09

CHAPTER

테이블과 뷰

Contents

1-1 테이블의 개요

- 테이블
  - 정보를 저장하는 데이터베이스의 개체
  - 데이터베이스를 구성하는 가장 기본적이고 핵심적인 요소
  - 테이블의 행은 로(row) 또는 레코드(record), 열은 칼럼(column) 또는 필드 (field)라고 함

1-2 테이블 생성

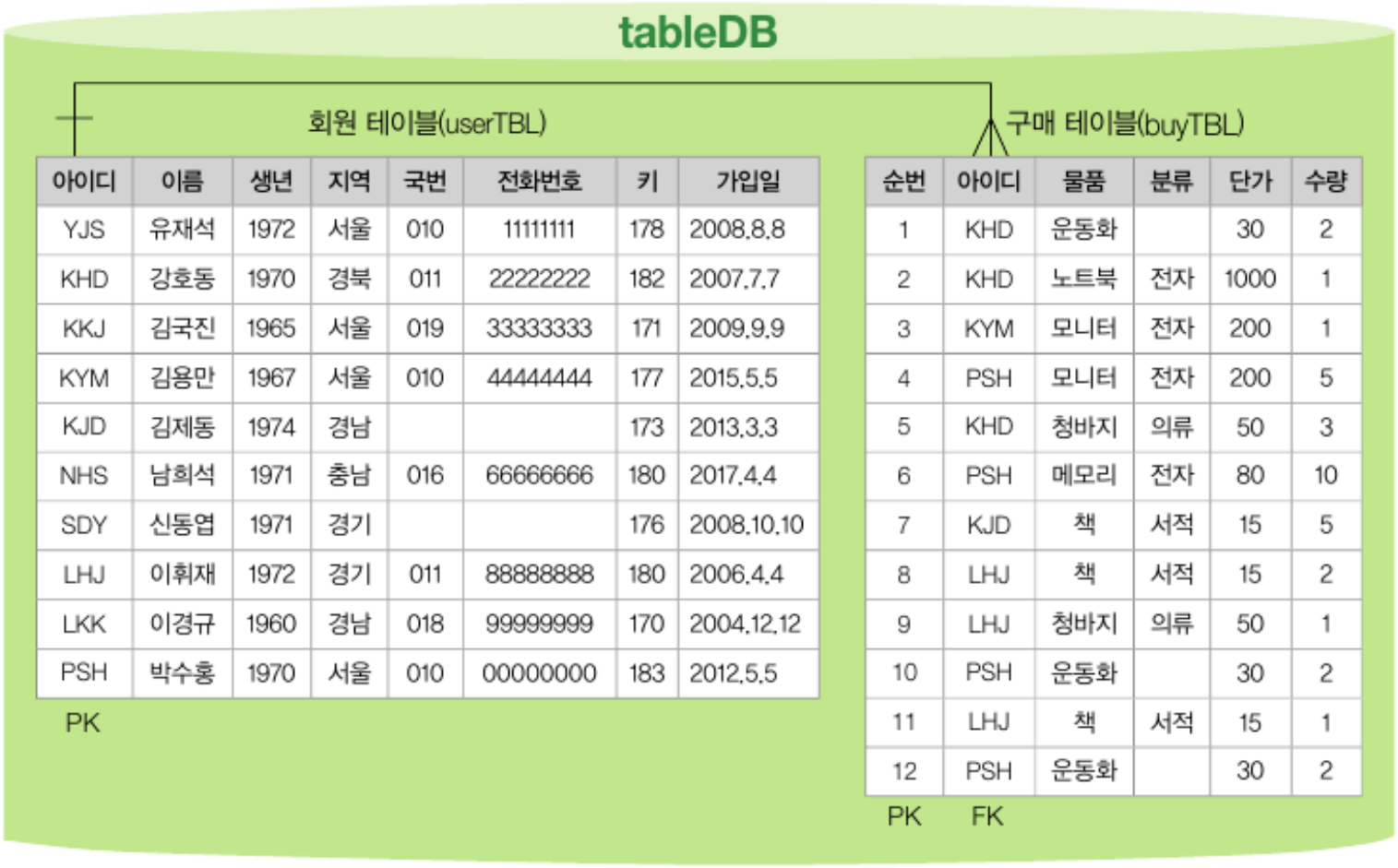
- 테이블을 생성하는 기본적인 형식

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
      col_name column_definition
   | [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_userName, ...)
        [index_option]
   | {INDEX | KEY} [index_name] [index_type] (index_col_userName, ...)
        [index_option]
   | [CONSTRAINT [symbol]] UNIQUE [INDEX | KEY]
        [index_name] [index_type] (index_col_userName, ...)
        [index_option]
   | {FULLTEXT | SPATIAL} [INDEX | KEY] [index_name] (index_col_userName, ...)
        [index_option]
   | [CONSTRAINT [symbol]] FOREIGN KEY
        [index_name] (index_col_userName, ...) reference_definition
   | CHECK (expr);

column_definition:
    data_type [NOT NULL | NULL] [DEFAULT default_value]
        [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
        [COMMENT 'string']
        [COLUMN_FORMAT {FIXED | DYNAMIC | DEFAULT}]
        [STORAGE {DISK | MEMORY | DEFAULT}]
        [reference_definition]
   | data_type [GENERATED ALWAYS] AS (expression)
        [VIRTUAL | STORED] [UNIQUE [KEY]] [COMMENT comment]
        [NOT NULL | NULL] [[PRIMARY] KEY]
```

1-2 테이블 생성

- tableDB의 구조



아이디	이름	생년	지역	국번	전화번호	키	가입일
YJS	유재석	1972	서울	010	11111111	178	2008.8.8
KHD	강호동	1970	경북	011	22222222	182	2007.7.7
KKJ	김국진	1965	서울	019	33333333	171	2009.9.9
KYM	김용만	1967	서울	010	44444444	177	2015.5.5
KJD	김제동	1974	경남			173	2013.3.3
NHS	남희석	1971	충남	016	66666666	180	2017.4.4
SDY	신동엽	1971	경기			176	2008.10.10
LHJ	이휘재	1972	경기	011	88888888	180	2006.4.4
LKK	이경규	1960	경남	018	99999999	170	2004.12.12
PSH	박수홍	1970	서울	010	00000000	183	2012.5.5

PK

순번	아이디	물품	분류	단가	수량
1	KHD	운동화		30	2
2	KHD	노트북	전자	1000	1
3	KYM	모니터	전자	200	1
4	PSH	모니터	전자	200	5
5	KHD	청바지	의류	50	3
6	PSH	메모리	전자	80	10
7	KJD	책	서적	15	5
8	LHJ	책	서적	15	2
9	LHJ	청바지	의류	50	1
10	PSH	운동화		30	2
11	LHJ	책	서적	15	1
12	PSH	운동화		30	2

PK FK

그림 9-1 샘플로 사용할 tableDB

[실습 9-1] SQL 문으로 테이블 생성하기

교재 296~300p 참고

1 데이터베이스 생성하기

1-1 앞 실습에서 사용했던 데이터베이스 제거

