



Database 프로그래밍을 위한  
오라클 명령어

05

## 데이터 정의어(DDL)

강 사 : 김 진 성



# 목 차

- 1 테이블 생성(CREATE TABLE) 과 자료형
- 2 테이블 구조변경(ALTER TABLE)
- 3 테이블 삭제(DROP TABLE)
- 4 테이블 레코드 삭제(TRUNCATE)
- 5 테이블 이름 변경(RENAME)
- 6 데이터 사전과 뷰

# 01. CREATE TABLE

- DDL(Data Definition Language)을 사용하여 테이블 구조 생성, 수정, 삭제 실습
- CREATE TABLE 명령어로 새로운 테이블 생성
- CREATE TABLE문 기본 형식

```
CREATE TABLE [schema.]table_name  
  (column datatype [DEFAULT expr] [column_constraint],  
   . . . . .  
   [table_constraint]);
```

➤schema	테이블의 소유자
➤table_name	생성하고자 하는 테이블 이름. 사용자 단위로 유일한 이름
➤column	테이블에서 사용하는 열 이름. 테이블 단위로 유일한 이름
➤datatype	열의 자료형
➤DEFAULT expr	INSERT문장에서 값 입력 생략 시 기본적으로 입력되는 값 지정
➤column_constraint	열정의 부분에서 무결성 제약 조건을 기술(예: primary key, not null)
➤table_constraint	테이블 정의 부분에서 무결성 제약 조건을 기술

# 1.1 데이터 형

- Oracle에서 제공되는 데이터 형의 종류

DATA TYPE	설 명
VARCHAR2(n)	가변 길이 문자 데이터(1~4000byte), 최소크기 1byte
CHAR(n)	고정 길이 문자 데이터(1~2000byte) , 최소크기 1byte
NUMBER(p,s)	전체 p자리 중 소수점 이하 s자리(p:1~38, s:-84~127)
DATE	7Byte(BC 4712년 1월 1일부터 AD 9999년 12월 31일)
RAW(n)	n Byte의 원시 이진 데이터(1~2000)
LONG	가변 길이 문자 데이터(1~2Gbyte)
LONG RAW	가변 길이 원시 이진 데이터(1~2Gbyte)

## 1.2 의사컬럼

### 의사컬럼(Pseudo Column)

- 실제 테이블에 존재하지 않은 가짜 컬럼(가상 공간에서 사용)
- SELECT와 WHERE절에서만 사용 가능하다.
- 대표적인 의사컬럼 : ROWNUM과 ROWID

```
SELECT ROWNUM, EMPNO, ENAME, ROWID  
FROM EMP WHERE ROWNUM <=10 ;
```

검색 순서  
대로 값 출력

	ROWNUM	EMPNO	ENAME	ROWID
1	1	7521	WARD	AAAD6VAAEAAAAO4AAA
2	2	7369	SMITH	AAAD6VAAEAAAAO4AAB
3	3	7499	ALLEN	AAAD6VAAEAAAAO4AAC
4	4	7566	JONES	AAAD6VAAEAAAAO4AAD
5	5	7654	MARTIN	AAAD6VAAEAAAAO4AAE
6	6	7698	BLAKE	AAAD6VAAEAAAAO4AAF
7	7	7782	CLARK	AAAD6VAAEAAAAO4AAG
8	8	7788	SCOTT	AAAD6VAAEAAAAO4AAH
9	9	7839	KING	AAAD6VAAEAAAAO4AAI
10	10	7844	TURNER	AAAD6VAAEAAAAO4AAJ

중복되지 않은  
유일한 값

# (1) ROWID

- ROWID 데이터 형은 테이블에서 행의 위치를 지정하는 논리적인 주소값
- 데이터베이스 전체에서 중복되지 않는 유일한 값으로 테이블에 새로운 행이 삽입되면 테이블 내부에서 의사 컬럼 형태로 자동적으로 생성
- ROWID는 테이블의 특정 레코드를 랜덤하게 접근하기 위해서 주로 사용



## (2) ROWNUM

- ROWNUM은 조회된 레코드의 순번을 갖는 의사칼럼
- 레코드가 삽입(INSERT)순에 대한 순번을 나타내는 번호
- 전체 레코드 순번 검색

**SELECT ROWNUM, EMPNO, ENAME, ROWID FROM EMP;**

- 처음 부터 특정 순번 검색

**SELECT ROWNUM, EMPNO, ENAME, ROWID  
FROM EMP WHERE ROWNUM <=10 ;**



## <실습하기> 새롭게 테이블 생성하기

- 사원 테이블과 유사한 구조의 사원번호, 사원이름, 급여 3개의 칼럼으로 구성된 EMP01 테이블을 생성하기
1. CREATE TABLE 명령어로 EMP01 테이블을 새롭게 생성

```
CREATE TABLE EMP01(  
  EMPNO NUMBER(4),  
  ENAME VARCHAR2(20),  
  SAL NUMBER(7, 2));
```





## <실습하기> 서브 쿼리로 테이블 생성하기

- CREATE TABLE 문에서 서브 쿼리를 사용하여 이미 존재하는 테이블과 동일한 구조와 내용을 갖는 새로운 테이블을 생성할 수 있다.
- 1. CREATE TABLE 명령어 다음에 컬럼을 일일이 정의하는 대신 AS 절을 추가하여 EMP 테이블과 동일한 내용과 구조를 갖는 EMP02 테이블을 생성해보자.

```
CREATE TABLE EMP02  
AS  
SELECT * FROM EMP;
```

데이터까지 복제된다.



