13장 데이터 무결성을 위한 제약 조건

13.1. 무결성 제약 조건의 개념과 종류

13.1.1. 데이터 무결성 제약 조건이란?

- 데이터 무결성 제약 조건(Data Integrity Constraint Rule)이란 테이블에 부적절한 자료가 입력되는 것을 방지하기 위해서 테이블을 생성할 때 각 컬럼에 대해서 정의하는 여러 가지 규칙을 말한다.

13.1.2. 무결한 데이터의 5가지 제약 조건

무결성 제약 조건	역할
NOT NULL	NULL을 허용하지 않는다.
UNIQUE	중복된 값을 허용하지 않는다. 항상 유일한 값을 갖도록 한다.
PRIMARY KEY	NULL을 허용하지 않고 중복된 값을 허용하지 않는다. NOT NULL 조건과 UNIQUE 조건을 결합한 형태이다.
FOREIGN KEY	참조되는 테이블의 칼럼의 값이 존재하면 허용한다.
CHECK	저장 가능한 데이터 값의 범위나 조건을 지정하여 설정한 값만을 허용한다.

13.1.3. 제약 조건 확인하기

- CONSTRAINT_NAME은 제약 조건 명이다.
- CONSTRAINT_TYPE는 제약 조건 유형을 저장하는 컬럼이다.

CONSTRAINT_TYPE	의미
P	PRIMARY KEY
R	FOREIGN KEY
U	UNIQUE
C	CHECK, NOT NULL

```
INSERT INTO DEPT
VALUES(10, 'TEST', 'SEOUL');
-- 오류발생: 무결성 제약 조건에 위배됩니다.

DESC DEPT
SELECT * FROM DEPT;

DESC USER_CONSTRAINTS;
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME='DEPT';
```

13.1.4. 제약 조건 살피기

```
COLUMN CONSTRAINT_TYPE FORMAT A18
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
FROM USER_CONSTRAINTS;
```

```
COLUMN OWNER FORMAT A10
COLUMN TABLE_NAME FORMAT A15
COLUMN COLUMN_NAME FORMAT A15
SELECT *
FROM USER_CONS_COLUMNS;
```

13.2. 필수 입력을 위한 NOT NULL 제약 조건

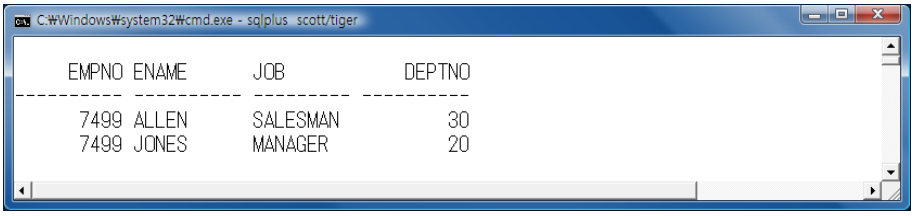
- NOT NULL 제한 조건은 해당 컬럼에 데이터를 추가하거나 수정할 때 NULL 값이 저장되지 않게 제약을 걸어주는 것으로서 사원번호와 사원명과 같이 자료가 꼭 입력되게 하고 싶을 때 사용한다.

```
-- NOT NULL 제약 조건을 설정하지 않고 테이블 생성하기
DROP TABLE EMP01;
CREATE TABLE EMP01(
EMPNO NUMBER(4),
ENAME VARCHAR2(10),
JOB VARCHAR2(9),
DEPTNO NUMBER(2)
);
SELECT * FROM EMP01;
INSERT INTO EMP01
VALUES(NULL, NULL, 'SALESMAN', 30);

-- NOT NULL 제약 조건을 설정하여 테이블 생성하기
DROP TABLE EMP02;
CREATE TABLE EMP02(
EMPNO NUMBER(4) NOT NULL,
ENAME VARCHAR2(10) NOT NULL,
JOB VARCHAR2(9),
DEPTNO NUMBER(2)
);
INSERT INTO EMP02
VALUES(NULL, NULL, 'SALESMAN', 30);  -- 오류발생: cannot insert NULL into ("SCOTT"."EMP02"."EMPNO")

DESC EMP02
INSERT INTO EMP02
VALUES(7499, 'ALLEN', 'SALESMAN', 30);
SELECT * FROM EMP02;
```

13.3. 유일한 값만 허용하는 UNIQUE 제약 조건

- UNIQUE 제약 조건이란 특정 칼럼에 대해 자료가 중복되지 않게 하는 것이다. 즉, 지정된 칼럼에는 유일한 값이 수록되게 하는 것이다.

