ORACLE

09.DDL

목차

1. DDL

- DDL(Data Definition Language)
  - 테이블과 같은 데이터 구조를 정의하는데 사용되는 언어로, 생성, 변경, 삭제, 이름 변경 등 데이터 구조와 관련된 명령어들을 말한다.
  - Schema, Domain, Table, View, Index를 정의하거나 변경 또는 삭제할 때 사용.

종류	의미
CREATE	데이터베이스, 테이블등을 생성하는 역할
ALTER	테이블을 수정하는 역할
DROP	데이터베이스, 테이블을 삭제하는 역할
TRUNCATE	테이블을 초기화 시키는 역할

2. CREATE TABLE

- 만들고자 하는 테이블의 구조(컬럼 이름, 컬럼 개수, 데이터 형태)등을 고려하여 DDL을 이용하여 테이블 구조 자체를 새롭게 생성, 수정, 삭제해보자.

형식

```
CREATE TABLE 테이블명(
  column_name1 data_type expr...,
  column_name2 data_type
)
```

2.1 데이터 형(1)

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. 최대 크기 2000Byte
VARCHAR2(size)	가변 길이 문자 데이터. 실제 입력된 문자열의 길이만큼 저장 영역을 차지. 최대 크기는 명시해야 하며, 최대 크기 2000Byte
NUMBER(p, s)	가변 길이 숫자 데이터. s (1~38, 디폴트:38) / s (-84~127, 디폴트:0). p는 소수점을 포함한 전체 자릿수를 의미하고, s는 소수점 자릿수를 의한다. p와s를 입력하지 않으면 데이터 크기에 맞게 자동 조절된다. 최대 22Byte
DATE	BC 4712년 1월 1일~AD 4712년 12월 31일까지의 날짜
LONG	가변 길이의 문자형 데이터 타입, 최대 크기는 2GB 검색기능은 지원하지 않는다.
LOB	2GB까지의 가변 길이 바이너리 데이터를 저장시킬 수 있다. 이미지 문서, 실행 파일을 저장할 수 있다.

2.1 데이터 형(2)

이 름	비 고
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
INTERVAL DAY TO SECOND	일, 시, 분, 초를 이용하여 기간을 저장 두 날짜 값의 정확한 차이를 표현한는데 유용하다.

- 문자 데이터 형 1)VARCHAR2 2)CHAR 3)LONG
  숫자 데이터 형 1)NUMBER
  날짜 데이터 형 1)DATE 2)TIMESTAMP
- 값의 변화(숫자의 증감)가 일어나는 것들은 NUMBER 타입으로 설정하지만,
  그렇지 않은 경우는 숫자값이어도 VARCHAR2를 사용하기도 한다.
- ex)핸드폰번호, 우편번호

2.1.1 LOB

- LOB(Large Object)
  - 데이터 형은 텍스트, 그래픽 이미지, 동영상, 사운드와 같이 구조화되지 않은 대용량의 텍스트나 멀티미디어 데이터를 저장하기 위한 데이터 형이며, 최대 4GB까지 저장 가능하다.
  - 오라클에서 제공되는 LOB 데이터 형은 BLOB, CLOB, NCLOB, BFILE 등이 있다.
- BLOB
  - 그래픽 이미지, 동영상, 사운드와 같은 구조화 되지 않은 데이터를 저장하기 위해 사용
- CLOB
  - e-BOOK과 같은 대용량의 텍스트 데이터를 저장하기 위해서 사용
- NCLOB
  - 국가별 문자셋 데이터를 저장하고, BFILE은 바이너리 데이터를 파일 형태로 저장

2.1.2 ROWID(1)

- ROWID
  - 테이블에서 행의 위치를 지정하는 논리적인 주소값
  - 데이터베이스 전체에서 중복되지 않는 유일한 값으로 테이블에 새로운 행이 삽입되면 테이블 내부에서 의사 컬럼 형태로 자동 생성된다.
  - 테이블의 특정 레코드를 랜덤하게 접근하기 위해서 주로 사용된다.

