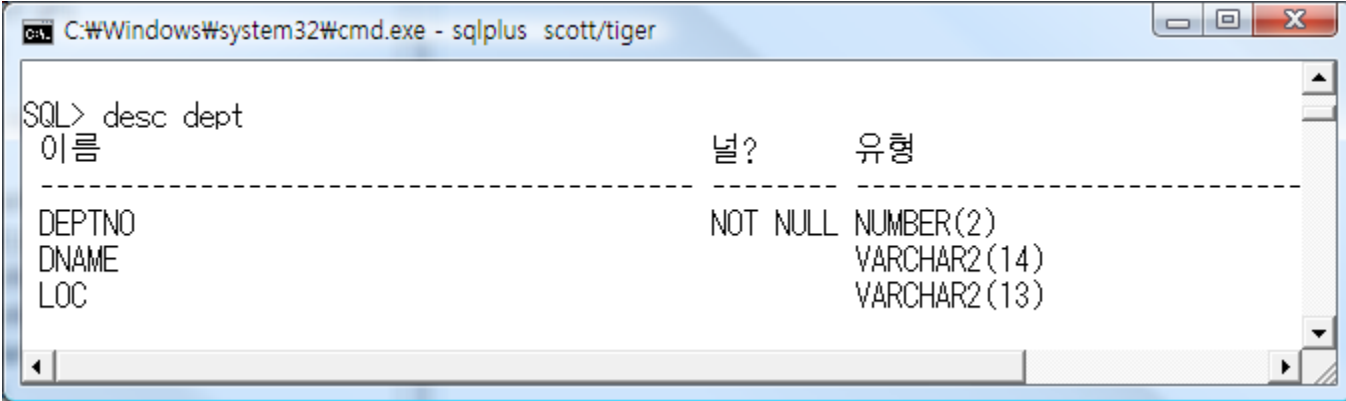
| 형식 | DESC[RIBE] 테이블명 |
|----|-----------------|
|----|-----------------|

02 테이블 구조를 살펴보기 위한 DESC

- 오라클을 설치하면 학습용으로 제공되는 DEPT 테이블은 부서의 정보를 저장하고 있으며, 이에 대한 구조를 살펴보기 위해서는 desc 명령어를 사용.

예

DESC DEPT ---- *DEPT 테이블의 구조 살피기*



```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger

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

- NUMBER

- NUMBER 데이터 형은 숫자 데이터를 저장하기 위해서 제공.

| | |
|----|--------------------------|
| 형식 | NUMBER(precision, scale) |
|----|--------------------------|

- precision은 소수점을 포함한 전체 자리 수를 의미하며 scale은 소수점 이하 자리 수를 지정.
- scale을 생략한 채 precision만 지정하면 소수점 이하는 반올림되어 정수 값만 저장.
- precision과 scale을 모두 생략하면 입력한 데이터 값만큼 공간이 할당.

- DATE

- DATE는 세기, 년, 월, 일, 시간, 분, 초의 날짜 및 시간 데이터를 저장하기 위한 데이터 형.
- 이렇듯 날짜 타입 안에는 세기, 년, 월, 일, 시, 분, 초, 요일 등 여러 가지 정보가 들어 있지만 별다른 설정이 없으면 년, 월, 일만 출력.
- 기본 날짜 형식은 "YY/MM/DD"형식으로 "년/월/일"로 출력.
- 2005년 12월 14일은 "05/12/14"로 출력.

- CHAR

- 문자 데이터를 저장하기 위한 자료형으로 CHAR가 있습니다. CHAR는 고정 길이 문자 데이터를 저장.
- 입력된 자료의 길이와는 상관없이 정해진 길이만큼 저장 영역 차지하며 최소 크기는 1.
- 주소를 저장하기 위해서 address 란 컬럼을 생성하되 저장될 데이터의 최대 크기를 고려해서 CHAR(20)이라고 주었고, 'seoul' 이란 데이터를 저장하였다고 하면 아래와 같이 저장됨.

address

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| s | e | o | u | l | | | | | | | | | | | | | | | |
|---|---|---|---|---|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|

- CHAR는 주어진 크기만큼 저장공간이 할당되므로 편차가 심한 데이터를 입력할 경우 위의 예와 같이 저장공간의 낭비를 초래하게 됨.

- VARCHAR2

- VARCHAR2 데이터 형은 가변적인 길이의 문자열을 저장하기 위해서 제공.
- 이번에는 주소를 저장하기 위해서 address 란 컬럼의 데이터형을 VARCHAR2(20)이라고 설정하고, 'seoul' 이란 데이터를 저장하였다고 하면 아래와 같이 저장됨.

address

| | | | | |
|---|---|---|---|---|
| s | e | o | u | l |
|---|---|---|---|---|

- VARCHAR2는 저장되는 데이터에 의해서 저장공간이 할당되므로 메모리 낭비를 줄일 수 있음.

04 데이터를 조회하기 위한 SELECT 문

- SELECT 문은 데이터를 조회하기 위한 SQL 명령어입니다.

| | |
|----|--|
| 형식 | SELECT [DISTINCT] {*, column[Alias], . . . } FROM table_name; |
|----|--|

- SQL 명령어는 하나의 문장으로 구성되어야 하는데 여러 개의 절이 모여서 문장이 되는 것이고 이러한 문장들은 반드시 세미콜론(;)으로 마쳐야 함.
- SELECT 문은 반드시 SELECT와 FROM 이라는 2개의 키워드로 구성.
- SELECT절은 출력하고자 하는 컬럼 이름을 기술 하고, FROM절 다음에는 조회하고자 하는 테이블 이름을 기술.
- 특정 컬럼 이름 대신 * 를 기술할 수 있는데, * 는 테이블 내의 모든 컬럼을 출력하고자 할 경우 사용.
- SQL 문에서 사용하는 명령어들은 대문자와 소문자를 구분하지 않는다는 특징이 있음.

