

# 5장 : 테이블과 뷰

데이터베이스에는 다양한 개체가 존재한다. 그중에서 가장 중요한 것이 테이블이다. 테이블이 있어야만 다른 개체들도 연계하여 사용할 수 있다. 이번 장에선 테이블과 뷰에 대해 살펴본다.

## 5-1 : 테이블 생성

테이블은 표 형태로 구성된 2차원 구조로, 행과 열로 구성되어 있다. 행은 레코드(record)라고도 부르며, 열은 필드(field)라고도 부른다. 테이블은 엑셀의 시트와 거의 비슷한 구조로 이루어져 있다.

### 데이터베이스와 테이블 설계

이미 2장에서 데이터베이스 모델링에 대한 간단한 개념과 MySQL을 이용해서 테이블을 만드는 방법을 알아봤다. 여기서는 앞에서 배운 내용을 복습하는 개념으로 테이블을 생성하고 관리하는 것에 초점을 맞춘다. 이전에 사용했던 인터넷 마켓 데이터베이스와 동일한 구조로 실습해본다.

다음과 같은 테이블을 설계하고 만드는 과정을 실습한다.

- 회원 테이블

열 이름	데이터 형식	Not Null	기타
MEM_ID	CHAR (8)	Y	PRIMARY KEY
MEM_NAME	VARCHAR (10)	Y	
MEM_NUMBER	TINYINT	Y	
ADDR	CHAR(2)	Y	
PHONE1	CHAR(3)	N	
PHONE2	CHAR(8)	N	
HEIGHT	TINYINT	N	UNSIGNED
DEBUT_DATE	DATE	N	

- 구매 테이블

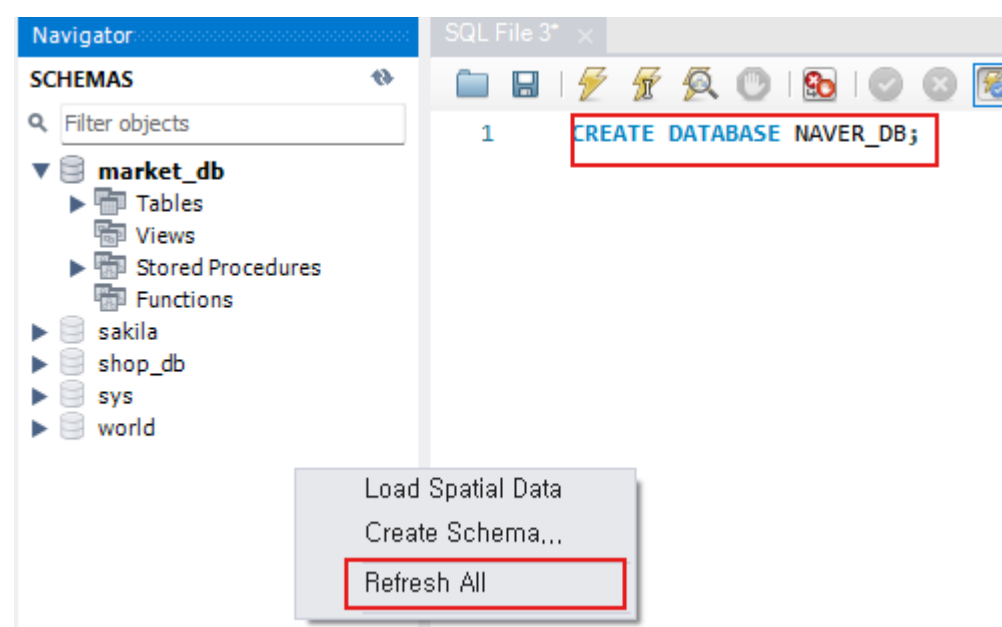
열 이름	데이터 형식	Not Null	기타
NUM	INT	Y	PRIMARY KEY, AUTO_INCREMENT
MEM_ID	CHAR(8)	Y	FOREIGN KEY
PROD_NAME	CHAR(6)	Y	
GROUP_NAME	CHAR(4)	N	
PRICE	INT	Y	UNSIGNED
AMOUNT	SMALLINT	Y	UNSIGNED

## GUI 환경에서 테이블 만들기

먼저 GUI 환경에서 테이블을 만드는 방법에 대해 배운다.

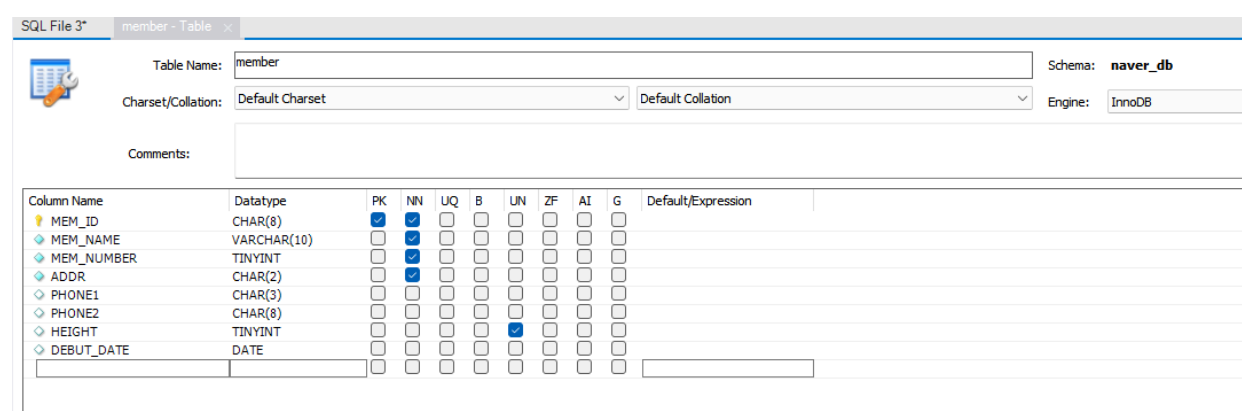
### 데이터베이스 생성

MySQL을 실행해서 새 쿼리 창을 생성한다. **CREATE DATABASE NAVER\_DB;** 를 입력, 실행한다. SQL로 만든 데이터베이스는 화면에 바로 적용되지 않기 때문에 스키마 패널에 보이지 않는다. 스키마 패널에서 마우스 오른쪽 버튼을 클릭하고 **[Refresh All]**을 선택한다.