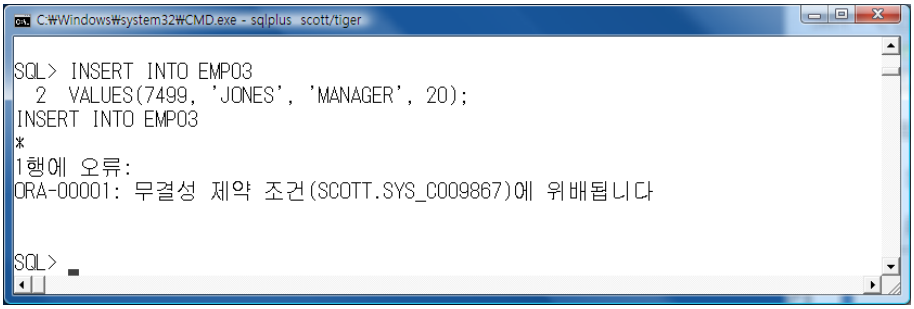
- 새로운 사원이 입사하여 이 사원의 정보를 입력했는데, 이미 존재하는 사원의 번호와 동일한 사원번호로 입력하였더니 성공적으로 추가된다면 어떻게 될까? -> 사원 번호로는 사원을 구분할 수 없게되기 때문에 큰 문제가 발생한다.




```
-- UNIQUE 제약 조건을 설정하여 테이블 생성하기
DROP TABLE EMP03;
CREATE TABLE EMP03(
EMPNO NUMBER(4) UNIQUE,
ENAME VARCHAR2(10) NOT NULL,
JOB VARCHAR2(9),
DEPTNO NUMBER(2)
);

INSERT INTO EMP03
VALUES(7499, 'ALLEN', 'SALESMAN', 30);
SELECT * FROM EMP03;

INSERT INTO EMP03
VALUES(7499, 'JONES', 'MANAGER', 20);
INSERT INTO EMP03
VALUES(NULL, 'JONES', 'MANAGER', 20);  -- 오류발생: unique constraint (SCOTT.SYS_C0011406) violated

INSERT INTO EMP03
VALUES(NULL, 'JONES', 'SALESMAN', 10);  -- UNIQUE는 NULL값은 예외로 간주한다.
SELECT * FROM EMP03;  -- 만약 NULL값도 입력되지 않게 제한하려면 UNIQUE NOT NULL처럼 기술하면 된다.
```

13.4. 컬럼 레벨로 제약 조건명을 명시하여 제약 조건 설정하기

■ 사용자가 제약조건을 주면 오라클은 SYS_ 다음에 숫자를 나열하여 제약 조건 명을 자동 부여한다.