## <실습하기> 원하는 컬럼으로 구성된 복제 테이블 생성하기

- 기존 테이블에서 원하는 컬럼만 선택적으로 복사해서 생성할 수 있다.
- 1. 서브 쿼리문의 SELECT 절에 \* 대신 원하는 컬럼명을 명시하면 기존 테이블에서 일부의 컬럼만 복사할 수 있다.

```
CREATE TABLE EMP03  
AS  
SELECT EMPNO, ENAME FROM EMP;
```



# <과제1>

- 1. EMP 테이블을 복사하되 사원번호, 사원이름, 급여 컬럼으로 구성된 테이블을 생성하시오.(단 테이블의 이름은 EMP04)



```
SQL> SELECT * FROM EMP04;
```

EMPNO	ENAME	SAL
7369	SMITH	800
7499	ALLEN	1600
7521	WARD	1250
7566	JONES	2975
7654	MARTIN	1250
7698	BLAKE	2850
7782	CLARK	2450
7788	SCOTT	3000
7839	KING	5000
7844	TURNER	1500
7876	ADAMS	1100

EMPNO	ENAME	SAL
7900	JAMES	950
7902	FORD	3000
7934	MILLER	1300

14 개의 행이 선택되었습니다.

## <실습하기> 원하는 행으로 구성된 복제 테이블 생성하기

- 기존 테이블에서 원하는 행만 선택적으로 복사해서 생성한다.
- 1. 서브 쿼리문의 SELECT 문을 구성할 때 WHERE 절을 추가하여 원하는 조건을 제시하면 기존 테이블에서 일부의 행만 복사한다.

```
CREATE TABLE EMP05  
AS  
SELECT * FROM EMP  
WHERE DEPTNO=10;
```



## 1.2 테이블의 구조만 복사하기

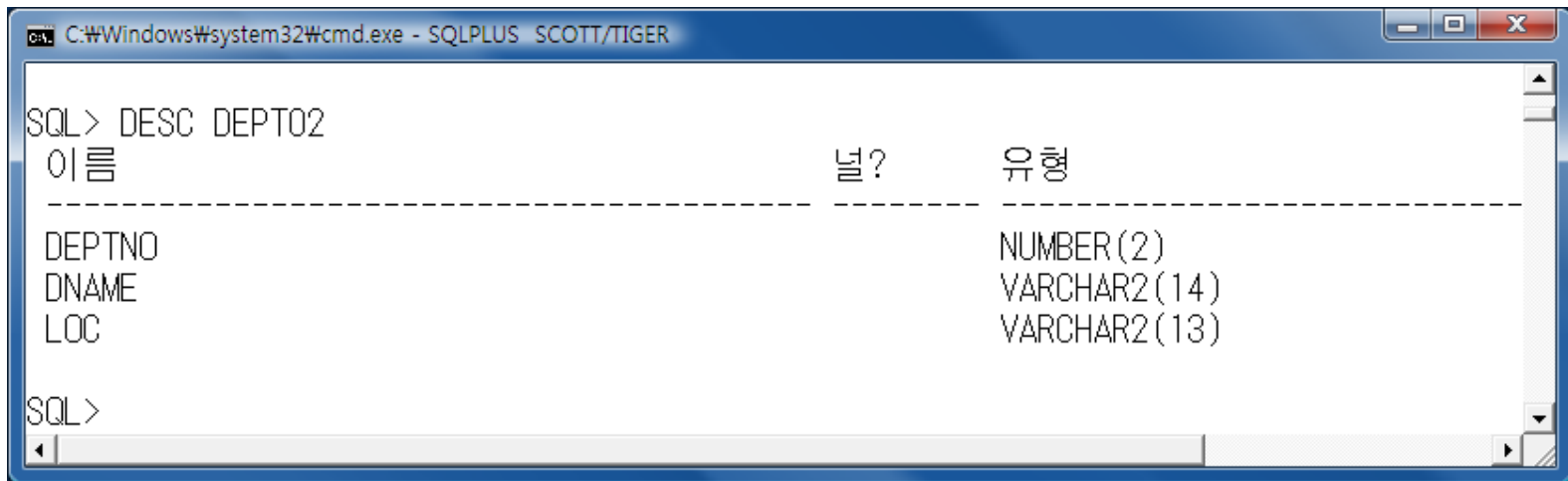
- 서브 쿼리를 이용하여 테이블을 복사하되 데이터는 복사하지 않고 기존 테이블의 구조만 복사하는 방법을 알아본다.
- WHERE 조건 절에 항상 거짓이 되는 조건을 지정하게 되면 테이블에서 얻어질 수 있는 로우가 없게 되므로 빈 테이블이 생성 된다.

```
CREATE TABLE EMP06  
AS  
SELECT * FROM EMP WHERE 1=0;
```