```
SCOTT>SELECT * FROM emp ;
```

```
SCOTT>SELECT *  
2 FROM emp ;
```

- 직원번호에 해당되는 컬럼 이름은 empno이고 직원명에 해당되는 컬럼 이름은 ename입니다.
- empno와 ename을 출력하기 위해서는 출력하고자 하는 순서대로 기술하되 컬럼과 컬럼 사이에 콤마를 기술합니다.

예

```
select empno, ename from emp;
```

- 급여로 연봉 계산을 해보도록 합시다.
- 일반적으로 연봉은 급여를 12번 곱한 것이므로 연봉을 구하기 위해서 산술 연산자를 사용합니다.

| 종류 | 예 |
|----|-----------------------------|
| + | SELECT sal + comm FROM emp; |
| - | SELECT sal - 100 FROM emp; |
| * | SELECT sal * 12 FROM emp; |
| / | SELECT sal / 2 FROM emp; |

| | |
|---|-------------------------------------|
| 예 | SELECT ename, sal, sal*12 FROM emp; |
|---|-------------------------------------|

- 오라클에서의 널은 매우 중요한 데이터입니다.
 - 오라클에서는 컬럼에 null 값이 저장되는 것을 허용하는데 널 값을 제대로 이해하지 못한 채 쿼리문을 사용하면 원하지 않는 결과를 얻을 수 있기 때문입니다.
 - 다양한 널의 정의.
 - 0(zero)도 아니고
 - 빈 공간도 아니다.
 - 미확정(해당 사항 없음), 알 수 없는(unknown) 값을 의미한다.
 - 어떤 값인지 알 수 없지만 어떤 값이 존재하고 있다.
 - ? 혹은 ∞ 의 의미이므로
 - 연산, 할당, 비교가 불가능하다.

널은 ? 혹은 ∞ 의 의미이기 때문에 연산, 할당, 비교가 불가능

$$100 + ? = ?$$

$$100 + \infty = \infty$$

예

```
SELECT ename, sal, job, comm, sal*12, sal*12+comm
FROM emp;
```

SQL> select ename, sal, job, comm, sal*12, sal*12+comm
2 from emp;

| ENAME | SAL | JOB | COMM | SAL*12 | SAL*12+COMM |
|--------|------|-----------|------|--------|-------------|
| SMITH | 800 | CLERK | | 9600 | |
| ALLEN | 1600 | SALESMAN | 300 | 19200 | 19500 |
| WARD | 1250 | SALESMAN | 500 | 15000 | 15500 |
| JONES | 2975 | MANAGER | | 35700 | |
| MARTIN | 1250 | SALESMAN | 1400 | 15000 | 16400 |
| BLAKE | 2850 | MANAGER | | 34200 | |
| CLARK | 2450 | MANAGER | | 29400 | |
| SCOTT | 3000 | ANALYST | | 36000 | |
| KING | 5000 | PRESIDENT | | 60000 | |
| TURNER | 1500 | SALESMAN | 0 | 18000 | 18000 |
| ADAMS | 1100 | CLERK | | 13200 | |
| JAMES | 950 | CLERK | | 11400 | |
| FORD | 3000 | ANALYST | | 36000 | |
| MILLER | 1300 | CLERK | | 15600 | |

14 개의 행이 선택되었습니다.

SQL>

영업직인 경우 커미션(comm) 컬럼에 값이 저장되어 있으므로 제대로 연봉 계산을 하게 된다.

영업직인 경우 무능력하여 받을 커미션(comm)이 없더라도 0으로 저장되어 있으므로 연봉 계산이 제대로 된다.

영업직인 아닌 경우에는 커미션에 널 값이 저장되어 있어서 연봉 계산 결과도 널값으로 구해지는 모순이 발생한다.

예