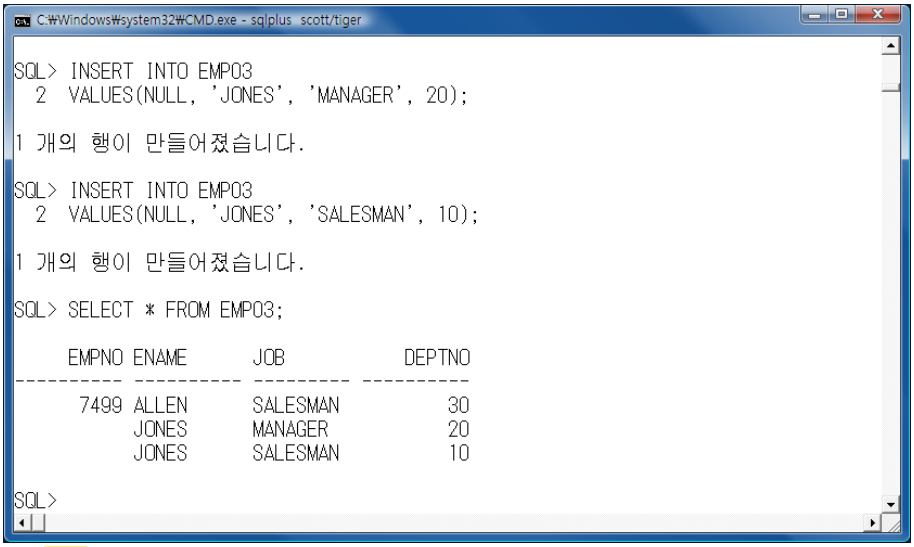
■ 사용자가 직접 제약 조건 명을 설정하려면 CONSTRAINT라는 키워드를 사용한다.

```
-- 형식
column_name data_type CONSTRAINT constraint_name constraint_type

-- 예: 컬럼 레벨로 제약 조건명 명시하기
DROP TABLE EMP04;
CREATE TABLE EMP04(
EMPNO NUMBER(4) CONSTRAINT EMP04_EMPNO_UK UNIQUE,
ENAME VARCHAR2(10));

SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
```

```
FROM USER_CONSTRAINTS
WHERE TABLE_NAME='EMP04';
```

13.5. 데이터 구분을 위한 PRIMARY KEY 제약 조건

- UNIQUE 제약 조건과 NOT NULL 제약 조건을 모두 갖는 것이 기본 키(PRIMARY KEY) 제약 조건이다.
- UNIQUE 제약 조건을 지정한 칼럼은 중복된 데이터를 저장하지는 못하지만 NULL 값을 저장하는 것은 허용한다. 이와 같이 동명이인이 입사를 했다면 이를 구분할 수 있는 유일한 키가 있어야 하는데 사원번호에 NULL 값이 저장되는 바람에 이들을 구분할 수 없게 된다.




```
-- PRIMARY KEY 제약 조건 설정하기
DROP TABLE EMP05;
CREATE TABLE EMP05(
EMPNO NUMBER(4) CONSTRAINT EMP05_EMPNO_PK PRIMARY KEY,
ENAME VARCHAR2(10) CONSTRAINT EMP05_ENAME_NN NOT NULL,
JOB VARCHAR2(9),
DEPTNO NUMBER(2)
);

INSERT INTO EMP05 VALUES(7499,'ALLEN','SALESMAN',30);
SELECT * FROM EMP05;

INSERT INTO EMP05 VALUES(7499,'JONES','MANAGER',20);     -- SQL Error: ORA-00001: unique constraint (SCOTT.EMP05_EMPNO_PK) violated
INSERT INTO EMP05 VALUES(NULL,'JONES','MANAGER',20);   -- SQL Error: ORA-01400: cannot insert NULL into ("SCOTT"."EMP05"."EMPNO")
```

13.6. 참조 무결성을 위한 FOREIGN KEY 제약 조건

- 참조의 무결성은 테이블 사이의 관계에서 발생하는 개념이므로 사원 테이블과 부서 테이블의 관계를 예를 들어 설명하면 ...

사원은 회사 내에 존재하는 부서에 소속되어 있어야 한다.

■ 테이블을 생성하기에 앞서 데이터베이스 모델링 과정에서 업무를 분석한 후 얻어낸 테이블의 관계(개체와 관계)를 다이어그램으로 나타낸 것이다. (ERD, Entity Relationship Diagram)

■ 외래 키(FOREIGN KEY) 제약조건은 사원 테이블의 부서 번호는 반드시 부서 테이블에 존재하는 부서 번호를 참조 가능하도록 하는 것이다. (참조의 무결성)

부서(dept) 테이블의 기본 키인 부서 번호(deptno) 컬럼을 부모 키라고 함

```
C:₩Windows₩system32₩CMD.exe - sqlplus scott/...

SQL> SELECT * FROM DEPT;

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

SQL>
```