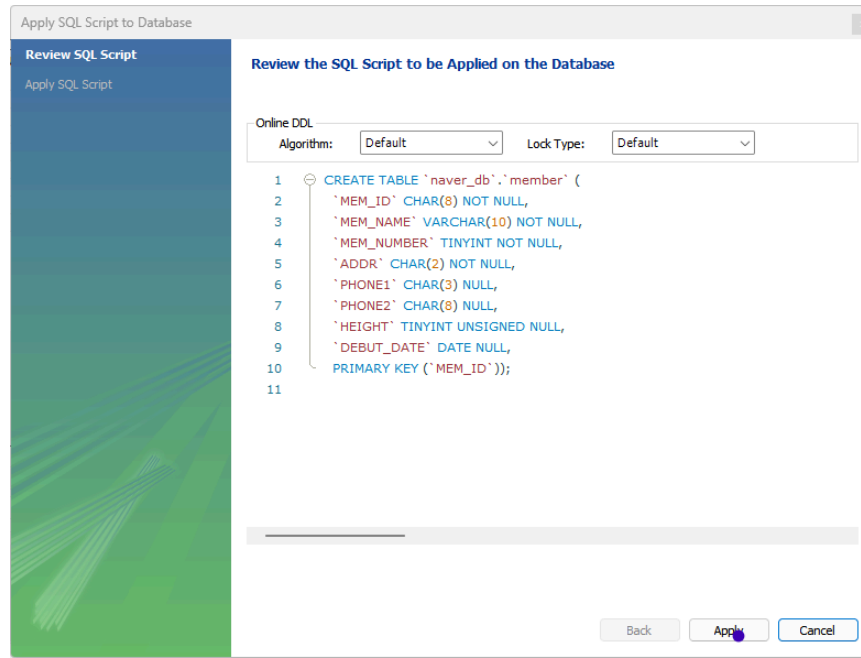


### 테이블 생성

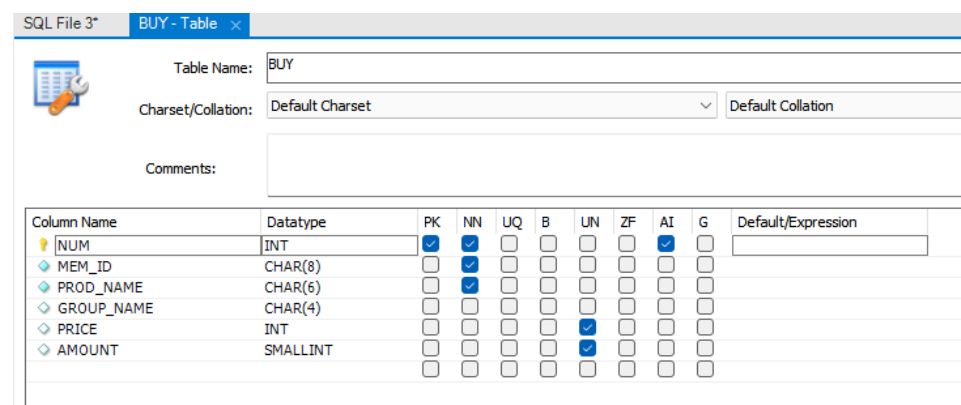
1. 먼저 회원 테이블을 생성한다. **NAVER\_DB** 데이터베이스를 확장해서 **Tables**를 선택하고 마우스 오른쪽 버튼을 클릭한 후 **[Create Table]**을 선택한다. 그 후 앞에서 설계한 대로 회원 테이블을 다음과 같이 구성한다. 완료되었으면 하단에 **[Apply]** 를 클릭하여 적용한다.



2. **Apply SQL scripts to Databse** 창에 생성된 CREATE TABLE 코드를 확인할 수 있다. **[Apply]→[Finish]** 버튼을 클릭하여 내용을 적용한 후 **[File] → [Close Tab]** 메뉴를 실행하여 쿼리 창을 종료한다.



3. 같은 방식으로 구매 테이블(buy)을 생성한다. 순번(num)은 자동 증가(**AUTO\_INCREMENT**)를 위해 **AI**로 지정하고 가격(PRICE)와 수량(AMOUNT)는 음수가 들어가지 않아 **UN(UNSIGNED)**로 처리한다. APPLY를 누르면 CREATE TABLE 코드가 생성된다.



4. GUI에서는 네이버 쇼핑 DB 구성도에서 **기본 키-외래 키 관계**를 선택할 수 없다. 그래서 코드를 다음과 같이 수정한다. 생성된 외래 키는 스키마 패널에서 확인할 수 있다.

```

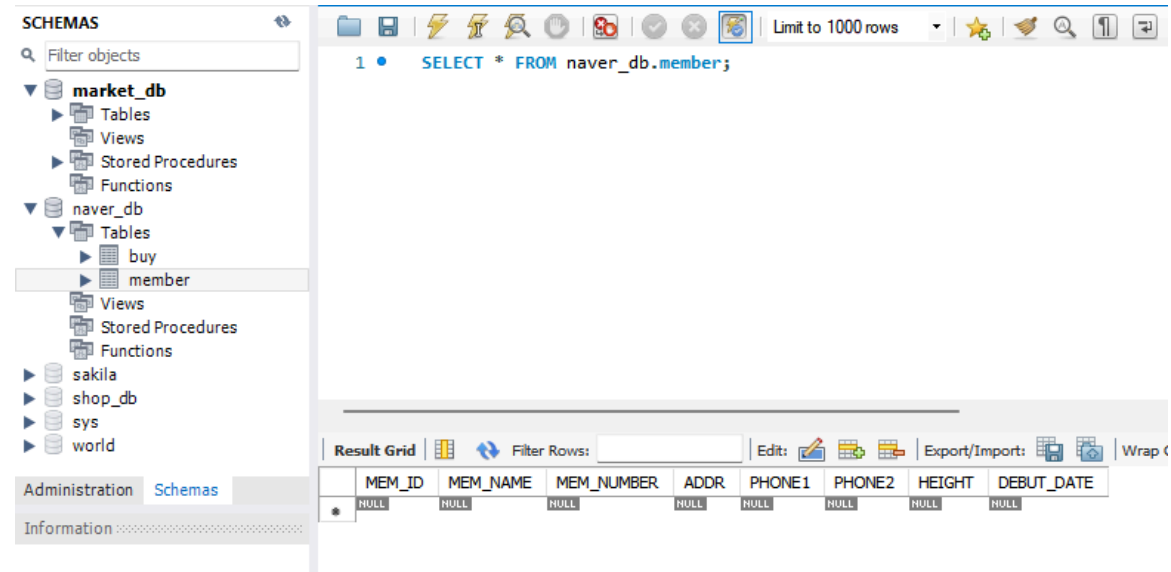
1 CREATE TABLE `naver_db`.`buy` (
2   `NUM` INT NOT NULL AUTO_INCREMENT,
3   `MEM_ID` CHAR(8) NOT NULL,
4   `PROD_NAME` CHAR(6) NOT NULL,
5   `GROUP_NAME` CHAR(4) NULL,
6   `PRICE` INT UNSIGNED NULL,
7   `AMOUNT` SMALLINT UNSIGNED NULL,
8   PRIMARY KEY (`NUM`),
9   FOREIGN KEY(MEM_ID) REFERENCE MEMBER(MEM_ID)
10 );
11

```

사진에는 REFERENCE라고 되어 있는데 뒤에 S를 붙여야 된다.

## 데이터 입력

1. 스키마 패널에서 **[naver\_db]→[Tables]→[member]**를 선택하고 마우스 오른쪽 버튼을 눌러 **[Select Rows- Limit 1000]**을 선택한다. 그러면 SELECT 문이 자동으로 생성되고 **[Result Grid]**창에 다음과 같은 결과가 보인다. 아직은 데이터를 입력하지 않아 행이 비어 있다.



2. **Insert new row** 아이콘(빨간색 박스 안)을 클릭하고 '네이버 쇼핑 DB 구성도의 값을 3개 행만 입력한다. 입력이 완료되면 우측 하단에 서 [Apply]→[Finish] 버튼을 차례대로 클릭하여 적용 후 [File]→[Close Tab] 메뉴를 실행해서 SQL 탭을 종료한다.

MEM_ID	MEM_NAME	MEM_NUMBER	ADDR	PHONE1	PHONE2	HEIGHT	DEBUT_DATE
TWC	트와이스	9	서울	02	11111111	167	2015.10.19
BLK	블랙핑크	4	경남	055	22222222	163	2016.08.08
WMN	여자친구	6	경기	031	33333333	166	2015.01.15
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. 이번에는 동일한 방식으로 구매 테이블에 3개 행만 입력하고 Apply 버튼을 연속으로 선택한다.

