

## 제약 조건

제약 조건(constraint)은 테이블에 저장할 데이터를 제약하는 특수한 규칙을 뜻합니다. 제약 조건을 설정한 열에는 조건에 맞지 않는 데이터를 저장할 수 없습니다. 이 제약 조건은 데이터 베이스 데이터의 정확성을 유지하기 위한 목적으로 사용하며 12장에서 살펴본 데이터 정의어 (DDL)로 설정할 수 있습니다. 이 장에서는 여러 제약 조건의 종류와 그 특성을 살펴보고 사용 방법을 알아보겠습니다.

### 14-1 제약 조건 종류

- 14-2 빈값을 허락하지 않는 NOT NULL
- 14-3 중복되지 않는 값 UNIQUE
- 14-4 유일하게 하나만 있는 값 PRIMARY KEY
- 14-5 다른 테이블과 관계를 맺는 FOREIGN KEY
- 14-6 데이터 형태와 범위를 정하는 CHECK
- 14-7 기본값을 정하는 DEFAULT

### 이 장에서 꼭 익혀야 할 것

- 제약 조건의 종류와 의미
- 테이블을 생성할 때 제약 조건 지정 방법

## 14-1 제약 조건 종류

### 제약 조건이란?

오라클에서 사용하는 제약 조건은 테이블의 특정 열에 지정합니다. 제약 조건을 지정한 열에 제약 조건에 부합하지 않는 데이터를 저장할 수 없습니다. 제약 조건 지정 방식에 따라 기존 데이터의 수정이나 삭제 가능 여부도 영향을 받습니다.

예를 들어 로그인에 사용할 아이디나 이메일 주소를 중복되지 않도록 설정할 수 있습니다. 또 는 회원 가입할 때 이름, 생년월일 등의 데이터는 필수 입력 항목으로 두어 빈값(NULL)을 허용하지 않도록 지정할 수 있습니다.

오라클 데이터베이스에서 사용하는 제약 조건은 다음과 같습니다.

종류	설명
NOT NULL	지정한 열에 NULL을 허용하지 않습니다. NULL을 제외한 데이터의 중복은 허용됩니다.
UNIQUE	지정한 열이 유일한 값을 가져야 합니다. 즉 중복될 수 없습니다. 단 NULL은 값의 중복에서 제외됩니다.
PRIMARY KEY	지정한 열이 유일한 값이면서 NULL을 허용하지 않습니다. PRIMARY KEY는 테이블에 하나만 지정 가능합니다.
FOREIGN KEY	다른 테이블의 열을 참조하여 존재하는 값만 입력할 수 있습니다.
CHECK	설정한 조건식을 만족하는 데이터만 입력 가능합니다.

### ▣ 한 발 더 나가기!! 데이터 무결성이란?

데이터 무결성(data integrity)은 데이터베이스에 저장되는 데이터의 정확성과 일관성을 보장한다는 의미이며 이를 위해 항상 유지해야 하는 기본 규칙을 가지고 있습니다. 제약 조건은 이러한 데이터 무결성을 지키기 위한 안전장치로서 중요한 역할을 합니다. 그리고 테이블 데이터의 삽입(insert), 수정(update), 삭제(delete) 등 모든 과정에서 지켜야 합니다. 다음 표는 데이터 무결성의 종류를 정리해 놓은 것입니다.

종류	설명
영역 무결성 (domain integrity)	열에 저장되는 값의 적정 여부를 확인. 자료형, 적절한 형식의 데이터, NULL 여부같은 정해 놓은 범위를 만족하는 데이터임을 규정
개체 무결성 (entity integrity)	테이블 데이터를 유일하게 식별할 수 있는 기본키는 반드시 값을 가지고 있어야 하며 NULL이 될 수 없고 중복될 수도 없음을 규정
참조 무결성 (referential integrity)	참조 테이블의 외래키 값은 참조 테이블의 기본키로서 존재해야 하며 NULL이 가능

☞ 데이터 무결성의 자세한 내용은 관계형 데이터 모델 관련 서적 및 자료를 참고하세요.

---

이러한 무결성을 보장하기 위해 오라클에서는 앞에서 살펴본 다섯 가지 제약 조건을 제공합니다. 제약 조건은 데이터베이스 설계 시점, 즉 테이블을 생성할 때 주로 지정합니다. 하지만 테이블 생성 후에도 추가·변경·삭제할 수 있습니다. 따라서 제약 조건은 데이터 정의어(DDL)에서 활용합니다.

## 14-2 빈값을 허락하지 않는 NOT NULL

### 테이블을 생성하며 제약 조건 지정

NOT NULL은 특정 열에 데이터의 중복 여부와는 상관없이 NULL의 저장을 허용하지 않는 제약 조건입니다. 반드시 열에 값이 존재해야만 하는 경우에 지정합니다. 우선, 다음과 같이 NOT NULL 제약조건을 지정하는 열을 포함된 테이블을 생성합시다. NOT NULL 제약 조건은 열 이름과 자료형 뒤에 NOT NULL 키워드를 명시하여 지정합니다.

#### 실습 14-1 테이블을 생성할 때 NOT NULL 설정하기

```
01 CREATE TABLE TABLE_NOTNULL(
02     LOGIN_ID VARCHAR2(20) NOT NULL,
03     LOGIN_PWD VARCHAR2(20) NOT NULL,
04     TEL        VARCHAR2(20)
05 );
06 DESC TABLE_NOTNULL;
```

:: 결과 화면(일부 열만 표시함)

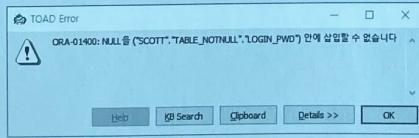
Column Name	ID	PK	Index Pos	Null?	Data Type	Default	Histogram
▶ LOGIN_ID	1			N	VARCHAR2 (20 Byte)	None	
LOGIN_PWD	2			N	VARCHAR2 (20 Byte)	None	
TEL	3			Y	VARCHAR2 (20 Byte)	None	

LOGIN\_ID, LOGIN\_PWD 열을 NOT NULL로 지정했습니다. 이 두 열은 INSERT문을 통해 신규 데이터를 삽입할 때 NULL 값을 입력할 경우 다음과 같이 오류가 발생합니다.

#### 실습 14-2 제약 조건이 NOT NULL인 열에 NULL 값 넣어보기

```
01 INSERT INTO TABLE_NOTNULL (LOGIN_ID, LOGIN_PWD, TEL)
02 VALUES ('TEST_ID_01', NULL, '010-1234-5678');
```

:: 결과 화면



즉 LOGIN\_ID, LOGIN\_PWD 열은 반드시 NULL이 아닌 값을 지정하도록 강제로 지정하고 있습니다. 반면에 TEL 열은 별다른 제약 조건을 지정하지 않았으므로 값을 지정하지 않아도 오류가 발생하지 않습니다.

#### 실습 14-3 제약 조건이 없는 TEL 열에 NULL 값 입력하기

```
01  INSERT INTO TABLE_NOTNULL (LOGIN_ID, LOGIN_PWD)
02    VALUES ('TEST_ID_01', '1234');

03  SELECT * FROM TABLE_NOTNULL;
```

:: 결과 화면

	LOGIN_ID	LOGIN_PWD	TEL
▶	TEST_ID_01	1234	

☞ TEL 열을 비워둠으로써 NULL을 암시적으로 삽입하고 있습니다.

열의 제약 조건으로 NOT NULL을 지정하면 UPDATE문을 사용하여 LOGIN\_ID 또는 LOGIN\_PWD 열 값을 NULL로 수정하는 것도 불가능합니다. 제약 조건을 지정한 열은 항상 해당 제약 조건을 만족해야 하므로 신규 데이터의 삽입뿐만 아니라 기존 데이터의 수정 및 삭제에도 영향을 줍니다.

#### 실습 14-4 NOT NULL 제약 조건이 지정된 열 데이터를 NULL 값으로 업데이트하기

```
01  UPDATE TABLE_NOTNULL
02    SET LOGIN_PWD = NULL
03  WHERE LOGIN_ID = 'TEST_ID_01';
```

:: 결과 화면



## 제약 조건 확인

지정한 제약 조건 정보를 확인하려면 다음과 같은 USER\_CONSTRAINTS 데이터 사전을 활용합니다.

열 이름	설명
OWNER	제약 조건 소유 계정
CONSTRAINT_NAME	제약 조건 이름(직접 지정하지 않을 경우 오라클이 자동으로 지정함)
CONSTRAINT_TYPE	제약 조건 종류 C : CHECK, NOT NULL U : UNIQUE P : PRIMARY KEY R : FOREIGN KEY
TABLE_NAME	제약 조건을 지정한 테이블 이름

다음 SELECT문을 사용하면 SCOTT 계정 소유의 제약 조건을 확인할 수 있습니다. 실습 14-1에서 제약 조건을 포함한 CREATE문을 통해 생성한 제약 조건을 살펴보면 실습 14-5의 결과화면(빨간색 상자로 표시한 부분)과 같습니다. CONSTRAINT\_NAME 열에 각 제약 조건 이름이 출력되며, 실습 14-1에서는 제약 조건의 이름을 따로 지정하지 않았으므로 오라클에 의해 자동으로 이름이 지정된 것을 확인할 수 있습니다.

