

Create Table

강사 : 강병준

SQL문의 종류

DDL (Data Definition Language) : 데이터와 그 구조를 정의 합니다.

SQL문	내 용
CREATE	데이터베이스 객체를 생성
DROP	데이터베이스 객체를 삭제
ALTER	기존에 존재하는 데이터베이스 객체를 다시 정의하는 역할

DML (Data Manipulation Language) : 데이터의 검색과 수정등의 처리

SQL문	내 용
INSERT	데이터베이스 객체에 데이터를 입력
DELETE	데이터베이스 객체에 데이터를 삭제
UPDATE	기존에 존재하는 데이터베이스 객체안의 데이터 수정
SELECT	데이터베이스 객체로부터 데이터를 검색

DCL (Data Control Language) : 데이터베이스 사용자의 권한을 제어

SQL문	내 용
GRANT	데이터베이스 객체에 권한을 부여
REVOKE	이미 부여된 데이터베이스객체의 권한을 취소

테이블이란 ?

- 테이블은 오라클 데이터베이스의 기본적인 데이터 저장 단위
- 데이터베이스 테이블은 사용자가 접근 가능한 모든 데이터를 보유하며 레코드와 컬럼으로 구성
 - "테이블"이라는 말이 더 많이 사용되지만 테이블의 형식어는 "릴레이션"
 - 컬럼(Column) : 테이블의 각 컬럼은 엔티티의 한 속성을 표현
 - 행(ROW, Record) : 테이블의 데이터는 행에 저장
 - 테이블의 정보 확인
 - **DESC[RIBE] [Schema.]Table명**

테이블 생성 (Create Table)

- CREATE TABLE SYNTAX

```
CREATE TABLE  
[schema.]table_name  
( column datatype  
  [, column datatype ...]  
)  
[TABLESPACE tablespace ]  
[ PCTFREE integer ]  
[ PCTUSED integer ]  
[ INITRANS integer ]  
[ MAXTRANS integer ]  
[ STORAGE storage-clause ]  
[ LOGGING | NOLOGGING ]  
[ CACHE | NOCACHE ] ;
```

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

테이블 생성 문법

- schema : 테이블의 소유자
- table_name: 테이블 이름
- column: 컬럼의 이름
- datatype: 컬럼의 데이터 유형
- **TABLESPACE**
: 테이블이 데이터를 저장 할 테이블스페이스를 지정
- **PCTFREE**
: 블록내에 이미 존재하고 있는 Row에 Update가 가능하도록 예약시켜 놓는 블록의 퍼센트 값을 지정
- **PCTUSED**
: 테이블 데이터가 저장될 블록의 행 데이터 부분의 크기를 퍼센트지로 지정
- **PCTFREE**에 의해 지정된 크기만큼 Block이 차면 **PCTUSED** 값보다 작아져야 새로운 행 삽입이 가능

테이블 생성 문법

- **INITRANS** : 하나의 데이터 블록에 지정될 초기 트랜잭션의 값을 지정. (기본값은 1)
- **MAXTRANS**: 하나의 데이터 블록에 지정될 수 있는 트랜잭션 최대 수를 지정. (기본값은 255)
- **STORAGE**: 익스텐트 스토리지에 대한 값을 지정 합니다.
- **LOGGING**: 테이블에 대해 이후의 모든 작업이 리두 로그 파일 내에 기록 되도록 지정. (default)
- **NOLOGGING**: 리두 로그 파일에 테이블의 생성과 특정 유형의 데이터 로드를 기록하지 않도록 지정
- **CACHE** : 전체 테이블 스캔(full table scan)이 수행될 때 읽어 들인 블록이 버퍼 캐쉬 내의 LRU 리스트의 가장 최근에 사용된 것의 자리에 위치 하도록 지정
- **NOCACHE** : 전체 테이블 스캔(full table scan)이 수행될때 읽어 들인 블록이 버퍼 캐쉬 내의 LRU 리스트의 가장 최근에 사용 되지 않은 것의 자리에 위치하도록 지정

```
CREATE TABLE [schema.]table_name  
( column datatype  
  [, column datatype ...]  
)  
[TABLESPACE tablespace ]  
[ PCTFREE integer ]  
[ PCTUSED integer ]  
[ INITRANS integer ]  
[ MAXTRANS integer ]  
[ STORAGE storage-clause ]  
[ LOGGING | NOLOGGING ]  
[ CACHE | NOCACHE ] ;
```