```
USE mysql;
DROP DATABASE IF EXISTS ShopDB;
DROP DATABASE IF EXISTS ModelDB;
DROP DATABASE IF EXISTS cookDB;
DROP DATABASE IF EXISTS movieDB;
```

1-2 tableDB 생성

```
CREATE DATABASE tableDB;
```

[실습 9-1] SQL 문으로 테이블 생성하기

교재 296~300p 참고

2 테이블 생성하기

2-1 회원 테이블과 구매 테이블 생성(테이블의 기본적인 틀만 구성)

```
USE tableDB;
DROP TABLE IF EXISTS buyTBL, userTBL;
CREATE TABLE userTBL -- 회원 테이블
( userID CHAR(8), -- 사용자 아이디
  userName VARCHAR(10), -- 이름
  birthYear INT, -- 출생 연도
  addr CHAR(2), -- 지역(경기, 서울, 경남 등으로 글자만 입력)
  mobile1 CHAR(3), -- 휴대폰의 국번(011, 016, 017, 018, 019, 010 등)
  mobile2 CHAR(8), -- 휴대폰의 나머지 전화번호(하이픈 제외)
  height SMALLINT, -- 키
  mDate DATE -- 회원 가입일
);
CREATE TABLE buyTBL -- 구매 테이블
( num INT, -- 순번(PK)
  userID CHAR(8), -- 아이디(FK)
  prodName CHAR(6), -- 물품
  groupName CHAR(4), -- 분류
  price INT, -- 단가
  amount SMALLINT -- 수량
);
```

[실습 9-1] SQL 문으로 테이블 생성하기

교재 296~300p 참고

2-2 몇 가지 옵션 추가

```
USE tableDB;
DROP TABLE IF EXISTS buyTBL, userTBL;
CREATE TABLE userTBL
( userID CHAR(8) NOT NULL,
  userName VARCHAR(10) NOT NULL,
  birthYear INT NOT NULL,
  addr CHAR(2) NOT NULL,
  mobile1 CHAR(3) NULL,
  mobile2 CHAR(8) NULL,
  height SMALLINT NULL,
  mDate DATE NULL
);
CREATE TABLE buyTBL
( num INT NOT NULL,
  userID CHAR(8) NOT NULL,
  prodName CHAR(6) NOT NULL,
  groupName CHAR(4) NULL,
  price INT NOT NULL,
  amount SMALLINT NOT NULL
);
```

[실습 9-1] SQL 문으로 테이블 생성하기

교재 296~300p 참고

2-3 각 테이블에 기본키 설정

```
DROP TABLE IF EXISTS buyTBL, userTBL;
CREATE TABLE userTBL
( userID CHAR(8) NOT NULL PRIMARY KEY,
  userName VARCHAR(10) NOT NULL,
  birthYear INT NOT NULL,
  addr CHAR(2) NOT NULL,
  mobile1 CHAR(3) NULL,
  mobile2 CHAR(8) NULL,
  height SMALLINT NULL,
  mDate DATE NULL
);
CREATE TABLE buyTBL
( num INT NOT NULL PRIMARY KEY,
  userID CHAR(8) NOT NULL,
  prodName CHAR(6) NOT NULL,
  groupName CHAR(4) NULL,
  price INT NOT NULL,
  amount SMALLINT NOT NULL
);
```

[실습 9-1] SQL 문으로 테이블 생성하기

교재 296~300p 참고

2-4 구매 테이블(buyTBL)의 순번(num) 열에 AUTO_INCREMENT 설정

```
DROP TABLE IF EXISTS buyTBL;
CREATE TABLE buyTBL
( num INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
  userID CHAR(8) NOT NULL,
  prodName CHAR(6) NOT NULL,
  groupName CHAR(4) NULL,
  price INT NOT NULL,
  amount SMALLINT NOT NULL
);
```

2-5 구매 테이블(buyTBL) 아이디(userID) 열을 회원 테이블(userTBL) 아이디(userID) 열의 외래키로 설정

```
DROP TABLE IF EXISTS buyTBL;
CREATE TABLE buyTBL
( num INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
  userID CHAR(8) NOT NULL,
  prodName CHAR(6) NOT NULL,
  groupName CHAR(4) NULL,
  price INT NOT NULL,
  amount SMALLINT NOT NULL,
  FOREIGN KEY(userID) REFERENCES userTBL(userID)
);
```

[실습 9-1] SQL 문으로 테이블 생성하기

교재 296~300p 참고

3 테이블에 데이터 입력하기

3-1 회원 테이블에 데이터 3건 입력

```
INSERT INTO userTBL VALUES ('YJS', '유재석', 1972, '서울', '010', '11111111', 178, '2008-8-8');
INSERT INTO userTBL VALUES ('KHD', '강호동', 1970, '경북', '011', '22222222', 182, '2007-7-7');
INSERT INTO userTBL VALUES ('KKJ', '김국진', 1965, '서울', '019', '33333333', 171, '2009-9-9');
```

3-2 구매 테이블에 데이터 3건 입력

```
INSERT INTO buyTBL VALUES (NULL, 'KHD', '운동화', NULL, 30, 2);
INSERT INTO buyTBL VALUES (NULL, 'KHD', '노트북', '전자', 1000, 1);
INSERT INTO buyTBL VALUES (NULL, 'KYM', '모니터', '전자', 200, 1);
```

실행 결과

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ...

[실습 9-1] SQL 문으로 테이블 생성하기

교재 296~300p 참고

3-3 회원 테이블(userTBL)에 나머지 데이터를 입력한 후 구매 테이블(buyTBL)의 세 번째 데이터 부터 다시 입력

```
INSERT INTO userTBL VALUES ('KYM', '김용만', 1967, '서울', '010', '44444444', 177, '2015-5-5');
INSERT INTO userTBL VALUES ('KJD', '김제동', 1974, '경남', NULL , NULL, 173, '2013-3-3');
INSERT INTO userTBL VALUES ('NHS', '남희석', 1971, '충남', '016', '66666666', 180, '2017-4-4');
INSERT INTO userTBL VALUES ('SDY', '신동엽', 1971, '경기', NULL, NULL, 176, '2008-10-10');
INSERT INTO userTBL VALUES ('LHJ', '이휘재', 1972, '경기', '011', '88888888', 180, '2006-4-4');
INSERT INTO userTBL VALUES ('LKK', '이경규', 1960, '경남', '018', '99999999', 170, '2004-12-12');
INSERT INTO userTBL VALUES ('PSH', '박수홍', 1970, '서울', '010', '00000000', 183, '2012-5-5');

