테이블 구조 생성, 변경 및 삭제하는 DDL

테이블 구조를 정의하는 CREATE TABLE

❖ CREATE TABLE 문

- 형식

```
CREATE TABLE 테이블명
(컬럼이름, 데이터 타입 표현, …)
```

테이블 구조를 정의하는 CREATE TABLE

❖ 데이터 타입

이름	비고
CHAR(size)	고정 길이 문자 데이터
VARCHAR2(size)	가변 길이 문자 데이터. 최대 2000byte
NUMBER	최대 40자리까지의 숫자 지정 가능(소수점, 부호는 미포함)
NUMBER(w)	w자리까지 최대 38자리까지 가능
NUMBER(w, d)	w는 전체 길이, d는 소수점 이하 자릿수(소수점은 미포함)
DATE	날짜
LONG	가변 길이의 문자형 타입. 최대 2GB
LOB	2GB까지의 가변 길이 바이너리 데이터 저장
ROWID	의사 컬럼. DB 데이터가 아님. 모든 레코드에게 자동 부여
BFILE	대용량 바이너리 데이터를 파일 형태로 저장. 최대 4GB
TIMESTAMP(n)	DATE의 확장형
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 유용

테이블 구조를 정의하는 CREATE TABLE

❖ LOB

- Large OBject
- 텍스트, 그래픽 이미지, 동영상, 사운드와 같이 구조화 되지 않은 대용량의 텍스트나 멀티미디어 데이터 저장
- BLOB, CLOB, NCLOB, BFILE 등

  - BLOB : 비정형 데이터 저장
  - CLOB : 대용량 텍스트
  - NCLOB : 국가별 문자셋 데이터 저장
  - BFILE : 바이너리 데이터를 파일 형태로 저장

테이블 구조를 정의하는 CREATE TABLE

❖ ROWID

- 행의 위치를 지정하는 논리적인 주소 값
- 데이터베이스 전체에서 중복되지 않는 유일한 값
- 새로운 행이 삽입되면 자동으로 생성

```
SELECT ROWID, employee_id, first_name, last_name
FROM employees
```

테이블 구조를 정의하는 CREATE TABLE

❖ TIMESTAMP

- DATE형의 확장된 형태
- 백만분의 일초단위까지 표현

❖ INTERVAL YEAR TO MONTH

- 년과 월을 사용하여 두 날짜 사이의 기간을 저장하기 위한 데이터 형

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

❖ INTERVAL DAY TO SECOND

- 일, 시, 분, 초를 사용하여 두 날짜 사이의 기간을 저장하기 위한 데이터형

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

자릿수 생략시 기본값 2

테이블 구조를 정의하는 CREATE TABLE

❖ 실습

- 컬럼에 기간(3년) 저장하기

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

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

SELECT YEAR01, SYSDATE, SYSDATE+YEAR01
FROM SAM02

→ +03-00 14/07/19            17/07/19
```

테이블 구조를 정의하는 CREATE TABLE

❖ **실습**

- 컬럼에 기간 100일 저장하기

```
CREATE TABLE SAM03(
DAYO1 INTERVAL DAY(3) TO SECOND);

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

SELECT * FROM SAM03
```

테이블 구조를 정의하는 CREATE TABLE

❖ 식별자의 명명 규칙

- 테이블 명과 컬럼 명과 같이 사용자가 이름을 부여하는 것 - 식별자
- 반드시 문자로 시작해야 한다.
- A~Z까지의 대소문자와 0~9까지의 숫자, 특수 기호는(_, $, #)만 가능
- 오라클에서 사용되는 예약어나 다른 객체명과 중복 불가

테이블 구조를 정의하는 CREATE TABLE

❖ **실습**

- EMP01이라는 이름으로 사원번호, 사원명, 급여 3개의 컬럼으로 구성된 테이블 생성

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

SELECT * FROM EMP01;

DESC EMP01;
```

테이블 구조를 정의하는 CREATE TABLE

❖ **실습**

- 다음과 같은 컬럼으로 구성되는 DEPT01 테이블을 생성하시오.