사원(emp) 테이블의 부서 번호(deptno) 컬럼은 외래 키로 지정해야만 참조의 무결성이 설정됨

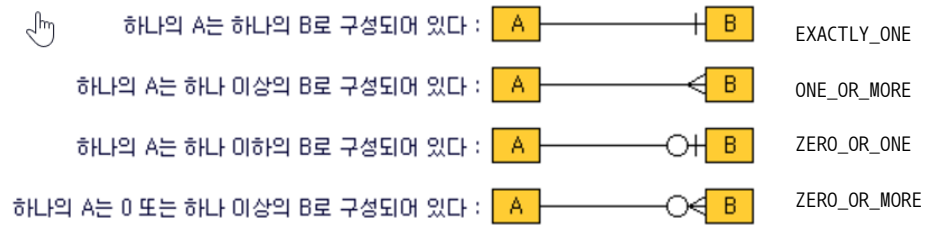
```
C:₩Windows₩system32₩CMD.exe - sqlplus scott/tiger

SQL> SELECT * FROM EMP;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 80/12/17        800                    20
      7499 ALLEN      SALESMAN        7698 81/02/20       1600        300         30
      7521 WARD       SALESMAN        7698 81/02/22       1250        500         30
      7566 JONES      MANAGER         7839 81/04/02       2975                    20
      7654 MARTIN     SALESMAN        7698 81/09/28       1250       1400         30
      7698 BLAKE      MANAGER         7839 81/05/01       2850                    30
      7782 CLARK      MANAGER         7839 81/06/09       2450                    10
      7788 SCOTT      ANALYST         7566 87/04/19       3000                    20
      7839 KING       PRESIDENT            81/11/17       5000                    10
      7844 TURNER     SALESMAN        7698 81/09/08       1500          0         30
      7876 ADAMS      CLERK           7788 87/05/23       1100                    20
      7900 JAMES      CLERK           7698 81/12/03        950                    30
      7902 FORD       ANALYST         7566 81/12/03       3000                    20
      7934 MILLER     CLERK           7782 82/01/23       1300                    10

14 개의 행이 선택되었습니다.

SQL>
```

```
-- 다음은 EMP 테이블과 DEPT 테이블의 제약 조건을 확인한다.
SELECT TABLE_NAME, CONSTRAINT_TYPE,
CONSTRAINT_NAME, R_CONSTRAINT_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN ('DEPT', 'EMP');

-- 왜래 키 제약 조건 설정하기
```

```
DROP TABLE EMP06;
CREATE TABLE EMP06(
EMPNO NUMBER(4) CONSTRAINT EMP06_EMPNO_PK PRIMARY KEY ,
ENAME VARCHAR2(10) CONSTRAINT EMP06_ENAME_NN NOT NULL,
JOB VARCHAR2(9),
DEPTNO NUMBER(2) CONSTRAINT EMP06_DEPTNO_FK REFERENCES DEPT(DEPTNO)
);

INSERT INTO EMP06 VALUES(7499,'ALLEN','SALESMAN',30);
SELECT * FROM EMP06;
INSERT INTO EMP06 VALUES(7499,'JONES','MANAGER',50);     -- SQL Error: ORA-00001: unique constraint
(SCOTT.EMP06_EMPNO_PK) violated

SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME, R_CONSTRAINT_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME='EMP06';
```

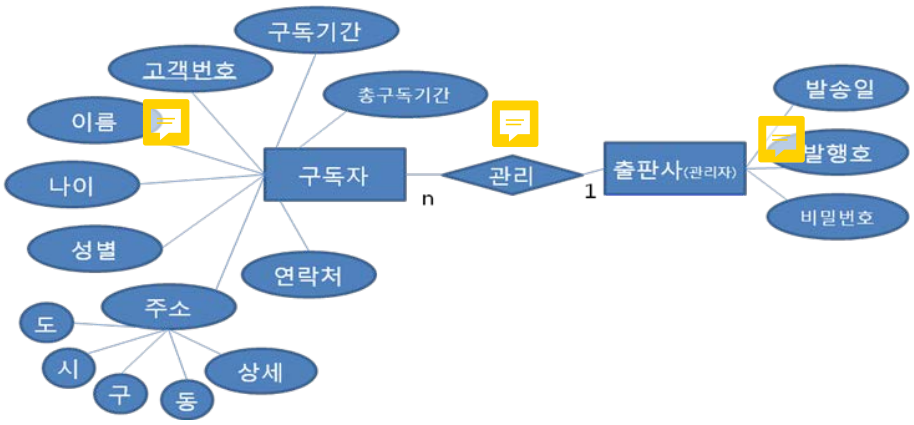
[참조] ERD(Entity Relationship Diagram)