### 실습 14-5 제약 조건 살펴보기(SCOTT 계정)

```
01  SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
02    FROM USER_CONSTRAINTS;
```

:: 결과 화면

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SCOTT	SYS_C0015544	C	TABLE_NOTNULL
SCOTT	SYS_C0015545	C	TABLE_NOTNULL
SCOTT	FK_DEPTNO	R	EMP
SCOTT	PK_DEPT	P	DEPT
SCOTT	PK_EMP	P	EMP

☞ 여러분이 실행한 실습의 결과 화면을 살펴보면 CONSTRAINT\_NAME 열 이름이 책의 결과와 다를 수 있습니다. 왜냐하면 EMP 테이블 및 DEPT 테이블에는 이미 제약 조건이 지정되어 있기 때문입니다.

## 제약 조건 이름 직접 지정

TABLE\_NOTNULL 테이블에 지정한 제약 조건은 이름을 따로 지정해 주지 않아 오라클에서 자동으로 이름이 지정되었습니다. 제약 조건에 이름을 직접 지정하려면 다음과 같이 CONSTRAINT 키워드를 사용합니다.

실습 14-6 테이블을 생성할 때 제약 조건에 이름 지정하기

```
01 CREATE TABLE TABLE_NOTNULL2(
02     LOGIN_ID VARCHAR2(20) CONSTRAINT TBLNN2_LGNID_NN NOT NULL,
03     LOGIN_PWD VARCHAR2(20) CONSTRAINT TBLNN2_LGPNW_NN NOT NULL,
04     TEL      VARCHAR2(20)
05 );
06
07     SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
08     FROM USER_CONSTRAINTS;
```

:: 결과 화면

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SCOTT	SYS_C0015544	C	TABLE_NOTNULL
SCOTT	SYS_C0015545	C	TABLE_NOTNULL
SCOTT	TBLNN2_LGNID_NN	C	TABLE_NOTNULL2
SCOTT	TBLNN2_LGPNW_NN	C	TABLE_NOTNULL2
SCOTT	FK_DEPTNO	R	EMP
SCOTT	PK_DEPT	P	DEPT
SCOTT	PK_EMP	P	EMP

USER\_CONSTRAINTS 데이터 사전을 조회하면 제약 조건 이름(CONSTRAINT\_NAME)이 실습 14-6에서 지정한 대로 저장되어 있는 것을 확인할 수 있습니다.



실무에서 제약 조건 이름을 지정할 때

실무 꿀팁!

오라클이 자동으로 지정해 주는 이름은 제약 조건이 많아진 후 찾기 어려워질 수 있으므로 실무에서는 이를 붙이는 규칙을 정하여 제약 조건 이름을 직접 지정하는 경우가 많습니다.

## 이미 생성한 테이블에 제약 조건 지정

앞에서 보았듯이 제약 조건은 저장할 데이터에 제한을 주는 규칙으로 작용합니다. 이러한 특성으로 인해 제약 조건은 데이터와 테이블을 설계하는 시점, 즉 데이터베이스 사용 주기에서 비교적 초기에 지정하는 것이 일반적입니다. 하지만 경우에 따라 이미 생성되어 있는 테이블에 제약 조건을 추가하거나 제약 조건을 변경 또는 삭제해야 하는 경우도 종종 생깁니다.

## 생성한 테이블에 제약 조건 추가하기

NOT NULL 제약 조건의 추가는 ALTER 명령어와 MODIFY 키워드를 사용합니다. 먼저 실습 14-1에서 생성한 TABLE\_NOTNULL 테이블의 TEL 열에 NOT NULL 제약 조건을 추가해 보겠습니다. 제약 조건 추가에 앞서 실습 14-3을 통해 TEL 열에 NULL이 저장된 데이터가 이미 존재하고 있음을 기억해 보세요.

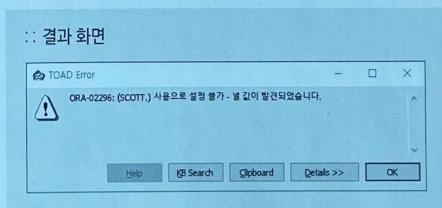
LOGIN_ID	LOGIN_PWD	TEL
TEST_ID_01	1234	

다음은 TABLE\_NOTNULL 테이블의 TEL 열에 NOT NULL 제약 조건을 추가하는 명령어입니다.

### 실습 14-7 TEL 열에 NOT NULL 제약 조건 추가하기

```
01 ALTER TABLE TABLE_NOTNULL  
02 MODIFY(TEL NOT NULL);
```

위 실습을 실행하면 다음과 같이 오류가 발생하고 제약 조건 추가는 실패합니다. 이는 제약 조건 대상이 되는 열이 가진 데이터 중 추가하려는 제약 조건에 맞지 않는 데이터가 존재하기 때문입니다. 여기에서는 열 데이터에 NULL을 허용하지 않는 NOT NULL 제약 조건을 추가하여 했는데, 이미 TEL 열의 데이터 중 NULL 값이 존재하기 때문에 제약 조건이 추가되지 않은 것입니다. 제약 조건 추가를 위한 명령어를 잘 작성했어도 제약 조건과 맞지 않는 데이터가 이미 있다면 제약 조건 지정이 실패한다는 사실을 잊지 마세요.



그러면 다음 UPDATE문으로 기존 TEL 열을 NULL이 아닌 데이터로 수정해 봅시다.

### 실습 14-8 TEL 열 데이터 수정하기

```
01 UPDATE TABLE_NOTNULL  
02     SET TEL = '010-1234-5678'  
03 WHERE LOGIN_ID = 'TEST_ID_01';  
  
04 SELECT * FROM TABLE_NOTNULL;
```

#### 결과 화면

LOGIN_ID	LOGIN_PWD	TEL
TEST_ID_01	1234	010-1234-5678

그리고 실습 14-7에서 실패했던 TEL 열에 NOT NULL 제약 조건을 다시 지정해 보겠습니다.

#### 실습 14-9 NOT NULL 제약 조건 추가하기

```
01 ALTER TABLE TABLE_NOTNULL
02   MODIFY(TEL NOT NULL);

03 SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
04   FROM USER_CONSTRAINTS;
```

TABLE\_NOTNULL 테이블의 TEL 열에 NULL을 가진 데이터가 없으므로 NOT NULL 제약 조건이 별다른 오류 없이 지정되는 것을 확인할 수 있습니다. USER\_CONSTRAINTS 데이터 사전을 통해 생성된 제약 조건을 확인해 주세요.

#### :: 결과 화면

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SCOTT	SYS_C0015544	C	TABLE_NOTNULL
SCOTT	SYS_C0015545	C	TABLE_NOTNULL
SCOTT	TBLNN2_LGNID_NN	C	TABLE_NOTNULL2
SCOTT	TBLNN2_LGPNW_NN	C	TABLE_NOTNULL2
SCOTT	SYS_C0015548	C	TABLE_NOTNULL



CONSTRAINT\_TYPE이  
C면 NOT NULL 또는  
CHECK라는 뜻입니다.

#### 생성한 테이블에 제약 조건 이름 직접 지정해서 추가하기

제약 조건 이름을 직접 지정하려면 CREATE와 마찬가지로 CONSTRAINT 키워드를 사용하면 됩니다. 실습 14-6에서 생성한 TABLE\_NOTNULL2 테이블의 TEL 열에 이름을 직접 지정하여 제약 조건(NOT NULL)을 추가해 보겠습니다.

#### 실습 14-10 제약 조건에 이름 지정해서 추가하기

```
01 ALTER TABLE TABLE_NOTNULL2
02   MODIFY(TEL CONSTRAINT TBLNN_TEL_NN NOT NULL);

03 SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
04   FROM USER_CONSTRAINTS;
```

:: 결과 화면

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SCOTT	SYS_C0015544	C	TABLE_NOTNULL
SCOTT	SYS_C0015545	C	TABLE_NOTNULL
SCOTT	TBLNN2_LGNID_NN	C	TABLE_NOTNULL2
SCOTT	TBLNN2_LGNPV_NN	C	TABLE_NOTNULL2
SCOTT	SYS_C0015548	C	TABLE_NOTNULL
SCOTT	TBLNN_TEL_NN	C	TABLE_NOTNULL2

#### 실습 14-11 TABLE\_NOTNULL2 테이블 열 구조 확인하기

01 DESC TABLE\_NOTNULL2;

:: 결과 화면(일부 열만 표시함)

Column Name	ID	PK	Index Pos	Null?	Data Type	Default	Histogram	Num Distinct
LOGIN_ID	1			N	VARCHAR2 (20 Byte)	None		
LOGIN_PWD	2			N	VARCHAR2 (20 Byte)	None		
TEL	3			N	VARCHAR2 (20 Byte)	None		

#### 생성한 제약 조건의 이름 변경하기

이미 생성한 제약 조건 이름을 변경하려면 ALTER 명령어에 RENAME CONSTRAINT 키워드를 사용합니다.