INSERT INTO buyTBL VALUES (NULL, 'KYM', '모니터', '전자', 200, 1);
INSERT INTO buyTBL VALUES (NULL, 'PSH', '모니터', '전자', 200, 5);
INSERT INTO buyTBL VALUES (NULL, 'KHD', '청바지', '의류', 50, 3);
INSERT INTO buyTBL VALUES (NULL, 'PSH', '메모리', '전자', 80, 10);
INSERT INTO buyTBL VALUES (NULL, 'KJD', '책', '서적', 15, 5);
INSERT INTO buyTBL VALUES (NULL, 'LHJ', '책', '서적', 15, 2);
INSERT INTO buyTBL VALUES (NULL, 'LHJ', '청바지', '의류', 50, 1);
INSERT INTO buyTBL VALUES (NULL, 'PSH', '운동화', NULL, 30, 2);
INSERT INTO buyTBL VALUES (NULL, 'LHJ', '책', '서적', 15, 1);
INSERT INTO buyTBL VALUES (NULL, 'PSH', '운동화', NULL, 30, 2);
```

2-1 제약 조건의 개요

- 제약 조건(constraint)
  - 데이터의 무결성을 지키기 위해 제한하는 조건
  - 특정 데이터를 입력할 때 무조건 입력되는 것이 아니라 제약 조건을 만족했을 때만 입력되도록 설정하는 것

- 대부분의 DBMS에서는 데이터의 무결성을 보장하기 위해 다음과 같은 제약 조건을 제공
  - 기본키 제약 조건
  - 외래키 제약 조건
  - UNIQUE 제약 조건
  - DEFAULT 제약 조건
  - NULL 값 허용

2-2 기본키 제약 조건

- 기본키 제약조건
  - 기본키에 입력되는 값은 중복될 수 없으며 NULL 값이 올 수도 있음
  - 회원 테이블에 입력되는 회원 아이디는 당연히 중복될 수 없고 비어 있을 수도 없음

```
DROP TABLE IF EXISTS buyTBL, userTBL;
CREATE TABLE userTBL
( userID CHAR(8) NOT NULL PRIMARY KEY,
  userName VARCHAR(10) NOT NULL,
  birthYear INT NOT NULL
);
```

- 기본키에 설정된 제약 조건의 이름을 DESCRIBE 문으로 확인

```
DESCRIBE userTBL;
```

Field	Type	Null	Key	Default	Extra
userID	char(8)	NO	PRI	NULL	
userName	varchar(10)	NO		NULL	
birthYear	int(11)	NO		NULL	

2-2 기본키 제약 조건

- 사용자가 기본키를 설정하면서 제약 조건의 이름을 직접 지정할 수도 있음

```
DROP TABLE IF EXISTS userTBL;
CREATE TABLE userTBL
( userID CHAR(8) NOT NULL,
  userName VARCHAR(10) NOT NULL,
  birthYear INT NOT NULL,
  CONSTRAINT PRIMARY KEY PK_userTBL_userID (userID)
);
```

- 이미 만들어진 테이블을 수정하는 ALTER TABLE 문을 사용하여 제약 조건 설정

```
DROP TABLE IF EXISTS userTBL;
CREATE TABLE userTBL
( userID CHAR(8) NOT NULL,
userName VARCHAR(10) NOT NULL,
birthYear INT NOT NULL
);
ALTER TABLE userTBL
ADD CONSTRAINT PK_userTBL_userID
PRIMARY KEY (userID);
```

  - ALTER TABLE userTBL: 회원 테이블(userTBL)을 변경
  - ADD CONSTRAINT PK_userTBL_userID: 제약 조건을 추가하고 제약 조건 이름을 ' PK_ userTBL_userID' 로 명명
  - PRIMARY KEY (userID): 추가할 제약 조건은 기본키 제약 조건이고 제약 조건을 설정할 열은 userID

2-2 기본키 제약 조건

- 2개 또는 그 이상의 열을 합쳐서 하나의 기본키로 설정하는 경우('제품 테이블' 예)
  - (제품 코드, 제품 일련번호)의 쌍을 기본키로 사용

표 9-1 제품 테이블

제품 코드	제품 일련번호	제조 일자	현 상태
AAA	0001	2019. 10. 10.	판매 완료
AAA	0002	2019. 10. 11.	매장 진열
BBB	0001	2019. 10. 12.	재고 창고
CCC	0001	2019. 10. 13.	판매 완료
CCC	0002	2019. 10. 14.	매장 진열

```
DROP TABLE IF EXISTS prodTBL;
CREATE TABLE prodTBL
( prodCode CHAR(3) NOT NULL,
  prodID CHAR(4) NOT NULL,
  prodDate DATETIME NOT NULL,
  prodState CHAR(10) NULL
 );
ALTER TABLE prodTBL
      ADD CONSTRAINT PK_prodTbl_proCode_prodID
      PRIMARY KEY (prodCode, prodID);
```

2-2 기본키 제약 조건

- 기본키를 CREATE TABLE 문 안에서 설정하는 방법

```
DROP TABLE IF EXISTS prodTBL;
CREATE TABLE prodTBL
( prodCode CHAR(3) NOT NULL,
  prodID CHAR(4) NOT NULL,
  prodDate DATETIME NOT NULL,
  prodState CHAR(10) NULL,
  CONSTRAINT PK_prodTbl_proCode_prodID
      PRIMARY KEY (prodCode, prodID)
);
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible
prodtbl	0	PRIMARY	1	prodCode	A	0	NULL	NULL		BTREE			YES
prodtbl	0	PRIMARY	2	prodID	A	0	NULL	NULL		BTREE			YES

2-3 외래키 제약 조건

- 외래키 제약 조건
  - 두 테이블 사이의 관계를 선언함으로써 데이터의 무결성을 보장하는 역할을 함
  - 테이블 사이에 외래키 관계를 설정하면 하나의 테이블이 다른 테이블에 의존하게 됨
  - 외래키를 정의하는 테이블을 '외래키 테이블'이라 하고, 외래키에 의해서 참조가 되는 테이블을 '기준 테이블'이라 함
  - 외래키 테이블에 데이터를 입력할 때는 기준 테이블을 참조
  - 외래키 테이블이 참조하는 기준 테이블의 열은 반드시 기본키(PK)이거나 UNIQUE 제약 조건으로 설정되어 있어야 함

2-3 외래키 제약 조건

- 외래키 설정 방법
  - 외래키는 CREATE TABLE 문 내부에 FOREIGN KEY 키워드를 이용하여 설정

```
DROP TABLE IF EXISTS buyTBL, userTBL;
CREATE TABLE userTBL
( userID CHAR(8) NOT NULL PRIMARY KEY,
  userName VARCHAR(10) NOT NULL,
  birthYear INT NOT NULL
);
CREATE TABLE buyTBL
( num INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
  userID CHAR(8) NOT NULL,
  prodName CHAR(6) NOT NULL,
  FOREIGN KEY (userID) REFERENCES userTBL (userID)
);
```

- 외래키 제약 조건의 이름 지정
  - 마지막 행에서 쉼표로 분리한 후 아래에 작성

```
DROP TABLE IF EXISTS buyTBL;
CREATE TABLE buyTBL
( num INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
  userID CHAR(8) NOT NULL,
 prodName CHAR(6) NOT NULL,
  CONSTRAINT FK_userTBL_buyTBL FOREIGN KEY (userID) REFERENCES userTBL (userID)
);
```

2-3 외래키 제약 조건

- ALTER TABLE문을 이용하여 외래키 제약 조건 지정

```
DROP TABLE IF EXISTS buyTBL;
CREATE TABLE buyTBL
( num INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
  userID CHAR(8) NOT NULL,
  prodName CHAR(6) NOT NULL
);
ALTER TABLE buyTBL
      ADD CONSTRAINT FK_userTBL_buyTBL
      FOREIGN KEY (userID)
      REFERENCES userTBL (userID);
```

