ORACLE

12. 제약조건

목차

1. 무결성 제약 조건

❖ 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.

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 컬럼의 값이 존재하면 허용한다.
CHECK	저장 가능한 데이터 값의 범위나 조건을 지정하여 설정한 값만을 허용한다.

2. 제약 조건 확인(1)

- 아래의 그림은 DEPT 테이블에 INSERT 작업 중 무결성 제약 조건을 위배했을 때 나타나는 에러 메시지이다.

```
INSERT INTO DEPT VALUES(10, 'TEST', 'TEST')
오류 보고 -
ORA-00001: unique constraint (JOONZIS.PK_DEPT) violated
```

```
INSERT INTO DEPT VALUES(NULL, 'TEST', 'TEST')
오류 보고 -
ORA-01400: cannot insert NULL into ("JOONZIS"."DEPT"."DEPTNO")
```

- DESC 명령어로는 NOT NULL 제약 조건만 확인할 수 있고 DEPTNO 컬럼에 기본 키 제약 조건이 지정된 것을 알 수 없다.

```
이름     널?       유형
------ -------- ------------
DEPTNO NOT NULL NUMBER(2)
DNAME           VARCHAR2(14)
LOC             VARCHAR2(13)
```

2. 제약 조건 확인(2)

- 아래의 그림은 DEPT 테이블에 INSERT 작업 중 무결성 제약 조건을 위배했을 때 나타나는 에러 메시지이다.

DEPT(부서 테이블)

DEPTNO	DNAME	LOC
10		
20		
30		
40		

EMP(사원 테이블)

DEPTNO	...	EMPNO
10		
20		
30		
40		

기본키

기본키(primary key)
=Unique + Not Null

참조

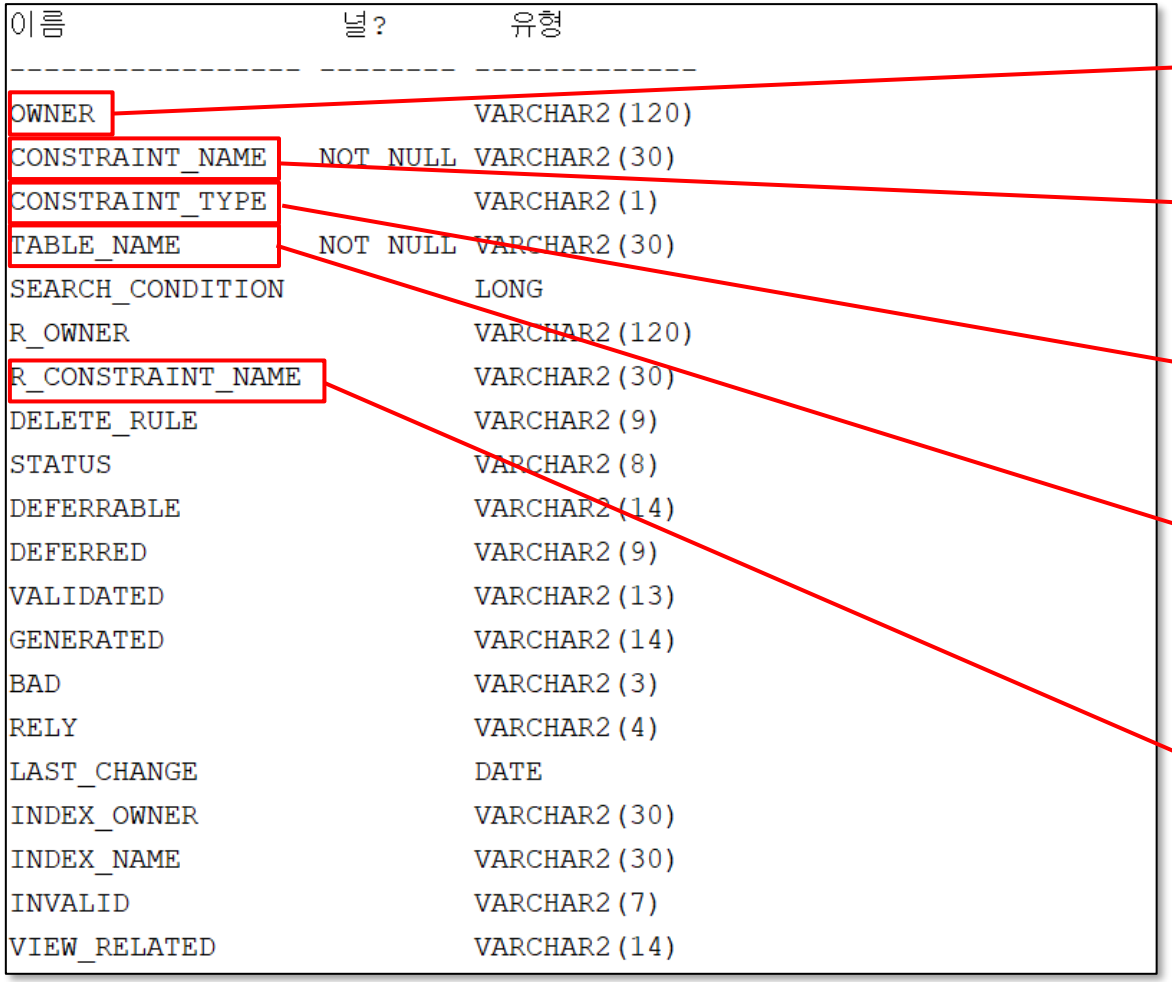
외래키(foreign key)

2. 제약 조건 확인(3)

- 오라클은 USER_CONSTRAINTS 데이터 딕셔너리 뷰로 제약 조건에 관한 정보를 알려준다.
- USER_CONSTRAINTS 데이터 딕셔너리 뷰를 조회하면 내가 만든(USER) 제약 조건(CONSTRAINTS)의 정보를 조회할 수 있다.