#### 실습 14-12 이미 생성된 제약 조건 이름 변경하기

01 ALTER TABLE TABLE\_NOTNULL2

02 RENAME CONSTRAINT TBLNN\_TEL\_NN TO TBLNN2\_TEL\_NN;

03 SELECT OWNER, CONSTRAINT\_NAME, CONSTRAINT\_TYPE, TABLE\_NAME

04 FROM USER\_CONSTRAINTS;

:: 결과 화면

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SCOTT	SYS_C0015544	C	TABLE_NOTNULL
SCOTT	SYS_C0015545	C	TABLE_NOTNULL
SCOTT	TBLNN2_LGNID_NN	C	TABLE_NOTNULL2
SCOTT	TBLNN2_LGNPV_NN	C	TABLE_NOTNULL2
SCOTT	SYS_C0015548	C	TABLE_NOTNULL
SCOTT	TBLNN2_TEL_NN	C	TABLE_NOTNULL2
SCOTT	FK_DEPTNO	R	EMP
SCOTT	PK_DEPT	P	DEPT
SCOTT	PK_EMP	P	EMP



제약 조건 이름이  
TBLNN\_TEL\_NN에서  
TBLNN2\_TEL\_NN으로 바  
뀌었습니다.

제약 조건 삭제

ALTER 명령어에 DROP CONSTRAINT 키워드를 사용하면 지정한 제약 조건을 삭제할 수 있습니다.

#### 실습 14-13 제약 조건 삭제하기

```
01 ALTER TABLE TABLE_NOTNULL2  
02 DROP CONSTRAINT TBLNN2_TEL_NN;  
03 DESC TABLE NOTNULL2;
```

제약 조건을 삭제하면 다음과 같이 TEL 열은 NULL이 저장될 수 있는 열이 됩니다.

:: 결과 화면(일부 열만 표시함)

Column Name	ID	PK	Index Pos	Null?	Data Type	Default	Histogram	Num Distinct
LOGIN_ID		1		N	VARCHAR2 (20 Byte)	None		
LOGIN_PWD		2		N	VARCHAR2 (20 Byte)	None		
TEL		3		Y	VARCHAR2 (20 Byte)	None		

1분  
복습

이미 존재하는 테이블의 특정 열에 NOT NULL 제약 조건을 추가하는 다음 코드의 빈칸을 채워 보세요.

1 테이블 이름  
2 (열 이름 3 )

1. ALTER TABLE 2. MODIFY 3. NOT NULL

## 14-3 중복되지 않는 값 UNIQUE

UNIQUE 제약 조건은 열에 저장할 데이터의 중복을 허용하지 않고자 할 때 사용합니다. NULL은 값이 존재하지 않음을 의미하기 때문에 중복 대상에서는 제외됩니다. 즉 UNIQUE 제약 조건을 지정한 열에 NULL은 여러 개 존재할 수 있습니다. UNIQUE 지정 방법은 NOT NULL 제약 조건과 동일합니다.

### 테이블을 생성하며 제약 조건 지정

UNIQUE 제약 조건 역시 CREATE문으로 테이블을 생성할 때 지정할 수 있습니다.

#### 실습 14-14 제약 조건 지정하기(테이블을 생성할 때)

```
01 CREATE TABLE TABLE_UNIQUE(
02     LOGIN_ID  VARCHAR2(20) UNIQUE,
03     LOGIN_PWD VARCHAR2(20) NOT NULL,
04     TEL        VARCHAR2(20)
05 );
06 DESC TABLE_UNIQUE;
```

:: 결과 화면

Column Name	ID	PK	Index Pos	Null?	Data Type	Default	Histogram	Num Distinct	Num Nulls	Density
LOGIN_ID	1	1	Y	VARCHAR2 (20 Byte)		None				
LOGIN_PWD	2		N	VARCHAR2 (20 Byte)		None				
TEL	3		Y	VARCHAR2 (20 Byte)		None				

### 제약 조건 확인

USER\_CONSTRAINTS 데이터 사전에서 CONSTRAINT\_TYPE 열 값이 U일 경우에 UNIQUE 제약 조건을 의미한다는 것을 기억해 보세요.

#### 실습 14-15 USER\_CONSTRAINTS 데이터 사전 뷰로 제약 조건 확인하기

```
01 SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
02   FROM USER_CONSTRAINTS
03  WHERE TABLE_NAME = 'TABLE_UNIQUE';
```

:: 결과 화면

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SCOTT	SYS_C0015610	C	TABLE_UNIQUE
SCOTT	SYS_C0015611	U	TABLE_UNIQUE

WHERE절을 통해 TABLE\_UNIQUE 테이블의 제약 조건만 조회합니다.

## 중복을 허락하지 않는 UNIQUE

UNIQUE 제약 조건을 지정한 LOGIN\_ID 열은 중복 값이 저장되지 않습니다. 그러면 이 내용을 확인하기 위해 먼저 INSERT문으로 데이터를 넣어 보죠.

실습 14-16 TABLE\_UNIQUE 테이블에 데이터 입력하기

```
01  INSERT INTO TABLE_UNIQUE(LOGIN_ID, LOGIN_PWD, TEL)
02  VALUES('TEST_ID_01', 'PWD01', '010-1234-5678');

03  SELECT * FROM TABLE_UNIQUE;
```

:: 결과 화면

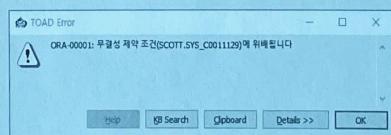
LOGIN_ID	LOGIN_PWD	TEL
TEST_ID_01	PWD01	010-1234-5678

그리고 LOGIN\_ID가 같은 값을 가진 다음 INSERT문을 실행해 보죠.

실습 14-17 LOGIN\_ID 열에 중복되는 데이터 넣기

```
01  INSERT INTO TABLE_UNIQUE (LOGIN_ID, LOGIN_PWD, TEL)
02  VALUES ('TEST_ID_01', 'PWD01', '010-1234-5678');
```

:: 결과 화면



실습 14-17에서는 실습 14-16과 같은 내용의 데이터를 넣고 있습니다. 하지만 UNIQUE 제약 조건이 지정되어 있는 LOGIN\_ID 열 때문에 위와 같이 오류가 발생합니다.

### 실습 14-18 TABLE\_UNIQUE 테이블에 데이터 입력하기

```
01 INSERT INTO TABLE_UNIQUE(LOGIN_ID, LOGIN_PWD, TEL)
02 VALUES('TEST_ID_02', 'PWD01', '010-1234-5678');
03 SELECT * FROM TABLE_UNIQUE;
```



LOGIN\_PWD 열은 NOT NULL 조건만  
지정되어 있어서 중복은 허용됩니다.

:: 결과 화면

	LOGIN_ID	LOGIN_PWD	TEL
▶	TEST_ID_01	PWD01	010-1234-5678
	TEST_ID_02	PWD01	010-2345-6789

실습 14-17과 달리 LOGIN\_ID에 다른 값을 지정한 경우는 문제없이 잘 실행됩니다. 여기에서 유심히 살펴봐야 하는 내용은 LOGIN\_PWD 열입니다. 이 열에 NOT NULL 제약 조건을 지정하여 중복 값이 있어도 문제가 발생하지 않습니다. NOT NULL은 열 값이 NULL만 아니면 되니까요.

### UNIQUE 제약 조건과 NULL 값

UNIQUE 제약 조건은 열 값의 중복은 허용하지 않지만 NULL 저장은 가능합니다. NULL은 존재하지 않는 값 또는 해당 사항이 없다는 의미로 사용되는 특수한 값이므로 NULL과 NULL을 비교했을 때 값이 같은지를 확인할 수 없습니다. 즉 NULL은 데이터 중복의 의미를 부여할 수 없습니다. 따라서 UNIQUE 제약 조건이 지정된 열에는 NULL이 여러 개 존재 할 수 있습니다.

### 실습 14-19 UNIQUE 제약 조건이 지정된 열에 NULL 값 입력하기

```
01 INSERT INTO TABLE_UNIQUE(LOGIN_ID, LOGIN_PWD, TEL)
02 VALUES(NULL, 'PWD01', '010-2345-6789');
03 SELECT * FROM TABLE_UNIQUE;
```

:: 결과 화면

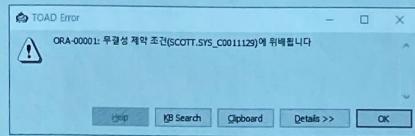
	LOGIN_ID	LOGIN_PWD	TEL
▶	TEST_ID_01	PWD01	010-1234-5678
	TEST_ID_02	PWD01	010-2345-6789
		PWD01	010-2345-6789

만약 UPDATE문을 사용하여 LOGIN\_ID 열에 이미 존재하는 값(TEST\_ID\_01)을 지정할 경우에 실행 후 중복 데이터가 발생하므로 다음과 같이 실행되지 않습니다.

#### 실습 14-20 TABLE\_UNIQUE 테이블 데이터 수정하기

```
01 UPDATE TABLE_UNIQUE  
02     SET LOGIN_ID='TEST_ID_01'  
03 WHERE LOGIN_ID IS NULL;
```

:: 결과 화면



#### 테이블을 생성하며 제약 조건 이름 직접 지정

UNIQUE 제약 조건 역시 제약 조건 이름을 지정할 수 있으며 지정하지 않으면 오라클이 자동으로 제약 조건 이름을 정해 줍니다.