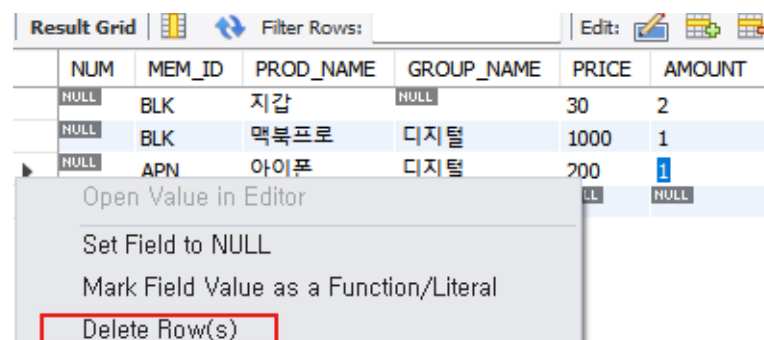
NUM	MEM_ID	PROD_NAME	GROUP_NAME	PRICE	AMOUNT
NULL	BLK	지갑	NULL	30	2
NULL	BLK	맥북프로	디지털	1000	1
NULL	APN	아이폰	디지털	200	1
NULL	NULL	NULL	NULL	NULL	NULL

NUM은 AUTO\_INCREMENT이므로 NULL로 둔다.

4. 오류가 발생했다. 네이버 쇼핑 DB 구성도의 회원 테이블과 구매 테이블은 **기본 키-외래 키**로 연결되어 있다. 이는 **구매 테이블의 MEM\_ID 값은 반드시 회원 테이블의 MEM\_ID로 존재해야 한다**는 의미이다. 여기서 아직 회원 테이블에 APN이라는 회원을 입력하지 않아 오류가 발생한 것이다. Cancel 버튼을 클릭해서 Apply SQL script to Database 창을 종료한다.

좀 더 쉽게 말하면 **회원 정보에 없는 사람의 물건 구매 기록이 남는 현상**이 나타난다는 것이다.

5. 행을 삭제하기 위해 **[Result Grid]** 창에서 APN 앞의 빈 부분을 클릭해서 선택하고 마우스 오른쪽 버튼을 클릭해서 **[Delete Row(s)]**를 선택한다. 그 후 다시 [Apply] 버튼을 클릭한다. 다음으로 Apply SQL script to Database 창에서 코드를 확인한 후 [Apply]와 [Finish] 버튼을 차례대로 클릭해서 데이터 입력을 완료한다.



이 정도로 MySQL 워크벤치에서 데이터를 입력하는 방법을 마친다. 다음엔 동일한 과정을 모두 SQL문을 사용해서 진행해본다.

## SQL 문으로 테이블 만들기

3장에서 SQL로 market\_db를 생성했다. 처음부터 다시 SQL을 이용해서 테이블을 생성한다. 먼저 위에서 만든 naver\_db를 삭제하고 스키마 패널에서 [Refresh All]을 선택한다.

### 데이터베이스 생성

새로운 쿼리 창을 열고, **CREATE DATABASE naver\_db;** 로 먼저 데이터베이스를 만든다.

### 테이블 생성

1. **CREATE TABLE** 구문으로 회원 테이블을 만든다. 열 이름과 형식을 지정하고, 나머지 조건들을 입력하면 어렵지 않게 테이블을 만들 수 있다. 입력 후 스키마 패널에서 [Refresh All]을 선택하면 생성한 테이블을 확인할 수 있다.

```
• CREATE TABLE MEMBER -- 회원 테이블
  (MEM_ID CHAR(8) NOT NULL PRIMARY KEY, -- 회원 아이디(PK)
  MEM_NAME VARCHAR(10) NOT NULL ,
  MEM_NUMBER TINYINT NOT NULL,
  ADDR CHAR(2) NOT NULL,
  PHONE1 CHAR(3) NULL,
  PHONE2 CHAR(8) NULL,
  HEIGHT TINYINT UNSIGNED NULL,
  DEBUT_DATE DATE NULL
  );
```

2. 구매 테이블도 동일한 방식으로 만들 수 있다. 순번(num) 열에 **AUTO\_INCREMENT**를 설정하고 기본 키(Primary key)로 지정하는 것과 아이디 열을 회원 테이블의 아이디 열의 외래 키로 설정하는 것만 주의하면 된다. **외래 키는 테이블을 만들 때 제일 마지막에 FOREIGN KEY 예약어로 지정한다.** 마지막에 Foreign key 문쪽의 의미를 설명하자면 '이 테이블의 MEM\_ID열을 MEMBER 테이블의 MEM\_ID열과 외래 키 관계로 연결해라' 라는 의미이다.

```
• CREATE TABLE BUY
  (NUM INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
  MEM_ID CHAR(8) NOT NULL,
  PROD_NAME CHAR(8) NOT NULL,
  GROUP_NAME CHAR(4) NULL,
  PRICE INT UNSIGNED NOT NULL,
  AMOUNT SMALLINT UNSIGNED NOT NULL,
  FOREIGN KEY(MEM_ID) REFERENCES MEMBER(MEM_ID)
  );
```

### 데이터 입력

1. 회원 테이블에 3건의 데이터를 입력한다.

```
INSERT INTO MEMBER VALUES('TWC','트와이스',9,'서울','02','11111111',167,'2015-10-19');
INSERT INTO MEMBER VALUES('BLK','블랙핑크',4,'경남','055','22222222',163,'2016-8-8');
INSERT INTO MEMBER VALUES('WMN','여자친구',6,'경기','031','33333333',166,'2015-1-15');
```

DATE로 지정된 열에는 연.월.일 또는 연-월-일 형식으로 값을 입력한다.

2. 이번에는 구매 테이블에 3건의 데이터를 입력한다. GUI와 마찬가지로, APN은 아직 회원 테이블에 존재하지 않아 오류가 발생한다.

```
INSERT INTO BUY VALUES(NULL, 'BLK','지갑',NULL,30,2 );
INSERT INTO BUY VALUES(NULL, 'BLK','맥북프로','디지털',1000,1);
INSERT INTO BUY VALUES(NULL, 'APN','아이폰','디지털',200,1);
```

```
Error Code: 1452. Cannot add or update a child row:
a foreign key constraint fails (`naver_db`.`buy`, CONSTRAINT `buy_ibfk_1` FOREIGN KEY (`MEM_ID`) REFERENCES `member` (`MEM_ID`))
```

지금까지 MySQL 워크벤치의 GUI 환경과 SQL 문으로 동일한 작업을 수행했다. 연습을 위해선 SQL 문으로 테이블 생성 방법을 익힌 후에 GUI를 사용하는 것을 추천한다.

## 5-2 : 제약조건

테이블을 만들 땐 테이블의 구조에 필요한 **제약조건**을 설정해줘야 한다. 앞에서 확인한 기본 키(Primary Key,pk)와 외래 키(Foreign Key,fk)가 대표적인 **제약조건**이다. 기본 키는 학번, 아이디, 사번 등과 같은 고유한 번호를 의미하는 열에, 외래 키는 기본 키와 연결되는 열에 지정한다.

이메일, 휴대폰과 같이 중복되지 않는 열에는 **고유 키(Unique)**를 지정할 수 있다. 사람의 키는 4미터를 넘지 않는다. 이때 실수로 400을 입력하는 것을 방지하는 제약조건이 **체크(Check)**이다. 매번 입력하기 귀찮은 데이터는 **기본값(Default)**를 설정할 수 있다. 마지막으로 값을 꼭 입력해야 하는 **NOT NULL**도 모두 제약조건이다.