1. ERD 표기 방법

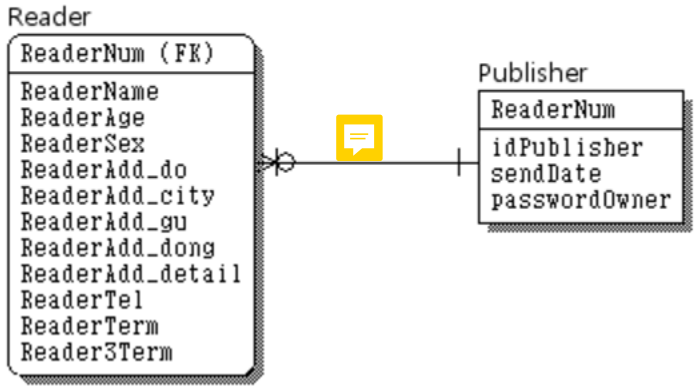


2. 논리적 설계

(1) 데이터베이스 모델링



(2) ERD 작성 (새발표기법)



3. 물리적 설계 (DDL 작성)

```
CREATE TABLE Publisher(
  sendDate DATE,
  passwordOwner NUMBER,
  idPublisher NUMBER,
  ReaderNum NUMBER NOT NULL
);

DROP INDEX XPKPublisher;
CREATE UNIQUE INDEX XPKPublisher ON Publisher (
  ReaderNum ASC
);

ALTER TABLE Publisher
ADD PRIMARY KEY(ReaderNum);
...
```

13.7. CHECK와 DEFAULT의 제약 조건

- CHECK 제약 조건은 입력되는 값을 체크하여 설정된 값 이외의 값이 들어오면 오류 메시지와 함께 명령이 수행되지 못하게 하는 것이다.
- 디폴트는 아무런 값을 입력 하지 않았을 때 디폴트제약의 값이 입력이 된다.

```
-- CHECK 제약 조건 설정하기
DROP TABLE EMP07;
CREATE TABLE EMP07(
EMPNO NUMBER(4) CONSTRAINT EMP07_EMPNO_PK PRIMARY KEY,
ENAME VARCHAR2(10) CONSTRAINT EMP07_ENAME_NN NOT NULL,
SAL NUMBER(7,2) CONSTRAINT EMP07_SAL_CK CHECK(SAL BETWEEN 500 AND 5000),
GENDER VARCHAR2(1) CONSTRAINT EMP07_GENDER_CK CHECK(GENDER IN('M','F'))
);

INSERT INTO EMP07 VALUES(7499,'ALLEN',500,'M');
INSERT INTO EMP07 VALUES(7499,'ALLEN',7000,'A');     -- SQL Error: ORA-02290: check constraint
(SCOTT.EMP07_GENDER_CK) violated

SELECT TABLE_NAME, CONSTRAINT_TYPE, CONSTRAINT_NAME, SEARCH_CONDITION
FROM USER_CONSTRAINTS
WHERE TABLE_NAME='EMP07';

-- DEFAULT 제약 조건 설정하기
```

```
DROP TABLE DEPT01;
CREATE TABLE DEPT01(
DEPTNO NUMBER(2) PRIMARY KEY,
DNAME VARCHAR2(14),
LOC VARCHAR2(13) DEFAULT 'SEOUL'
);

INSERT INTO DEPT01(DEPTNO,DNAME) VALUES(10,'ACCOUNTING');
SELECT * FROM DEPT01;
```

13.8. 테이블 레벨 방식으로 제약 조건 지정하기

■ 컬럼 레벨 제약 조건

- CREATE TABLE로 테이블을 생성하면서 컬럼을 정의하게 되는데 하나의 컬럼 정의가 다 마무리되기 전에 컬럼 명 다음에 타입을 지정하고 그 뒤에 연이어서 제약 조건을 지정하는 방식이다.