#### 실습 14-21 테이블을 생성할 때 UNIQUE 제약 조건 설정하기

```
01 CREATE TABLE TABLE_UNIQUE2(  
02     LOGIN_ID  VARCHAR2(20) CONSTRAINT TBLUNQ2_LGNID_UNQ UNIQUE,  
03     LOGIN_PWD VARCHAR2(20) CONSTRAINT TBLUNQ2_LGPNPW_NN NOT NULL,  
04     TEL        VARCHAR2(20)  
05 );
```

USER\_CONSTRAINTS 데이터 사전을 조회하면 제약 조건 이름이 앞에서 지정한 대로 저장되어 있는 것을 확인할 수 있습니다.

#### 실습 14-22 생성한 UNIQUE 제약 조건 확인하기(USER\_CONSTRAINTS 사용)

```
01 SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME  
02   FROM USER_CONSTRAINTS  
03 WHERE TABLE_NAME LIKE 'TABLE_UNIQUE%';
```

:: 결과 화면(실습 14-22)

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SCOTT	SYS_C0015610	C	TABLE_UNIQUE
SCOTT	SYS_C0015611	U	TABLE_UNIQUE
SCOTT	TBLUNQ2_LGPNPW_NN	C	TABLE_UNIQUE2
SCOTT	TBLUNQ2_LGNID_UNQ	U	TABLE_UNIQUE2

## 이미 생성한 테이블에 제약 조건 지정

ALTER 명령어로 이미 생성되어 있는 테이블에 UNIQUE 제약 조건을 추가할 수 있습니다.

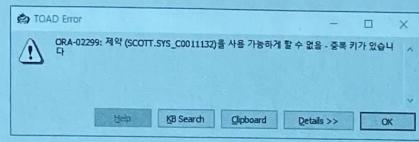
생성한 테이블에 제약 조건 추가하기

다음은 TABLE\_UNIQUE 테이블의 TEL 열에 UNIQUE 제약 조건을 추가합니다.

### 실습 14-23 이미 생성한 테이블 열에 UNIQUE 제약 조건 추가하기

```
01 ALTER TABLE TABLE_UNIQUE  
02 MODIFY(TEL UNIQUE);
```

:: 결과 화면



실습 14-23을 실행하면 오류가 발생하여 제약 조건 추가가 실패합니다. 왜냐하면 TEL 열에 이미 중복된 열이 있기 때문입니다. NOT NULL과 마찬가지로 제약 조건을 추가할 때 해당 열에 추가하려는 제약 조건에 맞지 않는 데이터가 존재할 경우 제약 조건 추가는 실행되지 못 합니다. 하지만 UPDATE문을 통해 TEL 열의 모든 값을 NULL로 수정한 후에 다시 ALTER 문을 실행해 보면 문제없이 실행됩니다. 왜냐하면 UNIQUE는 NULL 중복 여부를 따지지 않기 때문입니다. 그리고 NULL이 아닌 중복 값만 바꾸어도 결과는 같습니다.

### 실습 14-24 TEL 열 값을 모두 NULL 값으로 변경하기

```
01 UPDATE TABLE_UNIQUE  
02     SET TEL = NULL;  
  
03 SELECT * FROM TABLE_UNIQUE;
```

:: 결과 화면

LOGIN_ID	LOGIN_PWD	TEL
TEST_ID_01	PWD01	
TEST_ID_02	PWD01	
	PWD01	

### 실습 14-25 TEL 값에 UNIQUE 제약 조건 설정하기

```
01 ALTER TABLE TABLE_UNIQUE  
02 MODIFY(TEL UNIQUE);
```

:: 결과 화면

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SCOTT	SYS_C0015610	C	TABLE_UNIQUE
SCOTT	SYS_C0015611	U	TABLE_UNIQUE
SCOTT	SYS_C0015616	U	TABLE_UNIQUE
SCOTT	TBLUNQ2_LGNPW_NN	C	TABLE_UNIQUE2
SCOTT	TBLUNQ2_LGNID_UNQ	U	TABLE_UNIQUE2



CONSTRAINT\_TYPE에  
U가 설정되어 있습니다.  
U는 UNIQUE 제약 조건을  
의미합니다.

② 제약 조건을 확인할 때는 실습 14-22를 다시 한번 실행해 보세요.

생성한 테이블에 제약 조건 이름 직접 지정하거나 바꾸기

UNIQUE 제약 조건 역시 이름을 직접 지정할 수 있으며 이후에 이름을 바꿀 수도 있습니다.

### 실습 14-26 UNIQUE 제약 조건 이름 직접 지정하기

```
01 ALTER TABLE TABLE_UNIQUE2  
02 MODIFY(TEL CONSTRAINT TBLUNQ_TEL_UNQ UNIQUE);  
  
03 SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME  
04   FROM USER_CONSTRAINTS  
05 WHERE TABLE_NAME LIKE 'TABLE_UNIQUE%';
```

:: 결과 화면

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SCOTT	SYS_C0015610	C	TABLE_UNIQUE
SCOTT	SYS_C0015611	U	TABLE_UNIQUE
SCOTT	SYS_C0015616	U	TABLE_UNIQUE
SCOTT	TBLUNQ2_LGNPW_NN	C	TABLE_UNIQUE2
SCOTT	TBLUNQ2_LGNID_UNQ	U	TABLE_UNIQUE2
SCOTT	TBLUNQ_TEL_UNQ	U	TABLE_UNIQUE2

### 실습 14-27 이미 만들어져 있는 UNIQUE 제약 조건 이름 수정하기

```
01 ALTER TABLE TABLE_UNIQUE2  
02 RENAME CONSTRAINT TBLUNQ_TEL_UNQ TO TBLUNQ2_TEL_UNQ;  
  
03 SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME  
04   FROM USER_CONSTRAINTS  
05 WHERE TABLE_NAME LIKE 'TABLE_UNIQUE%';
```

:: 결과 화면

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SCOTT	SYS_C0015610	C	TABLE_UNIQUE
SCOTT	SYS_C0015611	U	TABLE_UNIQUE
SCOTT	SYS_C0015616	U	TABLE_UNIQUE
SCOTT	TBLUNQ2_LGNPW_NN	C	TABLE_UNIQUE2
SCOTT	TBLUNQ2_LGNID_UNQ	U	TABLE_UNIQUE2
SCOTT	TBLUNQ2_TEL_UNQ	U	TABLE_UNIQUE2



CONSTRAINT\_NAME이  
TBLUNQ\_TEL\_UNQ에서  
TBLUNQ2\_TEL\_UNQ로 수  
정되었습니다.

제약 조건 삭제

UNIQUE 제약 조건 삭제 역시 ALTER 명령어에 DROP CONSTRAINT 키워드를 사용합니다.

#### 실습 14-28 제약 조건 삭제하기

```
01 ALTER TABLE TABLE_UNIQUE2  
02 DROP CONSTRAINT TBLUNQ2_TEL_UNQ;  
  
03 SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME  
04 FROM USER_CONSTRAINTS  
05 WHERE TABLE_NAME LIKE 'TABLE_UNIQUE%';
```

TBLUNQ2\_TEL\_UNQ 제약 조건이 삭제되었음을 알 수 있습니다.

:: 결과 화면

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SCOTT	SYS_C001128	C	TABLE_UNIQUE
SCOTT	SYS_C001129	U	TABLE_UNIQUE
SCOTT	SYS_C001133	U	TABLE_UNIQUE
SCOTT	TBLUNQ2_LGNP_WN	C	TABLE_UNIQUE2
SCOTT	TBLUNQ2_LGNID_UNQ	U	TABLE_UNIQUE2

1부

이미 존재하는 테이블의 특정 열에 UNIQUE 제약 조건을 추가하는 다음 코드의 빈칸을 채워 보세요.

1 테이블 이름  
2 (열 이름 3 );

1. ALTER TABLE 2. MODIFY 3. UNIQUE

## 14-4 유일하게 하나만 있는 값 PRIMARY KEY

PRIMARY KEY 제약 조건은 UNIQUE와 NOT NULL 제약 조건의 특성을 모두 가지는 제약 조건입니다. 즉 데이터 중복을 허용하지 않고 NULL도 허용하지 않습니다. NULL이 아닌 유일한 값을 가지므로 주민등록번호나 EMP 테이블의 사원 번호같이 테이블의 각 행을 식별하는 데 활용됩니다. PRIMARY KEY 제약 조건은 테이블에 하나밖에 지정할 수 없습니다. 그리고 특정 열을 PRIMARY KEY로 지정하면 해당 열에는 자동으로 인덱스가 만들어집니다.

☞ 하지만 PRIMARY KEY로 적합한 특성을 가졌다 할지라도 주민등록번호와 같은 예민한 개인 정보를 의미하는 데이터는 PRIMARY KEY로 지정하지 않습니다.