  - ALTER TABLE buyTBL: 구매 테이블(buyTBL)을 수정
  - ADD CONSTRAINT FK_userTBL_buyTBL: 제약 조건을 추가하고 제약 조건 이름을 ' FK_ userTBL_buyTBL' 로 명명
  - FOREIGN KEY (userID): 외래키 제약 조건을 구매 테이블(buyTBL)의 아이디(userID) 열에 설정
  - REFERENCES userTBL (userID): 참조할 기준 테이블은 회원 테이블(userTBL)의 아이디 (userID) 열

2-3 외래키 제약 조건

- SHOW INDEX FROM buyTBL; 문으로 확인 결과

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible
buytbl	0	PRIMARY	1	num	A	0	NULL	NULL		BTREE			YES
buytbl	1	FK_userTbl_buyTbl	1	userID	A	0	NULL	NULL		BTREE			YES

- 외래키 설정시 ON DELETE CASCADE 또는 ON UPDATE CASCADE 옵션 사용

```
ALTER TABLE buyTBL
    DROP FOREIGN KEY FK_userTBL_buyTBL; -- 외래키 제거
ALTER TABLE buyTBL
    ADD CONSTRAINT FK_userTBL_buyTBL
    FOREIGN KEY (userID)
    REFERENCES userTBL (userID)
    ON UPDATE CASCADE;
```

2-4 UNIQUE 제약 조건

- UNIQUE 제약 조건
  - 중복되지 않는 유일한 값을 입력해야 하는 조건
  - 기본키 제약 조건과 거의 비슷하지만 NULL 값은 허용

- 기존 회원 테이블에 email 열을 추가한 경우(두 CREATE TABLE 문은 동일한 결과를 출력)

```
USE tableDB;
DROP TABLE IF EXISTS buyTBL, userTBL;

CREATE TABLE userTBL
( userID CHAR(8) NOT NULL PRIMARY KEY,
  userName VARCHAR(10) NOT NULL,
  birthYear INT NOT NULL,
  email CHAR(30) NULL UNIQUE
);

DROP TABLE IF EXISTS userTBL;

CREATE TABLE userTBL
( userID CHAR(8) NOT NULL PRIMARY KEY,
  userName VARCHAR(10) NOT NULL,
  birthYear INT NOT NULL,
  email CHAR(30) NULL,
  CONSTRAINT AK_email UNIQUE (email)
);
```

2-5 DEFAULT 제약 조건

- DEFAULT 제약 조건
  - 값을 입력하지 않았을 때 자동으로 입력되는 기본 값을 정의하는 조건

- 예
  - 출생 연도를 입력하지 않았다면 '-1', 주소를 입력하지 않았다면 '서울', 키를 입력하지 않았다면 '170'을 자동으로 입력하는 구문

```
DROP TABLE IF EXISTS userTBL;
CREATE TABLE userTBL
( userID CHAR(8) NOT NULL PRIMARY KEY,
  userName VARCHAR(10) NOT NULL,
  birthYear int NOT NULL DEFAULT -1,
  addr CHAR(2) NOT NULL DEFAULT '서울',
  mobile1 CHAR(3) NULL,
  mobile2 CHAR(8) NULL,
  height smallint NULL DEFAULT 170,
  mDate date NULL
);
```

2-5 DEFAULT 제약 조건

- ALTER TABLE 문에서 DEFAULT 제약 조건 지정시 ALTER COLUMN 구문 사용

```
DROP TABLE IF EXISTS userTBL;
CREATE TABLE userTBL
( userID CHAR(8) NOT NULL PRIMARY KEY,
userName VARCHAR(10) NOT NULL,
birthYear int NOT NULL,
addr CHAR(2) NOT NULL,
mobile1 CHAR(3) NULL,
mobile2 CHAR(8) NULL,
height smallint NULL,
mDate date NULL
);
ALTER TABLE userTBL
      ALTER COLUMN birthYear SET DEFAULT -1;
ALTER TABLE userTBL
     ALTER COLUMN addr SET DEFAULT '서울';
ALTER TABLE userTBL
     ALTER COLUMN height SET DEFAULT 170;
```

2-5 DEFAULT 제약 조건

- DEFAULT 제약 조건이 설정된 열에는 다음과 같은 방법으로 데이터 입력

```
-- 입력 데이터가 default이면 DEFAULT 문으로 설정된 값을 자동 입력한다.
INSERT INTO userTBL VALUES ('YBJ', '유병재', default, default, '010', '12345678', default, '2019.12.12');
-- 열 이름이 명시되지 않으면 DEFAULT 문으로 설정된 값을 자동 입력한다.
INSERT INTO userTBL (userID, userName) VALUES ('PNR', '박나래');
-- 값이 직접 명시되어 있으면 DEFAULT 문으로 설정된 값을 무시한다.
INSERT INTO userTBL VALUES ('WB', '원빈', 1982, '대전', '010', '98765432', 176, '2020.5.5');

SELECT * FROM userTBL;
```

userID	userName	birthYear	addr	mobile1	mobile2	height	mDate
PNR	박나래	-1	서울	NULL	NULL	170	NULL
WB	원빈	1982	대전	010	98765432	176	2020-05-05
YBJ	유병재	-1	서울	010	12345678	170	2019-12-12
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2-6 NULL 값 허용

- NULL 값 허용
  - 테이블을 정의할 때 특정 열에 NULL 값이 입력되는 것을 허용하려면 NULL로 설정하고, 허용하지 않으려면 NOT NULL로 설정하는 조건
  - 기본키로 설정된 열에는 NULL 값이 올 수 없으므로 특별히 설정하지 않아도 자동으로 NOT NULL로 인식
  - NULL 값은 '아무것도 없다'는 의미로 0, 빈 문자, 공백과 다르니 주의해야 함

[실습 9-2] 테이블 압축하기

교재 311~312p 참고

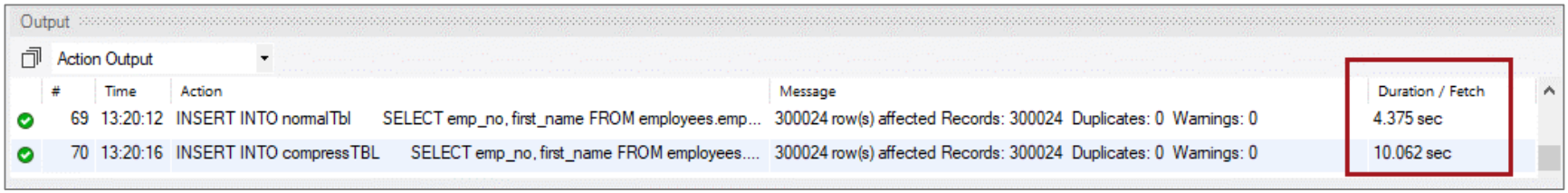
1 압축 효과를 비교할 두 테이블 만들기

1-1 테스트용 데이터베이스를 생성한 후 열이 동일한 2개의 테이블을 만들기

```
CREATE DATABASE IF NOT EXISTS compressDB;
USE compressDB;
CREATE TABLE normalTBL (emp_no INT, first_name VARCHAR(14));
CREATE TABLE compressTBL (emp_no INT, first_name VARCHAR(14))
      ROW_FORMAT=COMPRESSED;
```

