

## 12.1 DDL ( Data Definition Language )

- DDL 용도 및 종류

문장	설명
SELECT	데이터베이스로부터 데이터를 검색
INSERT UPDATE DELETE MERGE	데이터베이스 내의 테이블에 새로운 행을 입력하거나, 기존의 행을 수정 또는 삭제하는 명령어로 데이터 조작용어(DML : Data Manipulation Language)라고 함
CREATE ALTER DROP RENAME TRUNCATE	테이블을 생성, 변경, 삭제하는 명령어로 데이터 정의어(DDL : Data Definition Language)라고 함
COMMIT ROLLBACK SAVEPOINT	DML 문장에 의한 변경 사항을 관리하거나, 변경사항을 하나의 논리적 트랜잭션으로 포함시키는 명령어
GRANT REVOKE	데이터베이스와 데이터베이스를 구성하는 구조(테이블, 뷰 등)에 접근 권한을 부여하거나 회수하는 명령어로 데이터 제어어(DCL : Data Control Language)라고 함

## 12.2 DDL ( Data Definition Language )

### ■ 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

### ■ 데이터베이스 객체 이름 지정방법

- 테이블 및 컬럼명은 문자로 시작하며 1 ~ 30 문자 이내로 작성한다.
- 테이블 및 컬럼명은 A~Z , a~z , 0~9 , \_ , \$ , # 로 작성한다.  
한글 작성도 가능하지만 권장하지 않는다.
- 동일한 사용자의 다른 객체와 이름이 중복되지 않도록 한다.
- Oracle 의 예약어는 사용 불가.
- 대소문자 구별 안함.

## 12.3 DDL ( Data Definition Language )

### ■ 테이블 생성

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

```
SQL> CREATE TABLE DEPT_2  
2  ( DEPTNO NUMBER(2),  
3    DNAME VARCHAR2(10),  
4    LOC VARCHAR2(10) );
```

테이블이 생성되었습니다.

```
SQL> CREATE TABLE SCOTT.DEPT_3  
2  ( DEPTNO NUMBER(2),  
3    DNAME VARCHAR2(10),  
4    LOC VARCHAR2(10) );
```

테이블이 생성되었습니다.

### ■ 다른 사용자의 테이블 검색

```
SQL> show user;  
USER은 "SYS"입니다
```

```
SQL>
```

```
SQL> CONN / as sysdba  
연결되었습니다.
```

```
SQL>
```

```
SQL> SELECT * FROM DEPT;  
SELECT * FROM DEPT
```

\*

1행에 오류:

ORA-00942: 테이블 또는 뷰가 존재하지 않습니다

```
SQL> SELECT * FROM SCOTT.DEPT;
```

DEPTNO	DNAME	LOC
90	경리과	부산
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

## 12.4 DDL ( Data Definition Language )

### ■ DEFAULT 옵션

- 해당 테이블에 행을 입력할 때, 해당 컬럼에 값을 지정하지 않은 경우 자동으로 디폴트 값이 입력되어 NULL 값이 저장되는 것을 방지할 수 있다.
- 고정된 값만을 가지는 컬럼에 대해서도 사용 가능하다.

```
SQL> CREATE TABLE DEF_TABLE  
2  ( NUM NUMBER(2),  
3    WRITEDAY DATE  
4  );
```

테이블이 생성되었습니다.

```
SQL> CREATE TABLE DEF_TABLE2  
2  ( NUM NUMBER(2),  
3    WRITEDAY DATE DEFAULT SYSDATE  
4  );
```

테이블이 생성되었습니다.

```
SQL> INSERT INTO DEF_TABLE( NUM )  
2  VALUES ( 1 );
```

1 개의 행이 만들어졌습니다.

```
SQL> INSERT INTO DEF_TABLE2 ( NUM )  
2  VALUES ( 1 );
```

1 개의 행이 만들어졌습니다.

```
SQL> INSERT INTO DEF_TABLE2( NUM , WRITEDAY )  
2  VALUES ( 2, DEFAULT );
```

1 개의 행이 만들어졌습니다.