### 테이블을 생성하며 제약 조건 지정하기

테이블 PRIMARY KEY(기본키) 제약 조건은 앞에서 살펴본 제약 조건처럼 CREATE문으로 테이블을 생성하면서 지정할 수 있습니다.

실습 14-29 테이블을 생성할 때 특정 열에 PRIMARY KEY 설정하기

```
01 CREATE TABLE TABLE_PK(  
02     LOGIN_ID VARCHAR2(20) PRIMARY KEY,  
03     LOGIN_PWD VARCHAR2(20) NOT NULL,  
04     TEL        VARCHAR2(20)  
05 );  
  
06 DESC TABLE_PK;
```

:: 결과 화면

Column Name	ID	PK	Index Pos	Null?	Data Type	Default	Histogram	Num Distinct	Num Nulls	Density
▶ LOGIN_ID	1	1	1	N	VARCHAR2 (20 Byte)	None				
LOGIN_PWD	2			N	VARCHAR2 (20 Byte)	None				
TEL	3			Y	VARCHAR2 (20 Byte)	None				

테이블을 만들었으니 USER\_CONSTRAINTS 데이터 사전도 확인해 볼까요?

### 실습 14-30 생성한 PRIMARY KEY 확인하기

```
01 SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME  
02   FROM USER_CONSTRAINTS  
03 WHERE TABLE_NAME LIKE 'TABLE_PK%';
```

:: 결과 화면

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SCOTT	SYS_C0015692	C	TABLE_PK
SCOTT	SYS_C0015693	P	TABLE_PK



CONSTRAINT\_TYPE에 P가 설정되어 있습니다.  
P는 PRIMARY KEY 제약 조건을 의미합니다.

PRIMARY KEY 제약 조건은 특정 테이블의 데이터를 식별하는 유일한 값이라는 뜻입니다. 이 때문에 SELECT문을 통한 검색에 자주 활용되므로 PRIMARY KEY 제약 조건을 지정한 열에는 자동으로 인덱스가 만들어진다는 것을 기억하세요. 다음은 USER\_INDEXES 데이터 사전을 조회한 결과입니다.

### 실습 14-31 생성한 PRIMARY KEY를 통해 자동 생성된 INDEX 확인하기

```
01 SELECT INDEX_NAME, TABLE_OWNER, TABLE_NAME  
02   FROM USER_INDEXES  
03 WHERE TABLE_NAME LIKE 'TABLE_PK%';
```

:: 결과 화면

INDEX_NAME	TABLE_OWNER	TABLE_NAME
SYS_C0015693	SCOTT	TABLE_PK

USER\_INDEXES를 통해 TABLE\_PK 테이블에 인덱스가 생성되었음을 알 수 있습니다.

## 테이블을 생성하며 제약 조건 이름 직접 지정하기

다른 제약 조건과 마찬가지로 PRIMARY KEY 역시 제약 조건의 이름을 직접 지정할 수 있습니다.

### 실습 14-32 제약 조건의 이름을 직접 지정하여 테이블 생성하기

```
01 CREATE TABLE TABLE_PK2(  
02     LOGIN_ID VARCHAR2(20) CONSTRAINT TBLPK2_LGNID_PK PRIMARY KEY,  
03     LOGIN_PWD VARCHAR2(20) CONSTRAINT TBLPK2_LGNPWD_NN NOT NULL,  
04     TEL        VARCHAR2(20)  
05 );  
  
06 DESC TABLE_PK2;
```

:: 결과 화면

Column Name	ID	PK	Index Pos	Null?	Data Type	Default	Histogram	Num Distinct	Num Nulls	Density
▶ LOGIN_ID	1	1		N	VARCHAR2 (20 Byte)		None			
LOGIN_PWD	2			N	VARCHAR2 (20 Byte)		None			
TEL	3			Y	VARCHAR2 (20 Byte)		None			

오른쪽 화면은 실습 14-32에서 지정한 제약 조건을 확인하기 위해 실습 14-30을 다시 실행한 결과입니다.

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
▶ SCOTT	SYS_C0015692	C	TABLE_PK
SCOTT	SYS_C0015693	P	TABLE_PK
SCOTT	TBLPK2_LGNPW_NN	C	TABLE_PK2
SCOTT	TBLPK2_LGNID_PK	P	TABLE_PK2

오른쪽 아래 화면의 PRIMARY KEY 제약 조건 지정으로 자동 생성된 인덱스를 확인해 보세요(실습 14-31을 다시 실행한 결과). 그리고 제약 조건과 같은 이름이 붙여진 것도 눈여겨보세요. 사실 실습 14-29에서 제약 조건의 이름을 지정하지 않았을 때도 오라클이 자동으로 생성한 이름이 인덱스(INDEX\_NAME)에 사용되었음을 알 수 있습니다.

INDEX_NAME	TABLE_OWNER	TABLE_NAME
▶ TBLPK2_LGNID_PK	SCOTT	TABLE_PK2
SYS_C0015693	SCOTT	TABLE_PK

### PRIMARY KEY 제약 조건을 지정한 열 확인(중복 값을 입력했을 때)

PRIMARY KEY 제약 조건을 지정한 열에는 중복 값과 NULL이 허용되지 않습니다. 그러면 다음 데이터를 입력한 후 똑같은 값을 입력했을 때 중복 값의 허용 여부를 살펴보겠습니다.

실습 14-33 TABLE\_PK 테이블에 데이터 입력하기

```

01  INSERT INTO TABLE_PK(LOGIN_ID, LOGIN_PWD, TEL)
02  VALUES('TEST_ID_01', 'PWD01', '010-1234-5678');

03  SELECT * FROM TABLE_PK;

```

:: 결과 화면

LOGIN_ID	LOGIN_PWD	TEL
▶ TEST_ID_01	PWD01	010-1234-5678

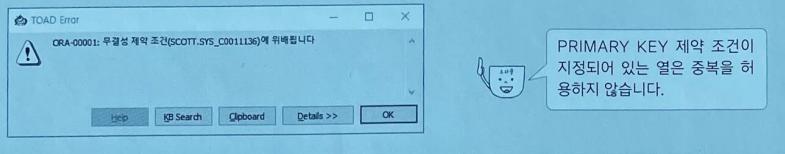
실습 14-34 TABLE\_PK 테이블에 중복되는 데이터 입력하기

```

01  INSERT INTO TABLE_PK(LOGIN_ID, LOGIN_PWD, TEL)
02  VALUES('TEST_ID_01', 'PWD02', '010-2345-6789');

```

:: 결과 화면



## PRIMARY KEY 제약 조건을 지정한 열 확인(NULL 값을 입력했을 때)

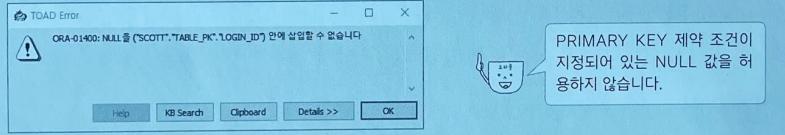
실습 14-35 | NULL 값을 명시적으로 입력하기

```
01  INSERT INTO TABLE_PK(LOGIN_ID, LOGIN_PWD, TEL)
02  VALUES(NULL, 'PWD02', '010-2345-6789');
```

실습 14-36 | NULL 값을 암시적으로 입력하기

```
01  INSERT INTO TABLE_PK(LOGIN_PWD, TEL)
02  VALUES('PWD02', '010-2345-6789');
```

:: 결과 화면(실습 14-35, 14-36의 실행 결과가 같음)



여기에서는 사용 방법을 생략했지만 PRIMARY KEY 제약 조건 역시 ALTER문의 MODIFY, RENAME, DROP을 통해 추가·수정·이름 변경·삭제 등의 수행이 가능합니다. PRIMARY KEY 제약 조건은 테이블 데이터를 식별하는 유일한 값을 뜻하므로 일반적으로 테이블의 생성 시점에 확정되는 경우가 대부분입니다. 즉 ALTER문을 사용하는 경우가 드물죠. 그리고 테이블에 이미 PRIMARY KEY 제약 조건이 지정되어 있다면 다른 열에는 추가할 수 없습니다. 또 PRIMARY KEY 제약 조건을 지정하려는 열에 중복 값이나 NULL이 있을 경우에도 동작하지 않는다는 점을 기억해 주세요.