1-2 두 테이블에 데이터를 30만 건 정도 입력

```
INSERT INTO normalTbl
      SELECT emp_no, first_name FROM employees.employees;
INSERT INTO compressTBL
      SELECT emp_no, first_name FROM employees.employees;
```



[실습 9-2] 테이블 압축하기

교재 311~312p 참고

2 압축 효과 보기

2-1 입력된 두 테이블의 상태 확인

```
SHOW TABLE STATUS FROM compressDB;
```

Name	Engine	Version	Row_format	Rows	Avg_row_length	Data_length	Max_data_length	Index_length	Data_free	Auto_increment
compresstbl	InnoDB	10	Compressed	299560	25	7626752	0	0	2097152	NULL
normaltbl	InnoDB	10	Dynamic	299533	40	12075008	0	0	4194304	NULL

2-2 실습한 데이터베이스 삭제

```
DROP DATABASE IF EXISTS compressDB;
```

3-2 임시 테이블

- 임시 테이블
  - 임시로 잠깐 사용하는 테이블

- 임시 테이블 생성 형식

```
CREATE TEMPORARY TABLE [IF NOT EXISTS] 테이블이름
(열정의 ...)
```

- 임시 테이블 사용법
  - 사용법은 일반 테이블과 동일
  - 단, 세션(session) 내에서만 존재하며 세션이 닫히면 자동으로 삭제
  - 테이블을 생성한 클라이언트만 접근할 수 있고 다른 클라이언트는 접근할 수 없음
  - 임시 테이블의 이름은 데이터베이스 내 다른 테이블의 이름과 동일하게 지을 수 있음(이름을 동일하게 하는 경우 임시 테이블이 있는 동안 기존 테이블에 접근할 수 없으며 무조건 임시 테이블에만 접근할 수 있음)

- 임시 테이블이 제거되는 시점
  - 사용자가 DROP TABLE로 직접 삭제하는 경우
  - Workbench를 종료하거나 MySQL 클라이언트를 종료하는 경우
  - MySQL 서비스를 재시작하는 경우

[실습 9-3] 테이블 압축하기

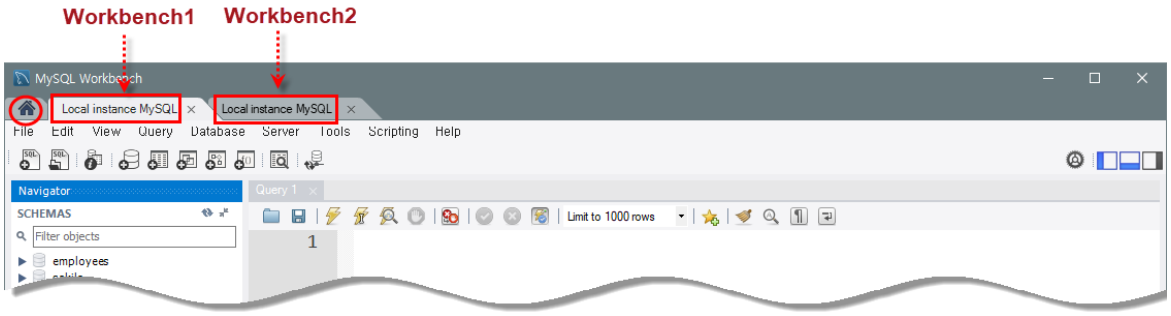
교재 313~315p 참고

1 2개의 세션 만들기

1-1 모든 쿼리 창을 닫고 Workbench를 종료

1-2 Workbench를 실행한 후 ' Local instance MySQL'에 접속(' Workbench 1')

1-3 [Home] 탭을 클릭하고 다시 ' Local instance MySQL'에 접속(' Workbench 2')



2 (Workbench 1) 임시 테이블 생성하기

2-1 임시 테이블 2개 생성

```
USE employees;
CREATE TEMPORARY TABLE IF NOT EXISTS tempTBL (id INT, userName CHAR(5));
CREATE TEMPORARY TABLE IF NOT EXISTS employees (id INT, userName CHAR(5));
DESCRIBE tempTBL;
DESCRIBE employees;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	YES		NULL	
userName	char(5)	YES		NULL	

[실습 9-3] 테이블 압축하기

교재 313~315p 참고

2-2 임시 테이블에 데이터 입력 후 확인

```
INSERT INTO tempTBL VALUES (1, 'Cook');
INSERT INTO employees VALUES (2, 'MySQL');
SELECT * FROM tempTBL;
SELECT * FROM employees;
```

	id	userName
	1	Cook

	id	userName
	2	MySQL

3 (Workbench 2) Workbench 1에서 생성한 테이블에 접근하기

3-1 Workbench 2로 이동하여 Workbench 1에서 생성한 테이블에 접근

```
1 USE employees;
2 SELECT * FROM tempTBL;
3 SELECT * FROM employees;
```

4 (Workbench 1) 임시 테이블 삭제하기

4-1 Workbench 1로 돌아와 DROP TABLE tempTBL; 문으로 임시 테이블 삭제

4-2 Workbench 1을 종료한 후 다시 접속하여 다음 쿼리 실행(기존의 employees 테이블이 조회)

```
USE employees;
SELECT * FROM employees;
```

4-1 테이블 삭제

- DROP TABLE 문
  - 테이블 삭제

- 테이블 삭제시 주의점
  - 외래키 제약 조건에 걸려 있는 기준 테이블은 삭제할 수 없음(외래키가 생성된 외래키 테이블을 삭제한 후 기준 테이블을 삭제해야 함)
  - 여러 개의 테이블을 동시에 삭제할 때는 DROP TABLE 테이블1, 테이블2, 테이블3;과 같이 계속 나열

4-2 테이블 수정

- ALTER TABLE 문
  - 이미 생성된 테이블의 구조에 무엇인가를 추가하거나 삭제하거나 변경

```
ALTER [IGNORE] TABLE tbl_name
    [alter_specification [, alter_specification] ...]
    [partition_options]

alter_specification:
    table_options
 | ADD [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
 | ADD [COLUMN] (col_name column_definition, ...)
 | ADD {INDEX | KEY} [index_name]
        [index_type] (index_col_userName, ...) [index_option] ...
 | ADD [CONSTRAINT [symbol]] PRIMARY KEY
        [index_type] (index_col_userName, ...) [index_option] ...
 | ADD [CONSTRAINT [symbol]]
        UNIQUE [INDEX | KEY] [index_name]
        [index_type] (index_col_userName, ...) [index_option] ...
 | ADD FULLTEXT [INDEX | KEY] [index_name]
        (index_col_userName, ...) [index_option]
 | ADD [CONSTRAINT [symbol]]
        FOREIGN KEY [index_name] (index_col_userName, ...)
        reference_definition
 | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
 | CHANGE [COLUMN] old_col_name new_col_name column_definition
        [FIRST | AFTER col_name]
 | MODIFY [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
 | DROP [COLUMN] col_name
 | DROP PRIMARY KEY
 | DROP {INDEX | KEY} index_name
 | DROP FOREIGN KEY fk_symbol
 | DISABLE KEYS
 | ENABLE KEYS
 | RENAME [TO | AS] new_tbl_name
 | RENAME {INDEX | KEY} old_index_name TO new_index_name
 | ORDER BY col_name [, col_name] ...
```

4-2 테이블 수정

- 회원 테이블(userTBL)에 회원의 홈페이지 주소 추가

