**1. 테이블 생성**

CREATE TABLE문장을 실행하여 테이블을 생성 합니다. 이 문장은 DDL문장으로 Oracle8 데이터베이스 구조를 생성,수정,삭제하는데 사용되는 SQL문장입니다. 이러한 문장은 데이터베이스에 즉각 영향을 미치며 데이터베이스 사전(DATA DICTIONARY)에 정보를 기록 합니다. CREATE TABLE문장을 실행 후 SQL\*Plus에서 “DESC table\_name”으로 학인 할 수 있습니다. 테이블을 생성하기 위해서는 SYSTEM PRIVILEGE(다음 과정)인 CREATE TABLE권한(SQL> SELECT \* FROM role\_sys\_privs;)을 가지고 있어야 합니다. 또한 사용자가 테이블을 만들 수 있는 공간(SQL> SELECT \* FROM user\_free\_space;)을 확보하여야 합니다.

♣ 참고

SYSTEM PRIVILEGE,OBJECT PRIVILEGE,사용자의 권한,사용 가능한 공간을 사용자는 DATA DICTIONARY를 검색하여 알 수 있다. 물론 뒤 장에서 좀더 자세히 다루도록 하겠다.

**1.1 ORACLE에서 사용하는 객체**

ORACLE 데이터베이스는 여러 개의 데이터 구조를 가지고 있습니다. 데이터베이스 설계에서 각각의 구조는 데이터베이스 개발 단계에서부터 생성할 수 있습니다.

|  |  |
| --- | --- |
| 객 체 | 설 명 |
| TABLE | 행과 열로 구성된 기본적인 저장 구조 |
| VIEW | 하나 이상의 테이블에서 데이터의 부분집합을 논리적으로 표현 |
| SEQUENCE | 고유한 번호를 자동으로 발생시키는 객체로 주로 PK 값 생성에 사용 |
| INDEX | 질의(SELECT) 성능을 향상시키기 위하여 사용하는 물리적인 저장 구조 |
| SYNONYM | 객체에 대한 이름을 부여 |

☞ Guidelines

1. 데이터베이스를 사용하고 있는 동안에도 언제든지 테이블을 생성할 수 있다.
2. 테이블의 크기는 명시할 필요가 없다.
3. 테이블 구조는 언제든지 수정 가능하다.

문제1) ORACLE DATA DICTIONARY에서 SYSTEM PRIVILEGE를 조회하여라.

|  |
| --- |
| SQL> **SELECT \***  2 **FROM system\_privilege\_map;**  PRIVILEGE NAME  --------- ----------------------------------------  -3 ALTER SYSTEM  -4 AUDIT SYSTEM  . . . . . . . . . .  86 rows selected. |

문제2) ORACLE DATA DICTIONARY에서 OBJECT PRIVILEGE를 조회하여라.

|  |
| --- |
| SQL> **SELECT \***  2 **FROM table\_privilege\_map;**  PRIVILEGE NAME  --------- ----------------------------------------  0 ALTER  1 AUDIT  2 COMMENT  . . . . . . . . . .  13 rows selected. |

문제3) 현재 SESSION을 이루고 있는 사용자의 SYSTEM PRIVILEGE중 ROLE에 관련된 사항을 ORACLE DATA DICTIONARY에서 조회하여라.

|  |
| --- |
| SQL> **CONN scott/tiger**  Connected.  SQL> **SELECT \***  2 **FROM role\_sys\_privs;**  ROLE PRIVILEGE ADM  ------------------------------ ---------------------------------------- ---  CONNECT ALTER SESSION NO  CONNECT CREATE CLUSTER NO  CONNECT CREATE DATABASE LINK NO  . . . . . . . . .  13 rows selected. |

문제4) 현재 SESSION을 이루고 있는 사용자가 사용할 수 있는 FREE SPACE가 얼마인지 ORACLE DATA DICTIONARY에서 조회하여라.

|  |
| --- |
| SQL> **CONN scott/tiger**  Connected.  SQL> **SELECT \***  2 **FROM user\_free\_space;**  TABLESPACE\_NAME FILE\_ID BLOCK\_ID BYTES BLOCKS  ------------------------------ --------- --------- --------- ---------  SYSTEM 1 8370 71680 35  . . . . . . . . . .  USER\_DATA 2 57 61440 30  ROLLBACK\_DATA 3 802 3602432 1759  TEMPORARY\_DATA 4 2 2095104 1023  10 rows selected. |

**1.1.1 Syntax**

|  |
| --- |
| **CREATE TABLE** [schema.]table\_name  ( column datatype [**DEFAULT** expr] [column\_constraint],  . . . . . . . .  [table\_constraint]); |

schema 테이블의 소유자

table\_name 생성하고자 하는 테이블 이름. 사용자 단위로 유일한 이름

column 테이블에서 사용하는 열 이름. 테이블 단위로 유일한 이름

datatype 열의 자료형

DEFAULT expr INSERT문장에서 값을 생략시 기본적으로 입력되는 값을 명시

column\_constraint 열정의 부분에서 무결성 제약 조건을 기술

table\_constraint 테이블 정의 부분에서 무결성 제약 조건을 기술

**1.1.2 이름 지정 규칙**

객체 이름을 지정하는 표준 규칙에 따라 데이터베이스 테이블과 열의 이름을 정합니다.

1. 문자로 시작하여야 한다
2. 문자의 길이는 1 ~ 30이내를 사용한다.
3. 오직 A ~ Z, a ~ z, 0 ~ 9, \_, $, # 만을 사용 가능하다.단 한글 데이터베이스에서는 한글 사용 가능하다.
4. 동일한 사용자가 소유한 객체 이름은 중복될 수 없다.
5. 예약어는 사용할 수 없다.

**1.1.3 DATA TYPE**

|  |  |
| --- | --- |
| DATA TYPE | 설 명 |
| VARCHAR2(n) | 가변 길이 문자 데이터(1~4000byte) |
| CHAR(n) | 고정 길이 문자 데이터(1~2000byte) |
| NUMBER(p,s) | 전체 p자리 중 소수점 이하 s자리(p:1~38, s:-84~127) |
| DATE | 7Byte(BC 4712년 1월 1일부터 AD 9999년 12월 31일) |
| LONG | 가변 길이 문자 데이터(1~2Gbyte) |
| CLOB | 단일 바이트 가변 길이 문자 데이터(1~4Gbyte) |
| RAW(n) | n Byte의 원시 이진 데이터(1~2000) |
| LONG RAW | 가변 길이 원시 이진 데이터(1~2Gbyte) |
| BLOB | 가변 길이 이진 데이터(1~4Gbyte) |
| BFILE | 가변 길이 외부 파일에 저장된 이진 데이터(1~4Gbyte) |

☞ Guidelines

1. 테이블이나 다른 데이터베이스 객체에 대한 서술적인 이름을 사용하여라.
2. 다른 테이블에도 일관되게 똑같은 이름을 지정하여라(예:EMP와DEPT의 DEPTNO)
3. 객체 이름은 대소문자를 구분하지 않는다.