DEPTNO	NUMBER(2)
DNAME	VARCHAR2(14)
LOC	VARCHAR2(13)

테이블 구조를 정의하는 CREATE TABLE

❖ 실습(서브 쿼리로 테이블 생성하기)

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

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

  - 서브 쿼리로 테이블을 만드는 경우 데이터는 복사되나 제약 조건은 복사되지 않음

테이블 구조를 정의하는 CREATE TABLE

❖ **실습**

- 원하는 컬럼으로 구성된 복제 테이블 생성하기

```
CREATE TABLE EMP03
AS
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME FROM EMPLOYEES;


SELECT * FROM EMP03;
```

테이블 구조를 정의하는 CREATE TABLE

❖ 실습

- 원하는 행으로 구성된 복제 테이블 생성하기

```
CREATE TABLE EMP04
AS
SELECT * FROM EMPLOYEES
WHERE DEPARTMENT_ID=80;

SELECT * FROM EMP04;
```

테이블 구조를 정의하는 CREATE TABLE

❖ 테이블의 구조만 복사하기

- 서브 쿼리를 이용

```
CREATE TABLE EMPLOYEES06
AS    SELECT * FROM EMPLOYEES WHERE 1=0;
```

  - WHERE 1=0은 항상 거짓(선택되는 로우가 없음)

테이블 구조를 정의하는 CREATE TABLE

❖ **실습**

- departments 테이블과 동일한 구조의 빈 테이블을 DEPT 이름으로 생성하시오

테이블 구조를 변경하는 ALTER TABLE

❖ ALTER TABLE

- ADD COLUMN절을 사용하여 새로운 컬럼을 추가
- MODIFY COLUMN 절을 사용하여 기존 컬럼을 수정
- DROP COLUMN 절을 사용하여 기존 컬럼을 삭제

❖ 새로운 컬럼 추가하기

- ALTER TABLE ADD 문
- 형식

```
ALTER TABLE 테이블이름
ADD (컬럼이름, 데이터타입 표현, …);
```

테이블 구조를 변경하는 ALTER TABLE

❖ EMP01 테이블에 JOB 컬럼 추가하기

```
DESC EMP01;

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

DESC EMP01;
```

테이블 구조를 변경하는 ALTER TABLE

❖ **실습**

- DEPT01 테이블에 문자 타입(10)의 부서장(DMGR) 컬럼을 추가한다

```
ALTER TABLE DEPT01
ADD(DMGR VARCHAR(10));

DESC DEPT01;
```

테이블 구조를 변경하는 ALTER TABLE

❖ 기존 컬럼 속성 변경하기

- ALTER TABLE MODIFY 문을 사용
- 형식

```
ALTER TABLE 테이블명
MODIFY (컬럼이름, 데이터 타입 표현, …)
```

- 규칙
  - 해당 컬럼에 자료가 없는 경우
    - 컬럼의 데이터 타입을 변경할 수 있다
    - 컬럼의 크기를 변경할 수 있다
  - 해당 컬럼에 자료가 있는 경우
    - 컬럼의 데이터 타입을 변경할 수 없다
    - 크기를 늘릴 수는 있지만 현재 가지고 있는 데이터의 크기보다 작은 크기로 변경할 수 없다.

테이블 구조를 변경하는 ALTER TABLE

❖ 기존 컬럼 속성 변경하기

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

DESC EMP01;
```

테이블 구조를 변경하는 ALTER TABLE

❖ **실습**

- DEPT01 테이블의 부서장(DMGR) 컬럼을 숫자 타입으로 변경해 보자

```
ALTER TABLE DEPT01
MODIFY(DMGR NUMBER(4));

DESC DEPT01;
```

테이블 구조를 변경하는 ALTER TABLE

❖ 기존 컬럼 삭제하기

- 형식

```
ALTER TABLE 테이블명
DROP COLUMN  컬럼명
```

```
ALTER TABLE EMP01
DROP COLUMN JOB;