예 **SELECT rowid, empno, ename FROM emp;**

	ROWID	EMPNO	ENAME
1	AAAE95AABAAALC5AAA	1001	김사랑
2	AAAE95AABAAALC5AAB	1002	한예슬
3	AAAE95AABAAALC5AAC	1003	오지호
4	AAAE95AABAAALC5AAD	1004	이병헌
5	AAAE95AABAAALC5AAE	1005	신농협
6	AAAE95AABAAALC5AAF	1006	장농건
7	AAAE95AABAAALC5AAG	1007	이분세
8	AAAE95AABAAALC5AAH	1008	감우성
9	AAAE95AABAAALC5AAI	1009	안성기
10	AAAE95AABAAALC5AAJ	1010	이병헌
11	AAAE95AABAAALC5AAK	1011	조향기
12	AAAE95AABAAALC5AAL	1012	강혜정
13	AAAE95AABAAALC5AAM	1013	박숭훈
14	AAAE95AABAAALC5AAN	1014	조인성

2.1.2 ROWID(2)

- ROWID는 다음과 같은 형식으로 데이터를 저장한다.

(총80비트:10바이트)

32bit	10bit	22bit	16bit
00000	FFF	BBBBB	RRR
데이터객체번호	상대적파일번호	블록번호	행번호

- 데이터 객체 번호는 테이블이나 인덱스와 같은 데이터 객체가 생성될 때 할당된다.
- 상대적인 파일 번호는 데이터가 저장되는 물리적인 데이터 파일 번호로서 유일한 값을 가진다.
- 블록 번호는 데이터 파일 내에서 행을 포함한 블록 위치이다.
- 행 번호는 블록 내에서 행 위치를 나타내는 번호이다.

2.1.3 날짜관련 데이터형(1)

- TIMESTAMP
  - DATE 형의 확장된 형태로서 백만분의 일초 단위까지 표현할 수 있다.
- INTERVAL YEAR TO MONTH
  - 년과 월을 사용하여 두 날짜 사이의 기간을 저장하기 위한 데이터 형이다.

형식	**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 데이터 형이다.

형식	**INTERVAL DAY(일수에 대한 자릿수) TO SECOND(초에 대한 자릿수)**

2.1.3 날짜관련 데이터형(2)

- 오라클에서 날짜의 년, 월을 빼거나 더할 때에 ADD_MONTHS 함수를 주로 사용한다.

형식	**ADD_MONTHS('날짜', '적용 숫자')**
예	**SELECT ADD_MONTHS(sysdate, -1) 지난달, ADD_MONTHS(sysdate, 1) 다음달 from dual;**

- INTERVAL
  - INTERVAL을 이용하면 조금 더 직관적으로 날짜 활용을 할 수 있다.

예

```
SYSDATE + (INTERVAL '1' YEAR)        --1년 더하기
SYSDATE + (INTERVAL '1' MONTH)       --1개월 더하기
SYSDATE + (INTERVAL '1' DAY)         --1일 더하기
SYSDATE + (INTERVAL '1' HOUR)        --1시간 더하기
SYSDATE + (INTERVAL '1' MINUTE)      --1분 더하기
SYSDATE + (INTERVAL '1' SECOND)      --1초 더하기
SYSDATE + (INTERVAL '02:10' HOUR TO MINUTE)     --2시간10분 더하기
SYSDATE + (INTERVAL '01:30' MINUTE TO SECOND)   --1분30초 더하기
```

문제

1. CREATE TABLE 명령어로 테이블을 생성하시오.**(길이 및 데이터 형태 자유)**

테이블 명 : EMP01

컬럼 : EMPNO – 사원번호

ENAME – 사원명

HEIGHT – 키

ADDRESS – 주소

PHONE – 핸드폰번호

2.2 서브쿼리로 테이블 복사하기(1)

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

예

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

- 기존 테이블에서 원하는 컬럼만 선택적으로 복사해서 생성할 수도 있다.

예

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

- 기존 테이블에서 원하는 행만 선택적으로 복사해서 생성할 수도 있다.

예

```
CREATE TABLE EMP04
AS
SELECT * FROM EMP WHERE DEPTNO=10;
```

2.2 서브쿼리로 테이블 복사하기(2)

- 서브 쿼리를 사용하여 이미 존재하는 테이블과 같은 구조의 테이블을 복사하되 데이터는 복사하지 않을 수 있다.
- 테이블의 구조만 복사하는 것은 별도의 명령이 있는 것이 아니다. WHERE 조건 절에 항상 거짓이 되는 조건을 지정하게 되면 테이블에서 얻어질 수 있는 데이터가 없게 되므로 빈 테이블이 생성된다.

