

MySQL & Maria DB

데이터 무결성을 위한 제약 조건

목차

- ▶ 무결성 제약 조건의 개념과 종류
- ▶ 제약 조건 확인하기
- ▶ 필수 입력을 위한 NOT NULL 제약 조건
- ▶ 유일한 값만 허용하는 UNIQUE 제약 조건
- ▶ 컬럼 레벨로 제약 조건명을 명시해서 제약 조건 설정하기
- ▶ 데이터 구분을 위한 PRIMARY KEY 제약 조건
- ▶ AUTO INCREMENT
- ▶ 참조 무결성을 위한 FOREIGN KEY 제약 조건
- ▶ CHECK 제약 조건
- ▶ DEFAULT 제약 조건
- ▶ 테이블 레벨 방식으로 제약 조건 지정하기
- ▶ 제약 조건 변경하기
- ▶ 제약 조건으로 인한 컬럼 삭제 불가능 예

무결성 제약 조건의 개념과 종류

▶ 무결성 제약 조건의 개념과 종류

- ▶ 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 말합니다.

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.

무결성 제약 조건의 개념과 종류

▶ 제약 조건 확인하기

- ▶ 아래의 그림은 EMP 테이블에 INSERT 작업 중 무결성 제약 조건을 위반했을 때 나타나는 에러 메시지입니다.

```
scott> INSERT INTO DEPT VALUES(10, "TEST", "SEOUL")  
[2021-10-20 13:25:01] [23000][1062] Duplicate entry '10' for key 'PRIMARY'
```

- ▶ DESC 명령어로 확인할 수 있는 제약조건은 다음과 같습니다.

	Field	Type	Null	Key	Default	Extra
1	DEPTNO	int(11)	NO	PRI	<null>	
2	DNAME	varchar(14)	YES		<null>	
3	LOC	varchar(13)	YES		<null>	

제약 조건 확인하기

▶ 제약 조건 확인하기

- ▶ MySQL/MariaDB는 information_schema.table_constraints 데이터 디렉터리 뷰로 제약 조건에 관한 정보를 알려 줍니다.
- ▶ information_schema.table_constraints 데이터 디렉터리 뷰를 조회하면 내가 만든(USER) 제약 조건(CONSTRAINTS)의 정보를 조회 할 수 있습니다.

형식

```
DESC information_schema.table_constraints;
```

제약 조건 확인하기

▶ 제약 조건 확인하기

▶ information_schema.table_constraints 컬럼

열	설명
CONSTRAINT_CATALOG	항상 def.
CONSTRAINT_SCHEMA	제약 조건이 포함된 데이터베이스 이름입니다.
CONSTRAINT_NAME	제약 조건 이름입니다.
TABLE_SCHEMA	데이터베이스 이름.
TABLE_NAME	테이블 이름입니다.
CONSTRAINT_TYPE	제약 유형. UNIQUE, PRIMARY KEY, FOREIGN KEY 또는 CHECK

제약 조건 확인하기

▶ 제약 조건 확인하기

- ▶ scott 계정으로 접속해 있는 상태에서 scott 소유의 테이블에 지정된 제약 조건을 살펴보도록 합시다.
- ▶ information_schema.table_constraints 을 통해 emp 테이블에 걸린 제약 조건의 내용을 살펴봅시다.

형식

```
select * from information_schema.table_constraints where table  
name='emp';
```

	CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE
1	def	scott	PRIMARY	scott	emp	PRIMARY KEY
2	def	scott	PK_EMP	scott	emp	FOREIGN KEY

필수 입력을 위한 NOT NULL 제약 조건

▶ 필수 입력을 위한 NOT NULL 제약 조건

- ▶ 새로운 사원이 입사하여 사원의 정보를 입력하는데 사원번호와 사원 명이 불 분명하여 데이터가 저장되지 않았다면 누구의 직급인지, 누구의 부서번호인지를 모르게 되므로 자료로서의 의미를 갖기 어렵습니다.
- ▶ 따라서 사원의 정보를 입력할 때 반드시 입력해야 하는 선택이 아닌 필수 입력을 요구하는 컬럼이 있다면 위와 같이 NULL 값이 저장되지 못하도록 제약 조건을 설정해야 합니다.
- ▶ 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 사용합니다.

필수 입력을 위한 NOT NULL 제약 조건

- ▶ NOT NULL 제약조건을 설정하지 않고 테이블 생성하기
 - ▶ NOT NULL 제약조건을 학습하지 전에 지금까지 실습에 사용했던 사원 테이블과 유사한 구조의 사원번호, 사원명, 직급, 부서번호 4개의 칼럼으로 구성된 EMP01 테이블에 아무런 제약 조건을 설정하지 않고 생성한 후에 이렇게 생성한 테이블에는 NULL 값이 저장됨을 확인해 봅시다.
 - ▶ 지금까지 실습에 사용했던 사원 테이블과 유사한 구조의 사원번호, 사원명, 직급, 부서번호 4개의 칼럼으로 구성된 EMP01 테이블을 생성해 봅시다.

```
CREATE TABLE EMP01(  
  EMPNO DECIMAL(4),  
  ENAME VARCHAR(10),  
  JOB VARCHAR(9),  
  DEPTNO DECIMAL(2)  
);
```

필수 입력을 위한 NOT NULL 제약 조건

- ▶ NOT NULL 제약조건을 설정하지 않고 테이블 생성하기
 - ▶ 생성된 테이블의 내용을 살펴보면 내용을 갖고 있지 않음을 알 수 있습니다. CREATE TABLE 명령은 테이블을 생성하면서 칼럼과 그 칼럼의 자료 형태 등의 구조를 정의하는 것이지 자료를 입력하는 것이 아니기 때문입니다.
 - ▶ 위에서 생성한 EMP01 테이블에는 데이터를 추가해 봅시다.

```
INSERT INTO EMP01  
VALUES(NULL, NULL, 'SALESMAN', 30);
```

```
SELECT * FROM EMP01;
```

	EMPNO	ENAME	JOB	DEPTNO
1	<null>	<null>	SALESMAN	30

필수 입력을 위한 NOT NULL 제약 조건

- ▶ NOT NULL 제약조건을 설정하지 않고 테이블 생성하기
 - ▶ EMP01 테이블에 사원번호와 사원명에 데이터를 저장하지 않더라도 해당 로우가 테이블에 추가됩니다.
 - ▶ 테이블을 생성하면서 아무런 제약 조건도 주지 않았기 때문입니다. DESC 명령어로도 NOT NULL 제약조건이 설정되어 있지 않음을 확인할 수 있습니다.