- WHERE 1=0; 조건은 항상 거짓이다. 이를 이용하여 테이블의 데이터는 가져오지 않고 구조만 복사하게 한다.

## <과제2>

- 2. DEPT 테이블과 동일한 구조의 빈 테이블을 생성하시오.  
(테이블의 이름은 DEPT02)



```
C:\Windows\system32\cmd.exe - SQLPLUS SCOTT/TIGER

SQL> DESC DEPT02
이름          DEPTNO          DNAME          LOC
-----
DEPTNO        NUMBER(2)
DNAME         VARCHAR2(14)
LOC           VARCHAR2(13)

SQL>
```

## 02. 테이블 제약조건

### 테이블 제약조건(constraint)

- 부적절한 자료 입력을 방지하기 위하여 컬럼에 제약조건을 사용한다.

제 약 조 건	설 명
PRIMARY KEY(PK)	유일하게 테이블의 각행을 식별(NOT NULL과 UNIQUE조건 만족)
FOREIGN KEY(FK)	열과 참조된 열 사이의 외래키 관계를 적용하고 설정
UNIQUE key(UK)	테이블의 모든 행을 유일하게 하는 값을 가진 열(NULL 허용)
NOT NULL(NN)	열은 NULL값을 포함할 수 없다.
CHECK(CK)	참이어야 하는 조건을 지정함(대부분 업무 규칙 설정)

## 02. 테이블 제약조건

- 제약 조건 정의 방법

- COLUMN LEVEL과 TABLE LEVEL 두 가지 방법이 있다.

### 가) 컬럼 LEVEL 제약 조건(COLUMN LEVEL CONSTRAINT)

- 열(컬럼)별로 제약 조건을 정의한다.
- 무결성 제약 조건 5가지를 모두 적용할 수 있다.
- NOT NULL 제약 조건은 컬럼 LEVEL에서만 가능하다.

```
Column datatype [CONSTRAINT constraint_name ] constraint_type
```

### 나) 테이블 LEVEL 제약 조건(TABLE LEVEL CONSTRAINT)

- 테이블의 칼럼 정의와는 개별적으로 정의한다.
- 하나 이상의 열을 참조할 경우에 사용한다.
- NOT NULL을 제외한 나머지 제약 조건만 정의

```
column datatype,  
[CONSTRAINT constraint_name] constraint_type (column1[,column2,.....])
```



## 02. 테이블 제약조건

### 가) PRIMARY KEY(PK)

- 테이블에 대한 기본 키를 생성한다.
- 하나의 기본 키만이 테이블에 존재할 수 있다.
- PRIMARY KEY 제약 조건은 테이블에서 각행을 유일하게 식별하는 열 또는 열의 집합이다.(UNIQUE와 NOT NULL조건을 만족)
- 이 제약 조건은 열 또는 열의 집합의 유일성을 요구하고 NULL값을 포함할 수 없다.
- UNIQUE INDEX가 자동 생성된다.

column datatype [CONSTRAINT constraint_name] PRIMARY KEY (col1[,col2,...])
column datatype, ....., [CONSTRAINT constraint_name] PRIMARY KEY (column1[,column2,...])

## 02. 테이블 제약조건

### PRIMARY KEY(PK) 예

```
SQL> CREATE TABLE test_tab1(  
    id          NUMBER(2) CONSTRAINT test_id_pk  PRIMARY KEY,  
    name        VARCHAR2(10));
```

Table created.

```
DESC test_tab1;  
이름  널   유형
```

```
-----  
ID  NOT NULL NUMBER(2)  
NAME      VARCHAR2(10)
```

```
SQL> CREATE TABLE test_tab2(  
    id          NUMBER(2),  
    name        VARCHAR2(10),  
    CONSTRAINT test_id_pk PRIMARY KEY (id));
```

Table created.