테이블명과 컬럼명 생성 규칙

- 되도록이면 영문으로 시작
- 길이는 1~30문자 까지..
- a~z, A~Z, 0~9, _, \$, # 만 사용 가능
- 동일한 사용자가 소유한 테이블명은 중복 불가
- 오라클에서 사용하는 예약어 사용 불가

테이블 생성

- 테이블 생성 시 주의 사항
 - 모든 컬럼들은 콤마로 구분
 - 각 컬럼들은 반드시 데이터 타입이 설정 되어야...
 - 컬럼 설정 시 좌우 괄호 설정
 - 최대 컬럼 수 1,000개

```
SQL> CREATE TABLE emp_test
      ( empid varchar2(5)
        , firstname varchar2(10)
        , lastname varchar2(10)
        , salary number(7)
      );
```


연 습 문 제

1. 아래의 구조를 만족하는 MY_DATA1 테이블을 생성하시오. 단 ID가 PRIMARY KEY이다.

Name	Null?	Type
-----	-----	-----
ID	NOT NULL	NUMBER(4)
NAME		VARCHAR2(10)
USERID		VARCHAR2(30)
SALARY		NUMBER(10,2)

2. 1번에서 생성한 테이블을 삭제하여라.

SubQuery를 이용한 테이블 생성

- 다른 테이블 복사
 - 특정 테이블의 전체 또는 특정 컬럼 복사 가능

<형식1>

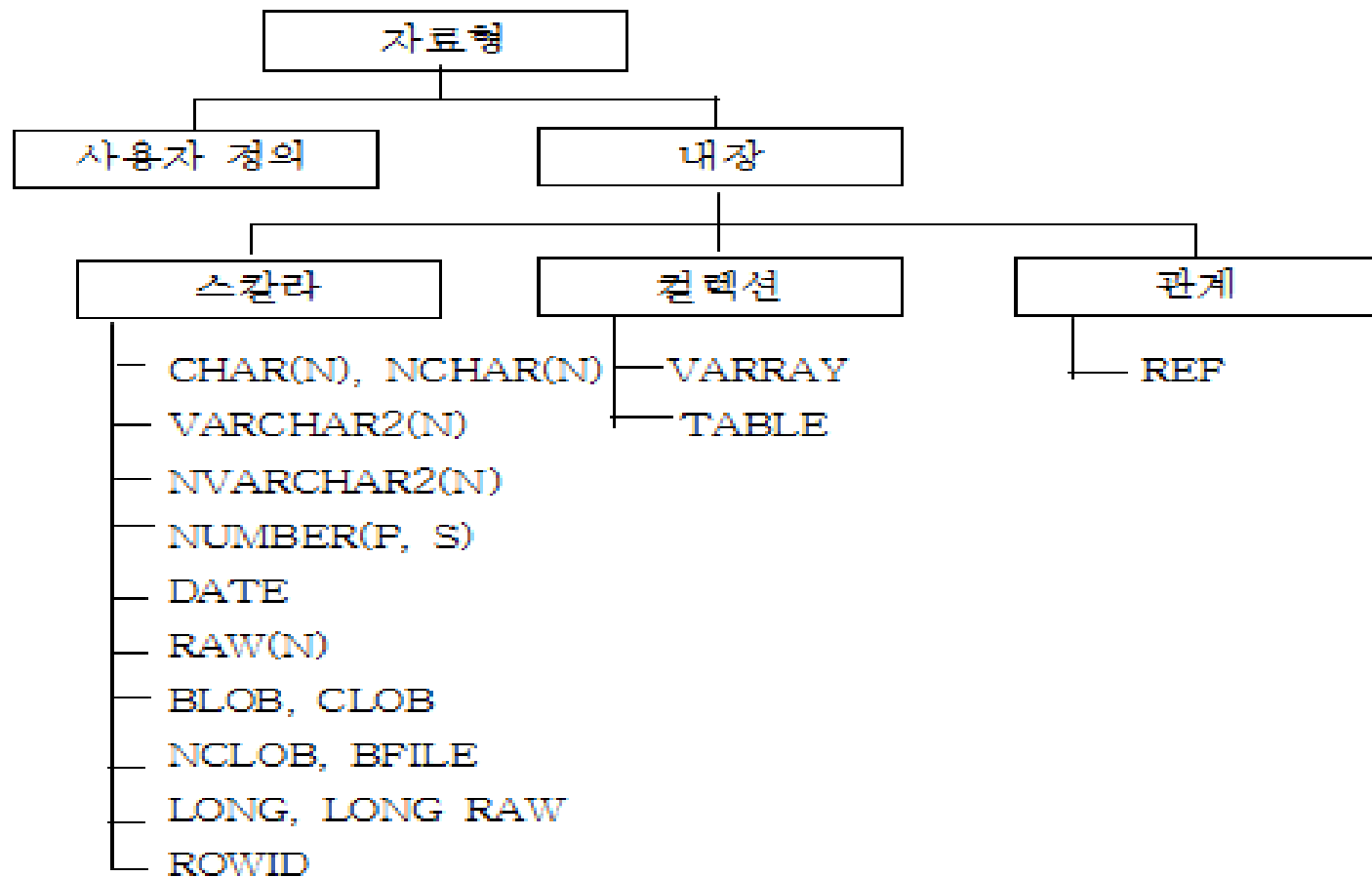
```
CREATE TABLE 테이블명  
AS SELECT * From 복사대상테이블;
```