예 **DESC USER_CONSTRAINTS;**

2. 제약 조건 확인(4)



이름	널?	유형
OWNER		VARCHAR2(120)
CONSTRAINT_NAME	NOT NULL	VARCHAR2(30)
CONSTRAINT_TYPE		VARCHAR2(1)
TABLE_NAME	NOT NULL	VARCHAR2(30)
SEARCH_CONDITION		LONG
R_OWNER		VARCHAR2(120)
R_CONSTRAINT_NAME		VARCHAR2(30)
DELETE_RULE		VARCHAR2(9)
STATUS		VARCHAR2(8)
DEFERRABLE		VARCHAR2(14)
DEFERRED		VARCHAR2(9)
VALIDATED		VARCHAR2(13)
GENERATED		VARCHAR2(14)
BAD		VARCHAR2(3)
RELY		VARCHAR2(4)
LAST_CHANGE		DATE
INDEX_OWNER		VARCHAR2(30)
INDEX_NAME		VARCHAR2(30)
INVALID		VARCHAR2(7)
VIEW_RELATED		VARCHAR2(14)

제약 조건을 소유한 사용자명

제약 조건 명

제약 조건 유형

제약 조건이 속한 테이블 명

외래키인 경우 어떤 기본키를 참조했는지에 대한 정보를 갖는다.

2. 제약 조건 확인(5)

- USER_CONSTRAINTS 데이터 딕셔너리는 제약 조건의 정보를 위해서 많은 컬럼으로 구성되어 있지만, 중요한 컬럼 몇 개만 살펴보도록 한다.
- OWNER는 제약 조건을 소유한 사용자명을 저장하는 컬럼이다.
- CONSTRAINT_NAME은 제약 조건 명을
- CONSTRAINT_TYPE은 제약 조건 유형을 저장하는 컬럼이다.
  - CONSTRAINT_TYPE은 P, R, U, C 4가지 값 중에 하나를 갖는다.

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

2. 제약 조건 확인(6)

- 제약 조건이 5개인데 제약 조건 유형이 4가지인 이유는 NOT NULL과 CHECK 제약 조건을 모두 C로 표현하기 때문이다.
- NOT NULL 제약 조건은 컬럼에 NOT NULL 조건을 체크할지 말지를 결정하기 때문에 CHECK를 나타내는 C로 표현한 것으로 이해하면 된다.
- 제약 조건 유형은 제약 조건의 이니셜로 표현되지만 FOREIGN KEY만은 R로 표현하는 것을 볼 수 있다.
- 이는 PRIMARY KEY를 참조(REFERENCE) 무결성의 이니셜인 R을 FOREIGN KEY의 제약 조건 유형으로 지정한 것으로 이해하면 된다.

2. 제약 조건 확인(7)

- ex) USER_CONSTRAINTS 테이블의 내용을 살펴보자.

예

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE,
TABLE_NAME FROM USER_CONSTRAINTS;
```

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SYS C007689	C	DEPT
PK DEPT	P	DEPT
PK EMP	P	EMP
FK DEPT	R	EMP
SYS C007697	C	DEPT01

- 어떤 테이블에 어떤 제약 조건이 설정되어 있는지 종류와 제약 조건 이름을 알 수 있다.

2. 제약 조건 확인(8)

- 하지만, USER_CONSTRAINTS 데이터 딕셔너리 뷰에는 어떤 컬럼에 제약 조건이 정의되었는지 컬럼 명이 보이질 않는다.
- 그리하여 어떤 컬럼에 어떤 제약 조건이 지정되었는지 알려주는 데이터 딕셔너리 뷰를 이요한다.
- USER_CONS_COLUMNS 데이터 딕셔너리 뷰는 제약 조건이 지정된 컬럼 명도 알려줍니다.

예 **SELECT * FROM USER_CONS_COLUMNS;**

OWNER	CONSTRAINT_NAME	TABLE_NAME	COLUMN_NAME	POSITION
JOONZIS	SYS_C007697	DEPT01	DEPTNO	(null)
JOONZIS	FK_DEPT	EMP	DEPTNO	1
JOONZIS	PK_EMP	EMP	EMPNO	1
JOONZIS	SYS_C007689	DEPT	DEPTNO	(null)
JOONZIS	PK_DEPT	DEPT	DEPTNO	1

3. NOT NULL(1)

- 새로운 사원이 입사하여 사원의 정보를 입력하는데 사원 번호와 사원 명이 불분명하여 데이터가 저장되지 않았다면 누구의 직급인지, 누구의 부서번호인지를 모르게 되므로 자료로서의 의미를 갖기 어렵다.

  * EMPNO, ENAME, JOB, DEPTNO의 컬럼 구조만을 갖는 EMP03 테이블 생성 후 진행

예 **INSERT INTO EMP03 VALUES(NULL, NULL, '영업부', 30);**

EMPNO	ENAME	JOB	DEPTNO
(null)	(null)	영업부	30

3. NOT NULL(2)

- 따라서 사원의 정보를 입력할 때 반드시 입력해야하는 선택이 아닌 필수 입력을 요구하는 컬럼이 있다면 위와 같이 NULL 값이 저장되지 못하도록 제약 조건을 설정해야 한다.
- 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.

3. NOT NULL(3)

- ex) EMP03 테이블에 사원 번호와 사원명에 데이터를 저장하지 않더라도 해당 로우가 테이블에 추가된다.
- 테이블을 생성하면서 아무런 제약 조건도 주지 않았기 때문에 DESC 명령어로도 NOT NULL 제약조건이 설정되지 않음을 확인할 수 있다.

예 **DESC EMP03;**

```
이름     널? 유형
------ -- ------------
EMPNO      NUMBER(4)
ENAME      VARCHAR2(10)
JOB        VARCHAR2(9)
DEPTNO     NUMBER(2)
```

3. NOT NULL(4)

- ex) EMP03과 유사한 구조의 테이블 EMP04를 생성하되 EMPNO와 ENAME 컬럼에 NOT NULL 제약 조건을 설정한 후 데이터를 추가한다.

예

```
CREATE TABLE EMP04(
EMPNO NUMBER(4) NOT NULL,
ENAME VARCHAR2(10) NOT NULL,
JOB VARCHAR2(9),
DEPTNO NUMBER(2)
);
```

예

```
INSERT INTO EMP04
VALUES(NULL, NULL, '영업부', 30);
```

- NOT NULL 조건을 지정하였기에 오류가 발생한다.