## 12.5 DDL ( Data Definition Language )

### ■ 데이터 타입 종류

: 테이블 생성시 컬럼에 지정할 수 있는 데이터 타입은 다음과 같다.

데이터 타입	설명
VARCHAR2( <i>size</i> )	가변 길이 문자열 데이터. (최소 길이 1, 최대 길이 4000 바이트)
CHAR[( <i>size</i> )]	고정 길이 문자열 데이터. (디폴트 및 최소 길이 1, 최대 길이 2000 바이트)
NUMBER( <i>p</i> , <i>s</i> )	가변 길이 숫자 데이터. (전체 자리수 <i>p</i> , 소수점 자리수 <i>s</i> . 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( <i>size</i> )	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 타입 사용시 주의할 점

: 서브쿼리에 의해 테이블을 생성할 때 복사되지 않는다.

: GROUP BY 또는 ORDER BY에 포함될 수 없다.

: 테이블에 오직 1개만 사용 가능하다.

: 제약조건을 정의 할 수 없다.

: CLOB 타입의 컬럼을 권장한다.

## 12.6 DDL ( Data Definition Language )

### ■ Datetime 데이터 타입

: Oracle 9i 부터 새로운 날짜 타입이 지원.

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

: DATE 타입이 확장된 것으로 년,월,일,시,분,초, $10^{-9}$  초까지 지정 가능.

`TIMESTAMP[( fractional_seconds_precision )]`

: precision은 초의 소수점 자릿수를 의미하며, 기본은 6자리이다.

```
SQL> CREATE TABLE new_tmp      SQL> insert into new_tmp
2  ( tmp_id NUMBER,              2  values ( 1 , sysdate , sysdate );
3    w_day DATE,
4    w_day2 TIMESTAMP ); 1 개의 행이 만들어졌습니다. SQL> SELECT * FROM new_tmp;
```

테이블이 생성되었습니다.

TMP_ID	W_DAY	W_DAY2
1	09/09/06	09/09/06 19:55:38.000000

## 12.7 DDL ( Data Definition Language )

### ■ TIMESTAMP WITH TIME ZONE

: TIMESTAMP 타입이 확장된 것으로 현재 시간과 그리니치 표준시(GMT)간의 시차를 추가로 저장한다.

: 다국적 기업에서 사용하는 데이터베이스에 날짜 및 시간 저장 시, 현지 국가의 시간으로 입력된 데이터는 지역별 시차로 인해 사용자에게 혼란을 일으킬 수 있다.

`TIMESTAMP[( fractional_seconds_precision )] WITH TIME ZONE`

```
SQL> CREATE TABLE new_tmp2
  2  ( tmp_id NUMBER,
  3    w_day TIMESTAMP WITH TIME ZONE );
```

테이블이 생성되었습니다.

```
SQL> SELECT * FROM new_tmp2;
```

TMP_ID	W_DAY
1	09/09/06 19:58:22.000000 <u>+09:00</u>

```
SQL> INSERT INTO new_tmp2
  2  VALUES( 1 , SYSDATE );
```

1 개의 행이 만들어졌습니다.

## 12.8 DDL ( Data Definition Language )

### ■ TIMESTAMP WITH LOCAL TIME ZONE

: 현재 지역 시간을 기준으로 변환되어 저장되므로 시차가 추가로 저장되지 않는다.

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

```
SQL> CREATE TABLE new_tmp3  
2  ( tmp_id NUMBER,  
3    w_day TIMESTAMP WITH LOCAL TIME ZONE );
```

```
SQL> INSERT INTO new_tmp3  
2  VALUES ( 1, SYSDATE );
```

Rome  
+01:00

Singapore  
+08:00

SEOUL  
+09:00

W\_DAY

09/09/06 13:16:06.000000

W\_DAY

09/09/06 20:16:06.000000

W\_DAY

09/09/06 21:16:06.000000



## 12.9 DDL ( Data Definition Language )

### ■ INTERVAL YEAR TO MONTH

- : 년과 월 단위로 저장.
- : 필요한 경우에 저장된 년과 월을 이용한다.