<형식2>

```
CREATE TABLE 테이블명  
(컬럼명1, 컬럼명2, ..)  
AS SELECT 컬럼1, 컬럼2 From 복사대상테이블;
```

데이터 형

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



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)	2000 Bytes 가변 길이 문자 데이터. 실제 입력된 문자열의 길이만큼 저장 영역을 차지. 최대 크기는 명시해야 하며, 최소 크기는 1
NUMBER(w)	W자리까지의 수치로 최대 38자리까지 가능하다. (38자리가 유효 숫자이다.) W를 지정하지 않으면 38
NUMBER(w, d)	W는 전체 길이, d는 소수점 이하 자릿수이다. 소수점은 자릿수에 포함되지 않는다.
DATE	BC 4712년 1월 1일~AD 4712년 12월 31일까지의 날짜

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.

데이터 타입

- 숫자 데이터 (Numeric Data)
 - 오라클 데이터베이스에서 숫자는 항상 가변 길이 데이터로 저장되며 유효 자릿수 38자리까지 저장할 수 있다.
- 날짜 데이터 (Date Data)
 - 오라클 서버는 날짜를 7 바이트, 고정 길이 필드(field)로 저장
 - 오라클 DATE는 항상 시간을 포함
- Long Raw
 - 그래픽 파일이나 사운드 파일을 저장하는데 유용 함

LOB

- 크기가 큰 오브젝트를 저장하기 위한 데이터 타입
 - **LONG과 LONG RAW, LOB** 데이터 유형이 있습니다.
 - **LONG** 데이터 유형은 **2GB**의 문자열 데이터를 저장
 - 오라클은 **LOB**을 저장하기 위한 여섯 가지 데이터 유형을 제공
 1. 큰 고정 폭(fixed-width) 문자 데이터를 위한 **CLOB**과 **LONG**
 2. 큰 고정 폭 국가 character set 데이터를 위한 **NCLOB**
 3. 구조화되지 않은 데이터를 저장하기 위한 **BLOB**과 **LONG RAW**
 4. 구조화되지 않은 데이터를 운영 체제 파일에 저장하기 위한 **BFILE**
- 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.