```
명령의 1 행에서 시작하는 중 오류 발생 -
INSERT INTO EMP04
VALUES(NULL, NULL, '영업부', 30)
오류 보고 -
ORA-01400: cannot insert NULL into ("JOONZIS"."EMP04"."EMPNO")
```

3. NOT NULL(4)

❖ DESC 명령어로 NOT NULL 제약 조건이 설정되어 있음을 확인할 수 있다.

예 **DESC EMP04;**

```
이름     널?       유형
------ -------- ------------
EMPNO  NOT NULL NUMBER(4)
ENAME  NOT NULL VARCHAR2(10)
JOB             VARCHAR2(9)
DEPTNO          NUMBER(2)
```

4. UNIQUE(1)

- 특정 컬럼에 대해 자료가 중복되지 않게 하는 제약 조건.
- 새로운 사원이 입사하여 이 사원의 정보를 입력하는데, 이미 존재하는 사원의 번호와 동일한 사원 번호가 추가된다면 문제가 발생할 수 있다.

EMPNO	ENAME	JOB	DEPTNO
7499	김씨	영업부	30
7499	박씨	인사부	40

4. UNIQUE(2)

- 기존과 동일한 구조를 갖는 EMP05테이블을 만들되 사원 번호를 유일키로 지정한다.

예

```
CREATE TABLE EMP05(
EMPNO NUMBER(4) UNIQUE,
ENAME VARCHAR2(10) NOT NULL,
JOB VARCHAR2(9),
DEPTNO NUMBER(2)
);
```

4. UNIQUE(3)

- 위에서 생성한 EMP05 테이블에 데이터를 추가한다.

예

```
INSERT INTO EMP05
VALUES(7499, '김씨', '영업부', 30);
```

- 앞에서 사원번호를 7499번의 자료를 입력하였는데 다시 동일한 사원번호를 입력하면 어떻게 되는지 확인한다.

예

```
INSERT INTO EMP05
VALUES(7499, '박씨', '인사부', 40);
```

```
명령의 1 행에서 시작하는 중 오류 발생 -
INSERT INTO EMP05
VALUES(7499, '박씨', '인사부', 40)
오류 보고 -
ORA-00001: unique constraint (JOONZIS.SYS_C007718) violated
```

5. PRIMARY KEY(1)

- UNIQUE 제약 조건과 NOT NULL 제약 조건을 모두 갖고 있는 제약 조건이다.
- 기존과 동일한 구조를 갖는 EMP06테이블을 만들되 사원 번호를 기본키로 지정한다.

예

```
CREATE TABLE EMP06(
EMPNO NUMBER(4) PRIMARY KEY,
ENAME VARCHAR2(10) NOT NULL,
JOB VARCHAR2(9),
DEPTNO NUMBER(2)
);
```

5. PRIMARY KEY(2)

- 위에서 생성한 EMP06 테이블에 데이터를 추가한다.

예

```
INSERT INTO EMP06
VALUES(7499, '김씨', '영업부', 30);
```

- 다음은 기본키로 지정된 사원 번호에 동일한 값을 저장해본다.

예

```
INSERT INTO EMP06
VALUES(7499, '박씨', '인사부', 40);
```

- UNIQUE 제약 조건에 의해 무결성 제약이 위배된다.

```
명령의 1 행에서 시작하는 중 오류 발생 -
INSERT INTO EMP06
VALUES(7499, '박씨', '인사부', 40)
오류 보고 -
ORA-00001: unique constraint (JOONZIS.SYS_C007720) violated
```

5. PRIMARY KEY(3)

- 이번에는 기본키로 지정된 사원번호에 NULL값을 입력해본다.

예