```
USE tableDB;
ALTER TABLE userTBL
    ADD homepage VARCHAR(30) -- 열 추가
        DEFAULT 'http://www.hanbit.co.kr' -- 디폴트 값
        NULL; -- NULL 허용
```

- 회원 테이블(userTBL)에서 전화번호 열 삭제

```
ALTER TABLE userTBL
    DROP COLUMN mobile1;
```

- 회원 테이블(userTBL)에서 회원 이름(name) 열의 이름을 uName으로, 데이터 형식을 VARCHAR(20)으로 변경하고 NULL 값도 허용

```
ALTER TABLE userTBL
    CHANGE COLUMN userName uName VARCHAR(20) NULL;
```

- 회원 테이블(userTBL)의 기본키 삭제

```
ALTER TABLE userTBL
    DROP PRIMARY KEY;
```

```
ALTER TABLE buyTBL
    DROP FOREIGN KEY 외래키이름;
```

[실습 9-4] 테이블 종합 실습하기

교재 318~326p 참고

1 테이블 새로 만들기

1-1 [그림 9-1]의 테이블 만들기

```
USE tableDB;
DROP TABLE IF EXISTS buyTBL, userTBL;

CREATE TABLE userTBL
( userID CHAR(8),
  userName VARCHAR(10),
  birthYear INT,
  addr CHAR(2),
  mobile1 CHAR(3),
  mobile2 CHAR(8),
  height SMALLINT,
  mDate DATE
);

CREATE TABLE buyTBL
( num INT AUTO_INCREMENT PRIMARY KEY,
  userID CHAR(8),
  prodName CHAR(6),
  groupName CHAR(4),
  price INT,
  amount SMALLINT
);
```

[실습 9-4] 테이블 종합 실습하기

교재 318~326p 참고

1 테이블 새로 만들기

1-2 각 테이블에 데이터 4건씩 입력(강호동의 출생 연도는 모른다고 가정하여 NULL 값을 넣고, 김국진의 출생 연도는 1865년으로 잘못 입력

```
INSERT INTO userTBL VALUES ('YJS', '유재석', 1972, '서울', '010', '11111111', 178, '2008-8-8');
INSERT INTO userTBL VALUES ('KHD', '강호동', NULL, '경북', '011', '22222222', 182, '2007-7-7');
INSERT INTO userTBL VALUES ('KKJ', '김국진', 1865, '서울', '019', '33333333', 171, '2009-9-9');
INSERT INTO userTBL VALUES ('KYM', '김용만', 1967, '서울', '010', '44444444', 177, '2015-5-5');

INSERT INTO buyTBL VALUES (NULL, 'KHD', '운동화', NULL, 30, 2);
INSERT INTO buyTBL VALUES (NULL, 'KHD', '노트북', '전자', 1000, 1);
INSERT INTO buyTBL VALUES (NULL, 'KYM', '모니터', '전자', 200, 1);
INSERT INTO buyTBL VALUES (NULL, 'PSH', '모니터', '전자', 200, 5);
```

2 기본키 제약 조건 설정하기

2-1 회원 테이블(userTBL)에 기본키 제약 조건 설정

```
ALTER TABLE userTBL
      ADD CONSTRAINT PK_userTBL_userID
      PRIMARY KEY (userID);
```

[실습 9-4] 테이블 종합 실습하기

교재 318~326p 참고

2-2 DESC userTBL; 문으로 테이블 정보 확인(기본키로 설정한 userID열이 NOT NULL로 바뀜)

Field	Type	Null	Key	Default	Extra
userID	char(8)	NO	PRI	NULL	
userName	varchar(10)	YES		NULL	
birthYear	int(11)	YES		NULL	
addr	char(2)	YES		NULL	
mobile1	char(3)	YES		NULL	
mobile2	char(8)	YES		NULL	
height	smallint(6)	YES		NULL	
mDate	date	YES		NULL	

3 외래키 제약 조건 설정하기

3-1 구매 테이블(buyTBL)의 userID 열에 외래키 설정

```
ALTER TABLE buyTBL
    ADD CONSTRAINT FK_userTBL_buyTBL
    FOREIGN KEY (userID)
    REFERENCES userTBL (userID);
```

실행 결과

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`tabledb`.`#sql-4f4_15`, CONSTRAINT `FK_userTBL_buyTBL` FOREIGN KEY (`userID`) REFERENCES `usertbl` (`userid`))

[실습 9-4] 테이블 종합 실습하기

교재 318~326p 참고

3-2 구매 테이블의 PSH 행 삭제하고 다시 외래키 설정

```
DELETE FROM buyTBL WHERE userID = 'PSH';
ALTER TABLE buyTBL
      ADD CONSTRAINT FK_userTBL_buyTBL
      FOREIGN KEY (userID)
      REFERENCES userTBL (userID);
```

3-3 구매 테이블의 네 번째 데이터 다시 입력

```
INSERT INTO buyTBL VALUES (NULL, 'PSH', '모니터', '전자', 200, 5);
```

실행 결과

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`tabledb`.`buytbl`, CONSTRAINT `FK_userTBL_buyTBL` FOREIGN KEY (`userID`) REFERENCES `usertbl` (`userID`))

[실습 9-4] 테이블 종합 실습하기

교재 318~326p 참고

3-4 외래키 제약 조건을 비활성화한 후 9건의 데이터를 입력하고 다시 외래키 제약 조건을 활성화

```
SET foreign_key_checks = 0;
INSERT INTO buyTBL VALUES (NULL, 'PSH', '모니터', '전자', 200, 5);
INSERT INTO buyTBL VALUES (NULL, 'KHD', '청바지', '의류', 50, 3);
INSERT INTO buyTBL VALUES (NULL, 'PSH', '메모리', '전자', 80, 10);
INSERT INTO buyTBL VALUES (NULL, 'KJD', '책', '서적', 15, 5);
INSERT INTO buyTBL VALUES (NULL, 'LHJ', '책', '서적', 15, 2);
INSERT INTO buyTBL VALUES (NULL, 'LHJ', '청바지', '의류', 50, 1);
INSERT INTO buyTBL VALUES (NULL, 'PSH', '운동화', NULL, 30, 2);
INSERT INTO buyTBL VALUES (NULL, 'LHJ', '책', '서적', 15, 1);
INSERT INTO buyTBL VALUES (NULL, 'PSH', '운동화', NULL, 30, 2);
SET foreign_key_checks = 1;
```

4 CHECK 제약 조건 설정하기

4-1 회원 테이블(userTBL)의 출생 연도를 1900년부터 현재까지만 설정 가능하도록 CHECK 제약 조건 설정

```
ALTER TABLE userTBL
      ADD CONSTRAINT CK_birthYear
      CHECK (birthYear >= 1900 AND birthYear <= YEAR(CURDATE()));
```

[실습 9-4] 테이블 종합 실습하기

교재 318~326p 참고

4-2 회원 테이블의 나머지 데이터도 입력

