

09장. 테이블 구조 생성, 변경 및 삭제하는 DDL

이 장에서 다룰 내용

- 1 **CREATE TABLE로 테이블 구조 정의하기**.....●
- 2 **ALTER TABLE로 테이블 구조 변경하기**.....●
- 3 **DROP TABLE로 테이블 구조 삭제하기**.....●
- 4 **테이블의 모든 로우를 제거하는 TRUNCATE**.....●
- 5 **테이블 명을 변경하는 RENAME**.....●
- 6 **데이터 디렉터리와 데이터 디렉터리 뷰**

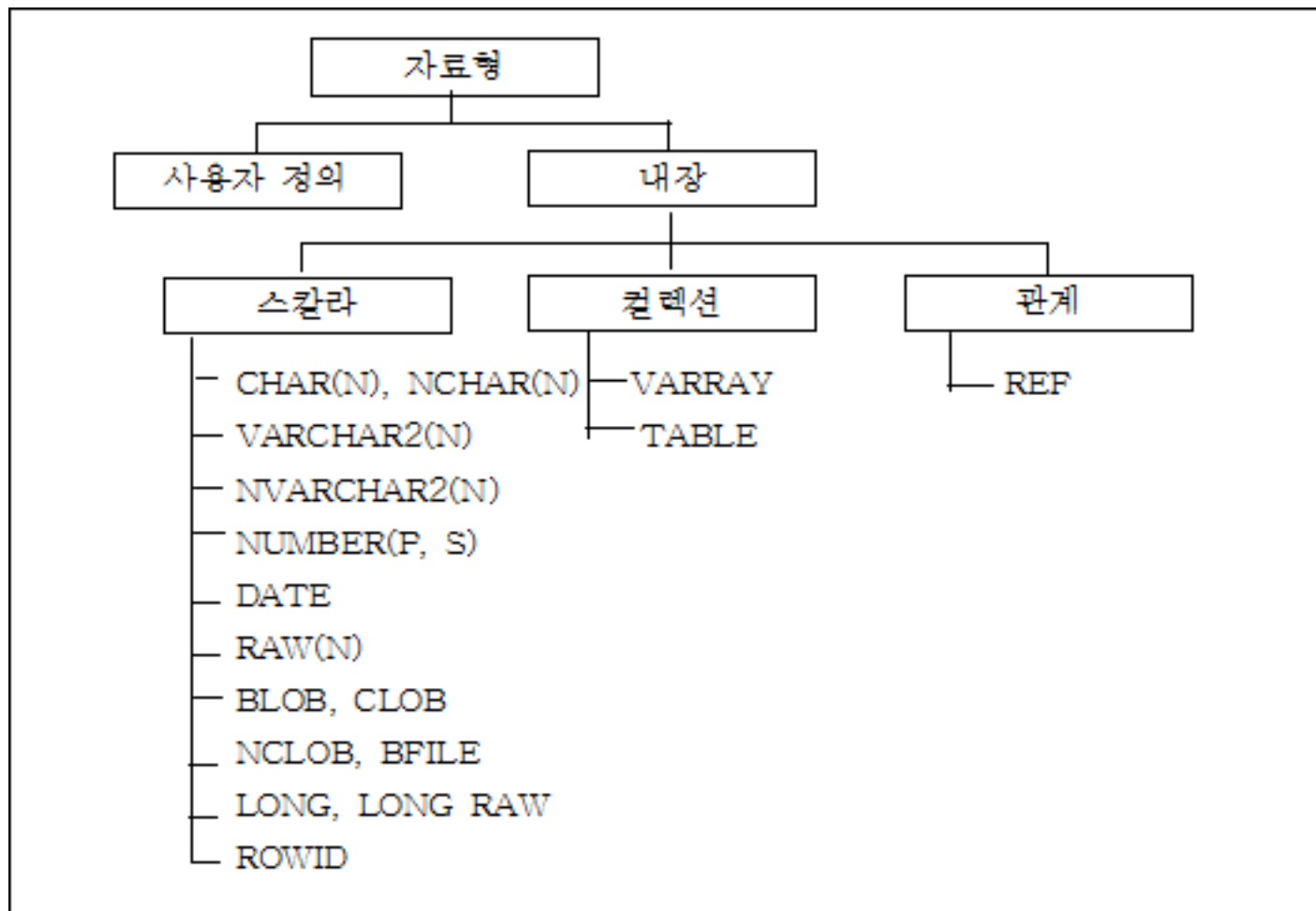
01. 테이블 구조 정의하는 CREATE TABLE

- ❖ 지금까지는 오라클에서 학습용으로 제공해 주는 테이블을 사용하여 다양한 조회를 해 보았습니다.
- ❖ 이번 장에서는 DDL(Data Definition Language)을 사용하여 테이블 구조 자체를 새롭게 생성, 수정, 삭제해 보겠습니다.
- ❖ 우선 CREATE TABLE 명령어로 새로운 테이블을 생성해 보겠습니다.
- ❖ 다음은 CREATE TABLE 문의 기본 형식입니다.

```
CREATE TABLE table_name  
(column_name, data_type expr, ...);
```

1.1 데이터 형

- ❖ 데이터베이스에는 문자, 숫자, 날짜, 이미지 등과 같은 다양한 형태의 데이터가 저장됩니다. 다음은 오라클에서 제공되는 데이터 형의 종류입니다.