```
INSERT INTO EMP06
VALUES(NULL, '이씨', '영업부', 20);
```

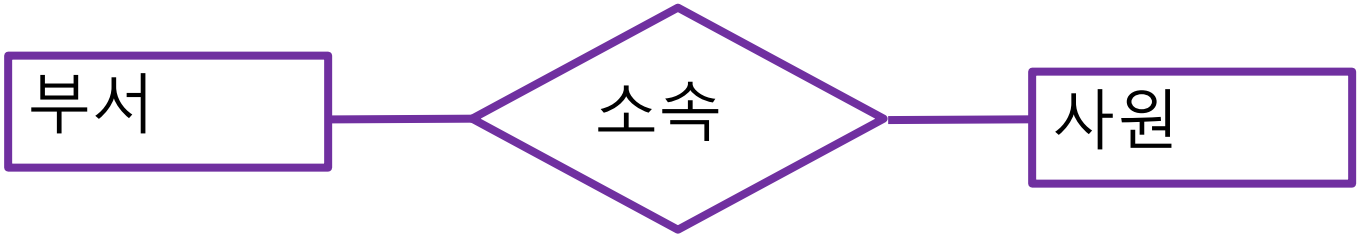
```
명령의 1 행에서 시작하는 중 오류 발생 -
INSERT INTO EMP06
VALUES(NULL, '이씨', '영업부', 20)
오류 보고 -
ORA-01400: cannot insert NULL into ("JOONZIS"."EMP06"."EMPNO")
```

6. FOREIGN KEY(1)

- 조인이나 서브 쿼리를 학습할 때 살펴보았듯이 사원 테이블에 없는 상세 정보는 부서 테이블에서 찾아오는데, 사원 테이블에 저장된 부서 번호가 테이블에 없다면 참조할 때 무결해야 한다는 조건(참조 무결성)에 위배되는 것이 된다.
- 그러므로 사원 테이블에 부서 번호를 입력할 때 부서 테이블에 존재하는 부서 번호만 입력하도록 하면 참조 무결성이 지켜진다.
- 이를 위해서 사원 테이블의 부서 번호 컬럼에 외래키 제약 조건을 명시해야 한다.
- 외래키 제약 조건은 사원 테이블의 부서 번호는 반드시 부서 테이블에 존재하는 부서 번호만 입력하도록 함으로서 사원 테이블이 부서 테이블을 부서 번호로 참조 가능하도록 하는 것을 의미한다.

6. FOREIGN KEY(2)

- 다음은 ERD(Entity Relation Diagram)로서 테이블을 생성하기에 앞서 데이터베이스 모델링 과정에서 업무를 분석한 후 얻어낸 개체와 관계를 다이어그램으로 나타낸 것이다.



- ERD를 보고 데이터베이스를 구현할 때에는 부서나 사원과 같은 개체는 테이블로 정의하고 소속이란 관계는 참조의 무결성을 위한 특정 컬럼에 외래 키 제약 조건으로 정의한다.

6. FOREIGN KEY(3)

- 참조의 무결성은 두 테이블 사이(사원 테이블, 부서 테이블)의 주종 관계에 의해서 결정되는데 주체가 되는 테이블은 부모 테이블이 되고 종속이 되는 테이블은 자식 테이블이 된다.

> 사원은 회사 내에 존재하는 부서에 소속되어 있어야 한다.

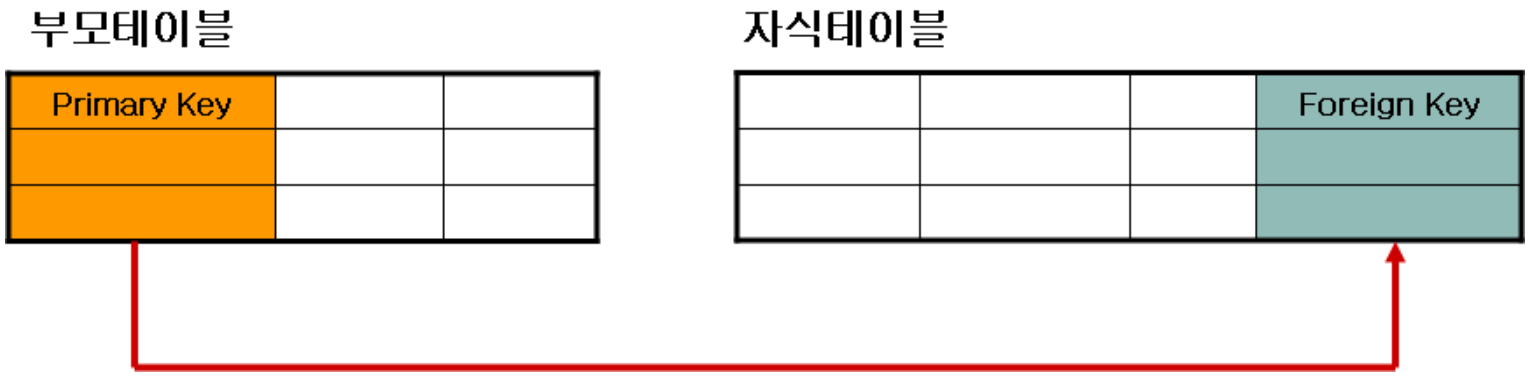
- 사원과 부서의 소속 관계가 위와 같이 표현된다면 부서가 주체(부모 테이블)이고 사원이 종속(자식 테이블)이 된다.

6. FOREIGN KEY(4)

- 주체 관계가 애매모호한 경우에는 어느 테이블의 데이터가 먼저 정의되어야 하는가를 기준으로 부모 테이블과 자식 테이블을 구분할 수 있다.
- 먼저 정의되어야 하는 테이블이 부모 테이블이고 나중에 정의되어야 하는 테이블이 자식 테이블이 된다.
- 회사를 설립하고 어떤 부서를 구성하여 운영할지 정한 후에, 그 부서에서 일할 사원을 뽑아야 소속이란 관계가 성립되므로 부서가 부모 테이블이 되고 사원이 자식 테이블이 된다.
- 외래 키(FOREIGN KEY) 제약 조건은 자식 테이블인 사원 테이블(EMP)의 부서 번호(DEPTNO) 컬럼에 부모 테이블인 부서 테이블(DEPT)의 부서 번호(DEPTNO)를 부모 키로 설정하는 것이다.

6. FOREIGN KEY(5)

- 이때 주의할 점은 부모 키가 되기 위한 컬럼은 반드시 부모 테이블의 기본 키(PRIMARY KEY)나 유일키(UNIQUE)로 설정되어 있어야 한다는 점이다.



- 우리가 지금까지 학습할 때 사용한 오라클이 제공해주는 EMP 테이블과 DEPT 테이블을 보면 부모 테이블인 부서 테이블(DEPT)의 부서 번호(DEPTNO)는 기본 키(PRIMARY KEY)로 설정되어 있고, 이를 참조할 수 있도록 하기 위해서 자식 테이블인 사원 테이블(EMP)에서 부서 번호(DEPTNO)에 외래 키(FOREIGN KEY) 제약조건을 설정해 놓은 상태이다.

6. FOREIGN KEY(6)

- 자식 테이블(EMP)에 참조의 무결성을 위해 특정 컬럼에 외래 키를 설정하였다면 새로운 데이터를 추가할 때마다 부모 테이블에 부모 키로 설정된 컬럼을 살핀다.
- 부모 키로 설정된 컬럼에 존재하는 값만 추가하고 존재하지 않는 값이라면 추가하지 않는다.
- 이렇게 함으로서 자식 테이블이 부모 테이블을 참조하는데 아무런 문제가 없도록 한다.

6. FOREIGN KEY(7)

- 외래 키 제약 조건이 지정된 사원 테이블에 부서 테이블에 존재하지 않은 50번 부서 번호를 저장해 보도록 한다.

예 **SELECT * FROM DEPT;**

DEPTNO	DNAME	LOC
10	경리부	서울
20	인사부	인천
30	영업부	용인
40	전산부	수원

예 **INSERT INTO EMP(EMPNO, ENAME, DEPTNO)**
VALUES(8000, '김씨', 50);

```
명령의 1 행에서 시작하는 중 오류 발생 -
INSERT INTO EMP(EMPNO, ENAME, DEPTNO)
VALUES(8000, '김씨', 50)
오류 보고 -
ORA-02291: integrity constraint (JOONZIS.FK_DEPT) violated - parent key not found
```

6. FOREIGN KEY(8)

- 다음은 오라클에서 제공해주는 EMP 테이블과 DEPT 테이블의 제약 조건을 살펴보도록 한다.

예