### ■ INTERVAL DAY TO SECOND

- : 일과 시 , 분 ,초 단위로 저장.
- : 필요한 경우에 저장된 일,시,분,초를 이용한다.

```
SQL> CREATE TABLE new_tmp4
2 ( time1 INTERVAL YEAR(4) TO MONTH,
3   time2 INTERVAL DAY(3) TO SECOND );
SQL> INSERT INTO new_tmp4
2 VALUES( INTERVAL '10-1' YEAR TO MONTH,
3         INTERVAL '10 12:10:10' DAY TO SECOND );
```

1 개의 행이 만들어졌습니다.

```
SELECT time1, time2 , sysdate, TO_CHAR( sysdate , 'YYYY-MM-DD HH24:MI:SS'),
TO_CHAR( SYSDATE+time1+time2, 'YYYY-MM-DD HH24:MI:SS')
FROM SCOTT.new_tmp4;
```

TIME1	TIME2	SYSDATE	TO_CHAR(SYSDATE,'YYYY-MM-DDHH24:MI:SS')	TO_CHAR(SYSDATE+TIME1+TIME2,'YYYY-MM-DDHH24:MI:SS')
10-1	10 12:10:10,0	09/09/06	2009-09-06 22:15:24	2019-10-17 10:25:34

## 12.10 DDL ( Data Definition Language )

### ■ 서브쿼리를 이용한 테이블 생성

- : 테이블을 생성하는 서브쿼리에서 리턴된 데이터를 입력할 수 있다.
- : CTAS 라고 한다.
- : 지정된 컬럼의 개수와 서브쿼리에서 리턴된 컬럼의 개수가 일치해야 한다.
- : 컬럼을 지정할 때 컬럼명과 디폴트 값만 지정 가능하다.
- : 지정된 컬럼을 연산했을때에는 반드시 alias 를 사용한다.
- : 제약 조건은 생성된 테이블에 만들어지지 않으며 오직 컬럼의 데이터 타입만 동일하게 생성된다.

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

```
SQL> CREATE TABLE deptA
  2  AS
  3  SELECT * FROM DEPT;
```

테이블이 생성되었습니다.

```
SQL> SELECT COUNT(*) FROM deptA;
```

```
COUNT(*)
-----
4
```

```
SQL> CREATE TABLE deptB( no , name )
  2  AS
  3  SELECT deptno , dname
  4  FROM DEPT;
```

테이블이 생성되었습니다.

```
SQL> SELECT count(*) FROM deptB;
```

```
COUNT(*)
-----
4
```

```
SQL> CREATE TABLE deptC
  2  AS
  3  SELECT * FROM DEPT
  4  WHERE 1= 2;
```

테이블이 생성되었습니다.

```
SQL> SELECT COUNT(*) FROM deptC;
```

```
COUNT(*)
-----
0
```

## 12.11 DDL ( Data Definition Language )

### ■ 제약조건 정의 ( Constraint )

: Oracle 서버는 제약조건을 이용하여 적절치 않은 데이터가 저장되는 것을 방지한다.

### ■ ( 데이터 무결성 ) 제약조건 종류

제약조건	기 술
NOT NULL	이 열은 null 값을 포함하지 않음을 지정
UNIQUE KEY	테이블의 모든 행에 대해 유일해야 하는 값을 가진 열 또는 열의 조합을 지정
PRIMARY KEY	유일하게 테이블의 각 행을 식별
FOREIGN KEY	열과 참조된 테이블의 열 사이의 외래키 관계를 적용하고 설정
CHECK	참이어야 하는 조건을 지정

### ■ 제약조건 특징

: 테이블에 데이터가 저장, 갱신, 삭제 될 때마다 지정된 제약조건을 반드시 만족해야 하므로 데이터 무결성을 강화시켜준다.

: 다른 테이블이 참조되고 있는 테이블이 삭제되는 것을 방지한다.

## 12.12 DDL ( Data Definition Language )

### ■ 제약조건 사용 지침

- : 적절한 이름을 지정하여 제약조건을 사용한다. 지정하지 않으면 자동으로 SYS\_Cn 형식으로 임의저장되기 때문에 관리가 어려워진다.
- : 테이블 생성과 동시에 지정할 수도 있고, 테이블 생성 후에 추가할 수도 있다.
- : 제약 조건은 테이블 수준 및 컬럼 수준, 2 가지 방법으로 지정한다.
- : 제약 조건은 USER\_CONSTRAINTS 데이터 디렉터리에서 검색할 수 있다.