1.1 데이터 형

이름	비고
CHAR(size)	고정 길이 문자 데이터. VARCHAR2와 동일한 형태의 자료를 저장할 수 있고, 입력된 자료의 길이와는 상관없이 정해진 길이만큼 저장 영역 차지. 최소 크기는 1
VARCHAR2(size)	Up to 2000 Bytes 가변 길이 문자 데이터. 실제 입력된 문자열의 길이만큼 저장 영역을 차지. 최대 크기는 명시해야 하며, 최소 크기는 1
NUMBER	Internal Number Format 최고 40자리까지의 숫자를 저장할 수 있습니다. 이때 소수점이나 부호는 길이에 포함되지 않는다.
NUMBER(w)	W자리까지의 수치로 최대 38자리까지 가능하다. (38자리가 유효 숫자이다.)
NUMVER(w, d)	W는 전체 길이, d는 소수점 이하 자릿수이다. 소수점은 자릿수에 포함되지 않는다.
DATE	BC 4712년 1월 1일~AD 4712년 12월 31일까지의 날짜

1.1 데이터 형

이름	비고
LONG	가변 길이의 문자형 데이터 타입, 최대 크기는 2GB
LOB	2GB까지의 가변 길이 바이너리 데이터를 저장시킬 수 있습니다 이미지 문서, 실행 파일을 저장할 수 있습니다.
ROWID	ROWID는 Tree-piece Format을 가짐. ROWID는 DB에 저장되어 있지 않으며, DB Data도 아니다.
BFILE	대용량의 바이너리 데이터를 파일 형태로 저장 최대 4GB
TIMESTAMP(n)	DATE 형의 확장된 형태
INTERVAL YEAR TO MONTH	년과 월을 이용하여 기간을 저장
INTERVAL DAY TO SECOND	일, 시, 분, 초를 이용하여 기간을 저장 두 날짜 값의 정확한 차이를 표현하는데 유용하다.

(1) LOB

- ❖ LOB(Large OBject) 데이터 형은 텍스트, 그래픽 이미지, 동영상, 사운드와 같이 구조화되지 않은 대용량의 텍스트나 멀티미디어 데이터를 저장하기 위한 데이터 형입니다.
- ❖ 최대 4GB 까지 저장 가능합니다. 오라클에서 제공되는 LOB 데이터 형은 BLOB, CLOB, NCLOB, BFILE 등이 있습니다.
- ❖ BLOB는 그래픽 이미지, 동영상, 사운드와 같은 구조화되지 않은 데이터를 저장하기 위해 사용됩니다.
- ❖ CLOB는 e-BOOK과 같은 대용량의 텍스트 데이터를 저장하기 위해서 사용됩니다.
- ❖ NCLOB은 국가별 문자셋 데이터를 저장하고, BFILE은 바이너리 데이터를 파일 형태로 저장합니다.