**1.1.4 DEFAULT OPTION**

열은 DEFAULT OPTION을 사용하여 DEFAULT 값을 부여할 수 있다. 이 OPTION은 열에 대한 값없이 어떤 행을 입력할 경우 NULL값이 입력되지 않게 해 줍니다.

1. 삽입시 열에 대한 Default값을 명시한다.
2. 기술 가능한 값은 literal 값, 표현식, SQL 함수(SYSDATE,USER등)이다.
3. 불가능한 값은 다른 열의 이름이나 의사(NEXTVAL,CURRVAL등)열 입니다.
4. DEFAULT DATA TYPE은 열의 DATA TYPE과 일치해야 한다.

**1.2 제약 조건**

ORACLE SERVER은 부적절한 자료가 입력되는 것을 방지하기 위하여 constraint을 사용한다.

1. 제약 조건은 테이블 LEVEL에서 규칙을 적용합니다.
2. 제약 조건은 종속성이 존재할 경우 테이블 삭제를 방지 합니다.
3. 테이블에서 행이 삽입,갱신,삭제될 때마다 테이블에서 규칙을 적용합니다.
4. Developer/2000 같은 ORACLE TOOL에 대한 규칙을 제공 합니다.
5. 제약 조건의 유형은 ORACLE에서 유효합니다.

☞ Guidelines

모든 제약 조건은 DATA DICTIONARY에 저장 됩니다. 제약 조건의 이름을 의미 있게 부여했다면 참조하기가 보다 쉽습니다. 제약 조건의 이름은 표준 객체 이름 규칙을 따릅니다. 제약 조건을 명명하지 않는다면 ORACLE SERVER이 SYS\_Cnnnnn의 형태로 이름을 생성합니다.

제약 조건은 테이블 생성시나 테이블이 생성된 후에 정의될 수 있습니다. 또는 일시적으로 DISABLE할 수 있고 ENABLE할 수도 있습니다. User\_constraints의 DATA DICTIONARY VIEW을 조회하므로 지정 테이블에 대해 정의된 제약 조건을 볼 수 있다.

**1.2.1 제약 조건 정의 방법**

제약 조건 정의하는 방법에는 COLUMN LEVEL과 TABLE LEVEL 두 가지 방법이 있다.

**가) 컬럼 LEVEL 제약 조건(COLUMN LEVEL CONSTRAINT)**

1. 열별로 제약 조건을 정의한다.
2. 무결성 제약 조건 5가지를 모두 적용할 수 있다.
3. NOT NULL제약 조건은 컬럼 LEVEL에서만 가능 하다.

* Syntax

|  |
| --- |
| Column datatype [**CONSTRAINT** constraint\_name ] constraint\_type |

**나) 테이블 LEVEL 제약 조건(TABLE LEVEL CONSTRAINT)**

1. 테이블의 칼럼 정의와는 개별적으로 정의한다.
2. 하나 이상의 열을 참조할 경우에 사용
3. NOT NULL을 제외한 나머지 제약 조건만 정의 가능하다.

* Syntax

|  |
| --- |
| column datatype,  [**CONSTRAINT** constraint\_name] constraint\_type (column1[,column2,......]) |

☞ Guidelines

constraint name만 보고도 어떤 용도의 CONSTRAINT인가를 식별할 수 있으면 사용자는 쉽게 데이터베이스를 운용할 것이다. 다음의 규칙에 따라 constraint name을 부여하기를 권장한다. tablename\_cloumnname\_constrainttype (예 : emp\_empno\_pk, emp\_deptno\_fk)

**1.2.2 데이터 무결성 제약 조건의 종류**

|  |  |
| --- | --- |
| 제 약 조 건 | 설 명 |
| PRIMARY KEY(PK) | 유일하게 테이블의 각행을 식별(NOT NULL과 UNIQUE조건을 만족) |
| FOREIGN KEY(FK) | 열과 참조된 열 사이의 외래키 관계를 적용하고 설정합니다. |
| UNIQUE key(UK) | 테이블의 모든 행을 유일하게 하는 값을 가진 열(NULL을 허용) |
| NOT NULL(NN) | 열은 NULL값을 포함할 수 없습니다. |
| CHECK(CK) | 참이어야 하는 조건을 지정함(대부분 업무 규칙을 설정) |

**가) PRIMARY KEY(PK)**

1. 테이블에 대한 기본 키를 생성합니다.
2. 하나의 기본 키만이 각 테이블에 대해 존재할 수 있다.
3. PRIMARY KEY 제약 조건은 테이블에서 각행을 유일하게 식별하는 열 또는 열의 집합입니다.(UNIQUE와 NOT NULL조건을 만족)
4. 이 제약 조건은 열 또는 열의 집합의 유일성을 요구하고 NULL값을 포함할 수 없음을 보증 합니다.
5. UNIQUE INDEX가 자동 생성된다.

* Syntax

|  |
| --- |
| column datatype [**CONSTRAINT** constraint\_name] **PRIMARY KEY** (col1[,col2,..]) |
| column datatype,  . . . . . . . ,  [CONSTRAINT constraint\_name] **PRIMARY KEY** (column1[,column2,..]) |

문제5) 아래의 두 문장의 차이점을 설명하여라.

|  |
| --- |
| SQL> **CREATE TABLE test\_tab1(**  2 **id NUMBER(2) CONSTRAINT test\_id\_pk PRIMARY KEY,**  3 **name VARCHAR2(10));**  Table created. |
| SQL> **CREATE TABLE test\_tab2(**  2 **id NUMBER(2),**  3 **name VARCHAR2(10),**  4 **CONSTRAINT test\_id\_pk PRIMARY KEY (id));**  Table created. |

**나) FOREIGN KEY(FK)**

1. FOREIGN KEY는 DETAIL쪽에서 정의한다.
2. MASTER TABLE의 PRIMARY KEY,UNIQUE KEY로 정의된 열을 지정할 수 있으며 열의 값과 일치하거나 NULL값이어야 한다.
3. FOREIGN KEY는 열 또는 열의 집합을 지정할 수 있으며 동일 테이블 또는 다른 테이블간의 관계를 지정할 수 있다.
4. ON DELETE CASCADE을 사용하여 DETAIL TABLE에서 관련된 행을 삭제하고 MASTER TABLE에서 삭제를 허용할 수 있다.