### 제약조건의 기본 개념과 종류

**제약조건(constrain)**은 데이터의 무결성을 지키기 위해 제한하는 조건이다. 데이터의 무결성이란 **데이터에 결함이 없다**는 의미이다. 만약 네이버 회원의 아이디가 중복되면 이메일, 블로그, 쇼핑 기록 등 상당한 혼란이 야기될 것이다. 이런 것이 바로 데이터의 결함이고 이런 결함이 없는 것을 **데이터의 무결성**이라고 한다.

이러한 결함을 미리 방지하기 위해서 회원 테이블의 아이디를 **기본 키(Primary Key)**로 지정할 수 있다. 기본 키가 되기 위한 **조건은 중복되지 않으며, 비어 있지도 않음**이므로, 실수로 중복된 아이디를 넣으려고 해도 입력이 안된다. 기본 키외에 MySQL에서 제공하는 대표적인 제약조건은 다음과 같으며, 이제부터 하나씩 살펴본다.

- **Primary Key** 제약조건
- **Foreign Key** 제약조건
- **Unique** 제약조건
- **Check** 제약조건
- **Default** 제약조건
- **Null** 제약조건

### 기본 키(Primary Key) 제약 조건

테이블에는 많은 행 데이터가 있다. 이 중에서 데이터를 구분할 수 있는 식별자를 **기본 키(Primary key)**라고 부른다. 기본 키에 입력되는 값은 중복될 수 없으며, NULL 값이 입력될 수 없다. 우리가 어느 사이트에 회원가입을 한다고 할 때, 아이디는 중복될 수 없다. 이는 회원 아이디가 기본 키로 설정되어 있기 때문이다.

대부분의 테이블은 기본 키를 가져야 한다. 기본 키가 없어도 테이블 구성이 가능하지만 실무에서 사용하는 테이블에는 기본 키를 설정해야 중복된 데이터가 입력되지 않는다.

마지막으로 기억해야 할 것은 테이블은 기본 키를 1개만 가질 수 있다는 점이다.

### CREATE TABLE에서 설정하는 기본 키 제약조건

CREATE TABLE 문에서 **PRIMARY KEY**를 넣어서 기본 키 설정을 할 수 있다.

```
CREATE TABLE MEMBER -- 회원 테이블
(MEM_ID CHAR(8) NOT NULL PRIMARY KEY, -- 회원 아이디(PK)
MEM_NAME VARCHAR(10) NOT NULL ,
MEM_NUMBER TINYINT NOT NULL,
ADDR CHAR(2) NOT NULL,
PHONE1 CHAR(3) NULL,
PHONE2 CHAR(8) NULL,
HEIGHT TINYINT UNSIGNED NULL,
DEBUT_DATE DATE NULL
);
```

**DESCRIBE** 문을 사용하여 테이블의 정보를 살펴볼 수 있다. mem\_id가 기본 키로 지정되어 있다.

	Field	Type	Null	Key	Default	Extra
▶	mem_id	char(8)	NO	PRI	NULL	
	mem_name	varchar(10)	NO		NULL	
	mem_number	int	NO		NULL	
	addr	char(2)	NO		NULL	
	phone1	char(3)	YES		NULL	
	phone2	char(8)	YES		NULL	
	height	smallint	YES		NULL	
	debut_date	date	YES		NULL	


CREATE TABLE 문에서 기본 키를 지정하는 방법은 제일 마지막 행에 **PRIMARY KEY(열\_이름)**을 추가하는 것이다.

```
CREATE TABLE MEMBER -- 회원 테이블
(
    MEM_ID CHAR(8) NOT NULL,
    MEM_NAME VARCHAR(10) NOT NULL ,
    MEM_NUMBER TINYINT NOT NULL,
    ADDR CHAR(2) NOT NULL,
    PRIMARY KEY(MEM_ID)
);
```

**ALTER TABLE에서 설정하는 기본 키 제약 조건**

제약조건을 설정하는 또 다른 방법은 이미 만들어진 테이블을 수정하는 ALTER TABLE 구문을 사용하는 것이다. ALTER TABLE은 다음과 같이 사용한다.

```
ALTER TABLE MEMBER -- MEMBER TABLE 변경
ADD CONSTRAINT -- 제약조건 추가
PRIMARY KEY(MEM_ID); -- MEM_ID 열에 기본 키 제약조건 설정
```

 테이블 삭제 순서

기본 키-외래 키 관계로 연결된 테이블 삭제를 할 때 외래 키가 설정된 테이블을 먼저 삭제한다.

**외래 키 제약조건**

**외래 키(Foreign Key)** 제약조건은 두 테이블 사이의 관계를 연결해주고, 그 결과 데이터의 무결성을 보장해주는 역할을 한다. 외래 키가 설정된 열은 꼭 다른 테이블의 기본 키와 연결되어야 한다.

기본 키가 있는 테이블을 **기준 테이블**이라고 부르며, 외래 키가 있는 테이블은 **참조 테이블**이라고도 부른다.

외래 키로 지정된 데이터는 반드시 기준 테이블에 존재해야 한다. 마지막으로 참조 테이블이 참조하는 기준 테이블의 열은 반드시 기본 키(PK)나 **고유 키(Unique)**로 설정되어 있어야 한다. 고유 키는 잠시 후에 살펴본다.

**CREATE TABLE에서 설정하는 외래 키 제약조건**

외래 키를 생성하는 방법은 CREATE TABLE 끝에 **FOREIGN KEY** 키워드를 설정하는 것이다.



```
CREATE TABLE BUY
(
  NUM INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
  MEM_ID CHAR(8) NOT NULL,
  PROD_NAME CHAR(8) NOT NULL,
  GROUP_NAME CHAR(4) NULL,
  PRICE INT UNSIGNED NOT NULL,
  AMOUNT SMALLINT UNSIGNED NOT NULL,
  FOREIGN KEY(MEM_ID) REFERENCES MEMBER(MEM_ID)
);
```

만약 기존 테이블의 열이 PRIMARY KEY 또는 UNIQUE가 아니라면 외래 키 관계는 설정되지 않는다.

### ALTER TABLE에서 설정하는 외래 키 제약조건

이미 만든 테이블에서 외래 키를 설정하는 다른 방법은 ALTER TABLE 구문을 이용하는 것이다.

```
ALTER TABLE BUY
ADD CONSTRAINT
FOREIGN KEY(MEM_ID) REFERENCES MEMBER(MEM_ID);
```

### 기존 테이블의 열이 변경될 경우

기존 테이블의 열이 변경되는 경우를 살펴보자. 회원 테이블의 BLK가 물품을 2건 구매한 상태에서 회원 아이디를 PINK로 변경하면 두 테이블의 정보가 일치하지 않게 된다.

5-1에서 우리는 구매 테이블을 만들고, 데이터를 입력했다.

```
CREATE TABLE BUY
(
  NUM INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
  MEM_ID CHAR(8) NOT NULL,
  PROD_NAME CHAR(8) NOT NULL,
  GROUP_NAME CHAR(4) NULL,
  PRICE INT UNSIGNED NOT NULL,
  AMOUNT SMALLINT UNSIGNED NOT NULL,
  FOREIGN KEY(MEM_ID) REFERENCES MEMBER(MEM_ID)
);

INSERT INTO MEMBER VALUES('TWC','트와이스',9,'서울','02','11111111',167,'2015-10-19');
INSERT INTO MEMBER VALUES('BLK','블랙핑크',4,'경남','055','22222222',163,'2016-8-8');
INSERT INTO MEMBER VALUES('WMN','여자친구',6,'경기','031','33333333',166,'2015-1-15');
```

내부 조인을 사용해 물품 정보 및 사용자 정보를 확인해보면 문제없이 잘 출력된다.