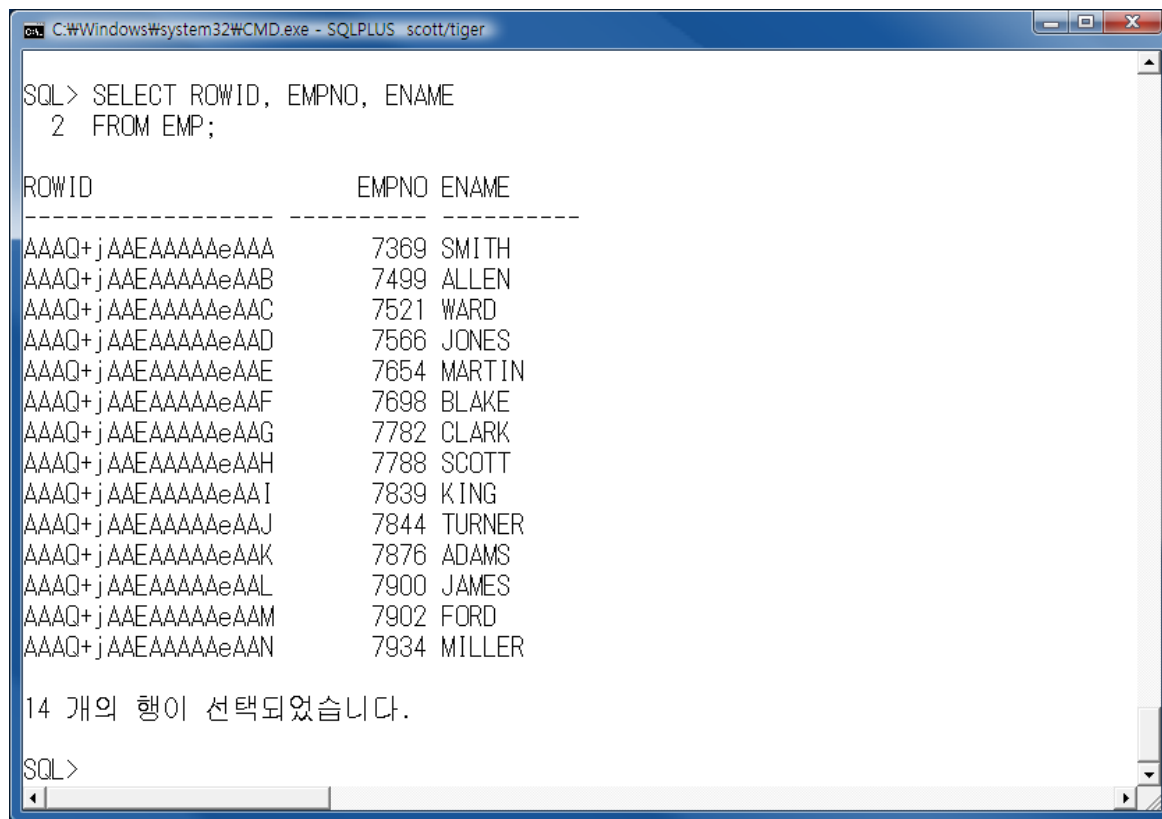
(2) ROWID

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

(2) ROWID

- ❖ 다음은 SELECT 문을 통해서 emp 테이블의 ROWID를 살펴보겠습니다.

```
SELECT ROWID, EMPNO, ENAME  
FROM EMP;
```



```
C:\Windows\system32\CMD.exe - SQLPLUS scott/tiger  
  
SQL> SELECT ROWID, EMPNO, ENAME  
2 FROM EMP;  
  
ROWID                EMPNO  ENAME  
-----  
AAAAQ+jAAEAAAAAeAAA  7369  SMITH  
AAAAQ+jAAEAAAAAeAAB  7499  ALLEN  
AAAAQ+jAAEAAAAAeAAC  7521  WARD  
AAAAQ+jAAEAAAAAeAAD  7566  JONES  
AAAAQ+jAAEAAAAAeAAE  7654  MARTIN  
AAAAQ+jAAEAAAAAeAAF  7698  BLAKE  
AAAAQ+jAAEAAAAAeAAG  7782  CLARK  
AAAAQ+jAAEAAAAAeAAH  7788  SCOTT  
AAAAQ+jAAEAAAAAeAAI  7839  KING  
AAAAQ+jAAEAAAAAeAAJ  7844  TURNER  
AAAAQ+jAAEAAAAAeAAK  7876  ADAMS  
AAAAQ+jAAEAAAAAeAAL  7900  JAMES  
AAAAQ+jAAEAAAAAeAAM  7902  FORD  
AAAAQ+jAAEAAAAAeAAN  7934  MILLER  
  
14 개의 행이 선택되었습니다.  
  
SQL>
```

(2) ROWID

- ❖ ROWID는 다음과 같은 형식으로 데이터를 저장합니다.

(총80비트:10바이트)			
32bit	10bit	22bit	16bit
00000	FFF	BBBBB	RRR
데이터객체번호	상대적파일번호	블록번호	행번호

- ❖ 데이터 객체번호는 테이블이나 인덱스와 같은 데이터 객체가 생성될 때 할당됩니다.
- ❖ 상대적인 파일번호는 데이터가 저장되는 물리적인 데이터 파일 번호로서 유일한 값을 가집니다.
- ❖ 블록번호 데이터 파일 내에서 행을 포함한 블록 위치입니다.
- ❖ 행 번호는 블록 내에서 행 위치를 나타내는 번호입니다.

(3) TIMESTAMP

- ❖ **TIMESTAMP 데이터 형은 DATE 형의 확장된 형태로서 백만분의 일초 단위까지 표현할 수 있습니다.**

(4) INTERVAL YEAR TO MONTH

- ❖ INTERVAL YEAR TO MONTH 형은 년과 월을 사용하여 두 날짜 사이의 기간을 저장하기 위한 데이터 형입니다.

INTERVAL YEAR(년도에 대한 자릿수) TO MONTH(달에 대한 자릿수)

- ❖ 자릿수를 지정하지 않으면 기본적으로 2자리 잡힙니다.

〈실습하기〉 컬럼에 기간(3년)을 저장하기

INTERVAL YEAR TO MONTH 형으로 컬럼을 생성하여 3년이란 기간을 저장해 봅시다.