```
SELECT TABLE_NAME, CONSTRAINT_TYPE,
CONSTRAINT_NAME, R_CONSTRAINT_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN ('DEPT', 'EMP');
```

TABLE_NAME	CONSTRAINT_TYPE	CONSTRAINT_NAME	R_CONSTRAINT_NAME
DEPT	C	SYS_C007689	(null)
DEPT	P	PK_DEPT	(null)
EMP	P	PK_EMP	(null)
EMP	R	FK_DEPT	PK_DEPT

6. FOREIGN KEY(9)

- R_CONSTRAINT_NAME 컬럼은 FOREIGN KEY인 경우 어떤 PRIMARY KEY를 참조했는지에 대한 정보를 갖는다.
- EMP 테이블의 제약조건 FK_DEPTNO의 R_CONSTRAINT_NAME 컬럼 값이 PK_DEPT으로 설정되어 있다.
- 이는 EMP 테이블의 FK_DEPTNO는 외래 키 제약 조건으로 PK_DEPT 제약 조건을 참조하고 있다는 내용이다. PK_DEPT 제약조건은 DEPT 테이블의 기본 키 제약 조건이므로 EMP 테이블은 DEPT 테이블을 참조하고 있는 셈이 된다.

6. FOREIGN KEY(10)

- ex) 지금까지 실습에 사용했던 사원 테이블과 유사한 구조의 사원 번호, 사원명, 직급, 부서 번호 4개의 컬럼으로 구성된 테이블을 생성하되 기본 키 제약 조건은 물론 외래키 제약 조건도 설정해본다.

예

```
CREATE TABLE EMP07(
EMPNO NUMBER(4) PRIMARY KEY ,
ENAME VARCHAR2(10) NOT NULL,
JOB VARCHAR2(9),
DEPTNO NUMBER(2) REFERENCES DEPT(DEPTNO)
);
```

6. FOREIGN KEY(11)

- 현재 EMP07 테이블에 부서 테이블에 존재하지 않는 부서 번호를 갖는 사원 정보를 추가해본다.

예

```
INSERT INTO EMP07
VALUES(1200, '김씨', '사원', 50);
```

```
명령의 1 행에서 시작하는 중 오류 발생 -
INSERT INTO EMP07
VALUES(1200, '김씨', '사원', 50)
오류 보고 -
ORA-02291: integrity constraint (JOONZIS.EMP07_DEPTNO_FK) violated - parent key not found
```

7. CHECK(1)

- CHECK 제약 조건은 입력되는 값을 체크하여 설정된 값 이외의 값이 들어오면 오류 메시지와 함께 명령이 수행되지 못하게 하는 것이다.
- 조건으로 데이터의 값의 범위나 특정 패턴의 숫자나 문자 값을 설정할 수 있다.
- 예를 들어 사원 테이블에 급여 컬럼을 생성하되 급여 컬럼 값은 200에서 1000사이의 값만 저장할 수 있도록 하거나 성별을 저장하는 컬럼으로 GENDER 를 정의하고, 이 컬럼에는 남자는 M, 여자는 F 둘 중의 하나만 저장할 수 있도록 제약을 주려면 CHECK 제약조건을 지정해야 한다.

7. CHECK(2)

- ex) 사원번호, 사원명, 급여, 성별, 부서 번호 등 5개의 컬럼으로 구성된 테이블을 생성하되 기본 키 제약 조건, 외래키 제약 조건은 물로 CHECK 제약 조건도 설정해보자.

예

```
CREATE TABLE EMP08(
EMPNO NUMBER(4) PRIMARY KEY ,
ENAME VARCHAR2(10) NOT NULL,
SAL NUMBER(7, 2) CHECK(SAL BETWEEN 200 AND 1000),
GENDER VARCHAR2(1) CHECK (GENDER IN('M', 'F')),
DEPTNO NUMBER(2) REFERENCES DEPT(DEPTNO)
);
```

8. DEFAULT

- 디폴트는 아무런 값을 입력하지 않았을 때 디폴트 제약의 갑이 입력된다.
- ex) 만약 지역명(LOC)라는 컬럼에 아무런 값도 입력 안했을 때 디폴트의 값인 '서울'이 들어가도록 하고 싶을 경우 디폴트 제약 조건을 지정한다.

예

```
CREATE TABLE DEPT02(
DEPTNO NUMBER(2) PRIMARY KEY,
DNAME VARCHAR2(14),
LOC VARCHAR2(13) DEFAULT '서울'
);
```

- 만약 지역명(LOC)라는 컬럼에 아무런 값도 입력하지 않았을 때 디폴트의 값인 ' 서울'이 들어감을 확인할 수 있다.

예

```
INSERT INTO DEPT02(DEPTNO, DNAME)
VALUES(10, '영업부');
```

DEPTNO	DNAME	LOC
10	영업부	서울

9. 제약 조건 지정하기(1)

❖ 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 레벨 제약 조건

- 컬럼을 모두 정의하고 나서 테이블 정의를 마무리 짓기 전에 따로 생성된 컬럼들에 대한 제약 조건을 한꺼번에 지정하는 것입니다.

9. 제약 조건 지정하기(2)

- 일반적으로 컬럼 레벨 방식으로 제약조건을 지정하는 것이 훨씬 간편할 텐데 굳이 테이블 레벨의 지정 방식을 사용하는 데에는 2가지 이유가 있다.

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

9. 제약 조건 지정하기(3)

- 다음은 테이블 레벨 정의 방식의 기본 형식이다.

형식	**CREATE TABLE *table_name*** **(*column_name1 datatype1,*** ***column_name2 datatype2,*** **. . .** **[CONSTRAINT *constraint_name*] *constraint_type*** ***(column_name)*** **)**

- 테이블 레벨에서 컬럼의 제약 조건을 정의할 때 주의할 것은 NOT NULL 조건은 테이블 레벨 정의 방법으로 제약 조건을 지정할 수 없다는 점이다.

9. 제약 조건 지정하기(4)

- 일반적으로 컬럼 레벨 방식으로 제약조건을 지정하는 것이 훨씬 간편할 텐데 굳이 테이블 레벨의 지정 방식을 사용하는 데에는 2가지 이유가 있다.

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

9. 제약 조건 지정하기(5)

- ex) 컬럼 레벨 제약 조건과 테이블 레벨 제약 조건을 지정하는 방법의 차이점을 살펴보자.
- 1. 컬럼 레벨 제약 조건

예

```
CREATE TABLE EMP10(
EMPNO NUMBER(4) PRIMARY KEY
ENAME VARCHAR2(10) NOT NULL,
JOB VARCHAR2(9) UNIQUE
DEPTNO NUMBER(4) REFERENCES DEPT(DEPTNO)
);
```

9. 제약 조건 지정하기(6)

❖ 2. 테이블 레벨 제약 조건

예

```
CREATE TABLE EMP11(
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)
);
```

9. 제약 조건 지정하기(7)

- 3. 명시적으로 제약 조건 명을 지정하여 테이블 레벨 방식 제약 조건 지정

예