```
select  ename, comm, sal*12+comm,
        nvl(comm, 0), sal*12+nvl(comm, 0)
from emp;
```

C:\Windows\system32\cmd.exe - sqlplus scott/tiger

```
SQL> select ename, comm, sal*12+comm, nvl(comm, 0), sal*12+nvl(comm, 0)
2 from emp;
```

| ENAME | COMM | SAL*12+COMM | NVL(COMM,0) | SAL*12+NVL(COMM,0) |
|--------|------|-------------|-------------|--------------------|
| SMITH | | | 0 | 9600 |
| ALLEN | 300 | 19500 | 300 | 19500 |
| WARD | 500 | 15500 | 500 | 15500 |
| JONES | | | 0 | 35700 |
| MARTIN | 1400 | 16400 | 1400 | 16400 |
| BLAKE | | | 0 | 34200 |
| CLARK | | | 0 | 29400 |
| SCOTT | | | 0 | 36000 |
| KING | | | 0 | 60000 |
| TURNER | 0 | 18000 | 0 | 18000 |
| ADAMS | | | 0 | 13200 |
| ----- | | | | |
| ENAME | COMM | SAL*12+COMM | NVL(COMM,0) | SAL*12+NVL(COMM,0) |
| JAMES | | | 0 | 11400 |
| FORD | | | 0 | 36000 |
| MILLER | | | 0 | 15600 |

14 개의 행이 선택되었습니다.

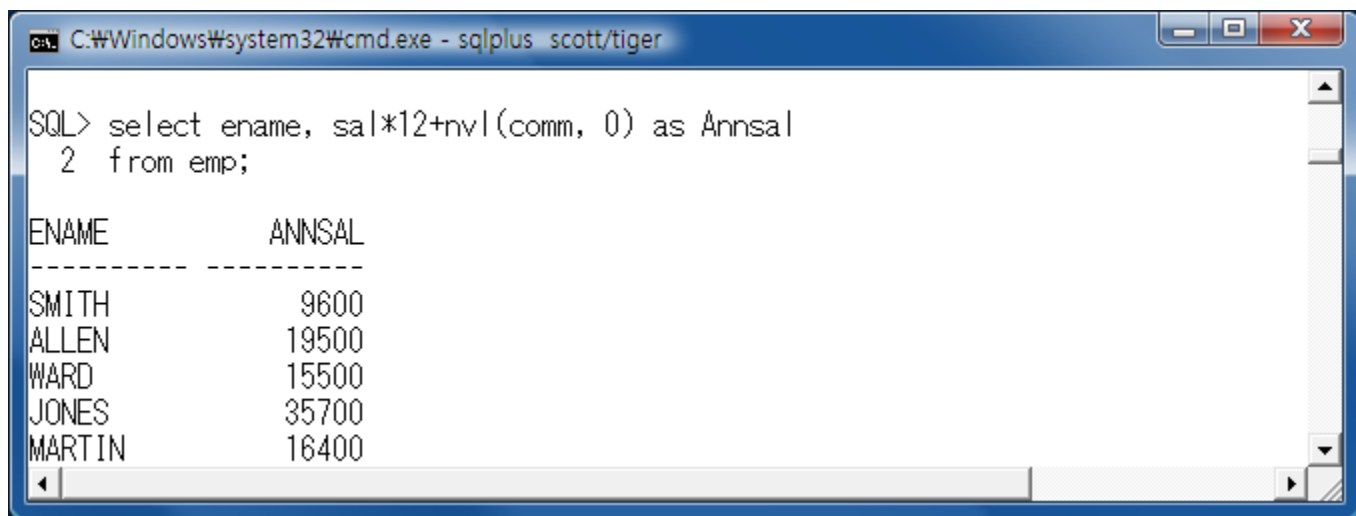
- 연봉 계산을 위해 사원 테이블에서 급여와 커미션 칼럼을 살펴본 결과 **영업사원이 아닌 직원들의 커미션은 NULL로 지정되어 있으므로** 연봉을 올바르게 계산하기 위해서는 **커미션이 NULL인 경우 0으로 변경하여 계산**에 되도록 해야 함.
- 오라클에서는 **NULL을 0 또는 다른 값으로 변환하기 위해서 사용하는 함수로 NVL을 제공**하고 있음.
- 커미션에 널이 저장되어 있더라도 널을 다른 값으로 변환하는 NVL 함수를 사용하면 제대로 된 계산 결과를 얻을 수 있음.

08 AS 로 컬럼에 별칭 부여하기

- 컬럼 이름 대신 별칭을 출력하고자 하면 컬럼을 기술한 바로 뒤에 AS 라는 키워드를 쓴 후 별칭을 기술합니다.

예