1. 다음은 SAM02란 이름의 테이블을 새롭게 생성하는 명령어입니다.

```
CREATE TABLE SAM02(  
YEAR01 INTERVAL YEAR(3) TO MONTH);
```

2. 생성된 테이블에 기간을 36개월로 저장합니다.

```
INSERT INTO SAM02  
VALUES(INTERVAL '36' MONTH(3));
```

3. 오늘 날짜를 출력하고 테이블 SAM02의 YEAR01 이라는 컬럼에 저장된 날짜만큼 지난 날짜를 계산하여 출력해 봅시다.

```
SELECT YEAR01, SYSDATE, SYSDATE+YEAR01 FROM SAM02;
```

(5) INTERVAL DAY TO SECOND

- ❖ INTERVAL DAY TO SECOND 형은 일, 시, 분, 초를 사용하여 두 날짜 사이의 기간을 저장하기 위한 데이터 형입니다.

INTERVAL DAY(일수에 대한 자릿수) TO SECOND(초에 대한 자릿수)

- ❖ 자릿수를 지정하지 않으면 기본적으로 2자리 잡힙니다.

〈실습하기〉 컬럼에 기간(100일)을 저장하기

INTERVAL DAY TO SECOND 형으로 컬럼을 생성하여 100일이란 기간을 저장해 봅시다.

1. 다음은 SAM03란 이름의 테이블을 새롭게 생성하는 명령어입니다.

```
CREATE TABLE SAM03(  
DAY01 INTERVAL DAY(3) TO SECOND);
```

2. 생성된 테이블에 기간을 100일을 저장합니다.

```
INSERT INTO SAM03  
VALUES(INTERVAL '100' DAY(3));
```

3. 오늘 날짜를 출력하고 테이블 SAM03의 DAY01 이라는 컬럼에 저장된 날짜만큼 지난 날짜를 계산하여 출력해 봅시다. .

```
SELECT DAY01, SYSDATE, SYSDATE+DAY01 FROM SAM03;
```

〈실습하기〉 새롭게 테이블 생성하기

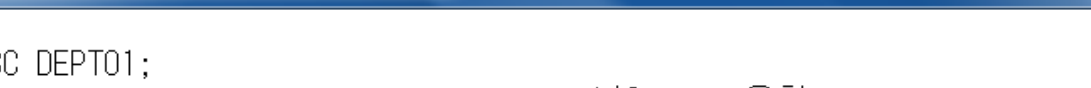
지금까지 실습에 사용했던 사원 테이블과 유사한 구조의 사원번호, 사원 이름, 급여 3개의 칼럼으로 구성된 EMP01 테이블을 생성해 봅시다.

1. CREATE TABLE 명령어로 EMP01 테이블을 새롭게 생성합니다.

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


〈스스로하기〉

1. 다음과 같은 구조의 테이블을 CREATE TABLE 명령어로 생성하되 테이블의 이름은 DEPT01 로 하시오.)



```

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

SQL> DESC DEPT01;

  이름                길이?      유형
-----
DEPTNO                NUMBER(2)
DNAME                 VARCHAR2(14)
LOC                  VARCHAR2(13)

SQL>
  
```

〈실습하기〉 서브 쿼리로 테이블 생성하기

CREATE TABLE 문에서 서브 쿼리를 사용하여 이미 존재하는 테이블과 동일한 구조와 내용을 갖는 새로운 테이블을 생성할 수 있습니다.

1. CREATE TABLE 명령어 다음에 컬럼을 일일이 정의하는 대신 AS 절을 추가하여 EMP 테이블과 동일한 내용과 구조를 갖는 EMP02 테이블을 생성해 봅시다.

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

〈실습하기〉 원하는 컬럼으로 구성된 복제 테이블 생성하기

기존 테이블에서 원하는 컬럼만 선택적으로 복사해서 생성할 수도 있습니다.

1. 서브 쿼리문의 SELECT 절에 * 대신 원하는 컬럼명을 명시하면 기존 테이블에서 일부의 컬럼만 복사할 수 있습니다.

