9장 DDL

9.1 테이블 구조를 정의하는 CREATE TABLE

■ CREATE TABLE 명령어는 새로운 테이블을 생성한다.

```
-- 형식
CREATE TABLE table_name
(column_name data_type expr, …);

-- 예
CREATE TABLE EX2_1 (
    COLUMN1    CHAR(10),
    COLUMN2    VARCHAR2(10),
    COLUMN3    VARCHAR2(10),
    COLUMN4    NUMBER
);
```

9.1.1 데이터형

■ 데이터베이스에는 문자, 숫자, 날짜, 이미지 등과 같은 다양한 형태의 데이터가 저장된다. 다음은 오라클에서 제공되는 데이터 형의 종류이다.

이 름	비 고
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 있다. 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
LONG	가변 길이의 문자형 데이터 타입, 최대 크기는 2GB
LOB	2GB까지의 가변 길이 바이너리 데이터를 저장시킬 수 있다. 이미지 문서, 실행 파일을 저장할 수 있다.
ROWID	ROWID는 Tree-piece Format을 갖음. 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 형식: INTERVAL YEAR(년도에 대한 자릿수) TO MONTH(달에 대한 자릿수)
INTERVAL DAY TO SECOND	일, 시, 분, 초를 이용하여 기간을 저장 두 날짜 값의 정확한 차이를 표현한는데 유용하다. 형식: INTERVAL DAY(일수에 대한 자릿수) TO SECOND(초에 대한 자릿수)

9.1.2 식별자 명명 규칙

■ 테이블 명과 컬럼 명과 같이 사용자가 이름을 부여하는 것을 식별자라고 하는데 식별자를 부여하기 위해서는 규칙을 준수해야 한다.
- 반드시 문자로 시작해야 함 (1~30자까지 가능함)
- A~Z까지의 대소문자와 0~9까지의 숫자, 특수 기호는 (_,$,#)만 포함 가능

- 오라클에서 사용되는 예약어나 다른 객체명과 중복 불가
- 공백 허용 안함

[실습] 새롭게 테이블 생성하기

■ 지금까지 실습에 사용했던 사원 테이블과 유사한 구조(사원번호, 사원명, 급여 3개의 컬럼으로 구성)의 EMP01 테이블을 생성해 보겠다.

[LAB-09-02.SQL]

```
01    -- 테이블 생성
02    DROP TABLE EMP01 CASCADE CONSTRAINTS;
03    CREATE TABLE EMP01(
04    EMPNO NUMBER(4),
05    ENAME VARCHAR2(20),
06    SAL NUMBER(7, 2));
07
08    -- 생성된 테이블의 내용을 확인한다.
09    SELECT * FROM EMP01;
10
11    -- 테이블의 구조를 DESC 명령어로 확인할 수 있다.
12    DESC EMP01
```

[실습] 서브 쿼리로 테이블 생성하기

■ CREATE TABLE문에서 서브 쿼리를 사용하여 이미 존재하는 테이블과 동일한 구조와 내용을 갖는 새로운 테이블을 생성할 수 있다.

[LAB-09-03.SQL]

```
01    -- EMP 테이블과 동일한 내용과 구조를 갖는 EMP02 테이블을 생성한다.
02    -- DROP TABLE EMP02 CASCADE CONSTRAINTS;
03    CREATE TABLE EMP02
04    AS
05    SELECT * FROM EMP;
06
07    -- 새롭게 생성된 EMP02 테이블의 구조는 EMP 테이블과 동일하다.
08    DESC EMP02
```

[실습] 원하는 컬럼으로 구성된 복제 테이블 생성하기

■ 기존 테이블에서 원하는 컬럼만 선택적으로 복사해서 생성할 수도 있다.

[LAB-09-04.SQL]

```
01    -- DROP TABLE EMP03 CASCADE CONSTRAINTS;
02    CREATE TABLE EMP03
03    AS
04    SELECT EMPNO, ENAME FROM EMP;
05
06    SELECT * FROM EMP03;
```

[실습] 원하는 행으로 구성된 복제 테이블 생성하기

- 기존 테이블에서 원하는 행만 선택적으로 복사해서 생성할 수도 있다.

[LAB-09-05.SQL]

```
01    -- DROP TABLE EMP05 CASCADE CONSTRAINTS;
02    CREATE TABLE EMP05
03    AS
04    SELECT * FROM EMP
05    WHERE DEPTNO=10;
06
07    SELECT * FROM EMP05;
```

9.1.3 테이블의 구조만 복사하기

- 테이블의 구조만 복사하는 것은 별도의 명령이 있는 것이 아니다. 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 된다.

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

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

9.2 테이블 구조를 변경하는 ALTER TABLE

9.2.1 새로운 컬럼 추가하기

- ALTER TABLE ADD 문은 기존 테이블에 새로운 컬럼을 추가한다.

- 새로운 컬럼은 테이블 맨 마지막에 추가되므로 자신이 원하는 위치에 만들어 넣을 수 없다.

- 이미 이전에 추가해 놓은 로우가 존재한다면 그 로우에도 컬럼이 추가되지만, 컬럼 값은 NULL 값으로 입력된다.

```
-- 형식
ALTER TABLE table_name
ADD (column_name, data_type expr, …);
```

[실습] EMP01 테이블에 JOB 컬럼 추가하기

- EMP01 테이블에 문자 타입의 직급(JOB) 컬럼을 추가해 보겠다.