## 02. 테이블 제약조건

### 나) FOREIGN KEY(FK)

- 외래키(FOREIGN KEY)는 참조하는 테이블에서 정의한다.
- MASTER TABLE의 PRIMARY KEY, UNIQUE KEY로 정의된 열을 지정할 수 있으며 열의 값과 일치하거나 NULL값이어야 한다.
- FOREIGN KEY는 열 또는 열의 집합을 지정할 수 있으며 동일 테이블 또는 다른 테이블간의 관계를 지정할 수 있다.
- ON DELETE CASCADE를 사용하여 DETAIL TABLE에서 관련된 행을 삭제하고 MASTER TABLE에서 삭제를 허용할 수 있다.

```
column datatype [CONSTRAINT constraint_name]
                REFERENCES table_name (column1[,column2,...] [ON DELETE CASCADE])
```

```
column datatype,
. . . . .
[CONSTRAINT constraint_name] FOREIGN KEY (column1[,column2,...])
                REFERENCES table_name (column1[,column2,...] [ON DELETE CASCADE])
```

## 02. 테이블 제약조건

### FOREIGN KEY(PK) 예

```
SQL> CREATE TABLE DEPT_TAB (  
    DEPTNO          NUMBER(2),  
    DNAME           CHAR(14),  
    LOC             CHAR(13),  
    CONSTRAINT DEPT_DEPTNO_PK PRIMARY KEY (DEPTNO));    // 기본키 지정  
Table created.
```

```
SQL> CREATE TABLE EMP_TAB (  
    EMPNO           NUMBER(4),  
    ENAME           VARCHAR2(10),  
    JOB             VARCHAR2(9),  
    MGR             NUMBER(4) REFERENCES EMP_TAB (EMPNO),    // 동일 테이블  
    HIREDATE        DATE,  
    SAL             NUMBER(7,2),  
    COMM            NUMBER(7,2),  
    DEPTNO          NUMBER(2) NOT NULL,  
    FOREIGN KEY (DEPTNO) REFERENCES DEPT_TAB (DEPTNO),    // 다른 테이블  
    PRIMARY KEY (EMPNO));    // CONSTRAINT절 생략 가능  
Table created.
```

## 02. 테이블 제약조건

### 다) UNIQUE KEY(UK)

- UNIQUE Key 무결성 제약 조건은 열 또는 열 집합의 값들이 유일해야 한다.
- 중복된 값을 가지는 행이 존재할 수 없음을 의미한다.
- PRIMARY KEY와 유사하나 NULL을 허용한다.
- 열이 하나 이상 포함되어 있다면 composite unique key라 부릅니다.
- UNIQUE Key에 대하여 UNIQUE INDEX가 자동 생성된다.

column datatype [CONSTRAINT constraint_name] UNIQUE
column datatype, [CONSTRAINT constraint_name] UNIQUE (column1[,column2,...])

## 02. 테이블 제약조건

### UNIQUE KEY(PK) 예

```
SQL> CREATE TABLE UNI_TAB1 (  
    DEPTNO          NUMBER(2) CONSTRAINT UNI_TAB_DEPTNO_UK UNIQUE,  
    DNAME           CHAR(14),  
    LOC             CHAR(13));
```

Table created.

```
INSERT INTO UNI_TAB1 VALUES(1,'AAAA','BBBB');
```

```
INSERT INTO UNI_TAB1 VALUES(1,'AAAA','BBBB');
```

무결성 제약 조건(SCOTT.UNI\_TAB\_DEPTNO\_UK)에 위배됩니다.

```
SQL> CREATE TABLE UNI_TAB2 (  
    DEPTNO          NUMBER(2),  
    DNAME           CHAR(14),  
    LOC             CHAR(13),  
    CONSTRAINT UNI_TAB_DEPTNO_UK UNIQUE (DEPTNO));
```

Table created.

## 02. 테이블 제약조건

### 라) NOT NULL(NN)

- NOT NULL 제약 조건은 열에서 NULL을 허용하지 않도록 보증한다.
- NOT NULL 제약 조건이 없는 열은 DEFAULT로 NULL을 허용한다.
- NOT NULL 제약 조건은 컬럼 제약조건에서만 가능하다.

column datatype [CONSTRAINT constraint_name] NOT NULL
column datatype, [CONSTRAINT constraint_name] NOT NULL (column1[,column2,...])

## 02. 테이블 제약조건

### NOT NULL 예

```
SQL> CREATE TABLE NN_TAB1 (  
    DEPTNO          NUMBER(2) CONSTRAINT UNI_TAB_DEPTNO_NN NOT NULL,  
    DNAME           CHAR(14),  
    LOC             CHAR(13));
```

Table created.

```
INSERT INTO NN_TAB1(DNAME,LOC) VALUES('AAAA','BBBB');
```

NULL을 ("SCOTT"."NN\_TAB1"."DEPTNO") 안에 삽입할 수 없습니다.

```
SQL> CREATE TABLE NN_TAB2 (  
    DEPTNO          NUMBER(2),  
    DNAME           CHAR(14),  
    LOC             CHAR(13),  
    CONSTRAINT UNI_TAB_DEPTNO_NN NOT NULL (DEPTNO)); // ERROR 발생  
    CONSTRAINT UNI_TAB_DEPTNO_NN NOT NULL (DEPTNO))
```