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

〈탄탄히 다지기〉

2. EMP 테이블을 복사하되 사원번호, 사원이름, 급여 컬럼으로 구성된 테이블을 생성하시오.(테이블의 이름은 EMP04 로 하시오.)



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe - sqlplus scott/tiger". The prompt is "SQL> SELECT * FROM EMP04;". The output displays two tables of employee data. The first table has 10 rows and the second table has 3 rows. Both tables have columns EMPNO, ENAME, and SAL. Below the tables, a message states "14 개의 행이 선택되었습니다." (14 rows selected).

```
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 생성할 수도 있습니다.

1. 서브 쿼리문의 SELECT 문을 구성할 때 WHERE 절을 추가하여 원하는 조건을 제시하면 기존 테이블에서 일부의 행만 복사합니다.

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

1.2 테이블의 구조만 복사하기

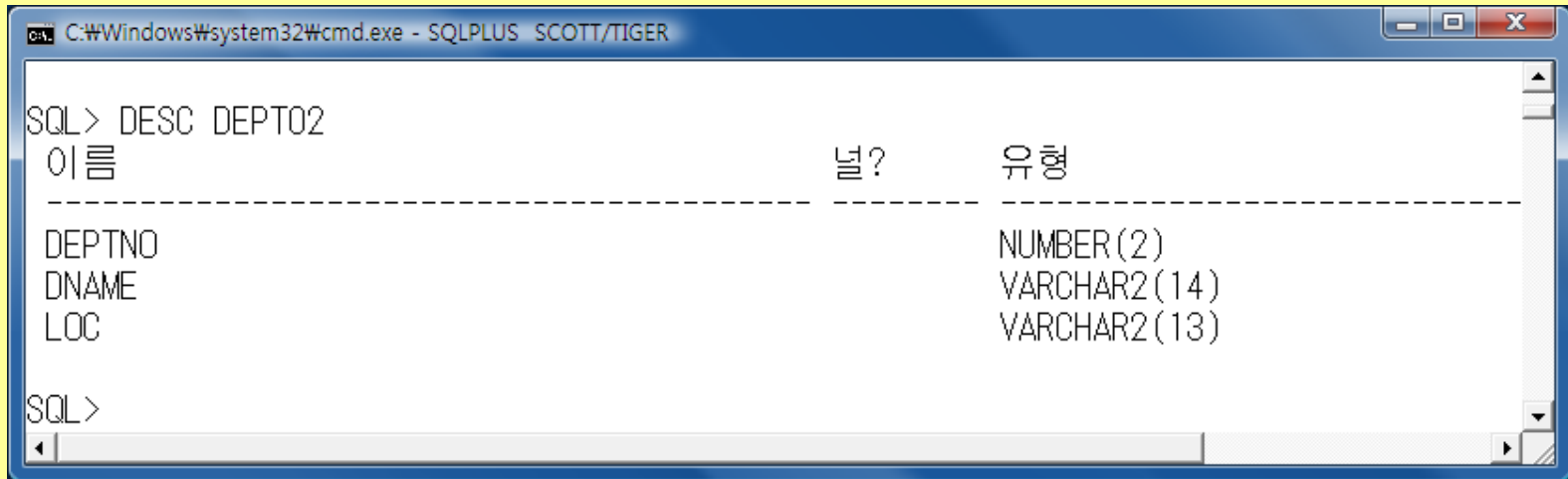
- ❖ 서브 쿼리를 이용하여 테이블을 복사하되 데이터는 복사하지 않고 기존 테이블의 구조만 복사하는 것을 살펴봅시다.
- ❖ 테이블의 구조만 복사하는 것은 별도의 명령이 있는 것이 아닙니다. 이 역시 서브 쿼리를 이용해야 하는데 WHERE 조건 절에 항상 거짓이 되는 조건을 지정하게 되면 테이블에서 얻어질 수 있는 로우가 없게 되므로 빈 테이블이 생성되게 됩니다.

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

- ❖ WHERE 1=0; 조건은 항상 거짓입니다. 이를 이용하여 테이블의 데이터는 가져오지 않고 구조만 복사하게 됩니다.

<탄탄히 다지기>

3. DEPT 테이블과 동일한 구조의 빈 테이블 생성하기 생성하시오.(테이블의 이름은 DEPT02 로 하시오.)



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - SQLPLUS SCOTT/TIGER". The user has entered the command "SQL> DESC DEPT02". The output displays the structure of the DEPT table, which is used as a template for creating DEPT02. The output is formatted as follows:

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

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

02. 테이블 구조 변경하는 ALTER TABLE

- ❖ ALTER TABLE 명령문은 기존 테이블의 구조를 변경하기 위한 DDL 명령문입니다. 테이블에 대한 구조 변경은 컬럼의 추가, 삭제, 컬럼의 타입이나 길이를 변경할 때 사용합니다. 테이블의 구조를 변경하게 되면 기존에 저장되어 있던 데이터에 영향을 주게 됩니다.
- ❖ ALTER TABLE로 칼럼 추가, 수정, 삭제하기 위해서는 다음과 같은 명령어를 사용합니다.
 - ADD COLUMN 절을 사용하여 새로운 칼럼을 추가한다.
 - MODIFY COLUMN 절을 사용하여 기존 칼럼을 수정한다.
 - DROP COLUMN 절을 사용하여 기존 칼럼을 삭제한다.

2.1 새로운 컬럼 추가하기

- ❖ ALTER TABLE ADD 문은 기존 테이블에 새로운 컬럼을 추가합니다.
- ❖ 새로운 컬럼은 테이블 맨 마지막에 추가되므로 자신이 원하는 위치에 만들어 넣을 수 없습니다.
- ❖ 또한 이미 이전에 추가해 놓은 로우가 존재한다면 그 로우에도 컬럼이 추가되지만, 컬럼 값은 NULL 값으로 입력됩니다.

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

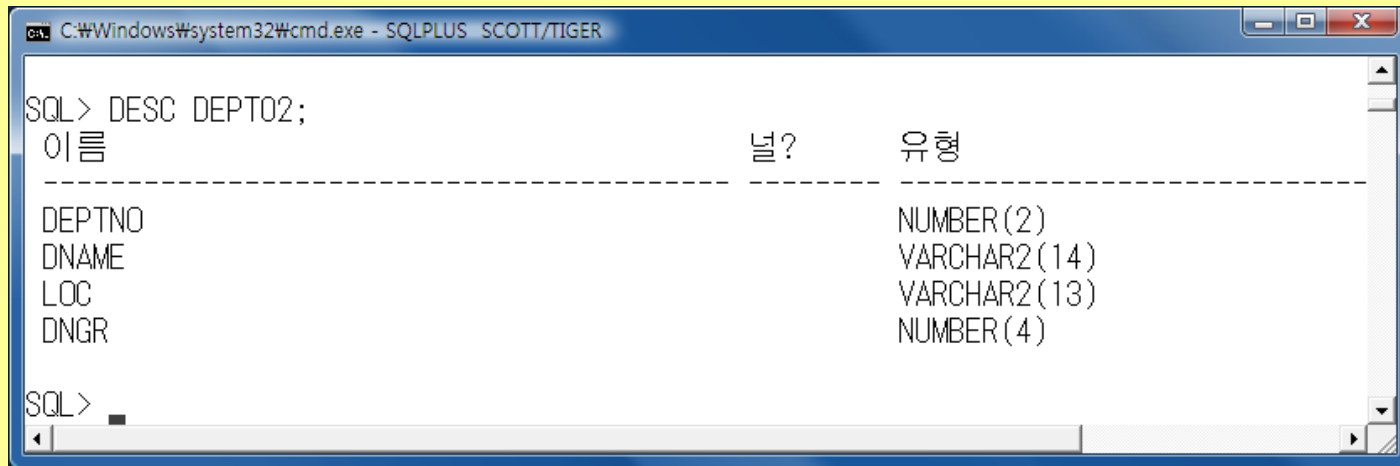
〈실습하기〉 EMP01 테이블에 JOB 컬럼 추가하기

EMP01 테이블에 문자 타입의 직급(JOB) 칼럼을 추가해 봅시다.

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

<탄탄히 다지기>

3. DEPT02 테이블에 문자 타입의 부서장(DMGR) 칼럼을 추가해 봅시다.



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

SQL> DESC DEPT02;
이름                널?       유형
-----
DEPTNO              NUMBER(2)
DNAME               VARCHAR2(14)
LOC                 VARCHAR2(13)
DMGR                NUMBER(4)

SQL>
```