```
CREATE TABLE EMP12(
EMPNO NUMBER(4) CONSTRAINT EMP12_ENAME_NN NOT NULL,
ENAME VARCHAR2(10),
JOB VARCHAR2(9),
DEPTNO NUMBER(4),
CONSTRAINT EMP12_EMPNO_PK PRIMARY KEY(EMPNO),
CONSTRAINT EMP12_JOB_UK UNIQUE(JOB),
CONSTRAINT EMP12_DEPTNO_FK FOREIGN KEY(DEPTNO)
REFERENCES DEPT(DEPTNO)
);
```

9.1 ALTER로 제약 조건 추가하기(1)

- 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.
- 다음은 제약 조건을 추가하기 위한 형식이다.

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

- 만일 새로운 제약 조건을 추가하려면 ALTER TABLE 문에 ADD 절을 사용해야한다.

9.1 ALTER로 제약 조건 추가하기(2)

- ex) 지금까지 실습에 사용했던 사원 테이블과 유사한 구조의 사원 번호, 사원명, 직급, 부서 번호 4개의 컬럼으로 구성된 EMP13 테이블을 제약조건을 하나도 설정하지 않은 채 생성해보자.

예

```
CREATE TABLE EMP13(
EMPNO NUMBER(4),
ENAME VARCHAR2(10),
JOB VARCHAR2(9),
DEPTNO NUMBER(4)
);
```

9.1 ALTER로 제약 조건 추가하기(3)

- 이제 이미 생성이 완료된 EMP13 테이블에 2가지 제약조건을 설정해 보도록한다. 첫 번째는 EMPNO 컬럼에 기본키를 설정하고 두 번째에는 DEPTNO 컬럼에 외래키를 설정한다.

예

```
ALTER TABLE EMP13
ADD CONSTRAINT EMP13_EMPNO_PK PRIMARY
KEY(EMPNO);

ALTER TABLE EMP13
ADD CONSTRAINT EMP13_DEPTNO_FK
FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO);
```

9.2 MODIFY로 제약 조건 추가하기(1)

- NOT NULL 제약 조건을 이미 존재하는 테이블에 추가해 보도록한다.
- 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 주의해야한다.
- 이는 'NULL을 허용하는 상태'에서 'NULL을 허용하지 않는 상태'로 변경하겠다는 의미로 이해하면 된다.

9.2 MODIFY로 제약 조건 추가하기(2)

- ex) 이미 존재하는 테이블에 NOT NULL 제약 조건을 추가한다.

예

```
ALTER TABLE EMP10
MODIFY ENAME CONSTRAINT EMP01_ENAME_NN NOT
NULL;
```

10. 제약 조건 제거하기(1)

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

형식	**ALTER TABLE *table_name*** **DROP [CONSTRAINT *constraint_name*];**

- 제약 조건을 제거하기 위해서는 제약 조건명을 반드시 제시해야 한다.
- 제약 조건을 CONSTRAINT 문을 사용하여 지정했을 경우에는 제약 조건명을 기억하기 쉽지만, CONSTRAINT 문을 사용하지 않았으면 특정 테이블의 특정 컬럼에 명시된 제약 조건을 USER_CONSTRAINTS 데이터 딕셔너리 뷰에서 찾아봐야 하는 불편함이 있다.
- 그렇기 때문에 제약 조건을 지정할 때에는 제약 조건명을 명시적으로 주는 것이 바람직하다.

10. 제약 조건 제거하기(2)

- ex) 사원 테이블에 지정한 제약 조건들을 제거해 보도록 합시다.
- 1. 기본 키 제약 조건을 제거한다.

예

```
ALTER TABLE EMP12
DROP CONSTRAINT EMP12_EMPNO_PK;
```

- 2. 사원명에 NULL이 저장될 수 있도록 NOT NULL 제약 조건을 제거해 보도록 한다.

예

```
ALTER TABLE EMP12
DROP CONSTRAINT EMP12_ENAME_NN;
```

11. 제약 조건 비활성화(1)

- 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 비활성화 시킴으로서 제약 조건을 삭제하지 않고도 제약 조건 사용을 잠시 보류할 수 있는 방법을 제공해 준다.
- 이렇게 비활성화 된 제약 조건은 원하는 작업을 한 후에는 다시 활성화 상태로 만들어 주어야 한다.
- 제약 조건을 비활성화, 활성화하는 방법을 살펴보도록 하자.

11. 제약 조건 비활성화(2)

- ex) 실습을 위해서 부서 테이블을 만든다. 그런 후에 부서 테이블을 부모 테이블로 하는 사원 테이블을 작성한다. 그러기 위해서는 부서 테이블의 부서 번호가 기본 키로 설정되어 있고, 사원 테이블의 부서 번호가 부서 테이블의 부서 번호를 참조할 수 있도록 외래 키를 설정해야 한다.
- 1. 부서 테이블 생성 후 DEPT 테이블의 내용을 복사 해온다.

예

```
CREATE TABLE DEPT01(
DEPTNO NUMBER(2) CONSTRAINT DEPT01_DEPTNO_PK
PRIMARY KEY,
DNAME VARCHAR2(14),
LOC VARCHAR2(13)
);
```

예

```
INSERT INTO DEPT01(DEPTNO, DNAME, LOC)
SELECT DEPTNO, DNAME, LOC FROM DEPT;
```

11. 제약 조건 비활성화(3)

- 2. 부서 테이블을 만들었으므로 이제 부서 테이블을 부모 테이블로 하는 사원 테이블을 작성하기 위해서 사원 테이블의 부서 번호가 부서 테이블의 부서 번호를 참조할 수 있도록 외래 키를 설정한다.

예

```
CREATE TABLE EMP01(
EMPNO NUMBER(4)
CONSTRAINT EMP01_EMPNO_PK PRIMARY KEY ,
ENAME VARCHAR2(10)
CONSTRAINT EMP01_ENAME_NN NOT NULL,
JOB VARCHAR2(9),
DEPTNO NUMBER(4)
CONSTRAINT EMP01_DEPTNO_FK REFERENCES
DEPT01(DEPTNO)
);
```

11. 제약 조건 비활성화(4)

- 3. 사원 테이블로서 사원의 정보를 추가할 때 부서 테이블을 참조하므로 부서 테이블에 존재하는 부서 번호를 입력한다.

예