DESC EMP01

	Field	Type	Null	Key	Default	Extra
1	EMPNO	decimal(4,0)	YES		<null>	
2	ENAME	varchar(10)	YES		<null>	
3	JOB	varchar(9)	YES		<null>	
4	DEPTNO	decimal(2,0)	YES		<null>	

필수 입력을 위한 NOT NULL 제약 조건

▶ NOT NULL 제약 조건 지정하기

- ▶ NOT NULL 제약 조건을 지정하지 않으면 위 예에서처럼 NULL 값이 저장됩니다.
- ▶ 특정 컬럼에 NULL 값이 저장되지 못하도록 하려면 NOT NULL 제한 조건을 설정해야 합니다.
- ▶ 이제 제약 조건을 설정하는 방법을 살펴봅시다.
- ▶ 제약 조건을 설정하는 방법은 컬럼 레벨과 테이블 레벨 두 가지 방식이 있습니다.

column_name data_type constraint_type

**ALTER TABLE *table_name* MODIFY COLUMN
column_name *column_type* NOT NULL**

필수 입력을 위한 NOT NULL 제약 조건

▶ NOT NULL 제약 조건 지정하기

- ▶ 사원 테이블(EMP02)을 사원번호, 사원명, 직급, 부서번호 4개의 칼럼으로 구성하되 이번에는 사원번호와 사원명에 NOT NULL 조건을 지정하도록 합시다. 제약 조건은 칼럼명과 자료형을 기술한 후에 연이어서 NOT NULL을 기술하면 됩니다.
- ▶ 지금까지 실습에 사용했던 사원 테이블과 유사한 구조의 사원번호, 사원명, 직급, 부서번호 4개의 칼럼으로 구성된 EMP02 테이블을 생성하되 EMPNO와 EMPNAME 컬럼에 NOT NULL 제약 조건 설정해 봅시다.

```
CREATE TABLE EMP02(  
  EMPNO DECIMAL(4),  
  ENAME VARCHAR(10) NOT NULL,  
  JOB VARCHAR(9),  
  DEPTNO DECIMAL(2)  
);
```

필수 입력을 위한 NOT NULL 제약 조건

- ▶ NOT NULL 제약 조건 지정하기

- ▶ 위에서 생성한 EMP02 테이블에는 데이터를 추가해 봅시다.

```
INSERT INTO EMP02  
VALUES(NULL, NULL, 'SALESMAN', 10);
```

```
[23000][1048] Column 'EMPNO' cannot be null
```

- ▶ EMP02 테이블은 사원번호와 사원명에 NOT NULL 조건을 지정하였기에 사원번호에 NULL을 추가하는 명령어에서 오류가 발생합니다.

필수 입력을 위한 NOT NULL 제약 조건

▶ NOT NULL 제약 조건 지정하기

- ▶ 이전에 배웠던 ALTER 명령어를 사용해 테이블에 not null 제약조건을 추가해 봅시다.

```
ALTER TABLE EMP02 MODIFY COLUMN EMPNO  
DECIMAL(4) NOT NULL
```

- ▶ DESC 명령어로 NOT NULL 제약조건이 설정되어 있음을 확인할 수 있습니다.

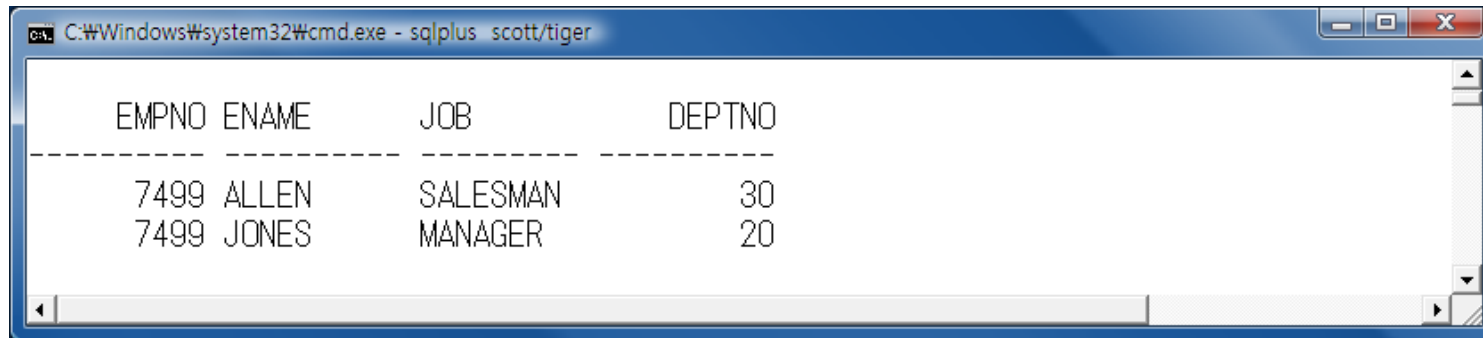
```
DESC EMP02
```

	Field	Type	Null	Key	Default	Extra
1	EMPNO	decimal(4,0)	NO		<null>	
2	ENAME	varchar(10)	NO		<null>	
3	JOB	varchar(9)	YES		<null>	
4	DEPTNO	decimal(2,0)	YES		<null>	

유일한 값만 허용하는 UNIQUE 제약 조건

▶ 유일한 값만 허용하는 UNIQUE 제약 조건

- ▶ UNIQUE 제약 조건이란 특정 칼럼에 대해 자료가 중복되지 않게 하는 것입니다.
- ▶ 즉, 지정된 칼럼에는 유일한 값이 수록되게 하는 것입니다.
- ▶ 새로운 사원이 입사하여 이 사원의 정보를 입력했는데, 이미 존재하는 사원의 번호와 동일한 사원번호로 입력하였더니 성공적으로 추가된다면 어떻게 될까요?



The screenshot shows a SQL*Plus window with the title bar 'C:\Windows\system32\cmd.exe - sqlplus scott/tiger'. The window displays a table with four columns: EMPNO, ENAME, JOB, and DEPTNO. The table contains two rows of data. The first row has EMPNO 7499, ENAME ALLEN, JOB SALESMAN, and DEPTNO 30. The second row has EMPNO 7499, ENAME JONES, JOB MANAGER, and DEPTNO 20. This illustrates a situation where a unique constraint on the EMPNO column would be violated by the second row.