---

## ❶ 한 번 더 나가기! CREATE문에서 제약 조건을 지정하는 다른 방식

CREATE문을 통해 제약 조건을 지정할 때 다음과 같이 열 바로 옆에 제약 조건을 지정하는 형식을 사용해 왔는데요. 이를 인라인(inline) 또는 열 레벨(column-level) 제약 조건 정의라고 합니다. 이 책에서 소개할 모든 제약 조건을 이 방식으로 지정할 수 있습니다.

```
CREATE TABLE TABLE_NAME(
    COL1 VARCHAR2(20) CONSTRAINT CONSTRAINT_NAME PRIMARY KEY,      -- 이름 지정함
    COL2 VARCHAR2(20) NOT NULL,                                     -- 이름 지정하지 않음
    COL3 VARCHAR2(20)
);
```

이와 달리 열을 정의한 후에 별도로 제약 조건을 정의할 수도 있습니다. 다음과 같이 열을 명시한 후 제약 조건을 테이블 단위에 지정하는 방식을 아웃오브라인(out-of-line) 또는 테이블(table-level) 제약 조건 정의라고 합니다. 이 방식은 NOT NULL 제약 조건을 제외한 제약 조건 지정이 가능합니다.

```
CREATE TABLE TABLE_NAME(
    COL1 VARCHAR2(20),
    COL2 VARCHAR2(20),
    COL3 VARCHAR2(20),
    PRIMARY KEY (COL1),                                         -- 이름 지정하지 않음(COL1 열에 PRIMARY KEY 지정)
    CONSTRAINT CONSTRAINT_NAME UNIQUE (COL2)                      -- 이름 지정함(COL2 열에 UNIQUE 지정)
);
```

## 14-5 다른 테이블과 관계를 맺는 FOREIGN KEY

외래키, 외부키로도 부르는 FOREIGN KEY는 서로 다른 테이블 간 관계를 정의하는 데 사용하는 제약 조건입니다. 특정 테이블에서 PRIMARY KEY 제약 조건을 지정한 열을 다른 테이블의 특정 열에서 참조하겠다는 의미로 지정할 수 있습니다. EMP 테이블의 DEPTNO 열이 DEPT 테이블의 DEPTNO 열을 참조하는 것이 그 예입니다. 그러면 FOREIGN KEY에 대해 알아보기 전에 USER\_CONSTRAINTS 데이터 사전에서 EMP 테이블과 DEPT 테이블의 제약 조건을 조회해 보죠.

### 실습 14-37 EMP 테이블과 DEPT 테이블의 제약 조건 살펴보기

```
01  SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME, R_OWNER, R_CONSTRAINT_NAME  
02    FROM USER_CONSTRAINTS  
03   WHERE TABLE_NAME IN ('EMP', 'DEPT');
```

CONSTRAINT\_TYPE 열 값이 R일 경우 외래키를 의미하며 R\_CONSTRAINT\_NAME의 PK\_DEPT는 DEPT 테이블의 PRIMARY KEY, 즉 DEPT 테이블의 DEPTNO 열을 참조한다는 뜻입니다.

#### :: 결과 화면

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	R_OWNER	R_CONSTRAINT_NAME
SCOTT	PK_DEPT	P	DEPT		
SCOTT	PK_EMP	P	EMP		
SCOTT	FK_DEPTNO	(R)	EMP	SCOTT	PK_DEPT

EMP 테이블의 DEPTNO 열은 다음과 같이 DEPT 테이블의 DEPTNO 열을 참조하여 저장 값의 범위를 정합니다. 이렇게 참조 관계를 정의하면 EMP 테이블의 DEPTNO 열에는 DEPT 테이블의 DEPTNO 열에 존재하는 값과 NULL만 저장할 수 있게 됩니다. 현재 DEPT 테이블의 DEPTNO 열에 저장되어 있는 값을 생각해 본다면 10, 20, 30, 40 그리고 NULL 외의 값은 저장이 불가능한 것이죠.

The diagram illustrates two tables: EMP and DEPT. The EMP table contains 14 rows of employee data, including columns for EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, and DEPTNO. The DEPT table contains 4 rows of department data, including columns for DEPTNO, DNAME, and LOC. An arrow points from the DEPTNO column in the EMP table to the DEPTNO column in the DEPT table, indicating a relationship between the two.

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶	7369	SMITH	CLERK	7902	1980-12-17	800		20
	7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30
	7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30
	7566	JONES	MANAGER	7839	1981-04-02	2975		20
	7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30
	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
	7782	CLARK	MANAGER	7839	1981-06-09	2450		10
	7788	SCOTT	ANALYST	7566	1987-04-19	3000		20
	7839	KING	PRESIDENT		1981-11-17	5000		10
	7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30
	7876	ADAMS	CLERK	7788	1987-05-23	1100		20
	7900	JAMES	CLERK	7698	1981-12-03	950		30
	7902	FORD	ANALYST	7566	1981-12-03	3000		20
	7934	MILLER	CLERK	7782	1982-01-23	1300		10

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

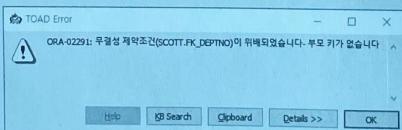
따라서 다음과 같이 EMP 테이블에 DEPTNO 열 값을 50으로 지정하고 INSERT문을 사용하면 오류가 나서 실행되지 않습니다. 50은 DEPT 테이블의 DEPTNO 열에 저장되어 있지 않은 값이니까요.

#### 실습 14-38 FOREIGN KEY가 참조하는 열에 존재하지 않는 데이터 입력하기

```
01  INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
02  VALUES(9999, '홍길동', 'CLERK', '7788', TO_DATE('2017/04/30', 'YYYY/MM/DD'), 1200, NULL, 50);
```

오류 메시지에 언급한 ‘부모 키가 없습니다’라는 말은 DEPT 테이블의 DEPTNO 열에 50이 존재하지 않는다는 뜻입니다. 참조 대상 테이블을 부모, 참조하는 테이블을 자식으로 표현한다는 것도 기억해 주세요.

#### :: 결과 화면



DEPT 테이블이 부모, EMP 테이블이 자식 테이블이 됩니다.

## FOREIGN KEY 지정하기

FOREIGN KEY 지정은 지금까지 살펴본 제약 조건을 지정하는 방법과 비슷합니다. 참조 대상을 지정하는 문법을 추가하여 다음과 같이 작성합니다.

```
CREATE TABLE 테이블 이름(  
...  
...  
);
```

기본 형식

오른쪽과 같이 제약 조건 이름을 지정하지 않고 FOREIGN KEY를 정의할 수 있습니다.

```
CREATE TABLE 테이블 이름(  
...  
...  
);
```

열을 모두 정의한 후 제약 조건을 지정하려면 오른쪽과 같이 마지막에 CONSTRAINT 키워드를 사용하면 됩니다.

```
CREATE TABLE 테이블 이름(  
...  
...  
CONSTRAINT [제약 조건 이름] FOREIGN KEY(열)  
REFERENCES 참조 테이블(참조할 열)  
);
```

그러면 EMP 및 DEPT 테이블과 열 구성이 같은 테이블을 만들어 볼까요? 먼저 참조 대상이 될 DEPT\_FK 테이블을 다음과 같이 생성합시다.

### 실습 14-39 DEPT\_FK 테이블 생성하기

```
01 CREATE TABLE DEPT_FK(  
02   DEPTNO NUMBER(2) CONSTRAINT DEPTFK_DEPTNO_PK PRIMARY KEY,  
03   DNAME  VARCHAR2(14),  
04   LOC    VARCHAR2(13)  
05 );  
  
06 DESC DEPT_FK;
```

:: 결과 화면(일부 열만 표시함)

Column Name	ID	PK	Index Pos	Null?	Data Type	Default	Histogram	Num Distinct	Num Nulls	Density
DEPTNO	1	1		N	NUMBER (2)		None			
DNAME	2			Y	VARCHAR2 (14 Byte)		None			
LOC	3			Y	VARCHAR2 (13 Byte)		None			

이제 DEPT\_FK 테이블의 DEPTNO 열을 참조하는 FOREIGN KEY 제약 조건을 정의한 EMP 테이블을 만들어 봅시다.

#### 실습 14-40 EMP\_FK 테이블 생성하기

```
01 CREATE TABLE EMP_FK(
02     EMPNO      NUMBER(4) CONSTRAINT EMPFK_EMPNO_PK PRIMARY KEY,
03     ENAME      VARCHAR2(10),
04     JOB        VARCHAR2(9),
05     MGR        NUMBER(4),
06     HIREDATE   DATE,
07     SAL         NUMBER(7,2),
08     COMM        NUMBER(7,2),
09     DEPTNO     NUMBER(2) CONSTRAINT EMPFK_DEPTNO_FK REFERENCES DEPT_FK (DEPTNO)
10 );
11 DESC EMP_FK;
```

:: 결과 확인(일부 열만 표시함)

Column Name	ID	PK	Index Pos	Null?	Data Type	Default	Histogram	Num Distinct	Num Nulls	Density
EMPNO	1	1	1	N	NUMBER (4)	None				
ENAME	2			Y	VARCHAR2 (10 Byte)	None				
JOB	3			Y	VARCHAR2 (9 Byte)	None				
MGR	4			Y	NUMBER (4)	None				
HIREDATE	5			Y	DATE	None				
SAL	6			Y	NUMBER (7,2)	None				
COMM	7			Y	NUMBER (7,2)	None				
DEPTNO	8			Y	NUMBER (2)	None				

EMP\_FK 테이블의 DEPTNO 열은 이제 DEPT\_FK 테이블의 DEPTNO 열을 참조하는 FOREIGN KEY 제약 조건이 지정되었습니다.