BLK	블랙핑크	지갑
BLK	블랙핑크	맥북프로
BLK	블랙핑크	청바지

이번엔 BLK의 아이디를 PINK로 변경해본다.

```
UPDATE MEMBER SET MEM_ID='PINK' WHERE MEM_ID='BLK';
```

**Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (naver\_db . buy , CONSTRAINT buy\_ibfk\_1 FOREIGN KEY ( MEM\_ID ) REFERENCES member ( MEM\_ID ))**

다음과 같은 오류가 발생한다. 기본 키-외래 키로 맺어진 후에는 기존 테이블의 이름이 변경되지 않는다. 이름이 변경되면 참조 테이블의 데이터에 문제가 발생하기 때문이다. 마찬가지로 삭제도 안된다.

(만약 BLK가 물건을 구매하지 않았다면, 즉 구매 테이블에 데이터가 없다면 회원 테이블의 BLK는 변경 가능)

만약 회원 테이블에서 이름이 변경, 삭제될 때, 구매 테이블에서도 자동으로 수정되게 만들고 싶다면 **ON UPDATE CASCADE(변경)** 문과 **ON DELETE CASCADE(삭제)** 문을 사용하면 된다.



```

USE NAVER_DB;
DROP TABLE IF EXISTS BUY;
CREATE TABLE BUY
(
    NUM INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
    MEM_ID CHAR(8) NOT NULL,
    PROD_NAME CHAR(8) NOT NULL
);

ALTER TABLE BUY
ADD CONSTRAINT
FOREIGN KEY(MEM_ID) REFERENCES MEMBER(MEM_ID)
ON UPDATE CASCADE
ON DELETE CASCADE;

```

이제 회원 테이블의 BLK를 PINK로 변경한다. 이번엔 오류 없이 잘 변경되었다. 마찬가지로 삭제도 정상적으로 실행된다.

```

UPDATE MEMBER SET MEM_ID = 'PINK' WHERE MEM_ID = 'BLK';

```

## 기타 제약 조건

앞에서 중요한 핵심 제약조건을 다루었다. 이 외에도 실무에서 데이터베이스를 운영하다 보면 필요한 여러 가지 제약 조건이 존재한다.

### 고유 키 제약조건

**고유 키(Unique) 제약조건**은 '중복되지 않는 유일한 값'을 입력해야 하는 조건이다. 이것은 기본 키 제약조건과 거의 비슷하지만, **고유 키 제약 조건은 NULL값을 허용한다**. 또한 기본 키는 테이블에 1개의 열에만 지정할 수 있지만 **고유 키는 테이블의 여러 열에 지정할 수 있다**.

만약 회원 테이블에 Email 주소가 있다면 중복되지 않으므로 고유 키로 설정할 수 있다.

```

DROP TABLE IF EXISTS BUY, MEMBER;
CREATE TABLE MEMBER
(
    MEM_ID CHAR(8) NOT NULL PRIMARY KEY,
    MEM_NAME VARCHAR(1) NOT NULL,
    HEIGHT TINYINT UNSIGNED NULL,
    EMAIL CHAR(30) NULL UNIQUE
);

```

### 체크 제약조건

**체크(Check) 제약조건**은 **입력되는 데이터를 점검하는 기능**을 한다. 예를 들어 평균 키(height)에 마이너스 값이 입력되지 않게 하거나, 연락처의 국번에 02,031,041 중 하나만 입력되도록 할 수 있다.

먼저 테이블을 정의하면서 CHECK 제약조건을 설정해본다. 평균 키는 반드시 100 이상의 값만 입력되도록 설정한다. 열의 정의 뒤에 **CHECK(조건)**을 추가해주면 된다.

```

DROP TABLE IF EXISTS MEMBER;
CREATE TABLE MEMBER
(
    MEM_ID CHAR(8) NOT NULL PRIMARY KEY,
    MEM_NAME VARCHAR(8) NOT NULL,
    HEIGHT TINYINT UNSIGNED NULL CHECK(HEIGHT >=100),
    PHONE1 CHAR(3) NULL
);

```

데이터를 입력해본다. 다음 두 번째 행은 체크 제약조건에 위배되므로 오류가 발생한다. CHECK CONSTRAINT 오류는 체크 제약조건에서 설정한 값의 범위를 벗어났기 때문에 발생한다.

```

INSERT INTO MEMBER VALUES('BLK', '블랙핑크', 163, NULL);
INSERT INTO MEMBER VALUES('TWC', '트와이스', 99, NULL);

```

**Error Code: 3819. Check constraint 'member\_chk\_1' is violated.**

필요하다면 테이블을 만든 후에 **ALTER TABLE** 문으로 제약조건을 추가할 수 있다. 연락처의 국번에 02, 031, 032, 054, 055, 061 중 하나만 입력되도록 설정한다.

```
ALTER TABLE MEMBER
ADD CONSTRAINT
CHECK (PHONE1 IN ('02','031','032','054','055','061'));
```

마찬가지로 체크 제약조건에 위배되는 데이터를 입력하면 오류가 발생한다.

### 기본값 정의

기본값(Default) 정의는 값을 입력하지 않았을 때 자동으로 입력될 값을 미리 지정해 놓는 방법이다.

예를 들어 키를 입력하지 않고 기본적으로 160이라고 입력되도록 하고 싶다면 다음과 같이 정의할 수 있다.

```
DROP TABLE IF EXISTS MEMBER;
CREATE TABLE MEMBER
(MEM_ID CHAR(8) NOT NULL PRIMARY KEY,
MEM_NAME VARCHAR(10) NOT NULL,
HEIGHT TINYINT UNSIGNED NULL DEFAULT 160,
PHONE1 CHAR(3) NULL
);
```

**ALTER TABLE** 사용 시 열에 DEFAULT를 지정하기 위해서는 **ALTER COLUMN** 문을 사용한다.

```
ALTER TABLE MEMBER
ALTER COLUMN PHONE1 SET DEFAULT '02';
```

기본값이 설정된 열에 기본값을 입력하려면 **DEFAULT**라고 써주고, 원하는 값을 입력하려면 해당 값을 써주면 된다.

```
INSERT INTO MEMBER VALUES('RED', '레드벨벳', 161, '054');
INSERT INTO MEMBER VALUES('SPC', '우주소녀', DEFAULT, DEFAULT);
```