2.2 기존 컬럼 속성 변경하기

- ❖ ALTER TABLE MODIFY 문을 다음과 같은 형식으로 사용하면 테이블에 이미 존재하는 컬럼을 변경할 수 있게 됩니다.

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

- ❖ 컬럼을 변경한다는 것은 컬럼에 대해서 데이터 타입이나 크기, 기본값들을 변경한다는 의미입니다.

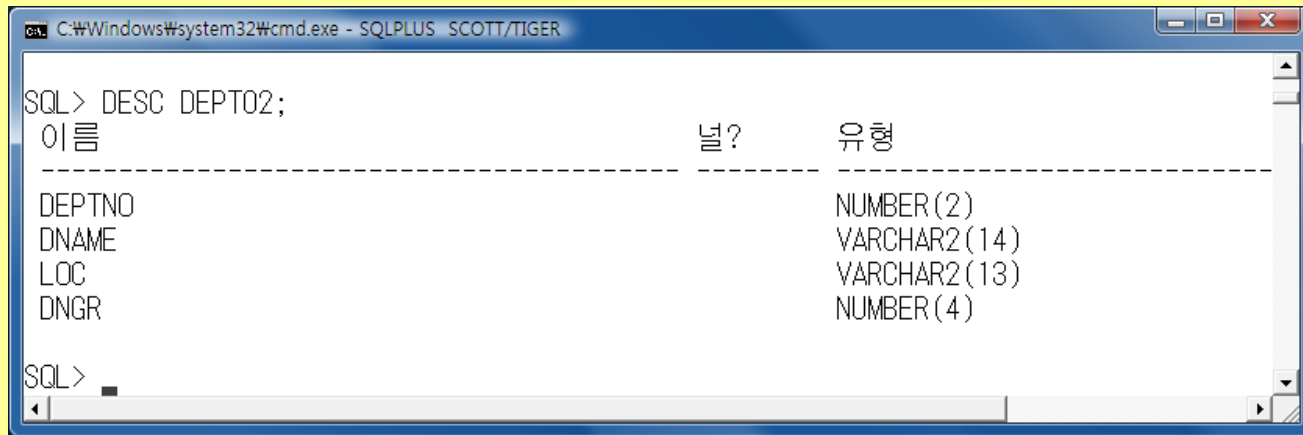
〈실습하기〉 컬럼의 크기 변경하기

1. 직급(JOB) 컬럼을 최대 30글자까지 저장할 수 있게 변경해 보도록 하자.