EMPNO	ENAME	JOB	DEPTNO
7499	ALLEN	SALESMAN	30
7499	JONES	MANAGER	20

유일한 값만 허용하는 UNIQUE 제약 조건

- ▶ UNIQUE 제약조건을 설정하여 테이블 생성하기
 - ▶ 다음은 사원 테이블의 사원번호를 유일키로 지정한 예입니다.
 - ▶ 지금까지 실습에 사용했던 사원 테이블과 유사한 구조의 사원번호, 사원명, 직급, 부서번호 4개의 칼럼으로 구성된 EMP03 테이블을 생성하되 사원번호를 유일키로 지정합시다. 제약 조건은 칼럼명과 자료형을 기술한 후에 연이어서 UNIQUE를 기술하면 됩니다.

```
CREATE TABLE EMP03(  
  EMPNO DECIMAL(4) UNIQUE,  
  ENAME VARCHAR(10) NOT NULL,  
  JOB VARCHAR(9),  
  DEPTNO DECIMAL(2)  
);
```

	Field	Type	Null	Key	Default	Extra
1	EMPNO	decimal(4,0)	YES	UNI	<null>	
2	ENAME	varchar(10)	NO		<null>	
3	JOB	varchar(9)	YES		<null>	
4	DEPTNO	decimal(2,0)	YES		<null>	

유일한 값만 허용하는 UNIQUE 제약 조건

- ▶ UNIQUE 제약조건을 설정하여 테이블 생성하기
 - ▶ 위에서 생성한 EMP03 테이블에 데이터를 추가해 봅시다.

```
INSERT INTO EMP03  
VALUES(7499, 'ALLEN', 'SALESMAN', 30);
```

- ▶ 앞에서 직원번호로 7499번의 자료를 입력하였는데 다시 동일한 직원번호를 입력하면 어떻게 될까요?

```
INSERT INTO EMP02  
VALUES(7499, 'JONES', 'MANAGER', 20);
```

```
[23000][1062] Duplicate entry '7499' for key 'EMPNO'
```

유일한 값만 허용하는 UNIQUE 제약 조건

▶ UNIQUE 제약조건을 설정하여 테이블 생성하기

- ▶ 하지만 NULL 값은 중복되어 저장할 수 있습니다. UNIQUE는 값(VALUE)이 유일함을 의미하는 것입니다. NULL은 값(VALUE)에서 제외되므로 유일한 조건인지를 체크하는 값에서 제외됩니다.

```
INSERT INTO EMP03  
VALUES(NULL, 'JONES', 'MANAGER', 20);
```

```
INSERT INTO EMP03  
VALUES(NULL, 'JONES', 'SALESMAN', 10);
```

	EMPNO	ENAME	JOB	DEPTNO
1	7499	ALLEN	SALESMAN	30
2	<null>	JONES	MANAGER	20
3	<null>	JONES	SALESMAN	10

컬럼 레벨로 제약 조건명을 명시해서 제약 조건 설정하기

- ▶ 컬럼 레벨로 제약 조건명을 명시해서 제약 조건 설정하기
 - ▶ 지금까지는 사용자가 제약 조건명을 지정하지 않고 제약 조건만을 명시했습니다.
 - ▶ 이럴 경우 DBMS 서버가 자동으로 제약 조건명을 부여합니다.
 - ▶ 제약 조건에 위배하면 오류 메시지에 제약 조건명만 출력되는데 DBMS가 부여한 제약 조건명으로는 어떤 제약 조건을 위반했는지 알 수 없기에 데이터 덱서너리를 검색해야만 어떤 제약 조건인지 확인 할 수 있습니다.
 - ▶ 만일 사용자가 의미 있게 제약 조건명을 명시한다면 제약 조건명만으로도 어떤 제약 조건을 위배했는지 알 수 있게 됩니다.

컬럼 레벨로 제약 조건명을 명시해서 제약 조건 설정하기

- ▶ 컬럼 레벨로 제약 조건명을 명시해서 제약 조건 설정하기
 - ▶ 이번에는 제약 조건명을 지정하는 방법을 살펴보도록 합시다.

```
CONSTRAINT constraint_type  
constraint_name(column_name)
```

- ▶ 사용자 제약 조건 명을 설정하기 위해서는 CONSTRAINT라는 키워드와 함께 제약 조건 명을 기술하면 된다는 것을 확인할 수 있습니다.
- ▶ 제약 조건 명(constraining_name)은 다음과 같은 명명 규칙을 준수해서 작성하는 것이 좋습니다.

```
[레이블명]_[컬럼명]_[제약 조건 유형]
```

컬럼 레벨로 제약 조건명을 명시해서 제약 조건 설정하기

- ▶ 컬럼 레벨로 제약 조건명을 명시해서 제약 조건 설정하기
 - ▶ 사원 테이블 EMP04에 대해서 사원 번호를 저장하는 컬럼 EMPNO에 대한 유일 키 제약 조건 명인 EMP04_EMPNO_UK를 지정합니다.

EMP04_EMPNO_UK

테이블명

컬럼명 제약 조건유형

- ▶ 사용자 제약 조건 명을 설정하기 위해서는 CONSTRAINT라는 키워드와 함께 제약 조건 명을 기술해야 합니다.

컬럼 레벨로 제약 조건명을 명시해서 제약 조건 설정하기

▶ 컬럼 레벨로 제약 조건 명 명시하기

- ▶ 지금까지 실습에 사용했던 사원 테이블과 유사한 구조의 사원번호, 사원명, 직급, 부서번호 4개의 컬럼으로 구성된 EMP04 테이블을 생성하되 사원번호에는 유일키로 사원명은 NOT NULL 제약조건을 설정해 보시다.

```
CREATE TABLE EMP04(  
  EMPNO DECIMAL(4),  
  ENAME VARCHAR(10) NOT NULL,  
  JOB VARCHAR(9),  
  DEPTNO DECIMAL(2),  
  CONSTRAINT UNIQUE EMP04_EMPNO_UK(EMPNO)  
);
```

컬럼 레벨로 제약 조건명을 명시해서 제약 조건 설정하기

▶ 컬럼 레벨로 제약 조건 명 명시하기

- ▶ 생성된 제약 조건 명을 확인하기 위해서 USER_CONSTRAINTS 데이터 디렉터리 뷰를 검색해 봅시다.

형식 `select * from information_schema.table_constraints where table_name='emp04';`

	CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE
1	def	scott	EMP04_EMPNO_UK	scott	emp04	UNIQUE

데이터 구분을 위한 PRIMARY KEY 제약 조건

▶ 데이터 구분을 위한 PRIMARY KEY 제약 조건

- ▶ 테이블 내의 해당 행을 다른 행과 구분할 수 있도록 하는 칼럼은 반드시 존재해야 합니다. 식별 기능을 갖는 칼럼은 유일하면서도 NULL 값을 허용하지 말아야 합니다.
- ▶ 즉, UNIQUE 제약 조건과 NOT NULL 제약 조건을 모두 갖고 있어야 하는데 이러한 두 가지 제약 조건을 모두 갖는 것이 기본 키(PRIMARY KEY) 제약 조건입니다.