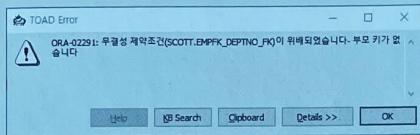
#### FOREIGN KEY 지정할 때 유의점

테이블을 만들고 나서 DEPT\_FK 테이블에는 데이터가 아직 없는 상태입니다. 이로 인해 EMP\_FK 테이블에 데이터를 추가할 때 부서 번호(DEPTNO)를 지정하면 오류가 나서 실행되지 않습니다. 앞에서 살펴본 대로 EMP\_FK 테이블의 DEPTNO 열은 DEPT\_FK 테이블의 DEPTNO를 참조하기 때문에 DEPT\_FK 테이블의 DEPTNO 열에 존재하지 않는 값을 사용하는 것은 불가능합니다.

#### 실습 14-41 | EMP\_FK 테이블에 데이터 삽입하기(DEPTNO 데이터가 아직 DEPT\_FK 테이블에 없을 때)

```
01 INSERT INTO EMP_FK  
02 VALUES(9999, 'TEST_NNAME', 'TEST_JOB', NULL, TO_DATE('2001/01/01', 'YYYY/MM/DD'),  
3000, NULL, 10);
```

:: 결과 화면



DEPT\_FK 테이블에 다음과 같이 데이터를 삽입한 후 다시 EMP\_FK에 데이터를 삽입해 볼까요? 먼저 DEPT\_FK에 10번 부서 데이터를 삽입합니다.

#### 실습 14-42 | DEPT\_FK에 데이터 삽입하기

```
01 INSERT INTO DEPT_FK  
02 VALUES(10, 'TEST_DNAME', 'TEST_LOC');  
  
03 SELECT * FROM DEPT_FK;
```

:: 결과 화면

	DEPTNO	DNAME	LOC
▶	10	TEST_DNAME	TEST_LOC

그리고 실습 14-41에서 실행에 실패한 EMP\_FK 테이블의 INSERT문을 다시 실행해 보면 이제 정상적으로 실행되는 것을 알 수 있습니다.

#### 실습 14-43 | EMP\_FK 테이블에 데이터 삽입하기

```
01 INSERT INTO EMP_FK  
02 VALUES(9999, 'TEST_NNAME', 'TEST_JOB', NULL, TO_DATE('2001/01/01', 'YYYY/MM/DD'),  
3000, NULL, 10);  
  
03 SELECT * FROM EMP_FK;
```

:: 결과 화면

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶	9999	TEST_NNAME	TEST_JOB		2001-01-01	3000		10

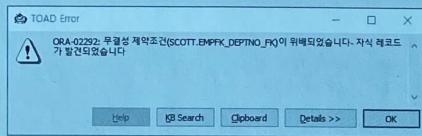
## FOREIGN KEY로 참조 행 데이터 삭제하기

현재 DEPT\_FK 테이블에는 10번 부서 데이터가 저장되어 있고 EMP\_FK 테이블에는 이 10번 부서를 참조하는 데이터가 있습니다. 이 경우에 DEPT\_FK 테이블의 DEPTNO 열에 저장된 10번 부서 데이터는 삭제할 수 없습니다.

### 실습 14-44 DEPT\_FK 테이블의 10번 부서 데이터 삭제하기

```
01  DELETE FROM DEPT_FK  
02  WHERE DEPTNO = 10;
```

:: 결과 화면



오류가 발생하는 이유는 자식 레코드, 즉 삭제하려는 DEPTNO 값을 참조하는 데이터가 존재하기 때문입니다. DEPT\_FK 테이블의 데이터를 삭제하려면 다음 방법 중 한 가지를 사용해야 합니다.

1. 현재 삭제하려는 열 값을 참조하는 데이터를 먼저 삭제한다.

ex) EMP\_FK 테이블의 DEPTNO가 10번인 데이터를 삭제한 후 DEPT\_FK 테이블의 10번 부서 삭제

2. 현재 삭제하려는 열 값을 참조하는 데이터를 수정한다.

ex) EMP\_FK 테이블의 DEPTNO가 10번인 데이터를 다른 부서 번호 또는 NULL로 변경한 후 DEPT\_FK 테이블의 10번 부서 삭제

3. 현재 삭제하려는 열을 참조하는 자식 테이블의 FOREIGN KEY 제약 조건을 해제한다.

하지만 위 방법은 삭제할 데이터를 참조하는 데이터의 수정 또는 삭제 작업을 선행해야 하므로 다소 귀찮은 작업이 될 것입니다. 그리고 FOREIGN KEY 제약 조건을 해제할 수 없는 경우도 종종 있으므로 이미 제약 조건으로 연결된 데이터의 삭제는 꽤나 까다로운 일입니다. 따라서 제약 조건을 처음 지정할 때 다음과 같이 추가 옵션을 지정하는 방법을 사용하기도 합니다. 이 방법은 데이터 삭제와 더불어 삭제할 데이터를 참조하는 처리를 어떻게 할지 정할 수 있습니다.

열 데이터를 삭제할 때 이 데이터를 참조하고 있는 데이터도 함께 삭제

CONSTRAINT [제약 조건 이름] REFERENCES 참조 테이블(참조할 열) ON DELETE CASCADE

기본 형식

DEPT\_FK 테이블의 DEPTNO 열 값이 10인 데이터를 삭제하면 이를 참조하는 EMP\_FK 테이블의 DEPTNO 열 값이 10인 데이터도 함께 삭제합니다.

DEPT\_FK 테이블

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

10번 부서를 삭제할 경우

EMP\_FK 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17	800		20
7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30
7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30
7566	JONES	MANAGER	7839	1981-04-02	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-06-09	2450		10
7788	SCOTT	ANALYST	7566	1987-04-19	3000		20
7839	KING	PRESIDENT		1981-11-17	5000		10
7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30
7876	ADAMS	CLERK	7788	1987-05-23	1100		20
7900	JAMES	CLERK	7698	1981-12-03	950		30
7902	FORD	ANALYST	7566	1981-12-03	3000		20
7934	MILLER	CLERK	7782	1982-01-23	1300		10

10번 부서에 속한 사원 데이터 함께 삭제

열 데이터를 삭제할 때 이 데이터를 참조하는 데이터를 NULL로 수정

CONSTRAINT [제약 조건 이름] REFERENCES 참조 테이블(참조할 열) ON DELETE SET NULL

기본 형식

DEPT\_FK 테이블의 DEPTNO 열 값이 10인 데이터를 삭제하면 이를 참조하는 EMP\_FK 테이블의 DEPTNO 열 값이 10인 데이터를 NULL로 수정합니다.

DEPT\_FK 테이블

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

10번 부서를 삭제할 경우

EMP\_FK 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17	800		20
7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30
7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30
7566	JONES	MANAGER	7839	1981-04-02	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-06-09	2450		10
7788	SCOTT	ANALYST	7566	1987-04-19	3000		20
7839	KING	PRESIDENT		1981-11-17	5000		10
7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30
7876	ADAMS	CLERK	7788	1987-05-23	1100		20
7900	JAMES	CLERK	7698	1981-12-03	950		30
7902	FORD	ANALYST	7566	1981-12-03	3000		20
7934	MILLER	CLERK	7782	1982-01-23	1300		10

10번 부서에 속한 사원의 DEPTNO 열 값을 NULL로 변경

참조 데이터를 지정하는 FOREIGN KEY 제약 조건도 PRIMARY KEY 제약 조건과 마찬가지로 테이블을 설계하는 시점에 결정이 나는 경우가 많습니다. 따라서 ALTER문을 사용한 제약 조건의 추가·변경·삭제 등 여러 기능을 수행할 수 있지만 이 책에서는 많이 쓰이는 CREATE문 사용법만 소개합니다. 좀 더 자세한 사용 방법을 알고 싶다면 오라클 공식 문서 ([docs.oracle.com/cd/E11882\\_01/server.112/e41084/clauses002.htm#SQLRF52180](http://docs.oracle.com/cd/E11882_01/server.112/e41084/clauses002.htm#SQLRF52180))를 참고하세요.

1분  
복습

다음 빈칸을 채우며 복습해 보세요.

<sup>1</sup> 제 은 어떤 테이블에 저장할 데이터 특성을 정의하는 데 사용하는 특수한 규칙을 뜻합니다.  
오라클 데이터베이스에서 사용할 수 있는 <sup>1</sup> 제 은 다섯 가지가 있는데요. 지정한 열에 NULL 값을 제외한 값의 중복이 불가능한 <sup>2</sup> U , NULL을 허용하지 않는 <sup>3</sup> N , 다른 테이블의 열을 참조하는 <sup>4</sup> F KEY, NULL 값과 데이터의 중복을 모두 허용하지 않는 <sup>5</sup> P KEY가 있습니다.

정답 1. NOT NULL 2. UNIQUE 3. NOT NULL 4. FOREIGN 5. PRIMARY

## 14-6 데이터 형태와 범위를 정하는 CHECK

CHECK 제약 조건은 열에 저장할 수 있는 값의 범위 또는 패턴을 정의할 때 사용합니다. 예를 들어 시간을 저장할 열 데이터는 0에서 23까지의 숫자만 허용합니다. CHECK 제약 조건 역시 다른 제약 조건과 마찬가지로 지정할 수 있습니다. 다음 CREATE문으로 LOGIN\_PWD 열에 이름을 직접 입력하여 CHECK 제약 조건을 지정해 보겠습니다.

