Chapter 10. 테이블 생성 및 관리

이번 장에서는 DDL(Data definition language) 문장을 이용하여 주요 데이터베이스 객체인 테이블의 생성에 대하여 알아보고, 테이블의 변경, 이름 바꾸기, 잘라내기 등의 테이블 관리 방법을 설명한다.

데이터베이스 객체의 종류

Oracle에서 지원하는 데이터베이스 객체는 다음과 같은 것들이 있다.

표 10-1. 데이터베이스 객체

객체명	설명
테이블(Table)	기본적인 저장 단위로 행과 컬럼으로 구성
뷰(View)	한개 이상의 테이블의 논리적인 부분 집합을 표시
시퀀스(Sequence)	숫자 값 생성기
인덱스(Index)	데이터 검색 성능 향상
동의어(Synonym)	객체에 대한 별칭

Oracle의 테이블 구조는 다음과 같은 특징을 갖는다.

- 테이블은 사용자들이 데이터베이스를 사용 중에 언제라도 생성할 수 있다.
- 테이블의 저장 용량을 지정 할 필요는 없지만 테이블에 얼마나 많은 데이터가 저장 될 것인가를 예측하는 것도 중요하다.
- 테이블 구조는 데이터베이스가 온라인 상태에서도 변경이 가능하다.

테이블 생성

테이블 생성시 테이블 및 컬럼명은 다음과 같은 명명 규칙을 준수하도록 한다.

- 테이블 및 컬럼명은 문자로 시작하며 1~30 문자 이내로 작성한다.
- 테이블 및 컬럼명은 A~Z, a~z, 0~9, _, $, #로 작성한다. 물론, 한글로 작성할 수도 있으나 권장하지 않는다.
- 동일한 사용자의 다른 객체와 이름이 중복되지 않도록 한다.
- Oracle에 의해 예약되어 있는 키워드는 사용 할 수 없다.

테이블명은 대소문자를 구분하지 않는다. 예를 들어, EMP, eMP, eMp는 동일한 테이블로 인식된다.

■ CREATE TABLE

테이블을 생성하려면 CREATE TABLE 명령을 사용해야 하며, 이 문장은 DDL 명령 중

하나이다. 이 문장을 실행하면 데이터베이스에 곧바로 적용되어 테이블이 생성되며, 관련 정보가 데이터 딕셔너리(Data dictionary)라는 객체 관련 정보가 저장되어 있는 테이블에 기록된다.

테이블을 생성하기 위해서는 CREATE TABLE 권한과 테이블을 생성 할 저장공간을 부여 받아야 한다. 이러한 권한들은 DCL(Data control language)에 의해 사용자에게 부여되고 회수 될 수 있다.

테이블을 생성하는 명령어는 다음과 같다.

```
CREATE TABLE [schema.]table
            (column datatype [DEFAULT expr][, ...]);
```

■ 다른 사용자의 테이블 검색

위 문장에서 스키마(schema)는 객체의 모음을 의미하며, 스키마 객체는 데이터베이스내 데이터를 직접 참조하는 논리적 구조로서 테이블, 뷰, 동의어, 시퀀스, 저장 프로시져, 인덱스, 클러스터, 데이터베이스 링크를 포함한다.

만약, 다른 사용자가 소유하고 있는 테이블을 검색하려면 테이블 이름 앞에 소유자의 이름을 반드시 붙여주어야 한다. 예를 들어, KIM으로 명명된 스키마가 존재하고 KIM이 EMP 테이블을 소유하고 있다면 해당 테이블은 다음과 같이 검색하여야만 한다.

```
SELECT * FROM KIM.EMP;
```

■ DEFAULT 옵션

테이블 작성시 해당 컬럼에 DEFAULT 옵션을 사용하여 디폴트 값을 지정할 수 있는데, 해당 테이블에 행을 입력할 때, 해당 컬럼에 값을 지정하지 않은 경우 자동으로 디폴트 값이 입력되어 NULL 값이 저장되는 것을 방지하기 위한 것이다. 또한, INSERT 문장에 DEFAULT 키워드를 지정하여 디폴트 값이 저장되도록 할 수 있다.

```
... HIREDATE DATE DEFAULT SYSDATE, ...
```

디폴트 값으로 리터럴 및 SQL 함수를 지정 할 수 있으며, 다른 컬럼의 이름 또는 가상 컬럼(Pseudocolumn)은 지정 할 수 없다. 물론, 컬럼의 데이터 타입과 디폴트 값의 타입은 일치되어야 한다.