\*

ERROR at line 5:

ORA-00904: invalid column name



## 02. 테이블 제약조건

### 마) CHECK(CK)

- 특정 열(컬럼)에 대해서 조건을 지정하여 조건에 만족하지는지를 점검한다.
- 조건에 만족하지 않은 값이 저장될 경우 오류를 반환한다.
- 다음과 같은 표현식은 허용되지 않는다.
  - CURRVAL, NEXTVAL, LEVEL, ROWNUM에 대한 참조
  - SYSDATE, UID, USER, USERENV 함수에 대한 호출
  - 다른 행에 있는 값을 참조하는 질의
  - ORACLE SERVER가 사용하는 예약어

```
column datatype [CONSTRAINT constraint_name] CHECK (condition)
```

```
column datatype,
```

```
.....  
[CONSTRAINT constraint_name] CHECK (condition)
```

## 02. 테이블 제약조건

### CHECK 예

```
SQL> CREATE TABLE CK_TAB1 (  
    DEPTNO      NUMBER(2) NOT NULL CHECK (DEPTNO IN (10,20,30,40,50)),  
    DNAME       CHAR(14),  
    LOC         CHAR(13));
```

Table created.

```
INSERT INTO CK_TAB3 VALUES(10,'AAAA','BBBB'); // 레코드 삽입
```

```
INSERT INTO CK_TAB3 VALUES(60,'AAAA','BBBB'); // 체크 제약조건 위배
```

```
SQL> CREATE TABLE CK_TAB2 (  
    DEPTNO      NUMBER(2) NOT NULL,  
    DNAME       CHAR(14),  
    LOC         CHAR(13),  
    CONSTRAINT CK_TAB_DEPTNO_CK CHECK (DEPTNO IN (10,20,30,40,50)));
```

Table created.

## 02. 테이블 구조 변경하는 ALTER TABLE

### ALTER TABLE 명령문

- 기존 테이블의 구조를 변경하기 위한 DDL 명령문
- 컬럼 추가, 삭제, 타입이나 길이를 변경할 때 사용
- 테이블의 구조를 변경하게 되면 기존에 저장되어 있던 데이터에 영향을 준다.
- 칼럼 추가, 수정, 삭제하기 위해서는 다음과 같은 명령어 사용
  - ADD COLUMN 절 : 새로운 칼럼 추가
  - MODIFY COLUMN 절 : 기존 칼럼 수정
  - DROP COLUMN 절 : 기존 칼럼 삭제



## 2.1 새로운 컬럼 추가하기

### ALTER TABLE ADD문

- 기존 테이블에 새로운 컬럼 추가
- 새로운 컬럼은 테이블 맨 마지막에 추가되므로 자신이 원하는 위치에 만들어 넣을 수 없다.
- 또한 이미 이전에 추가해 놓은 로우가 존재한다면 그 로우에도 컬럼이 추가되지만, 컬럼 값은 NULL 값으로 입력된다.

```
ALTER TABLE table_name  
ADD (column_name, data_type expr, ...);
```



## <실습하기> EMP01 테이블에 JOB 컬럼 추가하기

- EMP01 테이블에 문자 타입의 직급(JOB) 컬럼을 추가

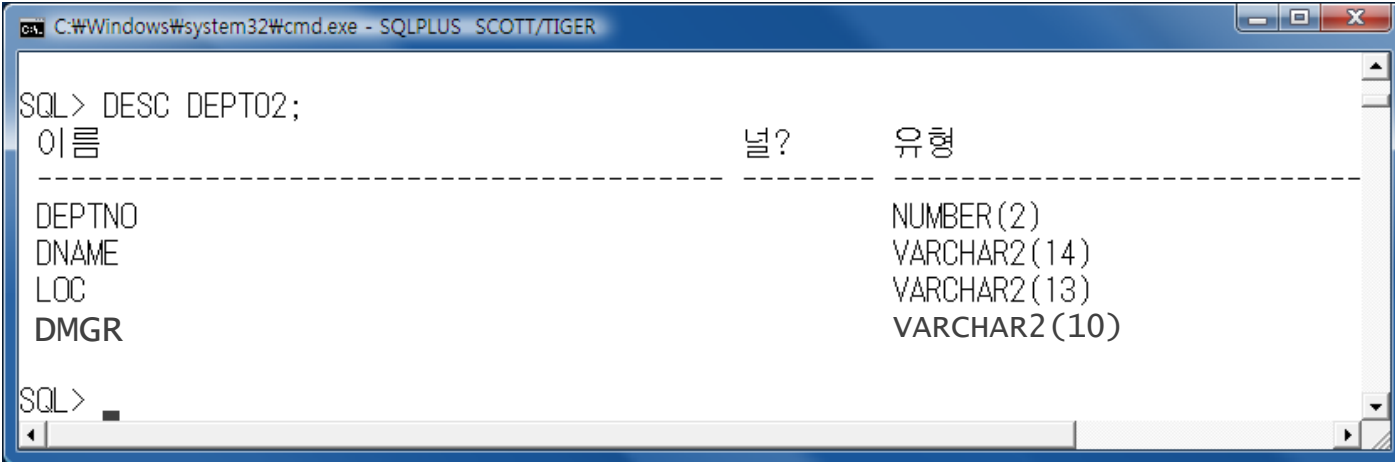
```
ALTER TABLE EMP01  
ADD(JOB VARCHAR2(9));
```

```
DESC emp01;  
이름      널  유형  
-----  
EMPNO     NUMBER(4)  
ENAME     VARCHAR2(20)  
SAL       NUMBER(7,2)
```

```
ALTER TABLE EMP01  
ADD(JOB VARCHAR2(9));  
DESC emp01;  
이름      널  유형  
-----  
EMPNO     NUMBER(4)  
ENAME     VARCHAR2(20)  
SAL       NUMBER(7,2)  
JOB       VARCHAR2(9)
```