* Syntax

|  |
| --- |
| column datatype [**CONSTRAINT** constraint\_name]  **REFERENCES** table\_ name (column1[,column2,..] [**ON DELETE CASCADE**]) |
| column datatype,  . . . . . . . ,  [**CONSTRAINT** constraint\_name] **FOREIGN KEY** (column1[,column2,..])  **REFERENCES** table\_name (column1[,column2,..] [**ON DELETE CASCADE**]) |

문제6) 아래의 두 문장의 차이점을 설명하여라.

|  |
| --- |
| SQL> **CREATE TABLE DEPT\_TAB (**  2 **DEPTNO NUMBER(2),**  3 **DNAME CHAR(14),**  4 **LOC CHAR(13),**  5 **CONSTRAINT DEPT\_DEPTNO\_PK PRIMARY KEY (DEPTNO));**  Table created. |
| SQL> **CREATE TABLE EMP\_TAB (**  2 **EMPNO NUMBER(4),**  3 **ENAME VARCHAR2(10),**  4 **JOB VARCHAR2(9),**  5 **MGR NUMBER(4) CONSTRAINT EMP\_SELF\_KEY REFERENCES EMP (EMPNO),**  6 **HIREDATE DATE,**  7 **SAL NUMBER(7,2),**  8 **COMM NUMBER(7,2),**  9 **DEPTNO NUMBER(2) NOT NULL,**  10 **CONSTRAINT EMP\_DEPTNO\_FK FOREIGN KEY (DEPTNO) REFERENCES DEPT (DEPTNO),**  11 **CONSTRAINT EMP\_EMPNO\_PK PRIMARY KEY (EMPNO));**  Table created. |

♣ 주의

FOREIGN KEY 값은 MASTER TABLE에서 존재하는 값과 일치해야 하거나 NULL이 되어야 한다. FOREIGN KEY 값은 데이터 값을 기초로 하여 순전히 논리적이지 문리적이거나 포인터가 아니다. MASTER TABLE(parent)은 참조 당하는 쪽(DEPT TABLE)을 테이블을 의미하고 DETAIL TABLE(child)은 참조하는 쪽(EMP TABLE)의 테이블을 의미한다.

☞ Guidelines

1. MASTER TABLE(참조 당하는 쪽)을 먼저 생성하여야 한다.
2. MASTER TABLE에 PRIMARY KEY 또는 UNIQUE KEY로 설정된 열을 DETAIL TABLE에서 참조하여야 한다.
3. MASTER TABLE과 DETAIL TABLE의 참조하는 열과 참조 당하는 쪽의 자료형과 크기가 일치해야 한다.

**다) UNIQUE key(UK)**

1. UNIQUE Key 무결성 제약 조건은 열 또는 열 집합의 모든 값들이 유일해야 한다.
2. 중복된 값을 가지는 행이 존재할 수 없음을 의미한다.
3. PRIMARY KEY와 유사하나 NULL을 허용한다.
4. 열이 하나 이상 포함되어 있다면 composite unique key라 부릅니다.
5. UNIQUE Key에 대하여 UNIQUE INDEX가 자동 생성된다.

* Syntax

|  |
| --- |
| column datatype [**CONSTRAINT** constraint\_name] **UNIQUE** |
| column datatype,  . . . . . . . ,  [**CONSTRAINT** constraint\_name] **UNIQUE** (column1[,column2,..]) |

문제7) 아래의 두 문장의 차이점을 설명하여라.

|  |
| --- |
| SQL> **CREATE TABLE UNI\_TAB1 (**  2 **DEPTNO NUMBER(2) CONSTRAINT UNI\_TAB\_DEPTNO\_UK UNIQUE,**  3 **DNAME CHAR(14),**  4 **LOC CHAR(13));**  Table created. |
| SQL> **CREATE TABLE UNI\_TAB2 (**  2 **DEPTNO NUMBER(2),**  3 **DNAME CHAR(14),**  4 **LOC CHAR(13),**  5 **CONSTRAINT UNI\_TAB\_DEPTNO\_UK UNIQUE (DEPTNO));**  Table created. |

**라) NOT NULL(NN)**

1. NOT NULL 제약 조건은 열에서 NULL을 허용하지 않도록 보증한다.
2. NOT NULL 제약 조건이 없는 열은 DEFAULT로 NULL을 허용한다.
3. NOT NULL 제약 조건은 COLUMN CONSTRAINT에서만 기술 가능하다.

* Syntax

|  |
| --- |
| column datatype [**CONSTRAINT** constraint\_name] **NOT NULL** |
| column datatype,  . . . . . . . ,  [**CONSTRAINT** constraint\_name] **NOT NULL** (column1[,column2,..]) |

문제8) 아래의 두 문장의 차이점을 설명하여라.

|  |
| --- |
| SQL> **CREATE TABLE NN\_TAB1 (**  2 **DEPTNO NUMBER(2) CONSTRAINT UNI\_TAB\_DEPTNO\_NN NOT NULL,**  3 **DNAME CHAR(14),**  4 **LOC CHAR(13));**  Table created. |
| SQL> **CREATE TABLE NN\_TAB2 (**  2 **DEPTNO NUMBER(2),**  3 **DNAME CHAR(14),**  4 **LOC CHAR(13),**  5 **CONSTRAINT UNI\_TAB\_DEPTNO\_NN NOT NULL (DEPTNO));**  CONSTRAINT UNI\_TAB\_DEPTNO\_NN NOT NULL (DEPTNO))  \*  ERROR at line 5:  ORA-00904: invalid column name |

**마) CHECK(CK)**

1. CHECK 제약 조건은 행이 만족해야 하는 조건을 정의한다.
2. 다음과 같은 표현식은 허용되지 않습니다.
3. CURRVAL, NEXTVAL, LEVEL, ROWNUM에 대한 참조
4. SYSDATE, UID, USER, USERENV 함수에 대한 호출
5. 다른 행에 있는 값을 참조하는 질의
6. ORACLE SERVER가 사용하는 예약어