### NULL 값 허용

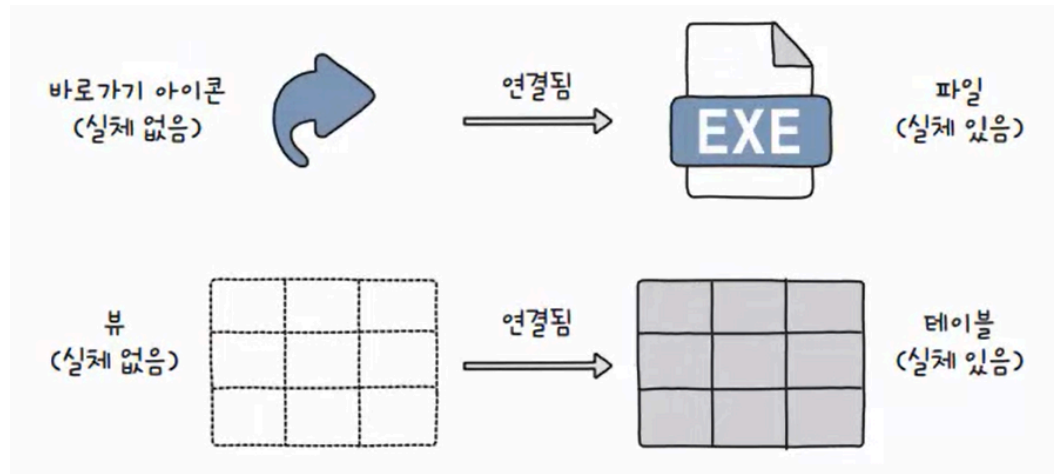
NULL 값을 허용하려면 생략하거나 NULL을 사용하고, 허용하지 않으려면 NOT NULL을 사용했다. 다만 PRIMARY KEY가 설정된 열에는 NULL값이 있을 수 없으므로 생략하면 자동으로 **NOT NULL**로 인식된다.

## 5-3 : 가상의 테이블: 뷰

**뷰(View)**는 데이터베이스 개체 중에 하나이다. 모든 데이터베이스 개체는 테이블과 관련이 있지만, 특히 뷰는 테이블과 아주 밀접하게 연관되어 있다. 뷰는 한번 생성해 놓으면 테이블이라고 생각하고 사용하고 사용해도 될 정도로 사용자들의 입장에서 테이블과 거의 동일한 개체로 취급한다.

뷰는 테이블처럼 데이터를 가지고 있지는 않다. 뷰의 실체는 SELECT 문으로 만들어져 있기 때문에 뷰에 접근하는 순간 SELECT가 실행되고 그 결과가 화면에 출력되는 방식이다. 비유하자면 **바탕 화면의 '바로 가기 아이콘' 과 비슷하다.**

뷰는 **단순 뷰**와 **복합 뷰**로 나뉘는데, 단순 뷰는 하나의 테이블과 연관된 뷰를 말하고, 복합 뷰는 2개 이상의 테이블과 연관된 뷰를 말한다.



뷰의 실습을 위해 먼저 인터넷 마켓 데이터베이스를 생성한 후 진행한다. 이전에 다운로드 받았던 소스파일을 사용한다.

MySQL을 실행해서 열린 쿼리 창은 모두 닫고 [File]→[Open SQL Script] 메뉴를 선택해서 다운로드한 market\_db.sql을 연다. 번개 아이콘을 클릭해서 모든 SQL 문을 실행한다.

## 뷰의 개념

새로운 쿼리 창을 열어 간단히 회원 테이블을 조회해본다. SELECT 문으로 아이디, 이름, 주소를 가져와서 출력한 결과를 보면 테이블의 형태를 하고 있다. 즉, SELECT 문으로 실행해서 나온 결과를 MEM\_ID, MEM\_NAME, ADDR 3개의 열을 가진 테이블의 형태로 볼 수 있다.

	MEM_ID	MEM_NAME	ADDR
▶	APN	에이핑크	경기
	BLK	블랙핑크	경남
	GRL	소녀시대	서울
	ITZ	잇지	경남
	MMU	마마무	전남
	OMY	오마이걸	서울
	RED	레드벨벳	경북
	SPC	우주소녀	서울

뷰가 바로 이런 개념이다. 그래서 **뷰의 실체가 SELECT 문**이 되는 것이다. 이 예제의 실행 결과를 V\_MEMBER라고 부른다면, 앞으로 V\_MEMBER를 그냥 테이블이라고 생각하고 접근하면 된다.



뷰의 이름만 보고도 뷰인지 알아볼 수 있도록 이름 앞에 **V\_**를 붙이는 것이 일반적이다.

뷰를 만드는 형식은 상당히 간단하며, 다음과 같다.

```
CREATE VIEW 뷰_이름
AS
SELECT 문;
```

뷰를 만든 후에 뷰에 접근하는 방식은 테이블과 동일하게 SELECT 문을 사용한다. 전체에 접근할 수도 있고, 필요하면 조건식도 테이블과 동일하게 사용할 수 있다.

```
SELECT 열_이름 FROM 뷰_이름
[WHERE 조건];
```

이제 회원 테이블의 아이디, 이름, 주소에 접근하는 뷰를 생성해본다.

```
CREATE VIEW V_MEMBER
AS
SELECT MEM_ID, MEM_NAME, ADDR FROM MEMBER;
```

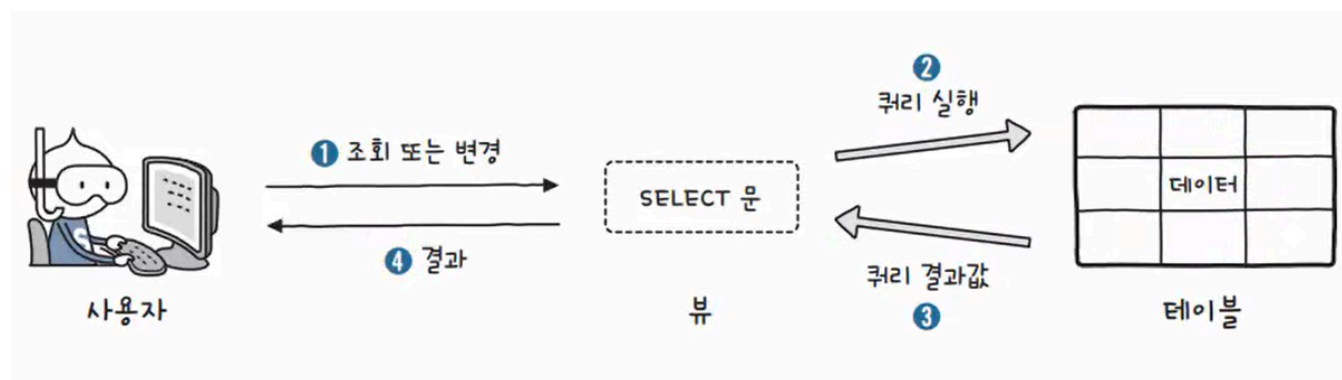
```
SELECT * FROM V_MEMBER;
```

	MEM_ID	MEM_NAME	ADDR
▶	APN	에이핑크	경기
	BLK	블랙핑크	경남
	GRL	소녀시대	서울
	ITZ	잇지	경남
	MMJ	마마무	전남
	OMY	오마이걸	서울
	RED	레드벨벳	경북
	SPC	우주소녀	서울

테이블과 마찬가지로 필요한 열만 보거나 조건식을 넣어서 사용할 수도 있다.

### 뷰의 작동 방식

사용자가 뷰에 접근하는 방식은 다음과 같다



그림에서 사용자는 뷰를 테이블이라고 생각하고 접근한다. 그러면 MySQL이 뷰 안에 있는 SELECT를 실행해서 그 결과를 사용자에게 보내주므로 사용자 입장에서는 1번과 4번만, 즉 뷰에서 모두 처리된 것으로 이해한다.

뷰는 기본적으로 **읽기 전용**으로 사용되지만, 뷰를 통해서 원본 테이블의 데이터를 수정할 수도 있다. 하지만 무조건 가능한 것은 아니고 몇 가지 조건을 만족해야 한다.