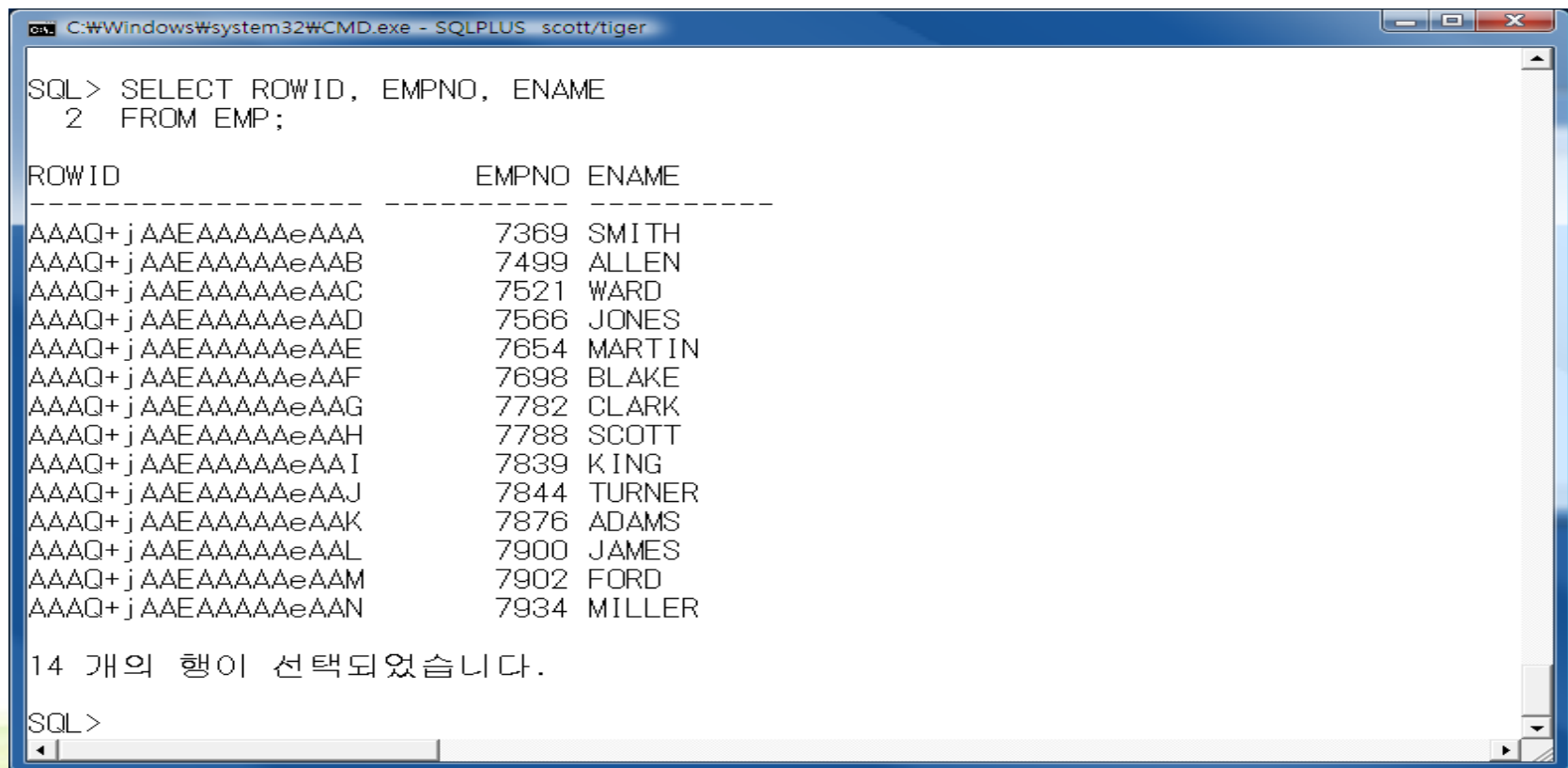
LONG과 LOB 데이터 유형 비교

LONG, LONG RAW	LOB
테이블에 컬럼 하나만 생성	테이블에 여러 개의 컬럼 생성
2GB	4GB
SELECT결과로 데이터를 리턴	SELECT결과로 위치를 리턴
데이터를 직접 저장	데이터를 직접 또는 간접 저장
오브젝트 유형을 지원하지 않음	오브젝트 유형 지원

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.
- 다음은 SELECT 문을 통해서 emp 테이블의 ROWID 확인