* Syntax

|  |
| --- |
| column datatype [**CONSTRAINT** constraint\_name] **CHECK** (condition) |
| column datatype,  . . . . . . . ,  [**CONSTRAINT** constraint\_name] **CHECK** (condition) |

문제9) 아래의 두 문장의 차이점을 설명하여라.

|  |
| --- |
| SQL> **CREATE TABLE CK\_TAB1 (**  2 **DEPTNO NUMBER(2)**  3 **CONSTRAINT UNI\_TAB\_DEPTNO\_CK CHECK (DEPTNO IN (10,20,30,40,50)),**  4 **DNAME CHAR(14),**  5 **LOC CHAR(13));**  Table created. |
| SQL> **CREATE TABLE CK\_TAB2 (**  2 **DEPTNO NUMBER(2),**  3 **DNAME CHAR(14),**  4 **LOC CHAR(13),**  5 **CONSTRAINT UNI\_TAB\_DEPTNO\_CK CHECK (DEPTNO IN (10,20,30,40,50)));**  Table created. |

**1.3 테이블 차트에 의한 테이블 생성**

**1.3.1 테이블 차트**

**가) TABLE NAME : POST**

|  |  |  |  |
| --- | --- | --- | --- |
| Column name | POST1 | POST2 | ADDR |
| Key Type | PK | |  |
| Nulls/Unique |  |  | NN |
| Data Type | CHAR | CHAR | VARCHAR2 |
| Maximun Length | 3 | 3 | 60 |
| Sample |  |  | 경기도 성낭시 분당구 정자동 |

**나) TABLE NAME : MEMBER**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Column name | ID | NAME | SEX | JUMIN1 | JUMIN2 | TEL | POST1 | POST2 | ADDR |
| Key Type | PK |  |  |  |  |  | FK | |  |
| Nulls/Unique |  | NN |  | UK | |  |  |  |  |
| FK Ref Table |  |  |  |  |  |  | POST | |  |
| FK Ref Column |  |  |  |  |  |  | POST1,POST2 | |  |
| Check |  |  | 1,2 |  |  |  |  |  |  |
| Data Type | NUM | VAR | CHAR | CHAR | CHAR | VAR | CHAR | CHAR | VAR |
| Maximun Length | 4 | 10 | 1 | 6 | 7 | 15 | 3 | 3 | 60 |
| Sample | 1234 | 홍길동 | 1 | 990101 | 1232344 | 712-1234 | 100 | 010 |  |

**1.3.2 테이블 생성 문**

**가) TABLE NAME : POST**

|  |
| --- |
| SQL> **CREATE TABLE post(**  2 **post1 CHAR(3),**  3 **post2 CHAR(3),**  4 **addr VARCHAR2(60) CONSTRAINT post\_addr\_nn NOT NULL,**  5 **CONSTRAINT post\_post12\_pk PRIMARY KEY (post1,post2));**  Table created. |

**나) TABLE NAME : MEMBER**

|  |
| --- |
| SQL> **CREATE TABLE member(**  2 **id NUMBER(4) CONSTRAINT member\_id\_pk PRIMARY KEY,**  3 **name VARCHAR(10) CONSTRAINT member\_name\_nn NOT NULL,**  4 **sex CHAR(1) CONSTRAINT member\_sex\_ck CHECK ( sex IN ('1','2')),**  5 **jumin1 CHAR(6),**  6 **jumin2 CHAR(7),**  7 **tel VARCHAR2(15),**  8 **post1 CHAR(3),**  9 **post2 CHAR(3),**  10 **addr VARCHAR2(60),**  11 **CONSTRAINT member\_jumin12\_uk UNIQUE (jumin1,jumin2),**  12 **CONSTRAINT member\_post12\_fk FOREIGN KEY (post1,post2)**  13 **REFERENCES post (post1,post2));**    Table created. |

**1.4 SUBQUERY을 사용한 테이블 생성**

테이블 생성시 이미 만들어져 있는 기존의 테이블을 이용하여 특정 열 또는 특정 행들만을 추출하여 사용자가 필요로 하는 새로운 테이블을 만들 수 있다.

**1.4.1 SUBQUERY을 이용하여 테이블 생성 방법**

1. CREATE TABLE문장과 AS SUBQUERY을 사용하여 테이블을 생성하고 행을 삽입합니다.
2. SUBQUERY의 열의 개수와 명시된 열의 개수를 좌측부터 일치시킨다.
3. 열 이름과 DEFAULT VALUE를 가진 열을 정의한다.

**1.4.2 Syntax**

|  |
| --- |
| **CREATE TABLE** table\_name [column1[,column2, . . . . . .]]  **AS** subquery |

table\_name 테이블의 이름

column1 열 이름, DEFAULT VALUE, 무결성 제약 조건

subquery 새로운 테이블에 삽입할 행의 집합을 정의한 SELECT문장

☞ Guidelines

1. 테이블은 명시된 열 이름으로 생성, SQL문장에 의해 RETURN된 행들이 테이블에 삽입.
2. 열 정의는 오직 열 이름과 DEFAULT VALUE만 정의 가능
3. 열이 기술되었다면 열의 수는 SUBQUERY의 열과 좌측부터 1대1 대응
4. 열이 기술되기 않았다면 테이블의 열 이름은 SUBQUERY의 열 이름과 동일
5. SUBQUERY에서 계산 식이나 함수를 사용하면 계산식과 함수를 열 이름으로 사용할 수 없기 때문에 반드시 Alias을 지정하거나 table\_name옆에 열 이름을 기술하여야 한다.