### 뷰를 사용하는 이유

뷰를 만들면 테이블과 동일하게 접근이 가능한데 굳이 테이블을 사용하지 않고 뷰를 사용하는 이유는 여러 가지가 있다.

#### 1. 보안에 도움이 된다.

앞의 예에서 만든 V\_MEMBER 뷰에는 사용자의 아이디, 이름, 주소만 있을 뿐 사용자의 다른 중요한 개인정보인 연락처, 평균 키, 데뷔 일자 등의 정보는 없다.

예를 들어, 인터넷 마켓 회원의 이름과 주소를 확인하는 작업을 진행하려고 한다. 작업량이 너무 많아 아르바이트생에게 요청할 계획이다. 그런데 아르바이트생이 회원 테이블에 접근할 수 있도록 한다면 더욱 중요한 개인 정보까지 모두 노출될 것이다. 이런 경우 정보가 공개될 수 있는 열의 데이터만 뽑아서 뷰를 생성하여 아르바이트가 전체 회원 테이블에 접근하지 못하도록 **권한을 제한**하고, 뷰에만 접근할 수 있도록 하면 보안 문제를 해결할 수 있다.



## 데이터베이스 보안

데이터베이스에서 보안은 상당히 중요한 주제이다. 기본적인 개념 정도는 알고 있는 것이 중요하다.

우리는 MySQL Workbench를 실행할 때 계속 root 사용자로 접근했다. root 사용자는 모든 권한이 있는 관리자로 테이블의 생성, 삭제는 물론 테이블의 데이터를 마음대로 조작할 수 있는 권한이 있다.

은행도 데이터베이스의 테이블에 정보를 저장한다. 통장 테이블에 고객의 예금을 관리한다고 가정했을 때, 은행에서 일하는 모든 직원에게 root의 권한을 부여한다면, 고의든 실수든 고객의 예금을 마음대로 사용할 수 있게 되며 큰 사고가 일어날 수 있다.

이런 사고를 미연에 방지하기 위해 직원의 등급에 따라 고객 정보에 접근할 수 있는 권한을 차등해서 부여한다.

## 2. 복잡한 SQL을 단순하게 만들 수 있다.

다음은 4장에서 학습했던 물건을 구매한 회원들에 대한 SQL이다.

```

SELECT B.MEM_ID,M.MEM_ID,B.PROD_NAME,M.ADDR,
       CONCAT(M.PHONE1,M.PHONE2) '연락처'
FROM BUY B
      INNER JOIN MEMBER M
      ON B.MEM_ID=M.MEM_ID;

```

내용이 길고 복잡하다. 만약 이 쿼리를 자주 사용해야 한다면, 사용자들은 매번 위와 같은 복잡한 쿼리를 입력해야 한다. 그런데 이 SQL을 다음과 같이 뷰로 생성하고 사용자들은 해당 뷰에만 접근하도록 하면 복잡한 SQL을 입력할 필요가 없다.

```

CREATE VIEW V_MEMBERBUY
AS
SELECT B.MEM_ID,M.MEM_ID,B.PROD_NAME,M.ADDR,
       CONCAT(M.PHONE1,M.PHONE2) '연락처'
FROM BUY B
      INNER JOIN MEMBER M
      ON B.MEM_ID=M.MEM_ID;

```

해당 뷰를 만들어 테이블이라 생각하고 접근하면 된다. 필요하다면 WHERE 절도 사용할 수 있다.

## 뷰의 실제 작동

뷰의 단순한 형태에 대해 살펴보았는데, 실무에서는 좀 더 복잡하게 사용한다. 실제로 사용하는 방법을 익혀본다.

### 뷰의 실제 생성, 수정, 삭제

기본적인 뷰를 생성하면서 뷰에서 사용될 열 이름을 테이블과 다르게 지정할 수 있다. 기존에 배운 **별칭**을 사용하면 되는데, 중간에 띄어쓰기 사용이 가능하다. 뷰를 조회할 때는 열 이름에 공백이 있으면 **백틱(`)**으로 묶어줘야 한다. 백틱은 키보드 1 옆에 있는 기호이다.

```

USE MARKET_DB;
CREATE VIEW V_VIEWTEST1
AS
SELECT B.MEM_ID 'MEMBER ID', M.MEM_NAME 'MEMBER NAME',
       B.PROD_NAME 'PRODUCT NAME',
       CONCAT(M.PHONE1,M.PHONE2) 'OFFICE PHONE'
FROM BUY B
      INNER JOIN MEMBER M
      ON B.MEM_ID = M.MEM_ID;

SELECT DISTINCT `MEMBER ID`, `MEMBER NAME` FROM V_VIEWTEST1; -- 백틱 사용

```

	MEMBER ID	MEMBER NAME
▶	APN	에이핑크
	BLK	블랙핑크
	GRL	소녀시대
	MMU	마마무

뷰의 수정은 **ALTER VIEW** 구문을 사용하며, 열 이름에 한글을 사용해도 된다.

```
ALTER VIEW V_VIEWTEST1
AS
SELECT B.MEM_ID '회원 아이디',M.MEM_NAME AS '회원 이름',
       B.PROD_NAME '제품 이름',
       CONCAT(M.PHONE1,M.PHONE2) '연락처'
FROM BUY B
      INNER JOIN MEMBER M
      ON B.MEM_ID = M.MEM_ID;

SELECT DISTINCT `회원 아이디`,`회원 이름` FROM V_VIEWTEST1; -- 백틱 사용
```

	회원 아이디	회원 이름
▶	APN	에이핑크
	BLK	블랙핑크
	GRL	소녀시대
	MMU	마마무

뷰의 삭제는 DROP VIEW를 사용한다.

```
DROP VIEW V_VIEWTEST1;
```



### 데이터베이스 개체의 생성/수정/삭제

데이터베이스 개체는 서로 완전히 다른 기능을 하지만 생성/수정/삭제하는 문법은 거의 동일하다.

#### 1. 생성

모든 데이터베이스 개체(테이블, 뷰, 인덱스, 스토어드 프로시저, 스토어드 함수, 트리거 등)을 생성할 때는 **CREAET 개체\_종류**를 사용한다.

#### 2.수정

이미 생성된 개체를 수정할 때는 **ALTER 개체\_종류**를 사용한다.

#### 3.삭제

기존의 개체를 삭제할 때는 **DROP 개체\_종류**를 사용한다.

## 뷰의 정보 확인

기존에 생성된 뷰에 대한 정보를 확인할 수 있다. 먼저 간단한 뷰를 생성해본다.

```
CREATE OR REPLACE VIEW V_VIEWTEST2
AS
SELECT MEM_ID, MEM_NAME, ADDR FROM MEMBER;
```



## CREATE OR REPLACE VIEW

뷰를 생성할 때 CREATE VIEW는 기존에 뷰가 있으면 오류가 발생하지만, **CREATE OR REPLACE VIEW**는 기존에 뷰가 있어도 덮어쓰이기 때문에 오류가 발생하지 않는다.