■ 테이블 만들기

다음과 같이 테이블을 생성하고 올바르게 생성되었는지 확인해보자.

```
SQL> CREATE TABLE BUSEO
  2  (DEPTNO NUMBER(2),
  3  DNAME VARCHAR2(14),
  4  LOC VARCHAR2(13));

테이블이 생성되었습니다.

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

CREATE 명령은 DDL 명령으로 실행된 후, 자동 커밋 된다.

■ 테이블의 종류

Oracle 서버 내에는 EMP와 같이 사용자가 생성한 사용자 테이블(User table)과 Oracle 서버가 생성하고 관리하는 테이블이 있는데, 이를 데이터 딕셔너리(Data dictionary)라고 부르며 데이터베이스 관련 정보가 저장되어 있다. 데이터 딕셔너리 테이블은 SYS가 소유하고 있으며, 사용자들이 알아보기 어려운 형태로 저장되어 있기 때문에 직접 검색하기는 어렵다. 그래서, Oracle 서버는 이러한 테이블들을 알아보기 쉽도록 뷰로 만들어 제공하는데, 이러한 뷰를 데이터 딕셔너리 뷰라고 부르며 이러한 뷰를 통해서 Oracle 서버의 사용자, 사용자에게 부여된 권한, 데이터베이스 객체 이름, 테이블 제약조건, 감사 정보 등을 검색할 수 있다. 데이터 딕셔너리 뷰에는 다음의 4가지 종류가 있다.

표 10-2. 데이터 딕셔너리 뷰의 종류

접두사	설명
USER_	사용자가 소유한 객체와 관련된 정보를 포함하고 있는 뷰
ALL_	사용자가 접근 가능한 객체와 관련된 정보를 포함하고 있는 뷰
DBA_	DBA 롤을 부여 받은 사용자들만 접근할 수 있는 제한적인 뷰
V$	데이터베이스 서버 성능, 메모리, 잠금 등의 정보를 포함하고 있는 동적 성능 뷰

데이터 딕셔너리를 검색하는 방법은 다음과 같다. 먼저 사용자가 소유하고 있는 테이블의 이름을 검색하면 다음과 같다.

```
SQL> SELECT TABLE_NAME
  2  FROM USER_TABLES;

TABLE_NAME
------------------------------
BONUS
BUSEO
DEPT
...
TEST1

7 개의 행이 선택되었습니다.
```

사용자가 소유하고 있는 객체의 종류를 검색하면 다음과 같다.

```
SQL> SELECT DISTINCT OBJECT_TYPE
  2  FROM USER_OBJECTS;

OBJECT_TYPE
------------------
INDEX
TABLE
```

사용자가 소유하고 있는 테이블, 뷰, 동의어, 시퀀스를 검색하면 다음과 같다.

```
SQL> SELECT *
  2  FROM USER_CATALOG;

TABLE_NAME                     TABLE_TYPE
------------------------------ -----------
BONUS                          TABLE
BUSEO                          TABLE
DEPT                           TABLE
...
TEST1                          TABLE

7 개의 행이 선택되었습니다.
```

사용자들에 의해 자주 검색되는 데이터 딕셔너리는 다음과 같다.

- USER_TABLES
- USER_OBJECTS
- USER_CATALOG

USER_CATALOG는 CAT라는 동의어가 생성되어 있으므로, 동의어를 이용하여 검색이 가능하다.

```
SQL> SELECT *
  2  FROM CAT;

TABLE_NAME                     TABLE_TYPE
------------------------------ -----------
BONUS                          TABLE
BUSEO                          TABLE
DEPT                           TABLE
...
TEST1                          TABLE