문제10) EMP 테이블에서30부서에 근무하는 사원의 정보만 추출하여 EMP\_30 테이블을 생성하여라. 단 열은 사원번호,이름,업무,입사일자,급여,보너스를 포함한다,

|  |
| --- |
| SQL> **CREATE TABLE emp\_30**  2 **AS SELECT empno,ename,job,hiredate,sal,comm**  3 **FROM emp**  4 **WHERE deptno = 30;**  Table created.  SQL> **SELECT \***  2 **FROM emp\_30;**  EMPNO ENAME JOB HIREDATE SAL COMM  --------- ---------- --------- ------------------ --------- ---------  7698 BLAKE MANAGER 01-MAY-81 2850  7654 MARTIN SALESMAN 28-SEP-81 1250 1400  . . . . . . . . . .  6 rows selected. |

문제11) EMP 테이블에서 부서별로 인원수,평균 급여,급여의 합,최소 급여,최대 급여를 포함하는 EMP\_DEPTNO 테이블을 생성하여라.

|  |
| --- |
| SQL> **CREATE TABLE emp\_deptno**  2 **AS SELECT deptno,COUNT(\*),AVG(sal),SUM(sal),MIN(sal),MAX(sal)**  3 **FROM emp**  4 **GROUP BY deptno;**  AS SELECT deptno,COUNT(\*),AVG(sal),SUM(sal),MIN(sal),MAX(sal)  \*  ERROR at line 2:  ORA-00998: must name this expression with a column alias  SQL> **CREATE TABLE emp\_deptno (deptno,e\_count,e\_avg,e\_sum,e\_min,e\_max)**  2 **AS SELECT deptno,COUNT(\*),AVG(sal),SUM(sal),MIN(sal),MAX(sal)**  3 **FROM emp**  4 **GROUP BY deptno;**  Table created.  SQL> **SELECT \***  2 **FROM emp\_deptno;**  DEPTNO E\_COUNT E\_AVG E\_SUM E\_MIN E\_MAX  --------- --------- --------- --------- --------- ---------  10 3 2983.3333 8950 1500 5000  20 5 2175 10875 800 3000  30 6 1566.6667 9400 950 2850 |

문제12) EMP 테이블에서 사원번호,이름,업무,입사일자,부서번호만 포함하는 EMP\_TEMP 테이블을 생성하는데 자료는 포함하지 않고 구조만 생성하여라.

|  |
| --- |
| SQL> **CREATE TABLE emp\_temp**  2 **AS SELECT empno,ename,job,hiredate,deptno**  3 **FROM emp**  4 **WHERE 1 = 2;**  Table created.  SQL> **SELECT \***  2 **FROM emp\_temp;**  no rows selected |

**1.5 데이터 사전(DATA DICTIONARY) 질의**

사용자가 소유한 다양한 데이터베이스 객체를 보기 위해서 데이터 사전을 질의하여 알 수 있다.

1. DDL문장을 실행하면 그 정보는 데이터 사전(DATA DICTIONARY)에 등록
2. 사용자가 소유한 테이블을 조회

문제13) 현재 SESSION을 이루고 있는 사용자가 소유하고 있는 TABLE을 조화하여라.

|  |
| --- |
| SQL> **SELECT table\_name,tablespace\_name**  2 **FROM user\_tables;**  TABLE\_NAME TABLESPACE\_NAME  ------------------------------ ------------------------------  BONUS USER\_DATA  CUSTOMER USER\_DATA  . . . . . . . . .  15 rows selected. |

문제14) 현재 SESSION을 이루고 있는 사용자가 소유한 모든 객체를 조회하여라

|  |
| --- |
| SQL> **COL object\_name FORMAT a20**  SQL> **COL timestamp FORMAT a25**  SQL> **SELECT object\_name,object\_type,timestamp**  2 **FROM user\_objects;**  OBJECT\_NAME OBJECT\_TYPE TIMESTAMP  -------------------- ------------- -------------------------  BONUS TABLE 1999-02-11:17:21:32  CUSTID SEQUENCE 1999-02-11:17:21:41  . . . . . . . . .  29 rows selected. |

문제15) 현재 SESSION을 이루고 있는 사용자가 소유한 테이블,뷰,동의어,시퀀스를 조회하여라

|  |
| --- |
| SQL> **SELECT \***  2 **FROM user\_catalog;**  TABLE\_NAME TABLE\_TYPE  ------------------------------ -----------  BONUS TABLE  CUSTID SEQUENCE  CUSTOMER TABLE  . . . . . . . . . .  19 rows selected. |

**2. 테이블을 수정**

테이블을 생성한 이후에 열이 생략 되었거나, 열 정의를 변경할 필요가 있을 수 있다. 테이블의 구조를 변경할 경우 ALTER TABLE명을 사용하여 변경 한다.

**2.1 새로운 열 추가**

1. 새로운 열을 추가 할 수는 있지만 테이블에 있는 기존의 열은 DROP할 수 없다.
2. 열이 위치를 기술할 수 없으며 항상 테이블에서 마지막에 위치 합니다.
3. 열을 추가할 때 테이블이 행을 포함하고 있다면 새로운 열은 이미 존재하는 열을 NULL로 초기화 한다.

**2.1.1 Syntax**

|  |
| --- |
| **ALTER TABLE** table\_name  **ADD** (column datatype [**DEFAULT** expr]  [,column datatype [**DEFAULT** expr] . . . . . . .] |

문제16) BONUS 테이블에 ETC COLUMN을 추가하여라. 단 자료형은 VARCHAR2(50) 사용하여라.

|  |
| --- |
| SQL> **ALTER TABLE bonus**  2 **ADD (etc VARCHAR2(50));**  Table altered.  SQL> **desc bonus**  Name Null? Type  ------------------------------- -------- ----  ENAME CHAR(10)  JOB CHAR(9)  SAL NUMBER  COMM NUMBER  ETC VARCHAR2(50) |

문제17) EMP\_30 테이블에 DEPTNO NUMBER(2)을 추가하여라.

|  |
| --- |
| SQL> **ALTER TABLE emp\_30**  2 **ADD (deptno number(2));**  Table altered.  SQL> **SELECT \***  2 **FROM emp\_30;**  EMPNO ENAME JOB HIREDATE SAL COMM  --------- ---------- --------- ------------------ --------- ---------  7698 BLAKE MANAGER 01-MAY-81 2850  7654 MARTIN SALESMAN 28-SEP-81 1250 1400  7499 ALLEN SALESMAN 20-FEB-81 1600 300  7844 TURNER SALESMAN 08-SEP-81 1500 0  7900 JAMES CLERK 03-DEC-81 2450  7521 WARD SALESMAN 22-FEB-81 1250 500  6 rows selected. |
| SQL> **ALTER TABLE emp\_30**  2 **ADD (deptno number(2));**  ALTER TABLE emp\_30  \*  ERROR at line 1:  ORA-00942: table or view does not exist  위 와 같이 ERROR가 발생되면 EMP\_30 테이블이 존재하지 않는 경우입니다. 이전 문제(EMP\_30 테이블을 생성)를 수행 후 다시 한번 위 예를 실행하여 보시오 |

**2.2 열 수정**

ALTER TABLE의 MODIFY절을 사용하여 열의 정의를 수정할 수 있습니다. 열의 수정은 열의 자료형, 크기,DEFAULT VALUE입니다.