데이터 구분을 위한 PRIMARY KEY 제약 조건

▶ PRIMARY KEY 제약 조건 설정하기

- ▶ 다음은 사원 테이블의 사원번호를 기본 키로 지정한 예입니다. 제약 조건은 칼럼명과 자료형을 기술한 후에 연이어서 PRIMARY KEY를 기술하면 됩니다.
- ▶ 지금까지 실습에 사용했던 사원 테이블과 유사한 구조의 사원번호, 사원명, 직급, 부서번호 4개의 칼럼으로 구성된 테이블을 생성하되 기본 키 제약 조건을 설정해 보시다.

```
CREATE TABLE EMP05(  
  EMPNO DECIMAL(4),  
  ENAME VARCHAR(10) NOT NULL,  
  JOB VARCHAR(9),  
  DEPTNO DECIMAL(2),  
  CONSTRAINT PRIMARY KEY EMP05_EMPNO_PK(EMPNO)  
);
```

데이터 구분을 위한 PRIMARY KEY 제약 조건

▶ PRIMARY KEY 제약 조건 설정하기

- ▶ 위에서 생성한 테이블에 데이터를 추가해 봅시다.

```
INSERT INTO EMP05  
VALUES(7499, 'ALLEN', 'SALESMAN', 30);
```

- ▶ 다음은 기본 키로 지정된 직원번호에 동일한 값을 저장해 보도록 하겠습니다.

```
INSERT INTO EMP05  
VALUES(7499, 'JONES', 'MANAGER', 20);
```

```
[23000][1062] Duplicate entry '7499' for key 'PRIMARY'
```

데이터 구분을 위한 PRIMARY KEY 제약 조건

▶ PRIMARY KEY 제약 조건 설정하기

- ▶ 이번에는 기본 키로 지정된 직원번호에 NULL 값을 저장해 보도록 하겠습니다.

```
INSERT INTO EMP05  
VALUES(NULL, 'JONES', 'MANAGER', 20);
```

```
[23000][1048] Column 'EMPNO' cannot be null
```

AUTO_INCREMENT

▶ AUTO_INCREMENT 를 활용한 기본 키 자동 삽입

- ▶ 단순한 번호로 PRIMARY KEY를 자동 저장할 수 있는 방법이 있습니다. AUTO_INCREMENT는 사용자에게 데이터를 받지 않는 이상 데이터를 자동적으로 1씩 증가시켜 삽입시켜 주는 속성입니다.
- ▶ 이 속성은 기본키(Primary Key)에만 옵션 부여가 가능합니다. 또한, INSERT 시에 일어나며 INSERT를 통해 값이 오게 되면 자동으로 PRIMARY 컬럼의 값으로 1씩 증가된 값이 오게 됩니다.
- ▶ 다만 이럴 경우 AUTO_INCREMENT로 오는 컬럼의 속성은 INT 타입으로 오는 것과 UNSIGNED를 붙여 음수가 오지 않는 형태를 권장합니다.

column type	Maximum unsigned value
TINYINT	255
SMALLINT	65,535
MEDIUMINT	16,777,215
INT	4,294,967,295
BIGINT	18,226,744,073,709,551,615

AUTO_INCREMENT

- ▶ AUTO_INCREMENT 를 활용한 기본 키 자동 삽입
 - ▶ 다음의 테이블을 실행시켜 보도록 하겠습니다.

```
CREATE TABLE EMP05(  
  EMPNO INT UNSIGNED AUTO_INCREMENT,  
  ENAME VARCHAR(10) NOT NULL,  
  JOB VARCHAR(9),  
  DEPTNO DECIMAL(2),  
  CONSTRAINT PRIMARY KEY EMP05_EMPNO_PK(EMPNO)  
);
```

AUTO_INCREMENT

- ▶ AUTO_INCREMENT 를 활용한 기본 키 자동 삽입
 - ▶ KEY 값에 NULL을 붙여 입력할 경우 그 부분에 자동적으로 1이 증가되어 삽입됩니다.

```
INSERT INTO EMP05  
VALUES(NULL, 'JONES', 'MANAGER', 20);
```

```
INSERT INTO EMP05  
VALUES(NULL, 'ALLEN', 'SALESMAN', 30);
```

	EMPNO	ENAME	JOB	DEPTNO
1	1	JONES	MANAGER	20
2	2	ALLEN	SALESMAN	30

AUTO_INCREMENT

▶ AUTO_INCREMENT 를 활용한 기본 키 자동 삽입

- ▶ AUTO_INCREMENT 값의 현재 상태는 다음의 쿼리로 확인이 가능합니다.

```
SHOW TABLE STATUS WHERE NAME = 'EMP05';
```

Data_free	Auto_increment	Create_time
0	3	2021-10-20 16:14:02

- ▶ AUTO_INCREMENT 값의 초기화는 다음처럼 가능합니다.

```
ALTER TABLE EMP05 AUTO_INCREMENT = 100;
```

Data_free	Auto_increment	Create_time
0	100	2021-10-20 16:43:30

참조 무결성을 위한 FOREIGN KEY 제약 조건

▶ 참조 무결성을 위한 FOREIGN KEY 제약 조건

- ▶ 참조의 무결성이란 개념을 알아야 FOREIGN KEY 제약 조건을 설명할 수 있습니다.
- ▶ 참조의 무결성은 테이블 사이의 관계에서 발생하는 개념이므로 우리가 지금까지 학습했던 사원 테이블과 부서 테이블의 관계를 예를 들어 설명하겠습니다.
- ▶ 우선 부서 테이블을 살펴봅시다.
- ▶ 부서 테이블에는 부서에 대한 정보를 구분하기 위해서 유일하고 NULL이 아닌 값만 저장하도록 부서 번호 컬럼(DEPTNO)을 기본 키로 설정하고 있습니다.
- ▶ 부서 테이블을 살펴보면 부서 번호가 10, 20, 30, 40인 부서만 존재합니다. 부서 테이블의 부서 번호 컬럼(DEPTNO)과 동일한 이름의 컬럼이 사원(EMP) 테이블에도 존재합니다.
- ▶ 사원 테이블에 존재하는 부서 번호는 부서 테이블에 존재하는 부서 번호인 10, 20, 30으로만 기록되어 있습니다.