```
INSERT INTO EMP01 VALUES(7499, '김씨', '영업부', 10);
INSERT INTO EMP01 VALUES(7369, '박씨', '인사부', 20);
```

- 4. DEPT01 테이블에서 10번 부서를 '김씨'란 사람이 참조하고 있는 상태에서 삭제해 봅시다.

예

```
DELETE FROM DEPT01 WHERE DEPTNO=10;
```

```
명령의 1 행에서 시작하는 중 오류 발생 -
DELETE FROM DEPT01 WHERE DEPTNO=10
오류 보고 -
ORA-02292: integrity constraint (JOONZIS.EMP01_DEPTNO_FK) violated - child record found
```

11. 제약 조건 비활성화(5)

- 자식 테이블인 사원 테이블(EMP01)은 부모 테이블인 부서 테이블(DEPT01)에 기본 키인 부서 번호를 참조하고 있을 때.
- 부서 테이블의 10번 부서는 사원 테이블에 근무하는 10번 사원이 존재하기 때문에 삭제할 수 없다.
- 부모 테이블(DEPT01)의 부서 번호 10번이 삭제되면 자식 테이블(EMP01)에서 자신이 참조하는 부모를 잃어버리게 되므로 삭제할 수 없는 것이다.
- 부서 번호가 10인 자료가 삭제되도록 하기 위해서는 아래 방법이 있다.

1) 사원 테이블(EMP01)의 10번 부서에서 근무하는 사원을 삭제한 후 부서 테이블(DEPT01)에서 10번 부서를 삭제한다.
2) 참조 무결성 때문에 삭제가 불가능하므로 EMP01 테이블의 외래키 제약 조건을 제거한 후에 10번 부서를 삭제한다.

11. 제약 조건 비활성화(6)

- 테이블에서 제약 조건을 삭제하지 않고 일시적으로 적용시키지 않도록 하는 방법으로 제약 조건을 비활성화하는 방법이 있다. 제약조건을 비활성화하는 방법을 살펴보도록 하자.

형식	**DISABLE CONSTRAINT : 제약 조건의 일시 비활성화** **ENABLE CONSTRAINT : 비활성화된 제약 조건을 다시 활성화**

- 비활성화는 DISABLE 예약어를 사용하여 다음과 같이 지정한다.

형식	**ALTER TABLE *table_name*** **DISABLE *[CONSTRAINT constraint_name];***

11. 제약 조건 비활성화(7)

- EMP01 테이블에 지정한 제약 조건 중에서 외래키 제약 조건이 있다.
- 이 제약 조건 때문에 DEPT01 테이블에서 10번 부서를 삭제할 수 없었다.
- 왜냐하면 EMP01 테이블의 '김씨'란 사람이 DEPT01 테이블에서 10번 부서를 참조하고 있는 상태였기 때문이다.
- EMP01 테이블에 지정한 외래키 제약 조건을 비활성화 시키고 나면 EMP01 테이블과 DEPT01 테이블이 아무런 관계도 없는 상태가 되기 때문에 DEPT01 테이블에서 10번 부서를 삭제하는 데 아무런 문제가 없다.

11. 제약 조건 비활성화(8)

- ex) EMP01 테이블에 지정한 외래키 제약 조건을 비활성화한 후에 DEPT01 테이블에서 10번 부서를 삭제해 본다.
- 1. EMP01 테이블에 지정한 외래키 제약 조건을 비활성화 시킨다.

예

```
ALTER TABLE EMP01
DISABLE CONSTRAINT EMP01_DEPTNO_FK;
```

- 제약 조건의 상태를 확인하기 위해서 USER_CONSTRAINTS 데이터 딕셔너리의 STATUS 컬럼값을 살펴보면 EMP01_DEPTNO_FK 제약 조건에 대해서 STATUS 컬럼값이 DISABLED로 지정되어 있음을 확인할 수 있다.

예

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE,
TABLE_NAME, R_CONSTRAINT_NAME, STATUS
FROM USER_CONSTRAINTS
WHERE TABLE_NAME='EMP01';
```

11. 제약 조건 비활성화(9)

- 2. 이제 EMP01 테이블에 지정한 외래키 제약 조건을 비활성화하였기 때문에 DEPT01 테이블에서 10번 부서를 삭제할 수 있게 되었다.

예

```
DELETE FROM DEPT01
WHERE DEPTNO=10;
```

11.1 제약 조건 활성화(1)

- 제약 조건을 비활성화 해 보았으므로 이번에는 제약 조건을 활성화 해보도록 하자.
- 활성화는 ENABLE 예약어를 사용하여 다음과 같이 지정한다.

형식

ALTER TABLE *table_name*
ENABLE *[CONSTRAINT constraint_name];*

11.1 제약 조건 활성화(2)

- ex) EMP01 테이블에 지정한 제약 조건 중에서 외래키 제약 조건을 비활성화 했습니다. 비활성화된 제약 조건은 다시 활성화해야 한다.
- 1. DISABLE CONSTRAINT 문에 의해 비활성화된 제약 조건을 되살리려면 다음과 같이 ENABLE 을 사용해야 한다

형식

```
ALTER TABLE EMP01
ENABLE CONSTRAINT EMP01_DEPTNO_FK;
```

- 하지만 부서 테이블의 10번 부서가 삭제된 상태에서는 외래키 제약 조건을 활성화 시킬 수 없다.
- 왜냐하면 외래키 제약 조건은 참조 무결성을 위배하지 않은 상태에서만 지정할 수 있는데, 사원 테이블(EMP01)에서 부서 테이블(DEPT01)의 10번 부서를 참조하고 있고 부서 테이블에 10번 부서가 존재하지 않기 때문에 참조 무결성에 위배되기 때문이다.

11.1 제약 조건 활성화(3)

- 2. 그러므로 외래키 제약 조건을 활성화시키기 전에 먼저 삭제된 부서 테이블의 10번 부서를 새로 입력해 놓아야 한다.

형식

```
INSERT INTO DEPT01 VALUES(10, '경리부', '서울');
```

- 3. 10번 부서를 새로 입력해 놓았으므로 이제 외래키 제약 조건을 활성화한다.

형식

```
ALTER TABLE EMP01
ENABLE CONSTRAINT EMP01_DEPTNO_FK;
```

12. CASCADE(1)

- 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 비활성화시켜 주는 옵션이다.