[LAB-09-06.SQL]

```
01    DESC EMP01
02
03    ALTER TABLE EMP01
04    ADD(JOB VARCHAR2(9));
05
06    DESC EMP01
```

9.2.2 기존 컬럼 속성 변경하기

- ALTER TABLE MODIFY 문을 다음과 같은 형식으로 사용하면 테이블에 이미 존재하는 컬럼을 변경할 수 있게 된다.
- 컬럼을 변경한다는 것은 컬럼에 대해서 데이터 타입이나 크기, 기본 값들을 변경한다는 의미이다.

```
-- 형식
ALTER TABLE table_name
MODIFY (column_name, data_type expr, …);

-- 예
ALTER TABLE EMP01
MODIFY(JOB VARCHAR2(30));
DESC EMP01
```

9.2.3 기존 컬럼 삭제하기

- ALTER TABLE ~ DROP COLUMN 명령어로 칼럼을 삭제할 수 있다.

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

-- 예
ALTER TABLE EMP01
DROP COLUMN JOB;
DESC EMP01
```

9.2.4 SET UNUSED 옵션 적용하기

- 특정 테이블(EMP02)에서 컬럼(JOB)을 삭제하는 경우 다음과 같이 무조건 삭제하는 것은 위험하다.
- 테이블에 저장된 내용이 많을 경우(몇 만 건에 대한 자료) 해당 테이블에서 컬럼을 삭제하는데 꽤 오랜 시간이 걸리게 될 것이다. 컬럼을 삭제하는 동안에 다른 사용자가 해당 컬럼을 사용하려고 접근하게 되면 지금 현재 테이블이 사용되고 있기 때문에 다른 사용자는 해당 테이블을 이용할 수 없게 된다. 이런 경우 작업이 원활하게 진행되지 않고 락(lock)이 발생하게 된다.
- ALTER TABLE 에 SET UNUSED 옵션을 지정하면 컬럼을 삭제하는 것은 아니지만 컬럼의 사용을

논리적으로 제한할 수 있게 된다.

- SET UNUSED 옵션은 사용을 논리적으로 제한할 뿐 실제로 컬럼을 삭제하지 않기 때문에 작업 시간이 오래 걸리지 않는다. 그렇기 때문에 락이 걸리는 일도 일어나지 않게 된다.

[실습] 직급 컬럼 사용 제한하기

SET UNUSED 옵션을 사용해 보겠습니다.

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

SELECT * FROM EMP02;
DESC EMP02;

ALTER TABLE EMP02
DROP UNUSED COLUMNS;
```

9.3 테이블 구조를 삭제하는 DROP TABLE

- DROP TABLE문은 기존 테이블을 제거한다.

```
-- 형식
DROP TABLE table_name;

-- 예
DROP TABLE EMP01;
DROP TABLE EMP01 CASCADE CONSTRAINTS;
SELECT * FROM EMP01;
```

9.4 테이블의 모든 로우를 제거하는 TRUNCATE

- 기존에 사용하던 테이블의 모든 로우를 제거하기 위한 명령어로 TRUNCATE가 제공된다.

```
-- 형식
TRUCATE TABLE
table_name

-- 예
SELECT * FROM EMP02;
TRUNCATE TABLE EMP02;
SELECT * FROM EMP02;
```

9.5 테이블명을 변경하는 RENAME

- 기존에 사용하던 테이블의 이름을 변경하기 위한 명령어로 RENAME이 제공된다.

```
-- 형식
RENAME old_name TO new_name

-- 예
RENAME EMP02 TO TEST;
SELECT * FROM EMP02;
SELECT * FROM TEST;
```

9.6 데이터 딕셔너리와 데이터 딕셔너리 뷰

- 데이터베이스 자원을 효율적으로 관리하기 위한 다양한 정보를 저장하는 시스템 테이블을 데이터 딕셔너리라고 한다.
- 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 없다.

9.6.1 USER_ 데이터 딕셔너리

- 접두어로 USER가 붙은 데이터 딕셔너리는 자신의 계정이 소유한 객체 등에 관한 정보를 조회한다.

```
DESC USER_TABLES;

SHOW USER

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

9.6.2 ALL_ 데이터 딕셔너리

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

```
DESC ALL_TABLES;

SELECT OWNER, TABLE_NAME FROM ALL_TABLES;
```

9.6.3 DBA_ 데이터 딕셔너리 뷰

- DBA_ 데이터딕셔너리는 DBA가 접근 가능한 객체 등을 조회 할 수 있는 뷰이다.
- 앞서도 언급했지만 DBA가 접근 불가능한 정보는 없기에 데이터베이스에 있는 모든 객체 등의 의미라 할 수 있다.
- USER_ 와 ALL_ 와 달리 DBA_ 데이터딕셔너리뷰는 DBA 시스템 권한을 가진 사용자만 접근할 수 있다.

```
SELECT TABLE_NAME, OWNER
FROM DBA_TABLES;

CONN system/manager
SELECT TABLE_NAME, OWNER
FROM DBA_TABLES;
```