예

```
CREATE TABLE EMP05
AS
SELECT * FROM EMP WHERE 1=0;
```

- WHERE 1=0 은 항상 거짓이기 때문에 이를 이용하여 테이블을 생성시 데이터를 제외한 테이블 구조만 복사할 수 있다.

문제

2. EMP 테이블을 복사하되 사원번호, 사원명, 급여 컬럼으로 구성된 테이블을 생성하시오. **(생성 테이블의 이름은 EMP06)**

EMPNO	ENAME	SAL
1001	김사랑	300
1002	한예슬	250
1003	오지호	500
1004	이병헌	600
1005	신농협	450
1006	장농건	480
1007	이눈세	520
1008	감우성	500
1009	안성기	1000
1010	이병헌	500
1011	조향기	280
1012	강혜정	300
1013	박숭훈	560
1014	조인성	250

3. ALTER TABLE

- 기존 테이블의 구조를 변경하기 위한 DDL 명령문의 하나로, 테이블에 대한 구조 변경은 컬럼의 추가, 삭제, 컬럼의 타입이나 길이를 변경할 때 사용한다. 테이블의 구조를 변경하게 되면 기존에 저장되어 있던 데이터에 영향을 주게 된다.
- ALTER TABLE로 컬럼 추가, 수정, 삭제하기 위해서는 다음과 같은 명령어를 사용
  - ADD COLUMN 절을 사용하여 새로운 컬럼을 추가
  - MODIFY COLUMN 절을 사용하여 기존 컬럼을 수정
  - DROP COLUMN 절을 사용하여 기존 컬럼을 삭제

3.1 컬럼 추가

- 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 입력됩니다

형식	**ALTER TABLE 테이블 명 ADD(컬럼 명 데이터 타입, ...);**

문제

3. EMP01 테이블에 문자 타입의 직급(JOB) 컬럼을 추가하시오. **(크기 9)**

```
이름       널? 유형
-------  -- -------------
EMPNO       NUMBER(4)
ENAME       VARCHAR2(20)
HEIGHT      NUMBER(5,2)
ADDRESS     VARCHAR2(200)
PHONE       VARCHAR2(20)
JOB         VARCHAR2(9)
```

3.2 컬럼 수정

- ALTER TABLE MODIFY 문을 다음과 같은 형식으로 사용하면 테이블에 이미 존재하는 컬럼을 변경할 수 있게 된다.

형식	**ALTER TABLE 테이블 명 MODIFY(컬럼 명 데이터 타입, ...);**

- 컬럼을 변경한다는 것은 컬럼에 대해서 데이터 타입이나 크기, 기본 값들을 변경한다는 의미.
- 특별한 경우가 아닐 때는 사용하지 말고, 테이블 설계를 잘 하도록 한다.

문제

4. EMP01 테이블의 직급(JOB)컬럼을 최대 30글자까지 저장할 수 있게 변경하시오.

```
이름       널? 유형
-------  -- -------------
EMPNO       NUMBER(4)
ENAME       VARCHAR2(20)
HEIGHT      NUMBER(5,2)
ADDRESS     VARCHAR2(200)
PHONE       VARCHAR2(20)
JOB         VARCHAR2(30)
```

3.3 컬럼 삭제

❖ ALTER TABLE DROP COLUMN 명령어로 컬럼을 삭제할 수 있다.

형식	**ALTER TABLE 테이블 명** **DROP COLUMN 컬럼 명**

문제

5. EMP01 테이블의 직급(JOB)컬럼을 삭제하시오.

```
이름          널? 유형
------- -- -------------
EMPNO         NUMBER(4)
ENAME         VARCHAR2(20)
HEIGHT        NUMBER(5,2)
ADDRESS       VARCHAR2(200)
PHONE         VARCHAR2(20)
```

3.4 SET UNUSED(1)

- 특정 테이블에서 특정 컬럼을 삭제하는 경우 다음과 같이 무조건 삭제하는 것은 위험할 수 있다.
- 테이블에 저장된 내용이 많을 경우(몇 만 건에 대한 자료) 해당 테이블에서 컬럼을 삭제하는 데 꽤 오랜 시간이 걸리게 된다. 컬럼을 삭제하는 동안에 다른 사용자가 해당 컬럼을 사용하려고 접근하게 되면 지금 현재 테이블이 사용되고 있기 때문에 다른 사용자는 해당 테이블을 이용할 수 없게 되고, 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 않습니다. 그렇기 때문에 락이 걸리는 일도 일어나지 않는다.