■ 테이블 레벨의 제약 조건

- 칼럼을 모두 정의하고 나서 테이블 정의를 마무리 짓기 전에 따로 생성된 칼럼들에 대한 제약 조건을 한꺼번에 지정하는 것이다.

■ **복합키로 기본키를 지정할 경우**

- 지금까지는 한 개의 컬럼으로 기본키를 지정했다. 하지만, 경우에 따라서는 2개 이상의 컬럼이 하나의 기본키를 구성하는 경우가 있는데 이를 복합키라고 한다. 복합키 형태로 제약조건을 지정할 경우에는 컬럼 레벨 형식으로는 불가능하고 반드시 테이블 레벨 방식을 사용해야 한다.

■ ALTER TABLE로 제약 조건을 추가할 때

- 테이블의 정의가 완료되어서 이미 테이블의 구조가 결정된 후에 나중에 테이블에 제약 조건을 추가하고 할 때에는 테이블 레벨 방식으로 제약 조건을 지정해야 한다.

[실습] 컬럼레벨 제약조건과 테이블레벨 제약조건 설정하기

■ 컬럼 레벨과 테이블 레벨로 제약 조건을 지정하는 방법의 차이점을 살펴본다.

```
-- 컬럼 레벨로 제약 조건을 지정하는 방법
DROP TABLE EMP01;
CREATE TABLE EMP01(
EMPNO NUMBER(4) PRIMARY KEY,
ENAME VARCHAR2(10) NOT NULL,
JOB VARCHAR2(9) UNIQUE,
DEPTNO NUMBER(4) REFERENCES DEPT(DEPTNO)
);

-- 테이블 레벨로 제약 조건을 지정하는 방법
DROP TABLE EMP02;
CREATE TABLE EMP02(
EMPNO NUMBER(4),
ENAME VARCHAR2(10) NOT NULL,
JOB VARCHAR2(9),
DEPTNO NUMBER(4),
PRIMARY KEY(EMPNO),
UNIQUE(JOB),
FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO)
```

```
);
```

13.9. 제약 조건 변경하기

13.9.1. 제약 조건 추가하기

- 테이블 구조를 결정하는 DDL을 학습하면서 테이블이 이미 생성된 이후에 테이블의 구조를 변경하기 위한 명령어로 ALTER TABLE을 사용한다는 것을 이미 학습하였다.
- 제약조건 역시 이미 테이블을 생성하면서 지정해주는 것이었기에 테이블 생성이 끝난 후에 제약 조건을 추가하기 위해서는 ALTER TABLE로 추가해 주어야 한다.

```
-- 형식
ALTER TABLE table_name
ADD [CONSTRAINT constraint_name]
constraint_type (column_name);
```

13.9.2. MODIFY로 NOT NULL 제약 조건 추가하기

- 이미 존재하는 테이블에 무결성 제약 조건을 추가로 생성하기 위해서 ALTER TABLE . . . ADD . . . 명령문을 사용하였다.
- 하지만 NOT NULL 제약 조건은 ADD 대신 MODIFY 명령문을 사용하므로 사용에 주의해야 한다. 이는 'NULL을 허용하는 상태'에서 'NULL을 허용하지 않는 상태'로 변경하겠다는 의미로 이해하기 바란다.

13.9.3. 제약 조건 제거하기

- 제약 조건을 제거하기 위해서 DROP CONSTRAINT 다음에 제거하고자 하는 제약 조건 명을 명시해야 한다.

```
-- 형식
ALTER TABLE table_name
DROP [CONSTRAINT constraint_name];
```

13.10. 제약 조건의 비활성화와 CASCADE

13.10.1. 제약 조건에 의해 작업이 어려운 경우

- 제약 조건이 설정되면 항상 그 규칙에 따라 데이터 무결성이 보장된다.
- 특별한 업무를 수행하는 과정에서 이러한 제약 조건 때문에 작업이 진행되지 못하는 경우가 생긴다.