SELECT ROWID, EMPNO, ENAME FROM EMP;



```
SQL> SELECT ROWID, EMPNO, ENAME
2  FROM EMP;

ROWID                                EMPNO  ENAME
-----
AAEQ+jAAEAAAAAeAAA                  7369  SMITH
AAEQ+jAAEAAAAAeAAB                  7499  ALLEN
AAEQ+jAAEAAAAAeAAC                  7521  WARD
AAEQ+jAAEAAAAAeAAD                  7566  JONES
AAEQ+jAAEAAAAAeAAE                  7654  MARTIN
AAEQ+jAAEAAAAAeAAF                  7698  BLAKE
AAEQ+jAAEAAAAAeAAG                  7782  CLARK
AAEQ+jAAEAAAAAeAAH                  7788  SCOTT
AAEQ+jAAEAAAAAeAAI                  7839  KING
AAEQ+jAAEAAAAAeAAJ                  7844  TURNER
AAEQ+jAAEAAAAAeAAK                  7876  ADAMS
AAEQ+jAAEAAAAAeAAL                  7900  JAMES
AAEQ+jAAEAAAAAeAAM                  7902  FORD
AAEQ+jAAEAAAAAeAAN                  7934  MILLER

14 개의 행이 선택되었습니다.

SQL>
```

실습하기

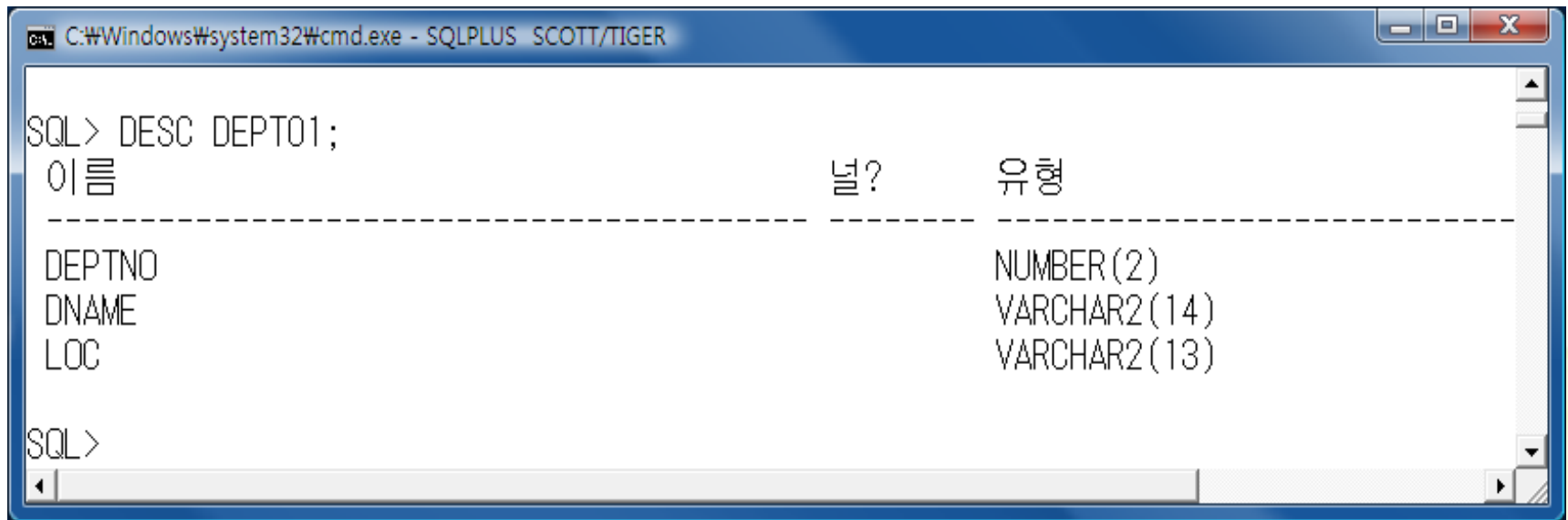
- 사원 테이블과 유사한 구조의 사원번호, 사원이름, 급여 3개의 칼럼으로 구성된 EMP01 테이블을 생성해 봅시다.