**2.2.1 Syntax**

|  |
| --- |
| **ALTER TABLE** table\_name  **MODIFY** (column datatype [**DEFAULT** expr]  [,column datatype [**DEFAULT** expr] . . . . . . .] |

문제18) 아래 구문들을 설명하여라.

|  |
| --- |
| SQL> **DESC emp\_30**  Name Null? Type  ------------------------------- -------- ----  EMPNO NOT NULL NUMBER(4)  ENAME VARCHAR2(10)  JOB VARCHAR2(9)  HIREDATE DATE  SAL NUMBER(7,2)  COMM NUMBER(7,2)  SQL> **ALTER TABLE emp\_30**  2 **MODIFY (ename VARCHAR2(15));**  Table altered.  SQL> **DESC emp\_30**  Name Null? Type  ------------------------------- -------- ----  EMPNO NOT NULL NUMBER(4)  ENAME VARCHAR2(15)  JOB VARCHAR2(9)  HIREDATE DATE  SAL NUMBER(7,2)  COMM NUMBER(7,2) |
| SQL> **ALTER TABLE emp\_30**  2 **MODIFY (empno CHAR(4));**  MODIFY (empno CHAR(4))  \*  ERROR at line 2:  ORA-01439: column to be modified must be empty to change datatype  SQL> **ALTER TABLE emp\_30**  2 **MODIFY (JOB char(9));**  Table altered.  SQL> **DESC emp\_30;**  Name Null? Type  ------------------------------- -------- ----  EMPNO NOT NULL NUMBER(4)  ENAME VARCHAR2(15)  JOB CHAR(9)  HIREDATE DATE  SAL NUMBER(7,2)  COMM NUMBER(7,2) |
| SQL> **DESC emp\_30**  Object does not exist.  위 와 같이 MESSAGE가 발생되면 EMP\_30 테이블이 존재하지 않는 경우입니다. 이전 문제(EMP\_30 테이블을 생성)를 수행 후 다시 한번 위 예를 실행하여 보시오 |

☞ Guidelines

1. 숫자 열의 정밀도나 폭을 증가할 수 있다.
2. 열이 모두 NULL이거나 테이블에 자료가 없으면 열의 폭을 감소시킬 수 있다.
3. 열이 NULL을 포함하면 열의 자료형을 변경할 수 있다.
4. 열이 NULL을 포함하거나 크기를 변경하지 않으면 CHAR을 VARCHAR2로 변경하거나 그 반대의 경우도 가능하다.
5. 열의 DEFAULT VALUE를 변경하는 것은 이후의 INSERT문장에만 영향을 미칩니다.

**2.3 제약 조건 추가**

ADD절을 가지는 ALTER TABLE문장을 사용하여 기존의 테이블에 대한 제약 조건을 추가할 수 있다.

**2.3.1 Syntax**

|  |
| --- |
| **ALTER TABLE** table\_name  **ADD** [**CONSTRAINT** constraint\_name] constraint\_type (column); |

제약 조건의 선택 사항이지만 기술하기를 권장한다. 기술하지 않을 경우는 ORACLE SERVER이 생성(SYS\_Cnnnnn)하여 부여한다.

문제19) EMP 테이블에서 이름 필드에 UNIQUE CONSTRAINT를 설정하고 DATA DICTIONARY에서 확인 하여라.

|  |
| --- |
| SQL> **ALTER TABLE emp**  2 **ADD CONSTRAINT emp\_ename\_uk UNIQUE (ename);**  Table altered.  SQL> **SELECT constraint\_name,table\_name,status**  2 **FROM user\_constraints;**  CONSTRAINT\_NAME TABLE\_NAME STATUS  ------------------------------ ------------------------------ --------  . . . . . . . . . .  EMP\_PRIMARY\_KEY EMP ENABLED  EMP\_SELF\_KEY EMP ENABLED  EMP\_FOREIGN\_KEY EMP ENABLED  EMP\_ENAME\_UK EMP ENABLED  . . . . . . . . . .  33 rows selected. |

☞ Guidelines

1. 제약 조건의 추가, 삭제는 가능하지만 변경은 불가능하다.
2. 제약 조건의 활성화 또는 비활성화가 가능하다.
3. MODIFY절을 사용하여 NOT NULL제약 조건을 추가한다.

♣ 주의

데이터는 열이 추가되는 시점에서 기존의 열에 대해 명시될 수 없기 때문에 테이블에 행이 하나도 없을 경우에만 NOT NULL열을 정의할 수 있다.

**2.4 제약 조건 삭제**

DROP 절을 가지는 ALTER TABLE문장을 사용하여 기존의 테이블에 대한 제약 조건을 삭제할 수 있다.

**2.4.1 Syntax**

|  |
| --- |
| **ALTER TABLE** table\_name  **DROP PRIMARY KEY** | UNIQUE (column) | **CONSTRAINT** constraint\_name [**CASCADE**]; |

♣ 참고

제약 조건을 삭제하면 USER\_CONSTRAINTS, USER\_CONS\_COLUMN 데이터 사전 뷰에서 제약 조건 이름을 삭제한다. DROP의 CASCADE문장은 모든 종속적인 제약 조건이 삭제됩니다.

문제20) 전 문제에서 생성한 EMP 테이블에 있는 emp\_ename\_uk을 삭제하고 DATA DICTIONARY에서 확인하여라.

|  |
| --- |
| SQL> **ALTER TABLE emp**  2 **DROP CONSTRAINT emp\_ename\_uk;**  Table altered.  SQL> **SELECT constraint\_name,table\_name,status**  2 **FROM user\_constraints;**  CONSTRAINT\_NAME TABLE\_NAME STATUS  ------------------------------ ------------------------------ -------  . . . . . . . . . . .  EMP\_PRIMARY\_KEY EMP ENABLED  EMP\_SELF\_KEY EMP ENABLED  EMP\_FOREIGN\_KEY EMP ENABLED  . . . . . . . . . . .  32 rows selected. |

♣ 참고

무결성 제약 조건을 삭제할 때, 그 제약 조건은 더 이상 ORACLE SERVER에 의해 적용되 않으며, 데이터 사전에서 확인할 수 없다.

**2.5 제약 조건 비활성화**

DISABLE절을 가지는 ALTER TABLE문장을 사용하여 삭제 또는 재생성 없이 제약 조건을 비활성화할 수 있다.