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

<탄탄히 다지기>

4. DEPT02 테이블의 부서장(DMGR) 칼럼을 숫자 타입으로 변경해 봅시다.



The screenshot shows a Windows command prompt 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 for DEPT02, with columns listed on the left and their data types on the right, separated by a dashed line. The columns are DEPTNO, DNAME, LOC, and DNGR. The data types are NUMBER(2), VARCHAR2(14), VARCHAR2(13), and NUMBER(4) respectively. The prompt "SQL>" is visible at the bottom left.

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

2.3 기존 컬럼 삭제

- ❖ 테이블에 이미 존재하는 컬럼을 삭제해 봅시다.
- ❖ ALTER TABLE ~ DROP COLUMN 명령어로 컬럼을 삭제할 수 있습니다.

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

〈실습하기〉 직급 컬럼 삭제하기

1. EMP01 테이블의 직급 컬럼을 삭제해 보도록 합시다.

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


<탄탄히 다지기>

5. DEPT02 테이블의 부서장(DMGR) 칼럼을 삭제해 봅시다.

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

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

SQL>
```

2.4 SET UNUSED 옵션 적용하기

- ❖ 특정 테이블(EMP02)에서 컬럼(JOB)을 삭제하는 경우 다음과 같이 무조건 삭제하는 것은 위험합니다.
- ❖ 테이블에 저장된 내용이 많을 경우(몇 만 건에 대한 자료) 해당 테이블에서 컬럼을 삭제하는 데 꽤 오랜 시간이 걸리게 될 것입니다. 컬럼을 삭제하는 동안에 다른 사용자가 해당 컬럼을 사용하려고 접근하게 되면 지금 현재 테이블이 사용되고 있기 때문에 다른 사용자는 해당 테이블을 이용할 수 없게 됩니다. 이런 경우 작업이 원활하게 진행되지 않고 락(lock)이 발생하게 됩니다.
- ❖ ALTER TABLE 에 SET UNUSED 옵션을 지정하면 컬럼을 삭제하는 것은 아니지만 컬럼의 사용을 논리적으로 제한할 수 있게 됩니다.
- ❖ SET UNUSED 옵션은 사용을 논리적으로 제한할 뿐 실제로 컬럼을 삭제하지 않기 때문에 작업 시간이 오래 걸리지 않습니다. 그렇기 때문에 락이 걸리는 일도 일어나지 않게 됩니다.

〈실습하기〉 직급 컬럼 사용 제한하기

1. SET UNUSED 옵션이 사용되는 용도를 살펴보았으므로 이제 EMP02 테이블의 JOB 컬럼의 사용을 논리적으로 제한해 봅시다.

```
ALTER TABLE EMP02  
SET UNUSED(JOB);
```

2. 가장 사용빈도가 적은 시간에 실제적인 삭제 작업을 진행합니다.

```
ALTER TABLE EMP02  
DROP UNUSED COLUMNS;
```

03. 테이블 구조 삭제하는 DROP TABLE

❖ DROP TABLE문은 기존 테이블을 제거합니다.

```
DROP TABLE table_name;
```

〈실습하기〉 테이블 삭제하기

1. CREATE TABLE을 학습할 때 만들어 놓았던 EMP01 테이블을 삭제해 봅시다.

```
DROP TABLE EMP01;
```

04. 테이블의 모든 로우를 제거하는 TRUNCATE

- ❖ 기존에 사용하던 테이블의 모든 로우를 제거하기 위한 명령어로 TRUNCATE가 제공됩니다.

```
TRUNCATE table_name
```

〈실습하기〉 테이블의 내용 전체 제거하기

테이블 EMP02 에 저장된 데이터를 확인하였으면 이번에는 테이블의 모든 로우를 제거해 보도록 하겠습니다.

```
TRUNCATE TABLE EMP02;
```

05. 테이블 명을 변경하는 RENAME

- ❖ 기존에 사용하던 테이블의 이름을 변경하기 위한 명령어로 RENAME이 제공됩니다.

```
RENAME old_name TO new_name
```

- ❖ EMP02 테이블의 이름을 TEST 란 이름으로 변경합니다.