CREATE TABLE 명령어로 EMP01 테이블을 새롭게 생성합니다.

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

연습문제

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



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - SQLPLUS SCOTT/TIGER". The command "SQL> DESC DEPT01;" has been entered, and the output is displayed as follows:

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

The command prompt ends with "SQL>" on a new line.

서브 쿼리로 테이블 생성하기

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

원하는 컬럼으로 구성된 복제 테이블 생성

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

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

연습문제

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

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

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.

원하는 행으로 구성된 복제 테이블 생성

- 기존 테이블에서 원하는 행만 선택적으로 복사해서 생성할 수도 있습니다.
- 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;
```

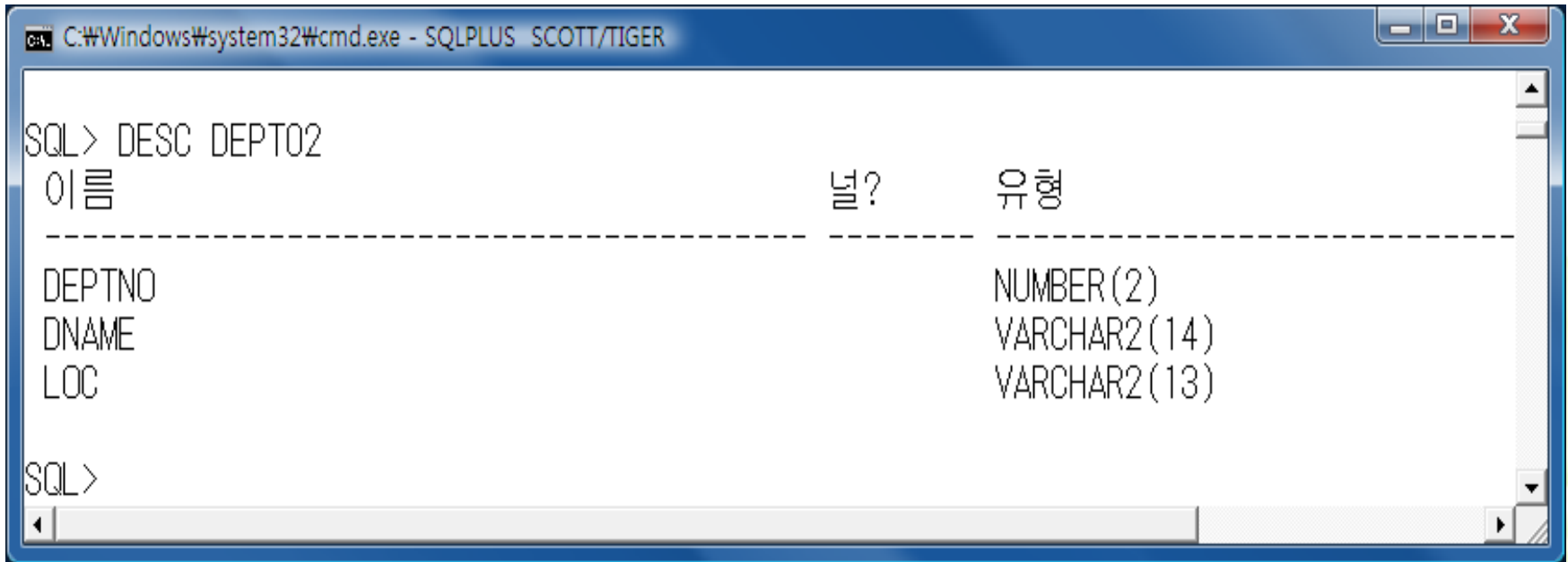
테이블의 구조만 복사하기

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

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


연습문제

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



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

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

SQL>
```

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 DEPT02 table, including column names (이름), nullability (널?), and data types (유형). The columns are DEPTNO (NUMBER(2)), DNAME (VARCHAR2(14)), and LOC (VARCHAR2(13)).