## <과제3>

- 3. DEPT02 테이블에 문자 타입의 부서장(DMGR) 칼럼을 추가하시오.



```
C:\Windows\system32\cmd.exe - SQLPLUS SCOTT/TIGER

SQL> DESC DEPT02;
이름                                널?       유형
-----
DEPTNO                             NUMBER(2)
DNAME                              VARCHAR2(14)
LOC                                VARCHAR2(13)
DMGR                                VARCHAR2(10)

SQL>
```

## 2.2 기존 컬럼 속성 변경하기

### ALTER TABLE MODIFY문

- 다음과 같은 형식으로 테이블에 존재하는 컬럼을 변경할 수 있다.

```
ALTER TABLE table_name  
MODIFY (column_name, data_type expr, ...);
```

- 컬럼을 변경한다는 것은 컬럼에 대해서 데이터 타입이나 크기, 기본값들을 변경한다는 의미



## <실습하기> 컬럼의 크기 변경하기

- 1. 직급(JOB) 컬럼을 최대 30글자까지 저장할 수 있게 변경하시오.

```
ALTER TABLE EMP01  
MODIFY(JOB VARCHAR2(30));
```

```
ALTER TABLE EMP01  
MODIFY(JOB VARCHAR2(30));  
DESC emp01;
```

table EMP01이(가) 변경되었습니다.

DESC emp01

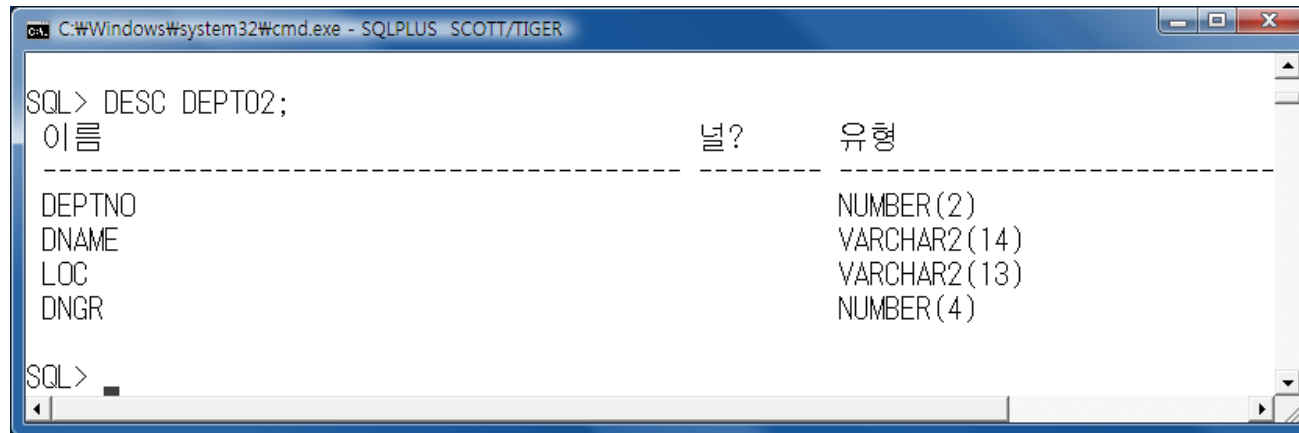
이름      널      유형

EMPNO		NUMBER(4)
ENAME		VARCHAR2(20)
SAL		NUMBER(7,2)
JOB		VARCHAR2(30)



## <과제4>

- 4. DEPT02 테이블의 부서장(DMGR) 칼럼을 숫자 타입으로 변경하시오.