☞ Guidelines

1. 무결성 제약 조건을 비활성화 하기 위하여 ALTER TABLE문장을 사용하여 DISABLE할 수 있다.
2. 종속적인 무결성 제약 조건을 비활성화 하기 위하여 CASCADE를 사용한다.
3. CREATE TABLE문장과 ALTER TABLE문장으로 DISABLE할 수 있다.

**2.5.1 Syntax**

|  |
| --- |
| **ALTER TABLE** table\_name  **DISABLE CONSTRAINT** constraint\_name [**CASCADE**]; |

문제21) EMP 테이블에 있는 PRIMARY KEY(EMP\_PRIMARY\_KEY)를 DISABLE하여라.

|  |
| --- |
| SQL> **SELECT constraint\_name,table\_name,status**  2 **FROM user\_constraints;**  CONSTRAINT\_NAME TABLE\_NAME STATUS  ------------------------------ ------------------------------ --------  . . . . . . .  EMP\_PRIMARY\_KEY EMP ENABLED  EMP\_SELF\_KEY EMP ENABLED  EMP\_FOREIGN\_KEY EMP ENABLED  . . . . . . . .  32 rows selected.  SQL> **ALTER TABLE emp**  2 **DISABLE CONSTRAINT emp\_primary\_key;**  ALTER TABLE emp  \*  ERROR at line 1:  ORA-02297: cannot disable constraint (SCOTT.EMP\_PRIMARY\_KEY) - dependencies exist  SQL> **ALTER TABLE emp**  2 **DISABLE CONSTRAINT emp\_primary\_key CASCADE;**  Table altered.  SQL> **SELECT constraint\_name,table\_name,status**  2 **FROM user\_constraints;**  CONSTRAINT\_NAME TABLE\_NAME STATUS  ------------------------------ ------------------------------ --------  . . . . . . . . . .  EMP\_PRIMARY\_KEY EMP DISABLED  EMP\_SELF\_KEY EMP DISABLED  EMP\_FOREIGN\_KEY EMP ENABLED  . . . . . . . . . .  32 rows selected. |

**2.6 제약 조건 활성화**

ENABLE절을 가지는 ALTER TABLE문장을 사용하여 삭제 또는 재생성 없이 제약 조건을 활성화할 수 있다.

☞ Guidelines

1. 제약 조건이 활성화 된다면 그 제약 조건은 테이블의 모든 데이터에 적용된다. 테이블의 모든 자료는 데이터의 제약 조건과 일치해야 한다.
2. UNIQUE key와 PRIMARY key는 제약 조건이 활성화 된다면 UNIQUE INDEX가 자동 생성된다.
3. CREATE TABLE문장과 ALTER TABLE문장으로 ENABLE할 수 있다.

**2.6.1 Syntax**

|  |
| --- |
| **ALTER TABLE** table\_name  **ENABLE CONSTRAINT** constraint\_name; |

문제22) EMP 테이블에 있는 PRIMARY KEY(EMP\_PRIMARY\_KEY), FOREIGN KEY(EMP\_SELF\_KEY)를 ENABLE하여라.

|  |
| --- |
| SQL> **SELECT constraint\_name,table\_name,status**  2 **FROM user\_constraints;**  CONSTRAINT\_NAME TABLE\_NAME STATUS  ------------------------------ ------------------------------ --------  . . . . . . . . . .  EMP\_PRIMARY\_KEY EMP DISABLED  EMP\_SELF\_KEY EMP DISABLED  EMP\_FOREIGN\_KEY EMP ENABLED  . . . . . . . . . .  32 rows selected.  SQL> **ALTER TABLE emp**  2 **ENABLE CONSTRAINT EMP\_PRIMARY\_KEY;**  Table altered.  SQL> **SELECT constraint\_name,table\_name,status**  2 **FROM user\_constraints;**  CONSTRAINT\_NAME TABLE\_NAME STATUS  ------------------------------ ------------------------------ --------  . . . . . . . . .  EMP\_PRIMARY\_KEY EMP ENABLED  EMP\_SELF\_KEY EMP DISABLED  EMP\_FOREIGN\_KEY EMP ENABLED  . . . . . . . . .  32 rows selected.  SQL> **ALTER TABLE emp**  2 **ENABLE CONSTRAINT emp\_self\_key;**  Table altered.  SQL> **SELECT constraint\_name,table\_name,status**  2 **FROM user\_constraints;**  CONSTRAINT\_NAME TABLE\_NAME STATUS  ------------------------------ ------------------------------ --------  . . . . . . . . .  EMP\_PRIMARY\_KEY EMP ENABLED  EMP\_SELF\_KEY EMP ENABLED  EMP\_FOREIGN\_KEY EMP ENABLED  . . . . . . . . .  32 rows selected. |

**2.7 제약 조건 조회**

테이블 소유자가 소유자 이름을 붙이지 않는 제약 조건은 시스템이 이름을 부여한다. 제약 조건 유형에서 C는 CHECK, P는 PRIMARY KEY, R은 REFERENCE, U는 UNIQUE를 담당하고 NOT NULL은 CHECK가 담당한다.

문제23) EMP 테이블에 있는 각종 제약 조건을 조회하여라.

|  |
| --- |
| SQL> **COL search\_condition FORMAT a30**  SQL> **SELECT constraint\_name,constraint\_type,search\_condition**  2 **FROM user\_constraints**  3 **WHERE table\_name = 'EMP';**  CONSTRAINT\_NAME C SEARCH\_CONDITION  ------------------------------ - ------------------------------  SYS\_C00613 C EMPNO IS NOT NULL  SYS\_C00614 C DEPTNO IS NOT NULL  EMP\_PRIMARY\_KEY P  EMP\_SELF\_KEY R  EMP\_FOREIGN\_KEY R  SQL> **SELECT constraint\_name, column\_name**  2 **FROM user\_cons\_columns**  3 **WHERE table\_name = 'EMP';**  CONSTRAINT\_NAME COLUMN\_NAME  ------------------------------ ------------------------------  EMP\_FOREIGN\_KEY DEPTNO  EMP\_PRIMARY\_KEY EMPNO  EMP\_SELF\_KEY MGR  SYS\_C00613 EMPNO  SYS\_C00614 DEPTNO |

**2.8 객체 이름 변경**

테이블,뷰,시퀀스,동의어를 변경하기 위해 RENAME문장을 실행 합니다. 단 객체 소유자 이어야 합니다.