기본값(default value) 사용

- 기본값(default value)
 - 테이블 정보에 컬럼에 대한 기본값을 정의 할 수 있다.
 - 데이터 입력(insert)시 입력값이 생략된 컬럼은 기본값이 입력된다.
 - 데이터 변경(update)시 **default** 키워드를 사용하면 기본값(**default value**)으로 변경이 가능하다.
- 기본값(default value)의 생성

```
SQL> create table order_status (  
2     order_status_id INTEGER  
3     constraint default_example_pk primary key,  
4     status varchar2(20) default 'Order placed'  
      not null,  
5     last_modified date default sysdate  
6 );
```

기본값(default value) 사용

```
SQL> insert into order_status(order_status_id)
2 values(1);
```

```
SQL> insert into order_status(order_status_id,
    status, last_modified)
2 values(2, 'Order shipped', '05/04/14');
```

```
SQL> select *
2 from order_status;
```

ORDER_STATUS_ID	STATUS	LAST_MODI
1	Order placed	05/04/13
2	Order shipped	05/04/14

기본값(default value) 사용

```
SQL> update order_status  
1  set status = default  
2  where order_status_id = 2;
```

```
SQL> select *  
2  from order_status;
```

ORDER_STATUS_ID	STATUS	LAST_MODI
1	Order placed	05/04/13
2	Order placed	05/04/14

테이블 정의 변경

- Column 추가
- Column 데이터 타입 변경
- Column 삭제
- 테이블, 컬럼에 주석(comment) 달기
- 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.

테이블에 컬럼 추가

<형식>

```
ALTER TABLE Table_name ADD  
(column1 DataType [default][Not Null]  
, column2 ...)
```

- 같은 이름의 컬럼 존재 할 수 없음
- LONG, LONG RAW 컬럼은 한 테이블 하나만!
- 최대 컬럼 수는 1,000
- 새로운 컬럼은 테이블의 맨 마지막에 위치한다

<예제>

```
ALTER TABLE emp_test ADD (ADDR VARCHAR2(30));
```

연습

1. EMP01 테이블에 문자 타입의 직급(JOB) 칼럼을 추가
2. DEPT02 테이블에 문자 타입의 부서장(DMGR) 칼럼을 추가

컬럼의 데이터 타입 변경

<형식>

```
ALTER TABLE Table_name MODIFY  
(컬럼명 [Default] [Not Null] 수정데이터타입)
```

- 제약 조건
 - 모든 행의 컬럼 값이 데이터 유형보다 작은 값이 있을 때만 가능!
 - 컬럼의 크기 확장
 - 컬럼에 Null 값이 없을 경우만 Not Null 조건 추가 가능

<예제>

```
ALTER TABLE emp_test MODIFY (ADDR VARCHAR2(25));
```

연습

1. emp01 테이블에서 직급(JOB) 칼럼을 최대 30글자까지 저장할 수 있게 변경해 보도록 하자.
2. DEPT02 테이블의 부서장(DMGR) 칼럼을 숫자 타입으로 변경해 봅시다.

컬럼 삭제

<형식1 논리적 삭제>

```
ALTER TABLE Table_name SET UNUSED COLUMN col_name
```

--해당 컬럼 무시 (실제로 데이터가 삭제되지는 않음)

```
ALTER TABLE Table_name DROP UNUSED COLUMNS
```

--컬럼의 내용을 삭제한 후 완전 삭제 가능

<논리적 삭제의 확인>

DBA_UNUSED_COL_TABS

<형식2 물리적 삭제>

```
ALTER TABLE Table_name DROP (col_name)
```

--내용과 함께 컬럼 완전 삭제

연습

EMP01 테이블의 업무 칼럼을 삭제해 보도록 합시다

DEPT02 테이블의 부서장(DMGR) 칼럼을 삭제해 봅시다

실습하기

1. 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;
```

테이블 관리

- 테이블 삭제
- 테이블의 이름 변경
- 테이블 절단
- 테이블, 컬럼에 주석 설정

테이블 삭제

<형식>