- 그렇다고 제약 조건을 삭제해 버리면 데이터 무결성을 보장받지 못하게 된다.
- 그렇기 때문에 오라클에서는 제약 조건을 비활성화시킴으로서 제약 조건을 삭제하지 않고도 제약 조건 사용을 잠시 보류할 수 있는 방법을 제공해준다.
- 이렇게 비활성화 된 제약 조건은 원하는 작업을 한 후에는 다시 활성화 상태로 만들어 주어야 한다.

13.10.2. 제약 조건의 비활성화

- 테이블에서 제약 조건을 삭제하지 않고 일시적으로 적용시키지 않도록 하는 방법으로 제약 조건을 비활성화하는 방법이 있다.

```
-- 형식
ALTER TABLE table_name
DISABLE [CONSTRAINT constraint_name];
```

13.10.3. 제약 조건의 활성화

- 제약 조건을 비활성화 해 보았으므로 이번에는 제약 조건을 활성화 해보도록 한다.

```
-- 형식
ALTER TABLE table_name
ENABLE [CONSTRAINT constraint_name];
```

13.10.4. CASCADE 옵션

- CASCADE 옵션은 부모 테이블과 자식 테이블 간의 참조 설정이 되어 있을 때 부모 테이블의 제약 조건을 비활성화하면 이를 참조하고 있는 자식 테이블의 제약 조건까지 같이 비활성화 시켜 주는 옵션이다.
- 또한 제약 조건의 비활성화뿐만 아니라 제약 조건이 삭제에도 활용되며, 역시 같은 이치로 부모 테이블의 제약 조건을 삭제하면 이를 참조하고 있는 자식 테이블의 제약 조건도 같이 삭제된다.

[실습] CASCADE 옵션으로 제약 조건 연속적으로 비활성화 / 제거하기

```
DROP TABLE DEPT01 CASCADE CONSTRAINTS;
CREATE TABLE DEPT01
AS
SELECT * FROM DEPT;
ALTER TABLE DEPT01
ADD CONSTRAINT DEPT01_DEPTNO_PK PRIMARY KEY (DEPTNO);
DESC DEPT01;

DROP TABLE EMP01 CASCADE CONSTRAINTS;
CREATE TABLE EMP01
```

```
AS
SELECT * FROM EMP WHERE 1=0;
ALTER TABLE EMP01
ADD CONSTRAINT EMP01_DEPTNO_FK FOREIGN KEY(DEPTNO) REFERENCES DEPT01(DEPTNO);

SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE,
TABLE_NAME, R_CONSTRAINT_NAME, STATUS
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('DEPT01', 'EMP01');

SELECT * FROM DEPT01;
SELECT * FROM EMP01;

INSERT INTO EMP01 (DEPTNO) VALUES (10);
SELECT * FROM EMP01;
INSERT INTO EMP01 (DEPTNO) VALUES (50);

ALTER TABLE DEPT01
DROP PRIMARY KEY CASCADE;
```

[과제] 과제-13-01.TXT

1. 다음과 같은 구조의 테이블을 생성해 보자. (단, 테이블 레벨방식 사용)
- 테이블 : ORDERS
- 컬럼 : ORDER_ID NUMBER(12,0)
ORDER_DATE DATE
ORDER_MODE VARCHAR2(8 BYTE)
CUSTOMER_ID NUMBER(6,0)
ORDER_STATUS NUMBER(2,0)
ORDER_TOTAL NUMBER(8,2)
SALES_REP_ID NUMBER(6,0)
PROMOTION_ID NUMBER(6,0)
- 제약사항 : 기본키는 ORDER_ID
ORDER_MODE에는 'direct', 'online'만 입력가능
ORDER_TOTAL의 디폴트 값은 0

<정답>

2. 다음과 같은 구조의 테이블을 생성해 보자. (단, 테이블 레벨방식 사용)
- 테이블 : ORDER_ITEMS
- 컬럼 : ORDER_ID NUMBER(12,0)
LINE_ITEM_ID NUMBER(3,0)
PRODUCT_ID NUMBER(3,0)
UNIT_PRICE NUMBER(8,2)
QUANTITY NUMBER(8,0)
- 제약사항 : 기본키는 ORDER_ID와 LINE_ITEM_ID
UNIT_PRICE, QUANTITY 의 디폴트 값은 0

<정답>