DESC EMP01;
```

테이블 구조를 변경하는 ALTER TABLE

❖ **실습**

- DEPT01 테이블에서 DMGR 컬럼을 삭제하시오.

테이블 구조를 변경하는 ALTER TABLE

❖ SET UNUSED 옵션 적용하기

- 컬럼을 삭제하지는 않지만 컬럼의 사용을 논리적으로 제한

- 컬럼삭제시 문제점
  - 기존에 데이터가 많은 경우 컬럼을 삭제하는데 시간이 소요
  - 타 사용자에의해 삭제 중인 컬럼에 대한 접근 가능

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

SELECT * FROM EMP02;

DESC EMP02;

ALTER TABLE EMP02
DROP UNUSED COLUMNS;
```

테이블 구조를 삭제하는 DROP TABLE

❖ 테이블 삭제

- 형식

```
DROP TABLE 테이블명
```

```
DROP TABLE EMP01;
```

- 테이블의 삭제와 무결성 제약 조건
  - 삭제하려는 테이블의 기본 키나 고유 키를 다른 테이블에서 사용하는 경우에 해당 테이블을 제거할 수 없다.

    예) EMPLOYEES 테이블보다 먼저 DEPARTMENTS 테이블을 삭제하는 경우

  - 참조하는 테이블을 먼저 제거한 후 해당 테이블을 삭제

테이블의 모든 로우를 제거하는 TRUNCATE

❖ TRUNCATE

- 형식

```
TRUNCATE  TABLE 테이블 명
```

```
TRUNCATE TABLE EMP02;
```

테이블명을 변경하는 RENAME

❖ RENAME

- 형식

```
RENAME 기존 이름 TO 새 이름
```

```
RENAME EMP02 TO TEST;

SELECT * FROM EMP02;
SELECT * FROM TEST;
```

데이터 딕셔너리와 데이터 딕셔너리 뷰

❖ 데이터 딕셔너리 테이블

- 데이터베이스 자원을 효율적으로 관리하기 위한 다양한 정보를 저장하는 시스템 테이블
- 사용자가 직접 데이터 딕셔너리의 내용을 수정할 수 없음

❖ 데이터 딕셔너리 뷰

- 일반 사용자는 데이터 딕셔너리 테이블을 볼 수 없음
- 일부를 일반 사용자에게 보이게끔 가공하여 제공하는 것이 데이터 딕셔너리 뷰

  - DBA_XXXX
  - ALL_XXXX
  - USER_XXXX

데이터 딕셔너리와 데이터 딕셔너리 뷰

❖ USER_ 데이터 딕셔너리

- 자신의 계정이 소유한 객체 등에 관한 정보 조회
  - 현재 접속 사용자 확인

```
SHOW USER;
```

  - USER_TABLES : 현재 접속한 사용자의 계정이 소유한 모든 테이블 조회 가능

```
DESC USER_TABLES;

SELECT TABLE_NAME FROM USER_TABLES
ORDER BY TABLE_NAME DESC
```

데이터 딕셔너리와 데이터 딕셔너리 뷰

❖ USER_ 데이터 딕셔너리

- USER_SEQUENCES
  - 계정이 소유한 시퀀스의 정보를 조회할 수 있는 데이터 딕셔너리 뷰
- USER_INDEXES
  - 계정이 소유한 인덱스 정보를 조회할 수 있는 데이터 딕셔너리 뷰
- USER_VIEWS
  - 계정이 소유한 뷰 정보를 조회할 수 있는 데이터 딕셔너리 뷰

데이터 딕셔너리와 데이터 딕셔너리 뷰

❖ ALL_ 데이터 딕셔너리

- 현재 접근할 수 있는 객체, 즉 자신 계정의 소유이거나 접근 권한을 부여받은 타 계정의 객체 등을 조회할 수 있다

```
DESC ALL_TABLES;

SELECT OWNER, TABLE_NAME
FROM ALL_TABLES;
```

- ALL_SEQUENCES
- ALL_INDEXES
- ALL_VIEWS

데이터 딕셔너리와 데이터 딕셔너리 뷰

❖ DBA_ 데이터 딕셔너리

- DBA에서 접근할 수 있는 객체 등을 조회할 수 있는 뷰
- DBA 시스템 권한을 가진 사용자만 접근 가능

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

- DBA_SEQUENCES
- DBA_INDEXES
- DBA_VIEWS