7 개의 행이 선택되었습니다.
```

데이터 타입의 종류

테이블 생성시 컬럼에 지정 할 수 있는 데이터 타입은 다음과 같다.

표 10-3. 데이터 타입

데이터 타입	설명
VARCHAR2(*size*)	가변 길이 문자열 데이터. (최소 길이 1, 최대 길이 4000 바이트)
CHAR[(*size*)]	고정 길이 문자열 데이터. (디폴트 및 최소 길이 1, 최대 길이 2000 바이트)
NUMBER(*p*, *s*)	가변 길이 숫자 데이터. (전체 자리수 *p*, 소수점 자리수 *s*. 전체 자리수는 1~38, 소수점 자리수는 -84~127)
DATE	날짜 및 시간 데이터. (B.C 4712년 1월 1일~ A.D 9999년 12월 31일)
LONG	가변 길이 문자열 데이터(2GB)
CLOB	문자 데이터(4GB)
RAW(*size*)	이진 데이터. (최대 2000 바이트)
LONG RAW	가변 길이 이진 데이터(2GB)
BLOB	이진 데이터(4GB)
BFILE	외부 파일에 저장된 이진 데이터(4GB)
ROWID	시스템에서 테이블내의 행들을 유일하게 식별할 수 있는 64 비트 숫자

LONG 타입과 관련하여 주의할 사항은 다음과 같다.

· LONG 타입의 컬럼은 서브쿼리에 의해 테이블을 생성할 때 복사 되지 않는다.
· LONG 타입의 컬럼은 GROUP BY 또는 ORDER BY에 포함 될 수 없다.
· LONG 타입의 컬럼은 테이블에 오직 1개만 사용 할 수 있다.
· LONG 타입의 컬럼에는 제약조건을 정의 할 수 없다.
· LONG 타입의 컬럼보다는 CLOB 타입의 컬럼을 사용하도록 한다.

추가된 날짜 타입

Oracle 9i에서는 아래와 같은 새로운 날짜 타입이 추가 되었다.

표 10-4 추가된 날짜 타입

날짜 데이터 타입	설명
TIMESTAMP	DATE 타입을 확장한 것으로 10^{-9} 초까지 지정 가능
INTERVAL YEAR TO MONTH	년, 월 단위로 시간을 저장(예, 1년 2개월). 즉, 두개의 날짜 타입 데이터 간의 간격을 저장하는데 사용
INTERVAL DAY TO SECOND	일, 시, 분, 초 단위로 시간을 저장(예, 1일 2시간 3분 4초). 즉, 두개의 날짜 타입 데이터 간의 간격을 저장하는데 사용

■ TIMESTAMP

TIMESTAMP 타입은 DATE 타입이 확장된 것으로 DATE 타입의 년, 월, 일, 시, 분, 초뿐만 아니라 10^{-9} 초까지 지정가능한 타입이다. TIMESTAMP 타입을 지정하는 방법은 다음과 같다.

```
TIMESTAMP[(fractional_seconds_precision)]
```

*fractional_seconds_precision*은 초의 소수점 자리 수를 의미하며, 디폴트 값은 소수점 6자리이다.

```
SQL> CREATE TABLE NEW_EMP
  2  (EMP_ID NUMBER,
  3  EMP_NAME VARCHAR2(15),
  4  START_DATE TIMESTAMP(7));

테이블이 생성되었습니다.

SQL> INSERT INTO NEW_EMP
  2  VALUES(1, '홍길동', SYSDATE);

1 개의 행이 만들어졌습니다.

SQL> INSERT INTO NEW_EMP
  2  VALUES(2, '콩쥐', '04/06/21 22:40:30.1234567');

1 개의 행이 만들어졌습니다.

SQL> SELECT * FROM NEW_EMP;

    EMP_ID EMP_NAME        START_DATE
---------- --------------- ---------------------------
         1 홍길동          04/06/21 21:33:22.0000000
         2 콩쥐            04/06/21 22:40:30.1234567
```

■ TIMESTAMP WITH TIME ZONE

TIMESTAMP WITH TIME ZONE 타입은 TIMESTAMP 타입이 확장 된 것으로 현재 지역시간과 그리니치 표준시(GMT : Greenwich Mean Time)간의 시차를 추가로 저장한다.

```
TIMESTAMP[(fractional_seconds_precision)] WITH TIME ZONE
```

즉, 다국적 기업에서 사용하는 데이터베이스에 날짜 및 시간을 저장하는 경우, 현지 국가의 시간으로 입력된 데이터는 지역별 시차로 인하여 사용자에게 혼란을 일으킬 수 있으므로 TIMESTAMP WITH TIME ZONE 타입을 사용하여 현재 지역시간에 GMT와 현재 지역시간과의 시차를 추가로 저장하면 해결 할 수 있다.

```
SQL> CREATE TABLE NEW_EMP2
  2  (EMP_ID NUMBER,
  3  EMP_NAME VARCHAR2(15),
  4  START_DATE TIMESTAMP WITH TIME ZONE);

테이블이 생성되었습니다.

SQL> INSERT INTO NEW_EMP2
  2  VALUES(1, '홍길동', SYSDATE);

1 개의 행이 만들어졌습니다.