```
INSERT INTO userTBL VALUES ('KJD', '김제동', 1974, '경남', NULL, NULL, 173, '2013-3-3');
INSERT INTO userTBL VALUES ('NHS', '남희석', 1971, '충남', '016', '66666666', 180, '2017-4-4');
INSERT INTO userTBL VALUES ('SDY', '신동엽', 1971, '경기', NULL, NULL, 176, '2008-10-10');
INSERT INTO userTBL VALUES ('LHJ', '이휘재', 1972, '경기', '011', '88888888', 180, '2006-4-4');
INSERT INTO userTBL VALUES ('LKK', '이경규', 1960, '경남', '018', '99999999', 170, '2004-12-12');
INSERT INTO userTBL VALUES ('PSH', '박수홍', 1970, '서울', '010', '00000000', 183, '2012-5-5');
```

5 데이터베이스 활용하기

5-1 박수홍 회원 아이디를 PSH에서 PARK로 변경

```
UPDATE userTBL SET userID = 'PARK' WHERE userID='PSH';
```

실행 결과

Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`tabledb`.`buytbl`, CONSTRAINT `FK_userTBL_buyTBL` FOREIGN KEY (`userID`) REFERENCES `usertbl` (`userID`))

5-2 시스템 변수 foreign_key_checks를 활용하여 외래키 제약 조건을 비활성화

```
SET foreign_key_checks = 0;
UPDATE userTBL SET userID = 'PARK' WHERE userID='PSH';
SET foreign_key_checks = 1;
```

[실습 9-4] 테이블 종합 실습하기

교재 318~326p 참고

5-3 구매 테이블과 회원 테이블 조인

```
SELECT B.userID, U.userName, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2) AS '연락처'
   FROM buyTBL B
      INNER JOIN userTBL U
         ON B.userID = U.userID;
```

userID	userName	prodName	addr	연락처
KHD	강호동	운동화	경북	22222233
KHD	강호동	노트북	경북	22222233
KHD	강호동	청바지	경북	22222233
KJD	김제동	책	경남	NULL
KYM	김용만	모니터	서울	44444454
LHJ	이휘재	책	경기	88888899
LHJ	이휘재	청바지	경기	88888899
LHJ	이휘재	책	경기	88888899

5-4 구매 테이블에 8건만 입력한 것은 아닌지 확인

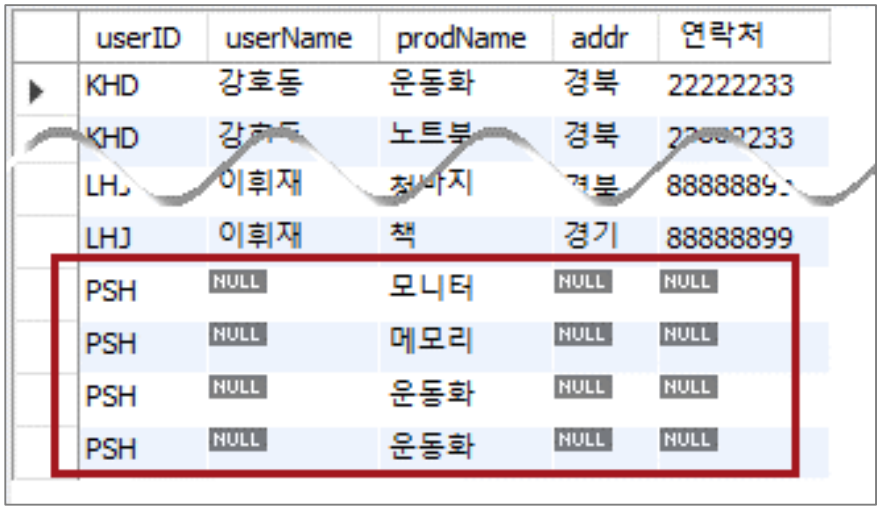
```
SELECT COUNT( * ) FROM buyTBL;
```

[실습 9-4] 테이블 종합 실습하기

교재 318~326p 참고

5-5 외부 조인으로 구매 테이블의 내용 모두 출력(아이디로 정렬)

```
SELECT B.userID, U.userName, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2) AS '연락처'
   FROM buyTBL B
      LEFT OUTER JOIN userTBL U
         ON B.userID = U.userID
   ORDER BY B.userID;
```



userID	userName	prodName	addr	연락처
KHD	강호동	운동화	경북	22222233
KHD	강호동	노트북	경북	22222233
LHJ	이휘재	청바지	경북	88888899
LHJ	이휘재	책	경기	88888899
PSH	NULL	모니터	NULL	NULL
PSH	NULL	메모리	NULL	NULL
PSH	NULL	운동화	NULL	NULL
PSH	NULL	운동화	NULL	NULL

5-6 박수홍의 아이디를 원래 아이디로 돌려놓음

```
SET foreign_key_checks = 0;
UPDATE userTBL SET userID = 'PSH' WHERE userID='PARK';
SET foreign_key_checks = 1;
```

[실습 9-4] 테이블 종합 실습하기

교재 318~326p 참고

5-7 외래키 제약 조건을 삭제한 후 다시 ON UPDATE CASCADE 옵션과 함께 설정

```
ALTER TABLE buyTBL
      DROP FOREIGN KEY FK_userTBL_buyTBL;
ALTER TABLE buyTBL
      ADD CONSTRAINT FK_userTBL_buyTBL
            FOREIGN KEY (userID)
            REFERENCES userTBL (userID)
            ON UPDATE CASCADE;
```

5-8 회원 테이블에서 박수홍의 아이디를 PSH에서 PARK로 변경한 후 구매 테이블도 바뀌었는지 확인

```
UPDATE userTBL SET userID = 'PARK' WHERE userID='PSH';
SELECT B.userID, U.userName, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2) AS '연락처'
   FROM buyTBL B
      INNER JOIN userTBL U
         ON B.userID = U.userID
   ORDER BY B.userID;
```

userID	userName	prodName	addr	연락처
KHD	강호동	운동화	경북	01122222222
KHD	강호동	노트북	경북	01122222222
LHJ	이휘재	청바지	경북	01188888888
LHJ	이휘재	책	경기	01188888888
PARK	박수홍	모니터	서울	01000000000
PARK	박수홍	메모리	서울	01000000000
PARK	박수홍	운동화	서울	01000000000
PARK	박수홍	운동화	서울	01000000000

[실습 9-4] 테이블 종합 실습하기

교재 318~326p 참고

5-9 박수홍(PARK)이 회원 탈퇴를 하면(회원 테이블에서 삭제되면) 구매한 기록도 삭제되는지 확인

```
DELETE FROM userTBL WHERE userID = 'PARK';
```

5-10 ON DELETE CASCADE 문을 추가하여, 기준 테이블의 행 데이터를 삭제할 때외래키 테이블의 연관된 행 데이터도 함께 삭제되도록 설정