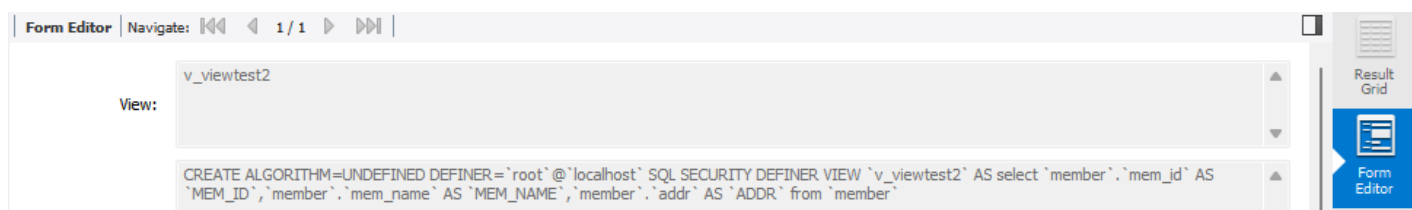
**DESCRIBE**문으로 기존 뷰의 정보를 확인할 수 있다. 다만 PRIMARY KEY 등의 KEY 정보는 확인되지 않는다.

```
DESCRIBE V_VIEWTEST2;
```

	Field	Type	Null	Key	Default	Extra
▶	MEM_ID	char(8)	NO		NULL	
	MEM_NAME	varchar(10)	NO		NULL	
	ADDR	char(2)	NO		NULL	

**SHOW CREATE VIEW** 문으로 뷰의 소스 코드도 확인할 수 있다. [RESULT GRID] 창에서 결과가 잘 보이지 않는다면 **[FORM EDITOR]** 창에서 확인하면 된다.

```
DESCRIBE V_VIEWTEST2;
```



SHOW 결과는 뷰를 생성할 때보다 훨씬 복잡해보인다. 하지만 핵심적인 코드는 생성할 때 사용한 코드와 동일하다.

## 뷰를 통한 데이터의 수정/삭제

뷰를 통해서 테이블의 데이터를 수정할 수도 있다. V\_MEMBER 뷰를 통해 데이터를 수정해본다.

```
UPDATE V_MEMBER SET ADDR = '부산' WHERE MEM_ID = 'BLK';
```

오류 없이 수정이 된다. 이번에는 데이터를 입력해본다.

```
INSERT INTO V_MEMBER(MEM_ID, MEM_NAME, ADDR) VALUES('BTS', '방탄소년단', '경기');
```

## Error Code: 1423. Field of view 'market\_db.v\_member' underlying table doesn't have a default value

다음과 같은 오류가 발생했다.

뷰가 참조하는 MEMBER 테이블의 열 중에서 MEM\_NUMBER 열은 NOT NULL로 설정되어서 반드시 입력해야 한다. 하지만 현재 V\_MEMBER에서는 MEM\_NUMBER 열을 참조하고 있지 않으므로 값을 입력할 방법이 없다.

만약 V\_MEMBER 뷰를 통해 MEMBER 테이블에 값을 입력하고 싶다면 V\_MEMBER에 MEM\_NUMBER 열을 포함하도록 뷰를 재정의하거나, 아니면 MEMBER에서 MEM\_NUMBER 열의 속성을 NULL로 변경하거나, 기본값(DEFAULT)를 지정해야 한다.

이번엔 지정한 범위로 뷰를 생성해본다. 평균 키가 167 이상인 뷰를 생성한다.

```
CREATE VIEW V_HEIGHT167
AS
SELECT * FROM MEMBER WHERE HEIGHT >= 167;
SELECT * FROM V_HEIGHT167;
```



	mem_id	mem_name	mem_number	addr	phone1	phone2	height	debut_date
▶	GRL	소녀시대	8	서울	02	44444444	168	2007-08-02
	ITZ	잇지	5	경남	NULL	NULL	167	2019-02-12
	TWC	트와이스	9	서울	02	11111111	167	2015-10-19

V\_HEIGHT167 뷰에서 키가 167 미만인 데이터를 삭제하면 당연히 167 미만인 데이터가 없으므로 삭제될 데이터도 없다.

## 뷰를 통한 데이터 입력

이번에는 V\_HEIGHT167 뷰에 167 미만인 데이터를 입력해본다.

```
INSERT INTO V_HEIGHT167 VALUES('TRA','티아라',6,'서울',NULL,NULL,159,'2005-01-01');
```

48 18:20:43 INSERT INTO V\_HEIGHT167 VALUES('TRA','티아라',6,'서울',NULL,NULL,159,'2005-01-01') 1 row(s) affected

일단 입력은 된다. 하지만 다시 뷰의 데이터를 살펴보면 새로 입력한 데이터가 보이지 않는다.

	mem_id	mem_name	mem_number	addr	phone1	phone2	height	debut_date
▶	GRL	소녀시대	8	서울	02	44444444	168	2007-08-02
	ITZ	잇지	5	경남	NULL	NULL	167	2019-02-12
	TWC	트와이스	9	서울	02	11111111	167	2015-10-19

해당 데이터는 뷰가 참조하는 **MEMBER 테이블에서는 입력이 된다**. 키가 16 이상인 데이터만 입력되도록 만들어진 뷰이기 때문이다. 이럴 때 **WITH CHECK OPTION**을 통해 뷰에 설정된 값의 범위가 벗어나는 값은 입력되지 않도록 할 수 있다. 뷰의 **WITH CHECK OPTION**은 **설정 한 범위의 데이터만 입력되도록 제한한다**.

```
ALTER VIEW V_HEIGHT167
AS
SELECT * FROM MEMBER WHERE HEIGHT >=167
WITH CHECK OPTION;

INSERT INTO V_HEIGHT167 VALUES('TOB','텔레트비',4,'영국',NULL,NULL,140,'1995-01-01');
```

ALTER 후에 조건을 벗어나는 데이터를 입력하면 오류가 발생한다.

**Error Code: 1369. CHECK OPTION failed 'market\_db.v\_height167'**

이렇게 하면 키가 167 미만인 데이터는 입력되지 않고 167 이상의 데이터만 입력된다.



### 단순 뷰와 복합 뷰

하나의 테이블로 만든 뷰를 **단순 뷰**라 하고, 두 개 이상의 테이블로 만든 뷰를 **복합 뷰**라고 한다.

위에서 봤던 JOIN한 결과를 뷰로 만든 것이 복합 뷰이다. 복합 뷰는 읽기 전용이며 데이터 입력/수정/삭제가 불가능하다.

## 뷰가 참조하는 테이블의 삭제

뷰가 참조하는 테이블을 삭제해본다. 회원 테이블과 구매 테이블 모두 삭제한다. 여러 개의 뷰가 두 테이블과 관련이 있는데도 테이블이 삭제되었다. 두 테이블 중 아무거나 연관되는 뷰를 조회해보면 다음과 같은 오류가 발생한다.

```
DROP TABLE IF EXISTS BUY, MEMBER;
SELECT * FROM V_HEIGHT167;
```

**Error Code: 1356. View 'market\_db.v\_height167' references invalid table(s) or column(s) or function(s) or definer/invoke of view lack rights to use them**

당연히 참조하는 테이블이 없기 때문에 조회할 수 없다는 메시지가 나온다. 바람직하진 않지만, **관련 뷰가 있더라도 테이블은 쉽게 삭제된다**.

뷰가 조회되지 않으면 **CHECK TABLE** 문으로 뷰의 상태를 확인해볼 수 있다. 뷰가 참조하는 테이블이 없어서 오류가 발생하는 것을 확인할 수 있다.

94 • CHECK TABLE V\_HEIGHT167;

Result Grid				
Filter Rows:		Export:		Wrap Cell Content: <a href="#">IA</a>
	Table	Op	Msg_type	Msg_text
▶	market_db.v_height167	check	Error	View 'market_db.v_height167' references invalid table(s) or column(s) or function(s) or definer/invoke of view lack rights to use them
	market_db.v_height167	check	error	Corrupt