SQL> SELECT * FROM NEW_EMP2;

    EMP_ID EMP_NAME        START_DATE
---------- --------------- ---------------------------------
         1 홍길동          04/06/21 21:52:08.000000 +09:00
```

위 결과에서 보면 홍길동의 입사일은 2004년 6월 21일 21시 52분 8초이며, GMT를 기준으로 9시간이 빠름을 보여준다.

■ TIMESTAMP WITH LOCAL TIME ZONE

TIMESTAMP WITH LOCAL TIME ZONE 타입은 현재 지역 시간을 기준으로 변환되어 저장되므로 시차가 추가로 저장되지 않는다.

```
TIMESTAMP[(fractional_seconds_precision)] WITH LOCAL TIME ZONE
```

```
SQL> CREATE TABLE NEW_EMP3
  2  (EMP_ID NUMBER,
  3  EMP_NAME VARCHAR2(15),
  4  START_DATE TIMESTAMP WITH LOCAL TIME ZONE);

테이블이 생성되었습니다.

SQL> INSERT INTO NEW_EMP3
  2  VALUES(1, '홍길동', SYSDATE);

1 개의 행이 만들어졌습니다.

SQL> SELECT * FROM NEW_EMP3;

    EMP_ID EMP_NAME        START_DATE
---------- --------------- -------------------------
         1 홍길동          04/06/21 22:01:49.000000
```

■ INTERVAL YEAR TO MONTH

INTERVAL YEAR TO MONTH 타입은 2년 6개월과 같이 년과 월 단위로 날짜 간격을 저장한다.

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

여기서, *year_precision*은 년도를 표시하는 자리수를 의미하며, 디폴트는 2이다.

INTERVAL YEAR TO MONTH 타입의 데이터 값은 다음과 같이 표현된다.
예) INTERVAL YEAR '123-2' YEAR(3) TO MONTH : 123년 2개월
예) INTERVAL '123' YEAR(3) : 123년
예) INTERVAL '300' MONTH(3) : 300개월

```
SQL> CREATE TABLE TIME1
  2  (DURATION INTERVAL YEAR(3) TO MONTH);

테이블이 생성되었습니다.

SQL> INSERT INTO TIME1 VALUES (INTERVAL '10' MONTH(2));

1 개의 행이 만들어졌습니다.

SQL> SELECT SYSDATE, SYSDATE+DURATION FROM TIME1;

SYSDATE  SYSDATE+
-------- --------
04/06/21 05/04/21
```

■ INTERVAL DAY TO SECOND

INTERVAL DAY TO SECOND 타입은 10일 1시간 30분 20초와 같이 년, 월, 시, 분, 초 단위로 시간 간격을 저장한다.

```
 INTERVAL DAY [(day_precision)]
 TO SECOND [(fractional_seconds_precision)]
```

여기서, day_precision은 일수를 표시하는 자리수를 의미하며, 디폴트는 2이고, fractional_seconds_precision은 소수점 자리수를 의미하며, 디폴트는 6이다.

INTERVAL DAY TO SECOND 타입의 데이터 값은 다음과 같이 표현된다.
예) INTERVAL '4 5:12:10.222' DAY TO SECOND(3) : 4일 5시간 12분 10.222초
예) INTERVAL '4 5:12' DAY TO MINUTE : 4일 5시간 12분
예) INTERVAL '400 5' DAY(3) TO HOUR : 400일 5시간
예) INTERVAL '11:12:10.2222222' HOUR TO SECOND(7)
: 11시간 12분 10.222222초

```
SQL> CREATE TABLE TIME2
  2  (DURATION INTERVAL DAY(3) TO SECOND);

테이블이 생성되었습니다.

SQL> INSERT INTO TIME2 VALUES(INTERVAL '60' DAY(2));

1 개의 행이 만들어졌습니다.

SQL> SELECT SYSDATE, SYSDATE+DURATION FROM TIME2;

SYSDATE  SYSDATE+
-------- --------
04/06/21 04/08/20
```

서브 쿼리를 이용한 테이블 생성

CREATE TABLE 명령과 서브 쿼리를 결합하면 테이블을 생성하는 동시에 데이터를 입력할 수 있다.