### ■ 제약조건 사용 문법

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

## 12.12 DDL ( Data Definition Language )

### ■ 제약조건 정의 방법

#### 1. 테이블 수준 지정 방법

- : 한 개 이상의 컬럼에 한 개의 제약 조건을 정의할 수 있다.
- : NOT NULL 제약조건을 제외한 모든 제약조건 정의 가능하다.
- : 컬럼 정의와 별도로 지정한다.

```
column, ...  
[CONSTRAINT constraint_name] constraint_type  
(column, ...),
```

#### 2. 컬럼 수준 지정 방법

- : 한 개의 컬럼에 한 개의 제약조건만 정의 가능하다.
- : 모든 제약조건 정의 가능하다.

```
column [CONSTRAINT constraint_name] constraint_type,
```

## 12.13 DDL ( Data Definition Language )

### ■ NOT NULL 제약 조건

- : 해당 컬럼에 NULL 값이 입력되는 것을 방지하는 제약조건이다.
- : NOT NULL 제약조건은 컬럼 수준에서만 지정 가능하다.

```
SQL> CREATE TABLE SAWON (  
  2  S_NO NUMBER(4),  
  3  S_NAME VARCHAR2(10) NOT NULL,  
  4  S_HIREDATE DATE CONSTRAINT SAWON_S_HIREDATE_NN NOT NULL);
```

테이블이 생성되었습니다.

```
SQL> INSERT INTO SAWON  
  2  VALUES(1, '길동', NULL);  
INSERT INTO SAWON
```

★

1행에 오류:

ORA-01400: NULL을 ("SCOTT"."SAWON"."S\_HIREDATE") 안에 삽입할 수 없습니다

```
SQL> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE  
  2  FROM USER CONSTRAINTS  
  3  WHERE TABLE_NAME = 'SAWON';
```

CONSTRAINT_NAME	C
SAWON_S_HIREDATE_NN	C
SYS_C003005	C

## 12.14 DDL ( Data Definition Language )

### ■ UNIQUE 제약 조건

- : 해당 컬럼에 중복된 값이 저장되지 않도록 제한한다.
- : 한 개 이상의 컬럼( 복합컬럼)으로 구성 할 수 있다.
- : NULL 값 저장 가능하다.
- : 테이블 수준 및 컬럼 수준, 모두 지정 가능하다. 하지만 복합컬럼 지정시에는 테이블 컬럼만 가능하다.
- : 해당 컬럼에 UNIQUE INDEX 가 자동 생성된다.

```
SQL> CREATE TABLE SAWON_2 (  
2  S_NO NUMBER(2),  
3  S_NAME VARCHAR2(10) NOT NULL,  
4  S_EMAIL VARCHAR2(20) CONSTRAINT SAWON_S_EMAIL_UK UNIQUE );
```

```
SQL> INSERT INTO SAWON_2 VALUES( 1, '홍길동', 'hong@abc.com' );
```

```
SQL> INSERT INTO SAWON_2 VALUES( 2, '유관순', 'hong@abc.com' );
```

```
INSERT INTO SAWON_2 VALUES( 2, '유관순', 'hong@abc.com' )
```

\*

1행에 오류:

ORA-00001: 무결성 제약 조건(SCOTT.SAWON\_S\_EMAIL\_UK)에 위반됩니다

---

```
SQL> CREATE TABLE SAWON_3  
2  ( S_NO NUMBER(2),  
3    S_NAME VARCHAR2(10) NOT NULL,  
4    S_EMAIL VARCHAR2(20),  
5    CONSTRAINT SAWON_3_S_EMAIL_UK UNIQUE (S_EMAIL ) );
```

테이블이 생성되었습니다.