- 또한 제약 조건의 비활성화뿐만 아니라 제약 조건의 삭제에도 활용되며, 역시 같은 이치로 부모 테이블의 제약 조건을 삭제하면 이를 참조하고 있는 자식 테이블의 제약 조건도 같이 삭제된다.

12. CASCADE(2)

- ex) 부서 테이블(DEPT01)의 기본 키 제약 조건을 비활성화해 보도록 하자.
- 1. 부서 테이블(DEPT01)의 기본 키 제약 조건을 "DISABLE PRIMARY KEY" 로 비 활성화하려고 시도한다.

형식

```
ALTER TABLE DEPT01
DISABLE PRIMARY KEY;
```

```
명령의 1 행에서 시작하는 중 오류 발생 -
ALTER TABLE DEPT01
DISABLE PRIMARY KEY
오류 보고 -
ORA-02297: cannot disable constraint (JOONZIS.DEPT01_DEPTNO_PK) - dependencies exist
02297. 00000 - "cannot disable constraint (%s.%s) - dependencies exist"
*Cause:    an alter table disable constraint failed becuase the table has
           foriegn keys that are dpendent on this constraint.
*Action:   Either disable the foreign key constraints or use disable cascade
```

- 부서 테이블의 기본 키는 사원 테이블(EMP01)의 외래 키에서 참조하고 있기 때문에 제약 조건을 비활성화할 수 없다.

12. CASCADE(3)

- 부모 테이블(부서)의 기본 키에 대한 제약조건을 비활성화하고자 하는 것인데 자식 테이블(사원)에서 이를 외래 키 제약조건으로 지정한 컬럼이라면 절대 비활성화할 수 없다.
- 만일 비활성화될 수 있다고 가정하면 기본 키가 더 이상 아닌 상태로 일반 컬럼을 자식 테이블이 외래 키 제약조건으로 지정하고 있는 아이러니 한 상태가 되기 때문이다.
- 그렇기 때문에 부모 테이블(부서)의 기본 키에 대한 제약조건을 비활성화 하려면 자식 테이블(사원)의 외래 키에 대한 제약조건을 비활성화하는 작업이 선행되어야 한다.
- 두 테이블 사이에 아무런 관련이 없어야 만 즉, 부서 테이블이 더 이상 부모 테이블로서의 역할을 하지 않고 있어야만 기본 키 제약 조건을 비활성화 시킬 수 있다.

12. CASCADE(4)

- 부모 테이블(부서)의 기본 키에 대한 제약조건을 비활성화하기 위한 작업을 순서대로 정리해보자.

1) 부모 테이블의 기본 키를 참조하는 자식 테이블의 외래 키에 대한 제약 조건을 비활성화해야 한다.
2) 부모 테이블의 기본 키에 대한 제약 조건을 비활성화해야 한다.

- 위 순서대로 제약조건을 여러 번에 걸쳐 비활성화 시키기는 번거로움을 없애주는 것이 CASCADE 옵션이다.
- CASCADE 옵션을 지정하여 기본 키 제약 조건을 비활성화하면 이를 참조하는 외래 키 제약 조건도 연속적으로 비활성화되기 때문에 한 번만 비활성화 해 주면 된다.

12. CASCADE(5)

- ex) CASCADE 옵션을 지정하여 기본 키 제약 조건을 비활성화하면 이를 참조하는 외래 키 제약 조건도 연속적으로 비활성화된다.
- 1. 부서 테이블(DEPT01)의 기본 키 제약 조건을 CASCADE 옵션을 지정하여 비활성화한다.

형식

```
ALTER TABLE DEPT01
DISABLE PRIMARY KEY CASCADE;
```

- 2. 데이터 딕셔너리 USER_CONSTRAINTS를 살펴본다.

형식

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

12. CASCADE(6)

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	R_CONSTRAINT_NAME	STATUS
DEPT01_DEPTNO_PK	P	DEPT01	(null)	DISABLED
EMP01_ENAME_NN	C	EMP01	(null)	ENABLED
EMP01_EMPNO_PK	P	EMP01	(null)	ENABLED
EMP01_DEPTNO_FK	R	EMP01	DEPT01_DEPTNO_PK	DISABLED

- DEPT01 테이블의 기본 키 제약 조건이 비활성화 된 것을 확인할 수 있다.
- DEPT01 테이블 뿐만 아니라 DEPT01 테이블의 기본키를 참조하고 있는 EMP01 테이블의 외래키 제약 조건도 비활성화 된 것을 확인 할 수 있다.

12. CASCADE(7)

- ex) CASCADE 옵션을 지정하여 기본 키 제약 조건을 제거하면 이를 참조하는 외래 키 제약 조건도 연속적으로 제거된다.
- 1. 이번에는 부서 테이블(DEPT01)의 기본키 제약 조건을 삭제해보도록 한다.

형식

```
ALTER TABLE DEPT01
DROP PRIMARY KEY;
```

```
명령의 1 행에서 시작하는 중 오류 발생 -
ALTER TABLE DEPT01
DROP PRIMARY KEY
오류 보고 -
ORA-02273: this unique/primary key is referenced by some foreign keys
02273. 00000 -  "this unique/primary key is referenced by some foreign keys"
*Cause:    Self-evident.
*Action:   Remove all references to the key before the key is to be dropped.
```

- 부서 테이블의 기본 키는 사원 테이블의 외래 키에서 참조하고 있기 때문에 제약 조건을 삭제할 수 없다.

12. CASCADE(8)

- 2. CASCADE 옵션을 지정하여 기본 키 제약 조건을 삭제하게 되면 이를 참조하는 외래 키 제약 조건도 연속적으로 삭제된다.

형식	**ALTER TABLE DEPT01** **DROP PRIMARY KEY CASCADE;**

- 데이터 딕셔너리 USER_CONSTRAINTS를 살펴본다.

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

12. CASCADE(9)

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	R_CONSTRAINT_NAME	STATUS
EMP01_ENAME_NN	C	EMP01	(null)	ENABLED
EMP01_EMPNO_PK	P	EMP01	(null)	ENABLED

- USER_CONSTRAINTS 데이터 딕셔너리를 살펴보면 DEPT01 테이블의 기본 키 제약 조건은 물론 이를 참조하는 EMP01 테이블의 외래 키 제약 조건도 삭제되었음을 확인할 수 있다.