The screenshot shows a command window titled "C:\Windows\system32\cmd.exe - SQLPLUS SCOTT/TIGER". The user has entered the command "SQL> DESC DEPT02;". The output displays the table structure with columns and their data types:

이름	널?	유형
DEPTNO		NUMBER (2)
DNAME		VARCHAR2 (14)
LOC		VARCHAR2 (13)
DNGR		NUMBER (4)

The prompt "SQL>" is visible at the bottom left of the window.

## 2.3 기존 컬럼 삭제

### ALTER TABLE DROP문

- 테이블에 이미 존재하는 컬럼을 삭제한다.
- ALTER TABLE ~ DROP COLUMN 명령어로 컬럼을 삭제할 수 있다.

```
ALTER TABLE table_name  
DROP COLUMN column_name;
```



## <실습하기> 직급 컬럼 삭제하기

- 1. EMP01 테이블의 직급 컬럼을 삭제하기

```
ALTER TABLE EMP01  
DROP COLUMN JOB;
```

```
ALTER TABLE EMP01  
DROP COLUMN JOB;  
DESC emp01;
```

table EMP01이(가) 변경되었습니다.

DESC emp01

이름	널	유형
----	---	----

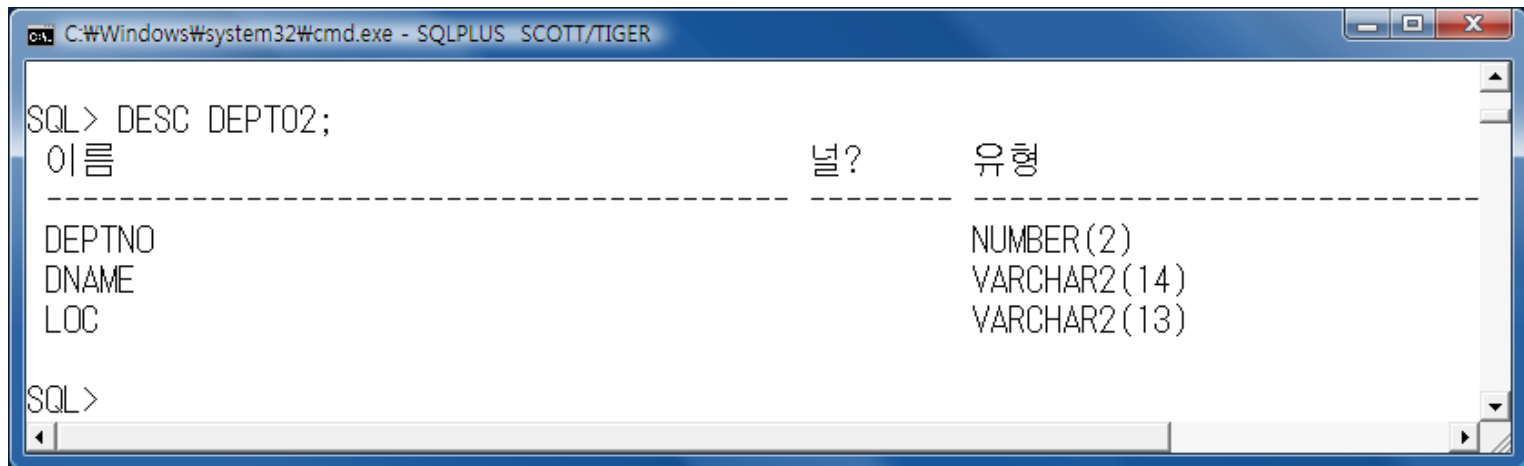
EMPNO		NUMBER(4)
-------	--	-----------

ENAME		VARCHAR2(20)
-------	--	--------------

SAL		NUMBER(7,2)
-----	--	-------------

## <과제5>

- 5. DEPT02 테이블의 부서장(DMGR) 칼럼을 삭제하시오.



```
C:\Windows\system32\cmd.exe - SQLPLUS SCOTT/TIGER

SQL> DESC DEPT02;
이름                널?       유형
-----
DEPTNO              NUMBER(2)
DNAME               VARCHAR2(14)
LOC                 VARCHAR2(13)

SQL>
```

### 03. 테이블의 모든 ROW 제거

- 기존에 사용하던 테이블의 모든 레코드(row)를 제거하기 위한 명령어로 TRUNCATE가 제공한다.

```
TRUNCATE table_name
```