참조 무결성을 위한 FOREIGN KEY 제약 조건

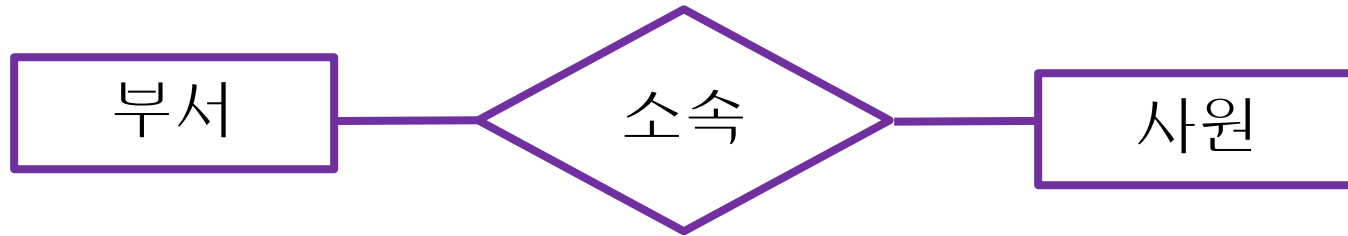
▶ 참조 무결성을 위한 FOREIGN KEY 제약 조건

- ▶ 해당 회사에 부서가 4개 존재한다면 그 회사에 다니는 직원들도 그 4개의 부서 중에 한곳에 소속이어야 하기 때문입니다.
- ▶ 만일 부서 테이블에 존재하지 않는 부서 번호가 특정 직원의 부서로 지정되어 있다면 이치에 맞지 않게 됩니다.
- ▶ 조인이나 서브 쿼리를 학습할 때 살펴보았듯이 직원 테이블에 없는 상세 정보는 부서 테이블에서 찾아오는데 직원 테이블에 저장된 부서번호가 테이블에 없다면 참조할 때 무결해야 한다는 조건(참조의 무결성)에 위배되는 것이 됩니다.
- ▶ 그러므로 직원 테이블에 부서번호 입력 할 때 부서 테이블에 존재하는 부서번호만 입력하도록 하면 참조의 무결성이 지켜지는 것입니다.
- ▶ 이를 위해서는 직원 테이블의 부서번호 컬럼에 외래 키 제약조건을 명시해야 합니다.
- ▶ 외래 키 제약조건은 직원 테이블의 부서 번호는 반드시 부서 테이블에 존재하는 부서 번호만 입력하도록 함으로서 직원 테이블이 부서 테이블을 부서 번호로 참조 가능하도록 하는 것을 의미합니다.

참조 무결성을 위한 FOREIGN KEY 제약 조건

- ▶ 참조 무결성을 위한 FOREIGN KEY 제약 조건

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



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

참조 무결성을 위한 FOREIGN KEY 제약 조건

▶ 참조 무결성을 위한 FOREIGN KEY 제약 조건

- ▶ 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 됩니다.

사원은 회사 내에 존재하는 부서에 소속되어 있어야 합니다.

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

참조 무결성을 위한 FOREIGN KEY 제약 조건

▶ 참조 무결성을 위한 FOREIGN KEY 제약 조건

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

참조 무결성을 위한 FOREIGN KEY 제약 조건

▶ 참조 무결성을 위한 FOREIGN KEY 제약 조건

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

부모테이블

Primary Key		

자식테이블

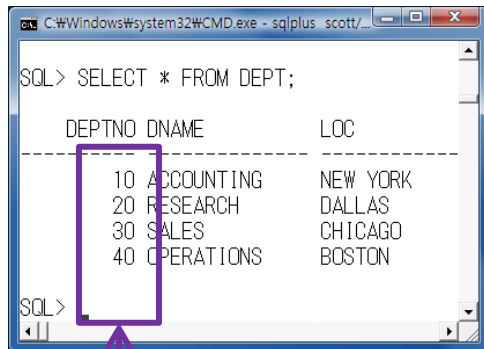
			Foreign Key



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

참조 무결성을 위한 FOREIGN KEY 제약 조건

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

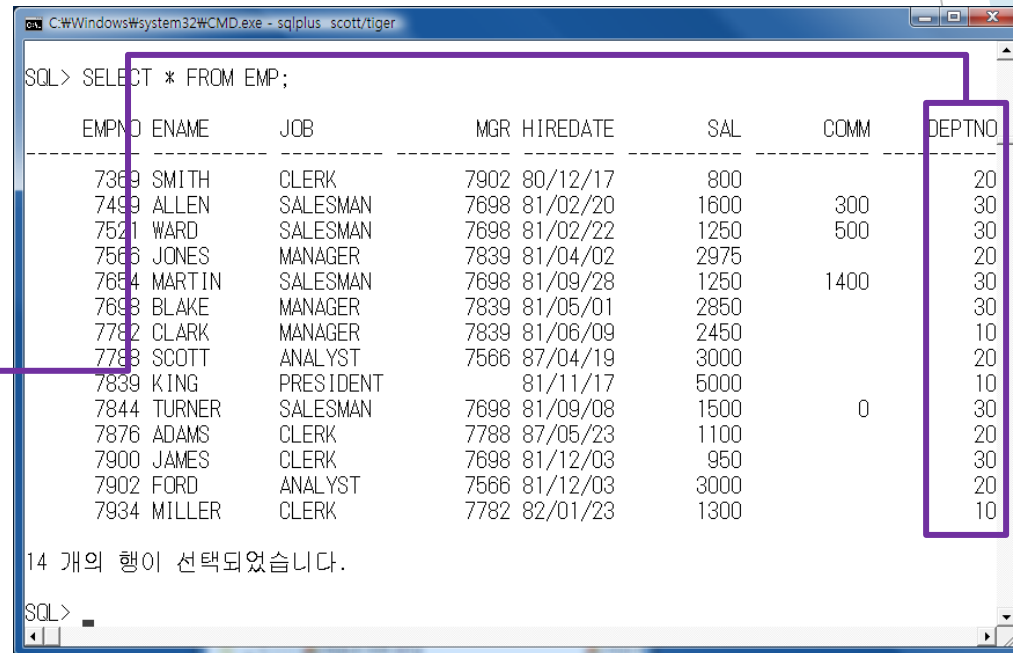


```
SQL> SELECT * FROM DEPT;
```

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



```
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>

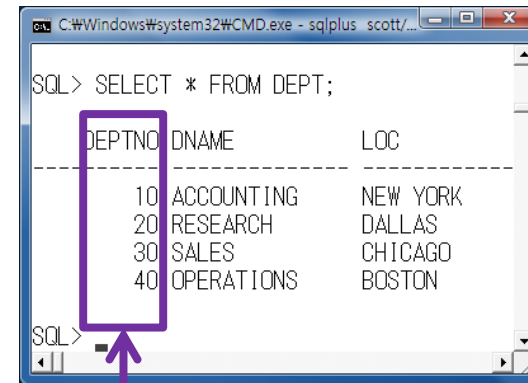
참조 무결성을 위한 FOREIGN KEY 제약 조건

▶ 참조 무결성을 위한 FOREIGN KEY 제약 조건

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

참조 무결성을 위한 FOREIGN KEY 제약 조건

- ▶ 참조 무결성을 위한 FOREIGN KEY 제약 조건
 - ▶ 외래 키 제약 조건이 지정된 사원 테이블에 부서 테이블에 존재하지 않은 50번 부서번호를 저장해 보도록 합시다.

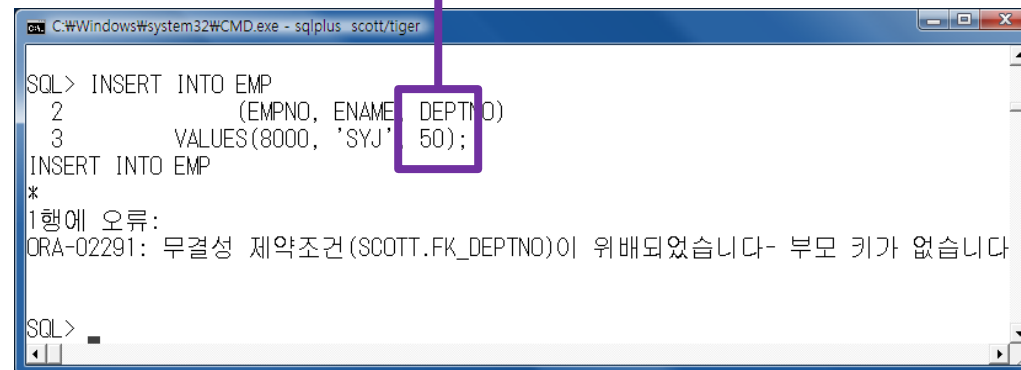