```
RENAME EMP02 TO TEST;
```


06. 데이터 디렉터리와 데이터 디렉터리 뷰

- ❖ 데이터베이스 자원을 효율적으로 관리하기 위한 다양한 정보를 저장하는 시스템 테이블을 데이터 디렉터리라고 합니다.
- ❖ 데이터 디렉터리는 사용자가 테이블을 생성하거나 사용자를 변경하는 등의 작업을 할 때 데이터베이스 서버에 의해 자동으로 갱신되는 테이블로 사용자는 데이터 디렉터리의 내용을 직접 수정하거나 삭제 할 수 없습니다.
- ❖ 이러한 데이터 디렉터를 사용자가 조회해 보면 시스템이 직접 관리하는 테이블이기에 암호 같은 기호만 보여질 뿐 내용을 알 수 없습니다.
- ❖ 데이터 디렉터리 원 테이블은 직접 조회하기란 거의 불가능한 일입니다.

06. 데이터 디렉터리와 데이터 디렉터리 뷰

- ❖ 의미 있는 자료 조회가 불가능하기에 오라클은 사용자가 이해할 수 있는 데이터를 산출해 줄 수 있도록 하기 위해서 데이터 디렉터리에서 파생한 데이터디렉터리 뷰를 제공합니다.
- ❖ 데이터디렉터리뷰는 접두어 따라 다음의 세 종류가 있습니다.

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

6.1. USER_ 데이터 디렉터리

- ❖ 접두어로 USER가 붙은 데이터 디렉터리는 자신의 계정이 소유한 객체 등에 관한 정보를 조회 합니다.
- ❖ USER가 붙은 데이터 디렉터리 중에서 자신이 생성한 테이블이나 인덱스나 뷰 등과 같은 자신 계정 소유의 객체 정보를 저장한 USER_TABLES 데이터 디렉터리 뷰를 살펴보도록 하겠습니다.

〈실습하기〉 USER_TABLES 데이터 디렉터리 뷰 살펴보기

1. DESC 명령어로 데이터 디렉터리 뷰 USER_TABLES의 구조를 살펴봅시다.

```
DESC USER_TABLES;
```

2. USER_TABLES 데이터 디렉터리 뷰는 현재 접속한 사용자 계정이 소유한 모든 테이블 정보를 조회 할 수 있는 뷰이기에 현재 사용자가 누구인지를 살펴봅시다.

```
SHOW USER
```

3. 데이터 디렉터리 USER_TABLES의 구조를 살펴보면 무수히 많은 컬럼으로 구성되었음을 알 수 있습니다. 이중에서 테이블의 이름을 알려주는 TABLE_NAME 컬럼의 내용을 살펴봅시다. 현재 사용자 계정이 SCOTT 이므로 SCOTT이 사용가능한 테이블의 이름만 알 수 있습니다.

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

6.2. ALL_ 데이터딕셔너리

- ❖ 사용자 계정이 소유한 객체는 자신의 소유이므로 당연히 접근이 가능합니다.
- ❖ 그러나 만일 자신의 계정이 아닌 다른 계정 소유의 테이블이나 시퀀스 등은 어떨까요?
- ❖ 오라클에서는 타계정의 객체는 원천적으로 접근 불가능합니다.
- ❖ 하지만 그 객체의 소유자가 접근할 수 있도록 권한을 부여하면 타계정의 객체에도 접근이 가능합니다.
- ❖ ALL_ 데이터 딕셔너리 뷰는 현재 계정이 접근 가능한 객체, 즉 자신 계정의 소유이거나 접근 권한을 부여 받은 타계정의 객체 등을 조회할 수 있는 데이터 딕셔너리 뷰입니다.
- ❖ 현재 계정이 접근 가능한 테이블의 정보 조회하는 뷰입니다.

〈실습하기〉 ALL_TABLES 데이터 딕셔너리 뷰 살펴보기

1. DESC 명령어로 데이터 딕셔너리 뷰 ALL_TABLES의 구조를 살펴봅시다.

```
DESC ALL_TABLES;
```

2. 데이터 딕셔너리 뷰 ALL_TABLES의 컬럼 역시 종류가 무수히 많습니다. 이 중에서 OWNER, TABLE_NAME 컬럼 값만 살펴보도록 합시다.

```
SELECT OWNER, TABLE_NAME FROM ALL_TABLES;
```

6.3. DBA_ 데이터 디렉터리 뷰

- ❖ DBA_ 데이터 디렉터리는 DBA가 접근 가능한 객체 등을 조회 할 수 있는 뷰입니다.
- ❖ 앞서도 언급했지만 DBA가 접근 불가능한 정보는 없기에 데이터베이스에 있는 모든 객체 등의 의미라 할 수 있습니다.
- ❖ USER_ 와 ALL_ 와 달리 DBA_ 데이터 디렉터리뷰는 DBA 시스템 권한을 가진 사용자만 접근할 수 있습니다.

〈실습하기〉 DBA_TABLES 데이터 딕셔너리 뷰 살펴보기

1. 다음은 테이블 정보를 살펴보기 위해서 DBA_TABLES 데이터 딕셔너리 뷰의 내용을 조회해 봅시다.

```
SELECT TABLE_NAME, OWNER  
FROM DBA_TABLES;
```

2. DBA 권한을 가진 SYSTEM 계정으로 접속하여 다시 한번 DBA_TABLES 데이터 딕셔너리 뷰의 내용을 조회해 봅시다. 사용자 계정과 암호를 소문자로 입력해야 인식한다는 점에 주의하기 바랍니다.

```
CONN system/manager
```

```
SELECT TABLE_NAME, OWNER  
FROM DBA_TABLES;
```


Thank You !