```
CREATE TABLE table
        [(column, column ...)]
AS subquery
```

위에서 지정된 컬럼의 개수와 서브 쿼리에서 리턴 된 컬럼의 개수가 일치해야하며 컬럼명과 디폴트 값을 이용하여 컬럼을 정의 할 수 있다.

서브 쿼리를 이용하여 테이블 생성시 주의 할 점은 다음과 같다.
· 지정된 컬럼명으로 테이블이 생성되고 서브 쿼리의 SELECT 문장에 의해 리턴된 행들이 테이블에 입력된다.
· 컬럼을 정의할 때 컬럼명과 디폴트 값만 지정할 수 있다.
· 컬럼을 지정할 때 컬럼의 개수와 서브 쿼리인 SELECT 문장의 컬럼 개수가 일치해야 한다.
· 제약조건은 생성된 테이블에 만들어지지 않으며, 오직 컬럼의 데이터 타입만 동일하게 생성된다.

```
SQL> CREATE TABLE EMP_YEAR
  2  AS SELECT EMPNO, ENAME, SAL*12+NVL(COMM, 0) ANNSAL, HIREDATE
  3  FROM EMP
  4  WHERE DEPTNO = 10;

테이블이 생성되었습니다.

SQL> DESC EMP_YEAR
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 ANNSAL                                             NUMBER
 HIREDATE                                           DATE

SQL> SELECT * FROM EMP_YEAR;

     EMPNO ENAME          ANNSAL HIREDATE
---------- ---------- ---------- --------
      7782 CLARK           29400 81/06/09
      7839 KING            60000 81/11/17
      7934 MILLER          15600 82/01/23
```

위 문장의 서브 쿼리에서 SAL*12+NVL(COMM, 0) 컬럼은 테이블의 컬럼명으로 부적합하기 때문에 컬럼 별칭을 반드시 지정하여야 한다.

테이블 변경

ALTER TABLE 명령을 이용하면 다음과 같은 작업을 수행 할 수 있다.
· 새로운 컬럼의 추가

· 컬럼 변경
· 신규 컬럼의 디폴트 값 지정
· 컬럼 삭제

■ 컬럼 추가

테이블에 컬럼을 추가하는 명령은 다음과 같다.

```
ALTER TABLE table
ADD (column datatype [DEFAULT expr]
    [, column datatype] ...);
```

앞서 생성한 EMP_YEAR 테이블에 JOB 컬럼을 추가하면 다음과 같다.

```
SQL> ALTER TABLE EMP_YEAR
  2  ADD (JOB VARCHAR2(10));

테이블이 변경되었습니다.

SQL> DESC EMP_YEAR
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 ANNSAL                                             NUMBER
 HIREDATE                                           DATE
 JOB                                                VARCHAR2(10)
```

■ 컬럼 변경

테이블의 컬럼을 변경하는 명령어는 다음과 같다.

```
ALTER TABLE table
MODIFY (column datatype [DEFAULT expr]
       [, column datatype] ...);
```

컬럼을 변경 할 때는 다음 사항을 주의하여야 한다.

· 숫자 컬럼의 전체 자리수를 증가 시킬 수 있다.
· 문자 컬럼의 전체 길이를 증가 시킬 수 있다.
· 컬럼의 길이를 축소시킬 수 있으나, 모든 행의 해당 컬럼 값이 전부 NULL 또는 행이 아직 입력되지 않은 경우에만 가능하다.
· 모든 행의 해당 컬럼 값이 NULL인 경우에만 데이터 타입을 변경 할 수 있다.
· CHAR 타입을 VARCHAR2 타입으로, VARCHAR2 타입을 CHAR 타입으로 변경이 가능하지만 해당 테이블에 행이 아직 입력되지 않았거나, 모든 행의 해당 컬럼 값이 전부 NULL인 경우만 가능하다.
· 디폴트 값을 변경하면 변경 이후부터 입력되는 행에 대해서만 적용 된다.

```
SQL> ALTER TABLE EMP_YEAR
  2  MODIFY (JOB VARCHAR2(20));

테이블이 변경되었습니다.

SQL> DESC EMP_YEAR
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 ANNSAL                                             NUMBER
 HIREDATE                                           DATE
 JOB                                                VARCHAR2(20)
```

■ 컬럼명 변경

테이블의 컬럼명을 변경하는 명령어는 다음과 같다.

```
ALTER TABLE table
RENAME COLUMN old_column TO new_column;
```

JOB 컬럼을 WORK로 변경해본다.

```
SQL> ALTER TABLE EMP_YEAR
  2  RENAME COLUMN JOB TO WORK;

테이블이 변경되었습니다.
```

■ 컬럼 삭제

테이블의 컬럼을 삭제하는 명령어는 다음과 같다.