```
SQL> SELECT * FROM DEPT;
```

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

```
SQL>
```

EMP 테이블에 DEPTNO 컬럼 값을 50으로 하여 새로운 사원을 추가하려 하면 참조의 무결성을 위배했다는 오류 메시지가 출력.
DEPT 테이블에는 DEPTNO 컬럼 값으로 10, 20, 30, 40 만 존재하고 50은 존재하지 않기 때문이다.
EMP 테이블에서 참조하는 DEPT 테이블의 DEPTNO 컬럼을 부모 키라고 하므로 부모 키가 없다는 오류 메시지도 함께 출력됨



```
SQL> INSERT INTO EMP
2      (EMPNO, ENAME, DEPTNO)
3      VALUES(8000, 'SYJ', 50);
INSERT INTO EMP
*
1행에 오류:
ORA-02291: 무결성 제약조건 (SCOTT.FK_DEPTNO)이 위배되었습니다- 부모 키가 없습니다

SQL>
```

참조 무결성을 위한 FOREIGN KEY 제약 조건

▶ 참조 무결성을 위한 FOREIGN KEY 제약 조건

- ▶ 다음은 MySQL/MariaDB에서 제공하는 EMP 테이블과 DEPT 테이블의 제약 조건을 살펴보도록 합시다.

형식

```
select * from information_schema.table_constraints where  
table_name in ('emp', 'dept');
```

	CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE
1	def	scott	PRIMARY	scott	dept	PRIMARY KEY
2	def	scott	PRIMARY	scott	emp	PRIMARY KEY
3	def	scott	PK_EMP	scott	emp	FOREIGN KEY

참조 무결성을 위한 FOREIGN KEY 제약 조건

▶ 외래키 제약 조건 설정하기

- ▶ 외래키 제약 조건 설정해 보도록 합시다.
- ▶ 지금까지 실습에 사용했던 사원 테이블과 유사한 구조의 사원번호, 사원명, 직급, 부서번호 4개의 칼럼으로 구성된 테이블을 생성하되 기본 키 제약 조건은 물론 외래키 제약 조건도 설정해 봅시다.

```
CREATE TABLE EMP06(  
  EMPNO DECIMAL(4),  
  CONSTRAINT PRIMARY KEY EMP05_EMPNO_PK(EMPNO),  
  ENAME VARCHAR(10) NOT NULL,  
  JOB VARCHAR(9),  
  DEPTNO DECIMAL(2),  
  [CONSTRAINT EMP06_DEPTNO_FK] FOREIGN KEY (DEPTNO) REFERENCES DEPT(DEPTNO)  
);
```

참조 무결성을 위한 FOREIGN KEY 제약 조건

▶ 외래키 제약 조건 설정하기

- ▶ 현재 **EMP06** 테이블에 부서 테이블에 존재하지 않는 부서번호를 갖는 사원 정보를 추가해 봅시다.

```
INSERT INTO EMP06  
VALUES(7566, 'JONES', 'MANAGER', 50);
```

```
[23000][1452] Cannot add or update a child row: a foreign key constraint fails ('scott'.emp06, CONSTRAINT 'EMP06_DEPTNO_FK' FOREIGN KEY ('DEPTNO') REFERENCES 'dept' ('DEPTNO'))
```

CHECK 제약 조건

▶ CHECK 제약 조건

- ▶ **CHECK** 제약 조건은 입력되는 값을 체크하여 설정된 값 이외의 값이 들어오면 오류 메시지와 함께 명령이 수행되지 못하게 하는 것입니다.
- ▶ 조건으로 데이터의 값의 범위나 특정 패턴의 숫자나 문자 값을 설정할 수 있습니다.
- ▶ 예를 들어 사원 테이블에 급여 컬럼을 생성하되 급여 컬럼 값은 500에서 5000사이의 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 합니다.

CHECK 제약 조건

▶ CHECK 제약 조건 설정 하기

- ▶ 사원 테이블에 급여 컬럼을 생성하되 급여 컬럼 값은 500에서 5000사이의 값만 저장할 수 있도록 하고 성별을 저장하는 컬럼으로 GENDER 를 정의하고, 이 컬럼에는 남자는 M, 여자는 F 둘 중의 하나만 저장할 수 있도록 CHECK 제약조건을 지정해 보시다.
- ▶ 사원번호, 사원명, 직급, 부서번호, 직급, 성별 6개의 칼럼으로 구성된 테이블을 생성하되 기본 키 제약 조건, 외래키 제약 조건은 물론 CHECK 제약 조건도 설정해 보시다.

```
CREATE TABLE EMP07(  
  EMPNO DECIMAL(4),  
  CONSTRAINT PRIMARY KEY EMP07_EMPNO_PK(EMPNO),  
  ENAME VARCHAR(10) NOT NULL,  
  SAL DECIMAL(7, 2),  
  CONSTRAINT EMP07_SAL_CK CHECK(SAL BETWEEN 500 AND 5000),  
  GENDER VARCHAR(1),  
  CONSTRAINT EMP07_GENDER_CK CHECK(GENDER IN('M', 'F'))  
);
```

DEFAULT 제약 조건

▶ DEFAULT 제약 조건

- ▶ 디폴트는 아무런 값을 입력 하지 않았을 때 디폴트제약의 값이 입력이 됩니다.
- ▶ 만약 지역명(LOC)라는 컬럼에 아무런 값도 입력 안했을 때 디폴트의 값인 'SEOUL'이 들어가도록 하고 싶을 경우 디폴트 제약 조건을 지정합니다. 다음과 같이 부서 테이블을 생성해 봅시다.