## 12.15 DDL ( Data Definition Language )

- UNIQUE 제약 조건

```
SQL> CREATE TABLE SAWON_4
  2  ( S_NO NUMBER(2),
  3    S_NAME VARCHAR2(10),
  4    S_EMAIL VARCHAR2(20),
  5    CONSTRAINT SAWON_4_UK UNIQUE( S_NAME , S_EMAIL ) );
```

테이블이 생성되었습니다.

```
INSERT INTO SAWON_4 VALUES ( 1, '홍길동', 'abc@abc.com');
```

```
INSERT INTO SAWON_4 VALUES ( 2, '홍길동', 'ddd@abc.com');
```

```
INSERT INTO SAWON_4 VALUES ( 3, '이순신', NULL );
```

```
INSERT INTO SAWON_4 VALUES ( 4, '유관순', NULL );
```

```
INSERT INTO SAWON_4 VALUES ( 5, '유관순', NULL );
```

```
INSERT INTO SAWON_4 VALUES ( 6, NULL , NULL );
```

```
INSERT INTO SAWON_4 VALUES ( 7, NULL , NULL );
```



## 12.16 DDL ( Data Definition Language )

### ■ PRIMARY KEY 제약 조건

- : 테이블에 기본키를 의미하며, 한 개의 테이블에는 오직 한 개의 기본키를 만들 수 있다.
- : UNIQUE + NOT NULL 제약조건 특성을 갖는다.
- : 테이블 수준 및 컬럼 수준 , 2 가지 방법 모두 사용 가능하다.
- : UNIQUE 와 마찬가지로 INDEX 가 자동 생성된다.

```
SQL> CREATE TABLE SAWON_5  
2  ( S_NO NUMBER(2) CONSTRAINT SAWON_5_S_NO_PK PRIMARY KEY,  
3    S_NAME VARCHAR2(10),  
4    S_SAL NUMBER(10) );
```

```
SQL> INSERT INTO SAWON_5 VALUES ( 1, '홍길동' , 2000 );
```

1 개의 행이 만들어졌습니다.

```
SQL> INSERT INTO SAWON_5 VALUES( 1, '이순신', 2000 );  
INSERT INTO SAWON_5 VALUES( 1, '이순신', 2000 )
```

\*

1행에 오류:

ORA-00001: 무결성 제약 조건(SCOTT.SAWON\_5\_S\_NO\_PK)에 위배됩니다


```
SQL> INSERT INTO SAWON_5 VALUES( NULL , '강감찬', 2000 );  
INSERT INTO SAWON_5 VALUES( NULL , '강감찬', 2000 )
```

\*

1행에 오류:

ORA-01400: NULL을 ("SCOTT"."SAWON\_5"."S\_NO") 안에 삽입할 수 없습니다

```
SQL> CREATE TABLE SAWON_6  
2  ( S_NO NUMBER(2),  
3    S_NAME VARCHAR2(10),  
4    S_SAL NUMBER(10),  
5    CONSTRAINT SAWON_6_S_NO_PK PRIMARY KEY ( S_NO ));
```



## 12.17 DDL ( Data Definition Language )

### ■ FOREIGN KEY 제약 조건

: 참조 무결성 제약조건이라고도 한다.

: 반드시 다른 테이블의 기본키나 또는 UNIQUE 컬럼의 값을 참조하도록 제한한다. ( NULL허용)

```
SQL> CREATE TABLE DEPT_2
  2  ( DEPTNO NUMBER(2) CONSTRAINT DEPT_2_DEPTNO_PK PRIMARY KEY,
  3    DNAME VARCHAR2(10),
  4    LOC VARCHAR2(10) );
```

테이블이 생성되었습니다.

```
SQL> INSERT INTO DEPT_2 VALUES ( 10, '인사', '서울' );
```

```
SQL> INSERT INTO DEPT_2 VALUES ( 20, '경리', '부산' );
```

```
SQL> INSERT INTO DEPT_2 VALUES ( 30, '관리', '대구' );
```

---

```
SQL> CREATE TABLE EMP_2
  2  ( EMPNO NUMBER(4) CONSTRAINT EMP_2_EMPNO_PK PRIMARY KEY,
  3    ENAME VARCHAR2(10),
  4    SAL  NUMBER(10),
  5    DEPTNO NUMBER(2) CONSTRAINT EMP_2_DEPTNO_FK REFERENCES DEPT_2(DEPTNO));
```

```
SQL> INSERT INTO EMP_2 VALUES ( 1111, '홍길동', 2000 , 40 );
```

```
INSERT INTO EMP_2 VALUES ( 1111, '홍길동', 2000 , 40 )
```