### 실습 14-45 테이블을 생성할 때 CHECK 제약 조건 설정하기

```
01 CREATE TABLE TABLE_CHECK(
02     LOGIN_ID  VARCHAR2(20) CONSTRAINT TBLCK_LOGINID_PK PRIMARY KEY,
03     LOGIN_PWD VARCHAR2(20) CONSTRAINT TBLCK_LOGINPW_CK CHECK (LENGTH(LOGIN_PWD) > 3),
04     TEL       VARCHAR2(20)
05 );
06 DESC TABLE_CHECK;
```

CHECK 키워드 다음의 LENGTH(LOGIN\_PWD) > 3은 LOGIN\_PWD 열 길이가 3 이상인 데이터만 저장 가능하다는 뜻입니다. 즉 비밀번호는 3글자 이상만 저장할 수 있도록 제한을 둔 것입니다.

② CHECK 제약 조건은 단순 연산뿐만 아니라 함수 활용도 가능합니다.

#### :: 결과 화면(일부 열만 표시함)

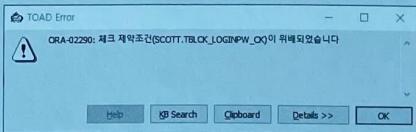
Column Name	ID	PK	Index Pos	Null?	Data Type	Default	Histogram	Num Distinct	Num Nulls	Density
LOGIN_ID	1	1	1	N	VARCHAR2 (20 Byte)		None			
LOGIN_PWD	2			Y	VARCHAR2 (20 Byte)		None			
TEL	3			Y	VARCHAR2 (20 Byte)		None			

지정한 CHECK 제약 조건 때문에 다음 INSERT문은 실행되지 않습니다. 왜냐하면 비밀번호가 CHECK 제약 조건에서 지정한 3자리를 넘지 않기 때문입니다.

### 실습 14-46 CHECK 제약 조건에 맞지 않는 예

```
01 INSERT INTO TABLE_CHECK
02     VALUES ('TEST_ID', '123', '010-1234-5678');
```

:: 결과 화면



하지만 INSERT문에 비밀번호를 4자리로 지정하면 오류 없이 실행됩니다.

#### 실습 14-47 CHECK 제약 조건에 맞는 예

```
01  INSERT INTO TABLE_CHECK  
02  VALUES ('TEST_ID', '1234', '010-1234-5678');  
  
03  SELECT * FROM TABLE_CHECK;
```

:: 결과 화면

	LOGIN_ID	LOGIN_PWD	TEL
▶	TEST_ID	1234	010-1234-5678

CHECK 제약 조건은 USER\_CONSTRAINTS 데이터 사전에서 확인할 수 있습니다. CONSTRAINT\_TYPE 열 값이 C이므로 NOT NULL, CHECK 제약 조건은 모두 C로 출력됩니다.

#### 실습 14-48 CHECK 제약 조건 확인하기

```
01  SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME  
02    FROM USER_CONSTRAINTS  
03   WHERE TABLE_NAME LIKE 'TABLE_CHECK';
```

:: 결과 화면

	OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
▶	SCOTT	TBLCK_LOGINPW_CK	C	TABLE_CHECK
	SCOTT	TBLCK_LOGINID_PK	P	TABLE_CHECK

## 14-7 기본값을 정하는 DEFAULT

제약 조건과는 별개로 특정 열에 저장할 값이 지정되지 않았을 경우에 기본값(default)을 지정 할 수 있는데요. 이때 사용되는 키워드가 DEFAULT입니다.

### 실습 14-49 테이블을 생성할 때 DEFAULT 제약 조건 설정하기

```
01 CREATE TABLE TABLE_DEFAULT(
02     LOGIN_ID VARCHAR2(20) CONSTRAINT TBLCK2_LOGINID_PK PRIMARY KEY,
03     LOGIN_PWD VARCHAR2(20) DEFAULT '1234',
04     TEL      VARCHAR2(20)
05 );
06 DESC TABLE_DEFAULT;
```

:: 결과 화면

Column Name	ID	PK	Index Pos	Null?	Data Type	Default	Histogram
▶ LOGIN_ID	1	1		N	VARCHAR2 (20 Byte)		None
LOGIN_PWD	2			Y	VARCHAR2 (20 Byte)	'1234'	None
TEL	3			Y	VARCHAR2 (20 Byte)		None

일부 열만 표시했습니다.

이제 다음 두 INSERT문을 실행하여 지정한 기본값이 잘 들어가는지 확인해 보죠.

### 실습 14-50 DEFAULT로 지정한 기본값이 입력되는 INSERT문 확인하기

```
01 INSERT INTO TABLE_DEFAULT VALUES ('TEST_ID', NULL, '010-1234-5678');
02 INSERT INTO TABLE_DEFAULT (LOGIN_ID, TEL) VALUES ('TEST_ID2', '010-1234-5678');
03 SELECT * FROM TABLE_DEFAULT;
```

:: 결과 화면

	LOGIN_ID	LOGIN_PWD	TEL
▶ TEST_ID		010-1234-5678	
TEST_ID2	1234	010-1234-5678	

☞ INSERT문도 SELECT문도 각각 실행합니다.

명시적으로 NULL을 지정한 첫 번째 INSERT문을 실행했을 때는 LOGIN\_PWD 열이 비어 있습니다. 그리고 두 번째 INSERT문처럼 LOGIN\_PWD 열 값을 지정하지 않으면 기본값인 1234가 들어가는 것을 확인할 수 있습니다.

### ▣ 한 발 더 나가기!! 제약 조건 비활성화, 활성화

제약 조건은 데이터 무결성을 보장하는 데 중요한 역할을 합니다. 하지만 신규 기능 개발 또는 테스트 같은 특정 업무를 수행해야 할 때 제약 조건이 걸림돌이 되는 경우가 종종 생깁니다. 이때 여러 필요에 의해 제약 조건을 비활성화하거나 비활성화되어 있는 제약 조건을 다시 활성화할 수 있습니다. 비활성화에는 DISABLE절을, 활성화에는 ENABLE절을 다음과 같이 사용합니다.

```
ALTER TABLE 테이블 이름  
DISABLE [NOVALIDATE / VALIDATE(선택)] CONSTRAINT 제약조건이름;
```

```
ALTER TABLE 테이블 이름  
ENABLE [NOVALIDATE / VALIDATE(선택)] CONSTRAINT 제약조건이름;
```

제약 조건의 비활성화와 활성화는 이 책에서 따로 다루지 않지만 제약 조건의 제한을 일시적으로 풀어 주는 방법이 존재한다는 점을 기억해 두면 도움이 됩니다. 좀 더 자세한 내용은 오라클 공식 문서 ([docs.oracle.com/cd/E11882\\_01/server.112/e41084/clauses002.htm#SQLRF52180](http://docs.oracle.com/cd/E11882_01/server.112/e41084/clauses002.htm#SQLRF52180))를 참고하세요.

**Q1** DEPT\_CONST 테이블과 EMP\_CONST 테이블을 다음과 같은 특성 및 제약 조건을 지정하여 만들어 보세요.

① DEPT\_CONST 테이블

열 이름	자료형	길이	제약 조건	제약 조건 이름
DEPTNO	정수형 숫자	2	PRIMARY KEY	DEPTCONST_DEPTNO_PK
DNAME	가변형 문자열	14	UNIQUE	DEPTCONST_DNAME_UNQ
LOC	가변형 문자열	13	NOT NULL	DEPTCONST_LOC_NN

② EMP\_CONST 테이블

열 이름	자료형	길이	제약 조건	제약 조건 이름
EMPNO	정수형 숫자	4	PRIMARY KEY	EMPCONST_EMPNO_PK
ENAME	가변형 문자열	10	NOT NULL	EMPCONST_ENAME_NN
JOB	가변형 문자열	9	-	-
TEL	가변형 문자열	20	UNIQUE	EMPCONST_TEL_UNQ
HIREDATE	날짜	-	-	-
SAL	소수점 둘째자리 숫자	7	CHECK : 급여는 1000~9999만 입력 가능	EMPCONST_SAL_CHK
COMM	소수점 둘째자리 숫자	7	-	-
DEPTNO	정수형 숫자	2	FOREIGN KEY	EMPCONST_DEPTNO_FK

③ 테이블 생성 후 데이터 사전 뷰를 사용하여 다음과 같이 두 테이블의 제약 조건을 확인해 보세요.

:: 결과 화면

■	TABLE_NAME	CONSTRAINT_NAME	CONSTRAINT_TYPE
▶	DEPT_CONST	DEPTCONST_DEPTNO_PK	P
	DEPT_CONST	DEPTCONST_DNAME_UNQ	U
	DEPT_CONST	DEPTCONST_LOC_NN	C
	EMP_CONST	EMPCONST_DEPTNO_FK	R
	EMP_CONST	EMPCONST_EMPNO_PK	P
	EMP_CONST	EMPCONST_ENAME_NN	C
	EMP_CONST	EMPCONST_SAL_CHK	C
	EMP_CONST	EMPCONST_TEL_UNQ	U

정답 이지스파클리싱  
홈페이지에서 확인하세요.