```
CREATE TABLE DEPT01(  
  DEPTNO DECIMAL(2) PRIMARY KEY,  
  DNAME VARCHAR(14),  
  LOC VARCHAR(13) DEFAULT 'SEOUL');
```

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

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

	DEPTNO	DNAME	LOC
1	10	ACCOUNTING	SEOUL

테이블 레벨 방식으로 제약 조건 지정하기

- ▶ 테이블 레벨 방식으로 제약 조건 지정하기
 - ▶ 지금까지 제약 조건을 지정하는 방식을 컬럼 레벨의 제약 조건 지정이라고 합니다.
 - ▶ 컬럼 레벨 제약 조건
 - ▶ **CREATE TABLE**로 테이블을 생성하면서 컬럼을 정의하게 되는데 하나의 컬럼 정의가 다 마무리되기 전에 컬럼 명 다음에 타입을 지정하고 그 뒤에 연이어서 제약 조건을 지정하는 방식입니다.
 - ▶ 테이블 레벨의 제약 조건
 - ▶ 컬럼을 모두 정의하고 나서 테이블 정의를 마무리 짓기 전에 따로 생성된 컬럼들에 대한 제약 조건을 한꺼번에 지정하는 것입니다.

테이블 레벨 방식으로 제약 조건 지정하기

▶ 테이블 레벨 방식으로 제약 조건 지정하기

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

▶ 복합키로 기본키를 지정할 경우

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

▶ 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 합니다.

테이블 레벨 방식으로 제약 조건 지정하기

- ▶ 테이블 레벨 방식으로 제약 조건 지정하기
 - ▶ 다음은 테이블 레벨 정의 방식의 기본 형식입니다.

```
CREATE TABLE table_name  
(column_name1 datatype1,  
 column_name2 datatype2,  
  . . .  
 CONSTRAINT constraint_name constraint_type(. . .)  
)
```

- ▶ 테이블 레벨에서 칼럼의 제약 조건을 정의할 때 주의할 것은 NOT NULL 조건은 테이블 레벨 정의 방법으로 제약 조건을 지정할 수 없다는 점입니다.

테이블 레벨 방식으로 제약 조건 지정하기

- ▶ 컬럼 레벨 제약 조건 설정과 테이블 레벨 제약 조건 설정하기
 - ▶ 컬럼 레벨로 제약 조건과 테이블 레벨로 제약 조건을 지정하는 방법의 차이점을 살펴보기로 합니다.
 - ▶ 테이블 레벨로 제약 조건을 지정하기에 앞서 컬럼 레벨로 제약 조건을 지정하는 방법을 살펴보기로 합니다.

```
CREATE TABLE EMP01(  
  EMPNO DECIMAL(4) PRIMARY KEY,  
  ENAME VARCHAR(10) NOT NULL,  
  JOB VARCHAR(9) UNIQUE,  
  DEPTNO INT(11),  
);
```

테이블 레벨 방식으로 제약 조건 지정하기

- ▶ 컬럼 레벨 제약 조건 설정과 테이블 레벨 제약 조건 설정하기
 - ▶ 위 사용 예는 컬럼 레벨 방식인데, 다음 사용 예인 테이블 레벨 방식과 비교 해보도록 합시다.

```
CREATE TABLE EMP02(  
  EMPNO DECIMAL(4),  
  ENAME VARCHAR(10) NOT NULL,  
  JOB VARCHAR(9),  
  DEPTNO INT(11),  
  PRIMARY KEY(EMPNO),  
  UNIQUE(JOB),  
  FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO)  
);
```

테이블 레벨 방식으로 제약 조건 지정하기

- ▶ 컬럼 레벨 제약 조건 설정과 테이블 레벨 제약 조건 설정하기
 - ▶ 명시적으로 제약 조건 명을 지정하여 테이블 레벨 방식으로 제약 조건을 지정해 봅시다.

```
CREATE TABLE EMP03(  
  EMPNO DECIMAL(4) NOT NULL,  
  ENAME VARCHAR(10),  
  JOB VARCHAR(9),  
  DEPTNO INT(11),  
  CONSTRAINT EMP03_EMPNO_PK PRIMARY KEY(EMPNO),  
  CONSTRAINT EMP03_JOB_UK UNIQUE(JOB),  
  CONSTRAINT EMP03_DEPTNO_FK FOREIGN KEY(DEPTNO)  
  REFERENCES DEPT(DEPTNO)  
);
```

테이블 레벨 방식으로 제약 조건 지정하기

- ▶ 컬럼 레벨 제약 조건 설정과 테이블 레벨 제약 조건 설정하기
 - ▶ 이번에는 복합키를 기본 키로 지정하는 방법을 살펴보도록 합시다.
 - ▶ 일련번호를 따로 두지 않고 이름과 핸드폰 번호를 복합하여 이를 기본 키로 지정하기로 합시다.

```
CREATE TABLE MEMBER01(  
  NAME VARCHAR(10),  
  ADDRESS VARCHAR(30),  
  HPHONE VARCHAR(16),  
  CONSTRAINT MEMBER01_COMBO_PK PRIMARY KEY(NAME, HPHONE)  
);
```

제약 조건 추가하기

▶ 제약 조건 추가하기

- ▶ 테이블 구조를 결정하는 **DDL**을 학습하면서 테이블이 이미 생성된 이후에 테이블의 구조를 변경하기 위한 명령어로 **ALTER TABLE**을 사용한다는 것을 이미 학습하였습니다.
- ▶ 제약조건 역시 이미 테이블을 생성하면서 지정해주는 것이었기에 테이블 생성이 끝난 후에 제약 조건을 추가하기 위해서는 **ALTER TABLE**로 추가해 주어야 합니다.
- ▶ 다음은 제약 조건을 추가하기 위한 형식입니다.

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

- ▶ 만일 새로운 제약 조건을 추가하려면 **ALTER TABLE** 문에 **ADD** 절을 사용해야 합니다.

제약 조건 추가하기

▶ 기존 테이블에 제약 조건 추가하기

- ▶ 제약조건 없이 테이블을 생성한 후 제약 조건을 추가해 봅시다.
- ▶ 지금까지 실습에 사용했던 사원 테이블과 유사한 구조의 사원번호, 사원명, 직급, 부서번호 4개의 칼럼으로 구성된 **EMP01** 테이블을 제약조건을 하나도 설정하지 않은 채 생성해 봅시다.

```
CREATE TABLE EMP01(  
  EMPNO DECIMAL(4),  
  ENAME VARCHAR(10),  
  JOB VARCHAR(9),  
  DEPTNO INT(11)  
);
```


제약 조건 추가하기

▶ 기존 테이블에 제약 조건 추가하기

- ▶ 이제 이미 생성이 완료된 **EMP01** 테이블에 2가지 제약조건을 설정해 보도록 합시다. 첫 번째는 **EMPNO** 컬럼에 기본키를 설정하고 두 번째에는 **DEPTNO** 컬럼에 외래키를 설정해 보도록 합시다.