\*

1행에 오류:

ORA-02291: 무결성 제약조건(SCOTT.EMP\_2\_DEPTNO\_FK)이 위반되었습니다- 부모 키가 없습니다

## 12.18 DDL ( Data Definition Language )

```
SQL> CREATE TABLE EMP_3
  2  ( EMPNO NUMBER(4) CONSTRAINT EMP_3_EMPNO_PK PRIMARY KEY,
  3    ENAME VARCHAR2(10),
  4    SAL  NUMBER(10),
  5    DEPTNO NUMBER(2),
  6  CONSTRAINT EMP_3 DEPTNO FK FOREIGN KEY ( DEPTNO ) REFERENCES DEPT_2(DEPTNO
  );
```

### ■ FOREIGN KEY 추가 옵션

– 부모 테이블의 행 삭제 시 문제될 수 있는 자식테이블 행 설정법.

: ON DELETE CASCADE

– FK 제약조건에 의해 참조되는 테이블( 부모 테이블)의 행이 삭제되면, 해당 행을 참조하는 테이블(자식 테이블)의 행도 같이 삭제되도록 한다.

: ON DELETE SET NULL

– FK 제약조건에 의해 참조되는 테이블( 부모 테이블)의 행이 삭제되면, 해당 행을 참조하는 테이블(자식 테이블)의 행을 NULL 로 설정한다.

## 12.19 DDL ( Data Definition Language )

### ■ CHECK 제약 조건

: 해당 컬럼에 반드시 만족해야 될 조건을 지정하는 제약 조건이다.

```
SQL> CREATE TABLE SAWON_7
  2  ( S_NO NUMBER(2),
  3    S_NAME VARCHAR2(10),
  4    S_SAL NUMBER(10) CONSTRAINT SAWON_7_S_SAL_CK CHECK( S_SAL < 500 ) );
```

테이블이 생성되었습니다.

```
SQL> INSERT INTO SAWON_7 VALUES ( 1, '홍길동', 600 );
INSERT INTO SAWON_7 VALUES ( 1, '홍길동', 600 )
```

\*

1행에 오류:

ORA-02290: 체크 제약조건(SCOTT.SAWON\_7\_S\_SAL\_CK)이 위반되었습니다

```
SQL> INSERT INTO SAWON_7 VALUES( 1 , '홍길도', 400 );
```

1 개의 행이 만들어졌습니다.

---

```
SQL> CREATE TABLE SAWON_8
  2  ( S_NO NUMBER(2),
  3    S_NAME VARCHAR2(10),
  4    S_SAL NUMBER(10),
  5    CONSTRAINT SAWON_8_S_SAL_CK CHECK ( S_SAL < 500 ) );
```

테이블이 생성되었습니다.

## 12.20 DDL ( Data Definition Language )

### ■ 테이블 삭제

- : 데이터베이스에서 해당 테이블을 제거하는 것이다.
- : 테이블에 저장된 모든 데이터와 관련 INDEX가 삭제된다.

```
DROP TABLE table
```

```
SQL> drop table sawon_8;
```

테이블이 삭제되었습니다.

### ■ 테이블 이름 변경

```
RENAME old_name TO new_name;
```

```
SQL> rename sawon_7 to sawon_77;
```

테이블 이름이 변경되었습니다.

### ■ 테이블 잘라내기

- : 테이블의 모든 행들을 삭제 할 수 있다.
- : 테이블이 사용하고 있던 저장 공간을 해제하여 다른 테이블들이 사용 할 수 있도록 한다.
- : DELETE 명령은 저장공간을 해제하지 않는다.
- : ROLLBACK 정보를 발생시키지 않아서 DELETE 보다 수행속도가 빠르다.
- : 단, DELETE와 달리 ROLLBACK 은 불가능하다.

```
TRUNCATE TABLE table;
```