```
DROP TABLE Table_name [CASCADE CONSTRAINT];
```

- 제약 조건
 - DROP TABLE 명령은 취소 불가 (rollback 안됨)
 - CASCADE CONSTRAINT는 종속되는 무결성 제약조건 까지 삭제 한다.
 - 삭제 권한이 있어야만 가능
 - 1. 테이블의 사용자 스키마의 일부
 - 2. DROP ANY TABLE 권한이 있어야 함

연습

CREATE TABLE을 학습할 때 만들어 놓았던 EMP01 테이블을 삭제해 보시다.

테이블 절단

<형식>

```
TRUNCATE TABLE Table_name;
```

- 제약 조건
 - 테이블의 있는 컬럼과 내용을 모두 해제(HWM삭제)
 - 내용만 삭제 : DELETE TABLE 수행
 - DELETE TABLE 권한이 있는 사용자만 가능
 - 데이터 복구 안됨!

연습

테이블 EMP02 에 저장된 데이터를 확인하였으면 이번에는 EMP02 테이블의 모든 데이터를 제거

테이블의 이름 변경

<형식1>

```
RENAME old_Tname TO new_Tname;
```

<형식2>

```
ALTER TABLE old_Tname RENAME TO new_Tname;
```

- 제약 조건
 - 테이블이 가지고 있는 정보는 변화 없음

연습

EMP02 테이블의 이름을 TEST 란 이름으로 변경합시다.

연습문제

1. 다음과 같은 테이블을 작성하시오.

테이블명 : JUSO

컬럼명	데이터 타입
NO	NUMBER(3)
NAME	VARCHAR2(10)
ADDR	VARCHAR2(20)
EMAIL	VARCHAR2(5)

2. 전화번호(PHONE) 컬럼을 VARCHAR2(10) 타입으로 추가하시오.
3. 이메일(EMAIL) 컬럼을 VARCHAR2(20) 타입으로 변경하시오.
4. 주소(ADDR) 컬럼을 삭제하시오.
5. 테이블을 삭제하시오.

연습문제

1. EMP 테이블의 SAL, COMM을 제외한 모든 COLUMN과 행을 포함하는 EMP_DEMO 테이블을 생성하는 SQL문을 작성하여라.
2. EMP 테이블과 DEPT 테이블을 이용하여 아래의 내용을 포함하는 테이블 (EMP_DEPT)을 생성하여라.

EMPNO	ENAME	JOB	DNAME	LOC
7839	KING	PRESIDENT	ACCOUNTING	NEW YORK
7698	BLAKE	MANAGER	SALES	CHICAGO

-
3. EMP 테이블과 SALGRADE 테이블을 이용하여 아래의 내용을 포함하는 테이블 (EMP_GRADE)을 생성하여라.

EMPNO	ENAME	JOB	SAL	COMM	GRADE
7839	KING	PRESIDENT	5000		5
7698	BLAKE	MANAGER	2850		4
7782	CLARK	MANAGER	2450		4
7566	JONES	MANAGER	2975		4

-
4. 3번에서 생성한 테이블에 SAL의 정밀도를 정수 부분을 12자리 소수 이하 4자리로 변경하는 SQL 문을 작성하여라.
 5. 2번과 3번에서 생성한 테이블을 모두 삭제하는 SQL문을 작성하여라

데이터 사전

- 데이터베이스 자원을 효율적으로 관리하기 위한 다양한 정보를 저장하는 시스템 테이블을 데이터 딕셔너리라고 합니다.
- 데이터 딕셔너리는 사용자가 테이블을 생성하거나 사용자를 변경하는 등의 작업을 할 때 데이터베이스 서버에 의해 자동으로 갱신되는 테이블로 사용자는 데이터 딕셔너리의 내용을 직접 수정하거나 삭제 할 수 없습니다.

데이터 사전

- 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.
- 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

USER_ 데이터 디렉터리

- 접두어로 USER가 붙은 데이터 디렉터리는 자신의 계정이 소유한 객체 등에 관한 정보를 조회합니다.
- 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

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;
```

ALL_ 데이터 딕셔너리

- 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 타 계정의 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

DESC 명령어로 데이터 디렉터리 뷰 ALL_TABLES의 구조를 살펴봅시다.

```
DESC ALL_TABLES;
```

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

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

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

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

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

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

```
CONN system/manager
```

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