3.4 SET UNUSED(2)

- 컬럼을 미사용 상태로 표시

형식	**ALTER TABLE 테이블명 SET UNUSED(컬럼명); ALTER TABLE 테이블명 SET UNUSED COLUMN 컬럼명[CASCADE CONSTRAINTS];**

- 미사용으로 표시된 컬럼을 드롭

형식	**ALTER TABLE 테이블명 DROP UNUSED COLUMNS;**

- 특징
  - 둘 다 사용 불가(삭제)
  - 롤백 불가
  - 일반적으로 테이블이 매우 크거나 사용량이 최대일 때 자원을 많이 소비하는 경우에 즉시 드롭하는 대신 미사용으로 표시해둔다.

3.4 SET UNUSED(3)

❖ * EMP02 = EMP COPY TABLE

예

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

	EMPNO	ENAME	MGR	HIREDATE	SAL	COMM	DEPTNO
1	1001	김사랑	1013	07/03/01	300	(null)	20
2	1002	한예슬	1005	07/04/02	250	80	30
3	1003	오지호	1005	05/02/10	500	100	30
4	1004	이병헌	1008	03/09/02	600	(null)	20
5	1005	신농협	1005	05/04/07	450	200	30
6	1006	장농건	1008	03/10/09	480	(null)	30
7	1007	이눈세	1008	04/01/08	520	(null)	10
8	1008	감우성	1003	04/03/08	500	0	30
9	1009	안성기	(null)	96/10/04	1000	(null)	20
10	1010	이병헌	1003	05/04/07	500	(null)	10
11	1011	조향기	1007	07/03/01	280	(null)	30
12	1012	강혜정	1006	07/08/09	300	(null)	20
13	1013	박숭훈	1003	02/10/09	560	(null)	20
14	1014	조인성	1006	07/11/09	250	(null)	10

4. DROP TABLE

❖ DROP TABLE문은 기존 테이블을 삭제한다.

형식 **DROP TABLE 테이블 명;**

문제

6. EMP01 테이블을 삭제하시오.

*** 복구 불가**

```
오류:
ORA-04043: EMP01 객체가 존재하지 않습니다.
```

5. TRUNCATE

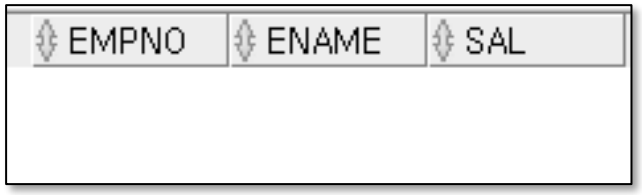
❖ TRUNCATE

- 기존에 사용하던 테이블의 모든 로우(데이터, 레코드)를 제거하기 위한 명령어
- 테이블을 최초 생성된 초기상태로 만든다.
- 용량이 줄어들고, 인덱스 등도 모두 삭제된다.
- Rollback이 불가능하다.(복구 불가)
- 무조건 전체 삭제만 가능하다.
- 삭제 행 수를 반환하지 않는다.

형식 **TRUNCATE TABLE 테이블 명;**

문제

7. TRUNCATE를 이용하여 EMP06 테이블의 모든 데이터를 삭제하시오.



6. RENAME

- RENAME
  - 기존에 사용하던 테이블의 이름을 변경하기 위한 명령어

형식 **RENAME *old_name* TO *new_name*;**

문제

8. EMP06 테이블의 이름을 TEST 란 이름으로 변경하시오.

7. 데이터 딕셔너리

- 데이터 딕셔너리(Data Dictionary)
  - 데이터 베이스 자원을 효율적으로 관리하기 위한 다양한 정보를 저장하는 '시스템 테이블'
  - 사용자가 테이블을 생성하거나 사용자를 변경하는 등의 작업을 할 때 데이터 베이스 서버에 의해 자동으로 갱신되는 테이블로 사용자는 데이터 딕셔너리의 내용을 직접 수정하거나 삭제 할 수 없다.
  - 데이터 딕셔너리를 사용자가 조회해 보면 시스템이 직접 관리하는 테이블이기에 암호 같은 기호만 보여질 뿐 내용을 알 수 없다.
  - 데이터 딕셔너리 원 테이블은 직접 조회할 수 없다.