```
select ename, sal*12+nvl(comm, 0) as Annsal  
from emp;
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - sqlplus scott/tiger". The prompt is "SQL>". The user has entered the following SQL query:

```
select ename, sal*12+nvl(comm, 0) as Annsal  
2  from emp;
```

The output of the query is displayed in a table format:

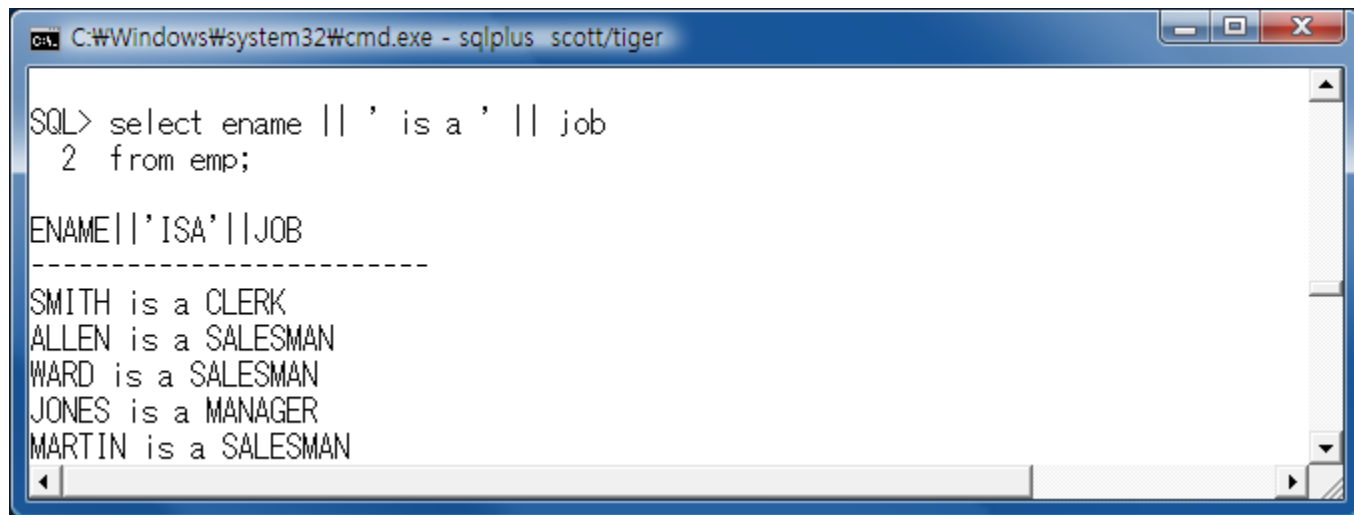
| ENAME | ANNSAL |
|--------|--------|
| SMITH | 9600 |
| ALLEN | 19500 |
| WARD | 15500 |
| JONES | 35700 |
| MARTIN | 16400 |

09 Concatenation 연산자의 정의와 사용

- 결과를 살펴보면 컬럼과 특정 값 사이에 공백이 생기는 것을 확인할 수 있습니다. 정말 영 문장처럼 보이도록 하기 위해서 "||"를 컬럼과 문자열 사이에 기술하여 하나로 연결하여 출력하면 됩니다.