```
ALTER TABLE table
DROP (column );
```

컬럼을 삭제하는 경우, 다음 사항을 주의하도록 한다.

· 컬럼은 값의 존재 유무에 상관 없이 삭제된다.
· ALTER TABLE 명령을 이용하여 한번에 하나의 컬럼만 삭제 할 수 있다.
· 테이블에는 최소한 하나의 컬럼이 남아 있어야 한다.
· 컬럼이 삭제되면 복구가 불가능하다.

```
SQL> ALTER TABLE EMP_YEAR
  2  DROP (JOB);

테이블이 변경되었습니다.

SQL> DESC EMP_YEAR
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 ANNSAL                                             NUMBER
 HIREDATE                                           DATE
```

위에서 ALTER TABLE ... DROP 명령에 의해 컬럼을 삭제하는 대신에 SET UNUSED 옵션을 사용하여 해당 컬럼을 사용하지 않는 것으로 설정 할 수 있다. 이렇게 되면 사용자는 해당 컬럼을 검색 할 수 없을뿐더러 DESC 명령으로도 확인이 불가능하기 때문에 해당 컬럼이 삭제된 것처럼 보여진다.

또한, ALTER TABLE ... DROP 명령은 테이블의 모든 행에 접근하여야 하므로 시간이 많이 소요되며 데이터베이스에 많은 부담을 주는 작업이므로 업무시간에는 해당 컬럼을 SET UNUSED 명령으로 숨겨두었다가 데이터베이스의 부담이 적은 시간에 해당 컬럼을 삭제하는 것이 바람직하다고 볼 수 있다. SET UNUSED 명령으로 표시해둔 컬럼의 갯수는 USER_UNUSED_COL_TABS 데이터 딕셔너리 뷰에서 확인 가능하다.

```
SQL> ALTER TABLE EMP_YEAR
  2  SET UNUSED (HIREDATE);

테이블이 변경되었습니다.

SQL> SELECT * FROM USER_UNUSED_COL_TABS;

TABLE_NAME                       COUNT
------------------------------ ----------
EMP_YEAR                              1

SQL> ALTER TABLE EMP_YEAR
  2  DROP UNUSED COLUMNS;

테이블이 변경되었습니다.
```

테이블 삭제

테이블 삭제는 데이터베이스에서 해당 테이블의 정의를 제거하는 것이다. 테이블을 삭제하면 테이블에 입력된 모든 행들과 관련 인덱스가 모두 삭제된다. 테이블을 삭제하는 명령어는 다음과 같다.

```
DROP TABLE table
```

테이블을 삭제하는 경우, 주의할 사항은 다음과 같다.

- 테이블의 모든 행들은 삭제된다.
- 뷰와 동의어는 삭제되지 않지만 유효하지 않다.
- 진행중인 트랜잭션은 커밋된다.
- 테이블의 소유자 또는 DROP ANY TABLE 권한을 가진 소유자만이 테이블을 삭제 할 수 있다.

```
SQL> DROP TABLE EMP_YEAR;

테이블이 삭제되었습니다.
```

기타 테이블 관련 명령어

■ 테이블 이름 변경

RENAME 명령을 이용하면 테이블, 뷰, 시퀀스, 동의어의 이름을 변경 할 수 있다. 해당 객체의 소유자만이 객체의 이름을 변경할 수 있다.

```
RENAME old_name TO new_name;
```

■ 테이블 잘라내기

DDL 문장인 TRUNCATE 명령을 사용하면 테이블의 모든 행들을 삭제 할 수 있으며 테이블이 사용하고 있던 저장 공간을 해제하여 다른 테이블들이 사용할 수 있도록 해준다. 반면, DELETE 명령은 저장 공간을 해제하지 않는다.

```
TRUNCATE TABLE table;
```

또한, TRUNCATE 명령어는 DELETE 문장과는 달리 롤백이 불가능하지만 다음과 같은 이유로 인하여 DELETE 명령보다 수행 속도가 빠르다.

· TRUNCATE 명령은 DDL 명령이므로 롤백 정보를 생성하지 않는다.
· TRUNCATE 명령은 삭제 트리거를 발생시키지 않는다.
· 외래키 제약조건에 의해 참조되는 테이블은 TRUNCATE 명령으로 테이블을 잘라낼 수 없으므로, 먼저 제약조건을 취소해야 한다.

복습

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.