**2.8.1) Syntax**

|  |
| --- |
| **RENAME** old\_name **TO** new\_name; |

문제24) 이전에 생성한 EMP\_30 테이블의 이름을 EMP\_TEMP30으로 변경하여라.

|  |
| --- |
| SQL> **RENAME emp\_30 TO emp\_temp30;**  Table renamed. |

**2.9 TRUNCATE TABLE문장**

테이블의 OWNER이거나 DELETE TABLE권한을 가진 사용자가 테이블의 모든 행을 삭제(구조는 삭제되지 않는다) 하고 사용하고 있던 기억 공간을 모두 해제할 경우에 사용합니다. 삭제된 행은 복구(ROLLBACK)할 수 없습니다.

**2.9.1 Syntax**

|  |
| --- |
| **TRUNCATE TABLE** table\_name; |

문제25) 이전에 생성한 EMP\_TEMP30 테이블의 모든 자료를 삭제하고 사용하고 있던 기억 공간 을 모두 해제하여라.

|  |
| --- |
| SQL> **TRUNCATE TABLE emp\_temp30;**  Table truncated. |

♣ 참고

DELETE문장은 테이블의 모든 행을 삭제할 수 있지만, 저장 공간을 해제할 수 없습니다.

**2.10 테이블에 주석문 추가**

COMMENT 문장을 사용하여 열,테이블,뷰,스냅샷에 대하여 2000Byte까지 주석을 추가할 수 있다. 주석은 데이터 사전 VIEW(All\_col\_comments, user\_col\_comments, all\_tab\_comments , user\_tab\_comments)를 통하여 볼 수 있다.

**2.10.1) Syntax**

|  |
| --- |
| **COMMENT ON TABLE** table\_name | **COLUMN** table.column  **IS**  ‘text’; |

문제26) EMP 테이블에 “Employee Information”라는 주석을 추가하여라.

|  |
| --- |
| SQL> **COMMENT ON TABLE emp**  2 **IS 'Employee Information';**  Comment created.  SQL> **COL comments FORMAT a30**  SQL> **SELECT \***  2 **FROM user\_tab\_comments;**  TABLE\_NAME TABLE\_TYPE COMMENTS  ------------------------------ ----------- ------------------------------  . . . . . . . . . .  DUMMY TABLE  EMP TABLE Employee Information  EMP\_DEPTNO TABLE  . . . . . . . . . .  15 rows selected. |

**3. 테이블 삭제**

DROP TABLE문장은 Oracle8 테이블의 정의를 삭제 합니다. 테이블을 삭제할 때 데이터베이스는 테이블에 있는 모든 자료와 그와 연관된 모든 INDEX를 DROP하고 사용하고 있던 공간을 돌려줍니다.

☞ Guidelines

1. 테이블의 모든 구조와 데이터가 삭제 됩니다.
2. DDL문장이기 때문에 TRANSACTION이 COMMIT됩니다.
3. 모든 인덱스가 삭제 됩니다.
4. VIEW나 SYNONYM은 남지만 사용시 ERROR가 발생합니다.
5. 테이블의 OWNER나 DROP ANY TABLE권한을 가진 사용자만이 테이블을 삭제할 수 있습니다.

**3.1 Syntax**

|  |
| --- |
| **DROP TABLE** table\_name [**CASCADE CONSTRAINT**] |

♣ 주의

일단 실행된 DROP TABLE문장은 복구(ROLLBACK)할 수 없습니다. ORACLE SERVER은 DROP TABLE문장을 실행할 때 삭제 여부를 질문하지 않습니다.

문제27) EMP\_TEMP30의 테이블을 삭제하여라.

|  |
| --- |
| SQL> **DROP TABLE emp\_30;**  Table dropped. |

문제28) DEPT 테이블을 삭제하여라.

|  |
| --- |
| SQL> **DROP TABLE dept;**  DROP TABLE dept  \*  ERROR at line 1:  ORA-02266: unique/primary keys in table referenced by enabled foreign keys  SQL> **DROP TABLE dept CASCADE CONSTRAINT;**  Table dropped.  SQL> @c:\orawin95\dbs\demobld |

♣ 참고

SQL> @c:\orawin95\dbs\demobld 은 DEMO SCRIPT가 있는 절대 패스를 사용하여 데모 테이블을 다시 생성합니다.

**◈ 연 습 문 제 ◈**

1. EMP 테이블에 있는 모든 CONSTRAINT를 조회하는 SELECT 문을 작성하여라.

2. EMP 테이블에 SAL,COMM을 제외한 모든 COLUMN과 행을 포함하는 EMP\_DEMO 테이블을 생성하는 SQL문을 작성하여라.

3. EMP 테이블과 DEPT 테이블을 이용하여 아래의 내용을 포함하는 테이블(EMP\_DEPT)을 생성하여라.

|  |
| --- |
| EMPNO ENAME JOB DNAME LOC  --------- ---------- --------- -------------- -------------  7839 KING PRESIDENT ACCOUNTING NEW YORK  7698 BLAKE MANAGER SALES CHICAGO  7782 CLARK MANAGER ACCOUNTING NEW YORK  . . . . . . . . . .  14 rows selected. |

4. EMP 테이블과 SALGRADE 테이블을 이용하여 아래의 내용을 포함하는 테이블(EMP\_GRADE)을 생성하여라.

|  |
| --- |
| EMPNO ENAME JOB SAL COMM GRADE  --------- ---------- --------- --------- --------- ---------  7839 KING PRESIDENT 5000 5  7698 BLAKE MANAGER 2850 4  7782 CLARK MANAGER 2450 4  7566 JONES MANAGER 2975 4  . . . . . . . . . .  14 rows selected. |

5. DEPT 테이블의 PRIMARY KEY를 DISABLE하는 SQL문을 작성하여라.

6. 3번에서 생성한 테이블에 EMPNO를 PRIMARY KEY로 설정하는 SQL문을 작성하여라.

7. 4번에서 생성한 테이블에 SAL의 정밀도를 정수 부분을 12자리 소수 이하 4자리를 기억할 수 있도록 변경하는SQL문을 작성하여라.

8. 3번에서 작성한 테이블의 내용과 기억 장소를 모두 해제하는 SQL문을 작성하여라.

9. 3번과 4번에서 생성한 테이블을 모두 삭제하는 SQL문을 작성하여라.