예

```
select ename || ' is a ' || job  
from emp;
```

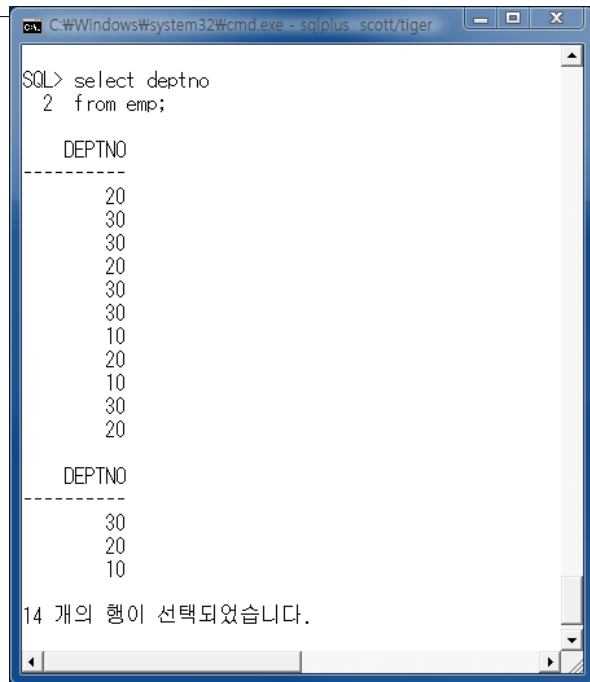


```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger  
  
SQL> select ename || ' is a ' || job  
2  from emp;  
  
ENAME||'ISA' ||JOB  
-----  
SMITH is a CLERK  
ALLEN is a SALESMAN  
WARD is a SALESMAN  
JONES is a MANAGER  
MARTIN is a SALESMAN
```


- 다음은 직원들이 소속되어 있는 부서 번호를 출력하기 위한 예입니다.

예

```
select deptno  
from emp;
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - sqlplus scott/tiger". The prompt is "SQL>". The user has entered the command "select deptno" on the first line and "2 from emp;" on the second line. The output displays a list of department numbers (DEPTNO) with a dashed line separator. The numbers are: 20, 30, 30, 20, 30, 30, 10, 20, 10, 30, 20. Below this, there is another dashed line separator followed by the numbers 30, 20, and 10. At the bottom of the window, a status bar indicates "14 개의 행이 선택되었습니다." (14 rows selected).

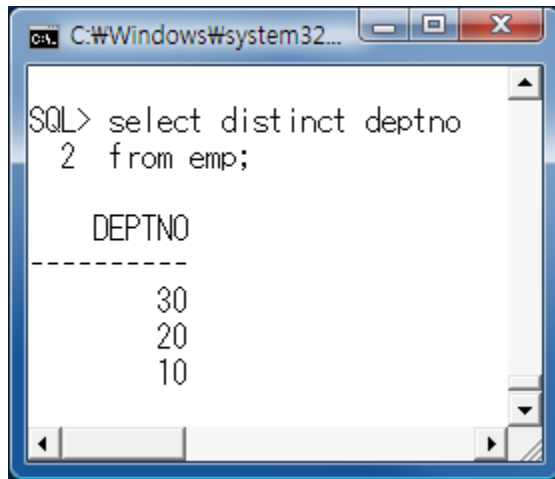
```
SQL> select deptno  
2 from emp;  
  
DEPTNO  
-----  
20  
30  
30  
20  
30  
30  
10  
20  
10  
30  
20  
  
DEPTNO  
-----  
30  
20  
10  
  
14 개의 행이 선택되었습니다.
```

10 DISTINCT 키워드

- 직원들이 소속되어 있는 부서의 목록을 얻기 위한 목적이라면 같은 부서의 번호가 중복되어 출력되는 것은 의미가 없습니다.
- 중복되는 부서 번호를 한 번씩만 출력하기 위해서는 키워드 DISTINCT를 사용합니다.

예

```
select distinct deptno  
from emp;
```



A screenshot of a Windows command prompt window. The title bar shows 'C:\Windows\system32...'. The command prompt displays the following SQL query and its output:

```
SQL> select distinct deptno  
2 from emp;
```

The output shows the column header 'DEPTNO' followed by a horizontal dashed line and three rows of data: 30, 20, and 10.

| DEPTNO |
|--------|
| 30 |
| 20 |
| 10 |

DISTINCT 사용 안 함

```

oracle@localhost:~
SCOTT>
SCOTT>SELECT deptno ,job
  2  FROM emp ;

DEPTNO JOB
-----
20 CLERK
30 SALESMAN
30 SALESMAN
20 MANAGER
30 SALESMAN
30 MANAGER
10 MANAGER
20 ANALYST
10 PRESIDENT
30 SALESMAN
20 CLERK
30 CLERK
20 ANALYST
10 CLERK

14 rows selected.

```

1

DISTINCT 사용 함

```

oracle@localhost:~
SCOTT>
SCOTT>SELECT DISTINCT deptno
  2  FROM emp ;

DEPTNO
-----
30
20
10

2

```

두 컬럼 DISTINCT 사용 함

```

oracle@localhost:~
SCOTT>
SCOTT>SELECT DISTINCT deptno , job
  2  FROM emp ;

DEPTNO JOB
-----
20 CLERK
30 SALESMAN
20 MANAGER
30 CLERK
10 PRESIDENT
30 MANAGER
10 CLERK
10 MANAGER
20 ANALYST

9 rows selected.

3

```