```
ALTER TABLE EMP01  
ADD CONSTRAINT EMP01_EMPNO_PK PRIMARY KEY(EMPNO);
```

```
ALTER TABLE EMP01  
ADD CONSTRAINT EMP01_DEPTNO_FK  
FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO);
```

제약 조건 추가하기

▶ MODIFY로 NOT NULL 제약 조건 추가하기

- ▶ NOT NULL 제약 조건을 이미 존재하는 테이블에 추가해 보도록 합시다.
- ▶ 이미 존재하는 테이블에 무결성 제약 조건을 추가로 생성하기 위해서 **ALTER TABLE ... ADD ...** 명령문을 사용하였습니다.
- ▶ 하지만 NOT NULL 제약 조건은 **ADD** 대신 **MODIFY** 명령문을 사용하므로 사용에 주의해야 합니다.
- ▶ 이는 'NULL을 허용하는 상태'에서 'NULL을 허용하지 않는 상태'로 변경하겠다는 의미로 이해하기 바랍니다.

제약 조건 추가하기

- ▶ **MODIFY로 NOT NULL 제약 조건 추가하기**
 - ▶ 이미 존재하는 테이블에 **NOT NULL** 제약 조건을 추가해 보도록 합시다.
 - ▶ **MODIFY** 명령어로 **NOT NULL** 제약 조건을 설정해 봅시다.

```
ALTER TABLE EMP01  
MODIFY COLUMN ENAME VARCHAR(10) NOT NULL;
```

제약 조건 제거하기

▶ 제약 조건 제거하기

- ▶ 제약 조건을 제거하기 위해서 **DROP CONSTRAINT** 다음에 제거하고자 하는 제약 조건 명을 명시해야 합니다.
- ▶ 다음은 제약조건을 제거하기 위한 형식입니다.

```
ALTER TABLE table_name  
DROP [CONSTRAINT TYPE [, constraint_name]];
```

- ▶ 제약 조건을 제거하기 위해서는 제약 조건명을 반드시 제시해야 합니다.
- ▶ 제약 조건을 **CONSTRAINT** 문을 사용하여 지정했을 경우에는 제약 조건 명을 기억하기 쉽지만, **CONSTRAINT** 문을 사용하지 않았으면 특정 테이블의 특정 컬럼에 명시된 제약 조건을 **USER_CONSTRAINTS** 데이터 디렉터리 뷰에서 찾아봐야 하는 불편함이 있습니다.
- ▶ 그렇기 때문에 제약 조건을 지정할 때에는 제약 조건명을 명시적으로 주는 것이 바람직합니다.

제약 조건 제거하기

▶ 제약 조건 제거하기

- ▶ 사원 테이블에 지정한 제약 조건들을 제거해 보도록 합시다.
- ▶ 기본 키 제약 조건을 제거합니다.

```
ALTER TABLE EMP01 DROP PRIMARY KEY;
```

- ▶ 외래 키 제약 조건을 제거 합니다.

```
ALTER TABLE EMP01  
DROP FOREIGN KEY EMP01_DEPTNO_FK;
```

제약 조건으로 인한 컬럼 삭제 불가능 예

▶ 제약 조건으로 인한 컬럼 삭제 불가능 예

- ▶ 제약 조건을 비활성화를 위한 실습을 위해서 부서 테이블을 만듭시다. 그런 후에 부서 테이블을 부모 테이블로 하는 사원 테이블을 작성합시다. 그러기 위해서는 부서 테이블의 부서번호가 기본 키로 설정되어 있고, 사원 테이블의 부서번호가 부서 테이블의 부서번호를 참조할 수 있도록 외래 키를 설정해야 합니다.
- ▶ 1.부서 테이블에 부서번호를 기본 키로 지정하여 새로운 테이블을 생성해 봅시다.

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

제약 조건으로 인한 컬럼 삭제 불가능 예

▶ 제약 조건으로 인한 컬럼 삭제 불가능 예

- ▶ 부서 테이블을 만들었으므로 이제 부서를 부모 테이블로 하는 직원 테이블을 작성하기 위해서 직원 테이블의 부서번호가 부서 테이블의 부서번호를 참조할 수 있도록 외래 키를 설정합니다.

```
CREATE TABLE EMP01(  
  EMPNO DECIMAL(4),  
  CONSTRAINT EMP01_EMPNO_PK PRIMARY KEY(EMPNO),  
  ENAME VARCHAR(10) NOT NULL,  
  JOB VARCHAR(9),  
  DEPTNO DECIMAL(2),  
  CONSTRAINT EMP01_DEPTNO_FK FOREIGN KEY(DEPTNO)  
  REFERENCES DEPT01(DEPTNO)  
);
```

제약 조건으로 인한 컬럼 삭제 불가능 예

▶ 제약 조건으로 인한 컬럼 삭제 불가능 예

- ▶ DEPT01에 DEPT 테이블의 데이터를 넣어줍니다.

```
INSERT INTO DEPT01 SELECT * FROM DEPT;  
SELECT * FROM DEPT01;
```

- ▶ 사원 테이블로서 사원의 정보를 추가할 때 부서 테이블을 참조하므로 부서 테이블에 존재하는 부서번호를 입력합니다.

```
INSERT INTO EMP01 VALUES(7499, 'ALLEN', 'SALESMAN', 10);  
INSERT INTO EMP01 VALUES(7369, 'SMITH', 'CLERK', 20);
```

- ▶ DEPT01 테이블에서 10번 부서를 ALLEN이란 사람이 참조하고 있는 상태에서 삭제해 봅시다.

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


제약 조건으로 인한 컬럼 삭제 불가능 예

▶ 제약 조건으로 인한 컬럼 삭제 불가능 예

- ▶ 자식 테이블인 사원 테이블(**EMP01**)은 부모 테이블인 부서 테이블(**DEPT01**)에 기본 키인 부서 번호를 참조하고 있습니다.
- ▶ 부서 테이블의 **10번** 부서는 사원 테이블에 근무하는 **10번** 사원이 존재하기 때문에 삭제할 수 없습니다.
- ▶ 부모 테이블(**DEPT01**)의 부서번호 **10번**이 삭제되면 자식 테이블(**EMP01**)에서 자신이 참조하는 부모를 잃어버리게 되므로 삭제할 수 없는 것입니다.
- ▶ 부서번호가 **10**인 자료가 삭제되도록 하기 위해서는 부서 테이블(**EMP01**)의 **10번** 부서에서 근무하는 사원을 삭제한 후 부서 테이블(**DEPT01**)에서 **10번** 부서를 삭제합니다.