## <실습하기> 테이블의 내용 전체 제거하기

- 테이블 EMP02 에 저장된 데이터를 확인하였으면 테이블의 모든 레코드를 제거한다.

```
TRUNCATE TABLE EMP02;
```

```
// 레코드 검색  
SELECT * FROM EMP02;
```

```
// 레코드 제거  
TRUNCATE TABLE EMP02;
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7521	WARD	SALESMAN	7698	81/02/22	1250	500	30
2	7369	SMITH	CLERK	7902	80/12/17	800	(null)	20
3	7499	ALLEN	SALESMAN	7698	81/02/20	1600	300	30
4	7566	JONES	MANAGER	7839	81/04/02	2975	(null)	20
5	7654	MARTIN	SALESMAN	7698	81/09/28	1250	1400	30
6	7698	BLAKE	MANAGER	7839	81/05/01	2850	(null)	30
7	7782	CLARK	MANAGER	7839	81/06/09	2450	(null)	10
8	7788	SCOTT	ANALYST	7566	87/04/19	3000	(null)	20
9	7839	KING	PRESIDENT	(null)	81/11/17	5000	(null)	10
10	7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30
11	7876	ADAMS	CLERK	7788	87/05/23	1100	(null)	20
12	7900	JAMES	CLERK	7698	81/12/03	950	(null)	30
13	7902	FORD	ANALYST	7566	81/12/03	3000	(null)	20
14	7934	MILLER	CLERK	7782	82/01/23	1300	(null)	10

## 04. 테이블 구조 삭제하는 DROP TABLE

- DROP TABLE문은 기존 테이블을 제거한다.

```
DROP TABLE table_name;
```



## <실습하기> 테이블 삭제하기

- 1. CREATE TABLE을 학습할 때 만들어 놓았던 EMP01 테이블을 삭제한다.

```
DROP TABLE EMP01;
```

```
DROP TABLE EMP01;  
  
SELECT * FROM TAB;
```



```
// 삭제 시 임시파일 제거  
DROP TABLE EMP01 PURGE;  
  
// 전체 임시파일 제거  
purge recyclebin;
```

	TNAME	TABTYPE	CLUSTERID
42	BBS	TABLE	(null)
43	EMP02	TABLE	(null)
44	EMP03	TABLE	(null)
45	EMP05	TABLE	(null)
46	EMP06	TABLE	(null)
47	DEPT_TAB	TABLE	(null)
48	UNI_TAB1	TABLE	(null)
49	EMP_TAB	TABLE	(null)
50	SPRINGGUEST	TABLE	(null)
51	GONGJI	TABLE	(null)
52	SPRINGBOARD	TABLE	(null)
53	SPRINGBBS	TABLE	(null)
54	SPRINGADMIN	TABLE	(null)
55	SPRINGGONGJI	TABLE	(null)
56	BIN\$QIQmTu9tSS2B0XBx8iVVag==\$0	TABLE	(null)
57	BIN\$ym75sHnoTyqBb5FWJwz5Rg==\$0	TABLE	(null)

임시파일



## 05. 데이터 사전과 뷰

### 데이터 사전(Data Dictionary)

- 데이터베이스 자원을 효율적으로 관리하기 위한 다양한 정보를 저장하는 시스템 테이블
- 사용자가 테이블을 생성하거나 사용자를 변경하는 등의 작업을 할 때 데이터베이스 서버에 의해 자동으로 갱신되는 테이블
- 사용자는 데이터 디렉터리의 내용을 직접 수정하거나 삭제 할 수 없다.(조회 불가)



## 05. 데이터 사전과 뷰

### 데이터 사전 뷰(DATA Dictionary View)

- 사용자가 이해할 수 있는 데이터를 산출해 줄 수 있도록 하기 위해  
서 데이터 사전에 만들어진 데이터사전 뷰 제공
- 데이터 사전 뷰는 접두어에 따라 다음의 세 종류로 분류

접두어	의미
DBA_XXXX	데이터베이스 관리자만 접근 가능한 객체 등의 정보 조회 (DBA는 모두 접근 가능하므로 결국 디비에 있는 모든 객체 에 관한 조회)
ALL_XXXX	자신 계정 소유 또는 권한을 부여 받은 객체 등에 관한 정 보 조회
USER_XXXX	자신의 계정이 소유한 객체 등에 관한 정보 조회

## 6.1. USER\_ 데이터사전

- 접두어로 USER가 붙은 데이터 사전은 사용자 계정이 소유한 객체 등에 관한 정보 조회
- USER가 붙은 데이터 사전 중에서 자신이 생성한 테이블, 인덱스, 뷰 등과 같은 사용자 계정 소유의 객체 정보를 저장한 USER\_TABLES 데이터 사전 뷰를 살펴본다.



## <실습하기> USER\_TABLES 데이터 딕셔너리 뷰 살펴보기

1. DESC 명령어로 데이터 사전 뷰(USER\_TABLES) 구조보기

```
DESC USER_TABLES;
```

2. USER\_TABLES 데이터 사전 뷰는 현재 접속한 사용자 계정이 소유한 모든 테이블 정보를 조회 할 수 있는 뷰 이기에 현재 사용자가 누구 인지를 볼 수 있다.

```
SHOW USER;
```

3. 데이터 사전 USER\_TABLES에서 테이블의 이름만 내림차순으로 볼 수 있다.

```
SELECT TABLE_NAME FROM USER_TABLES  
ORDER BY TABLE_NAME DESC;
```