7.1 데이터 딕셔너리 뷰

- 의미 있는 자료 조회가 불가능하기에 오라클은 사용자가 이해할 수 있는 데이터를 산출해 줄 수 있도록 하기 위해서 데이터 딕셔너리에서 파생한 데이터 딕셔너리 뷰를 제공한다.
- 데이터 딕셔너리 뷰는 접두어에 따라 다음의 세가지 종류가 있다.

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

7.2 USER_데이터 딕셔너리(1)

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

- USER가 붙은 데이터 딕셔너리 중에서 자신이 생성한 테이블이나 인덱스나 뷰 등과 같은 자신 계정 소유의 객체 정보를 저장한 USER_TABLES 데이터 딕셔너리 뷰를 살펴보도록 하자.

7.2 USER_데이터 딕셔너리(2)

- 1.DESC 명령어로 데이터 딕셔너리 뷰 USER_TABLES의 구조 확인

  예 **DESC USER_TABLES;**

- 2. USER_TABLES 데이터 딕셔너리 뷰는 현재 접속한 사용자 계정이 소유한 모든 테이블 정보를 조회 할 수 있는 뷰이기에 현재 사용자가 누구인지를 살펴본다.

  예 **SHOW USER;**

- 3. 데이터 딕셔너리 USER_TABLES의 구조를 살펴보면 무수히 많은 컬럼으로 구성되었음을 알 수 있다. 이중에서 테이블의 이름을 알려주는 TABLE_NAME 컬럼의 내용을 확인한다.

  예

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

7.3 ALL_데이터 딕셔너리(1)

- 사용자 계정이 소유한 객체는 자신의 소유이므로 당연히 접근이 가능하다.
- 오라클에서는 타계정의 객체는 원천적으로 접근이 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.
- 즉 현재 계정이 접근 가능한 테이블의 정보 조회하는 뷰이다.

7.3 ALL_데이터 딕셔너리(2)

- 1. DESC 명령어로 데이터 딕셔너리 뷰 ALL_TABLES의 구조를 살펴보도록 하자.

  예 **DESC ALL_TABLES;**

- 2. 데이터 딕셔너리 뷰 ALL_TABLES의 컬럼 역시 종류가 무수히 많다. 이 중에서 OWNER, TABLE_NAME 컬럼 값만 살펴보도록 하자.

  예 **SELECT OWNER, TABLE_NAME FROM ALL_TABLES;**

7.4 DBA_데이터 딕셔너리(1)

- DBA가 접근 가능한 객체 등을 조회할 수 있는 뷰이다.
- DBA가 접근 불가능한 정보는 없기에 데이터베이스에 있는 모든 객체 등의 의미라고 생각할 수 있다.
- USER_, ALL_와 달리 DBA_ 데이터딕셔너리뷰는 DBA 시스템 권한을 가진 사용자만 접근할 수 있다.

7.4 DBA_데이터 딕셔너리(2)

- 1. 다음은 테이블 정보를 살펴보기 위해서 DBA_TABLES 데이터 딕셔너리 뷰의 내용을 조회 해보도록 하자.

예

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

- 2. DBA 권한을 가진 SYSTEM 계정으로 접속하여 다시 한번 DBA_TABLES 데이터 딕셔너리 뷰의 내용을 조회한다. 사용자 계정과 암호를 소문자로 입력해야 인식한다는 점에 주의한다.

문제

9. 다음 표에 명시된 대로 DEPT_MISSION 테이블을 생성하시오

컬럼명	데이터 타입	크기
DNO	NUMBER	2
DNAME	VARCHAR2	14
LOC	VARCHAR2	13

10. 다음 표에 명시된 대로 EMP_MISSION 테이블을 생성하시오

컬럼명	데이터 타입	크기
ENO	NUMBER	4
ENAME	VARCHAR2	10
DNO	NUMBER	2

문제

11. 긴 이름을 저장할 수 있도록 EMP_MISSION 테이블을 수정하시오.

컬럼명	데이터 타입	크기
ENO	NUMBER	4
ENAME	VARCHAR2	25
DNO	NUMBER	2

12. EMP_MISSION 테이블을 제거하시오.

문제

13. DEPT_MISSION 테이블에서 DNAME 컬럼을 제거하시오.

14. DEPT_MISSION 테이블에서 LOC 컬럼을 UNUSED로 표시하시오.

15. DEPT_MISSION 테이블에서 UNUSED 컬럼을 제거하시오.