```
ALTER TABLE buyTBL
    DROP FOREIGN KEY FK_userTBL_buyTBL;
ALTER TABLE buyTBL
    ADD CONSTRAINT FK_userTBL_buyTBL
        FOREIGN KEY (userID)
        REFERENCES userTBL (userID)
        ON UPDATE CASCADE
        ON DELETE CASCADE;
```

5-11 박수홍(PARK)이 회원 탈퇴를 하면(회원 테이블에서 삭제되면) 구매한 기록도 삭제되는지 확인

```
DELETE FROM userTBL WHERE userID = 'PARK';
SELECT * FROM buyTBL;
```

num	userID	prodName	groupName	price	amount
1	KHD	운동화	NULL	30	2
2	KHD	노트북	전자	1000	1
3	KYM	모니터	전자	200	1
7	KHD	청바지	의류	50	3
9	KJD	책	서적	15	5
10	LHJ	책	서적	15	2
11	LHJ	청바지	의류	50	1
13	LHJ	책	서적	15	1

[실습 9-4] 테이블 종합 실습하기

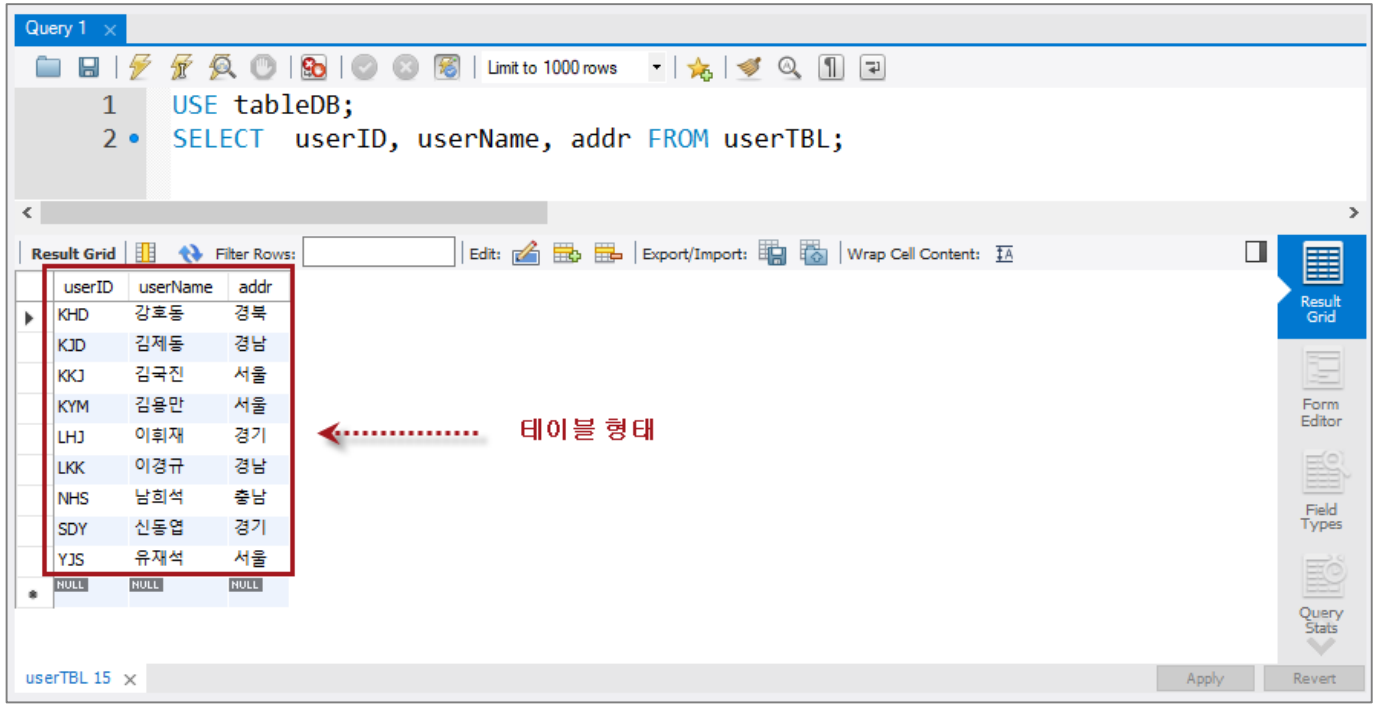
교재 318~326p 참고

5-12 원 테이블에서 CHECK 제약 조건이 걸려 있는 출생 연도(birthYear) 열을 ALTER TABLE 문으로 삭제

```
ALTER TABLE userTBL
      DROP COLUMN birthYear;
```

5-1 뷰의 개요

- 뷰(view)
  - '가상의 테이블'
  - 한 번 생성해 놓으면 테이블로 생각하고 사용해도 될 만큼 사용자가 볼 때 테이블과 거의 동일한 개체로 여겨짐
  - 쿼리 창에서 SELECT 문을 실행하여 나온 내용도 결국 테이블 모양 -> 뷰의 실체는 SELECT 문
  - SELECT userID, userName, addr FROM userTBL 문의 결과를 v_userTBL이라고 부른다면 앞으로는 v_userTBL을 그냥 테이블로 생각하고 접근하면 됨



5-2 뷰 생성

- 뷰의 생성과 활용

```
USE tableDB;
CREATE VIEW v_userTBL
AS
	SELECT userID, userName, addr FROM userTBL;
```

```
SELECT * FROM v_userTBL; -- 뷰를 테이블이라고 생각해도 무방
```

userID	userName	addr
KHD	강호동	경북
KJD	김제동	경남
KKJ	김국진	서울
KYM	김용만	서울
LHJ	이휘재	경기
LKK	이경규	경남
NHS	남희석	충남
SDY	신동엽	경기
YJS	유재석	서울

5-2 뷰 생성

- 뷰의 작동 방식
  - 뷰를 테이블로 여기고 접근해도 원래 테이블을 이용하여 접근한 것과 동일한 결과를 얻을 수 있음
  - 사용자가 뷰를 통해 접근하면 MySQL이 나머지는 알아서 처리해줌

그림 9-18 뷰의 작동 방식

5-3 뷰의 장점

- 뷰의 장점
  - 뷰는 보안에 도움이 됨
  - 뷰는 복잡한 쿼리를 단순화해줌
    - 물건을 구매한 회원에 대한 쿼리

```
SELECT U.userID, U.userName, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2) AS '연락처'
   FROM userTBL U
      INNER JOIN buyTBL B
         ON U.userID = B.userID;
```

    - 위 쿼리를 뷰로 생성

```
CREATE VIEW v_userbuyTBL
AS
SELECT U.userID, U.userName, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2) AS '연락처'
   FROM userTBL U
      INNER JOIN buyTBL B
         ON U.userID = B.userID;
```

    - 생성한 v_userbuyTBL 뷰를 이용하여 강호동의 구매 기록을 조회하는 쿼리

```
SELECT * FROM v_userbuyTBL WHERE userName = '강호동';
```

userID	userName	prodName	addr	연락처
KHD	강호동	운동화	경북	01122222222
KHD	강호동	노트북	경북	01122222222
KHD	강호동	청바지	경북	01122222222

[실습 9-5] 뷰 활용하기

교재 330~336p 참고

1 cookDB 초기화하기

1-1 C:₩SQL₩ cookDB.sql 파일을 열어 실행

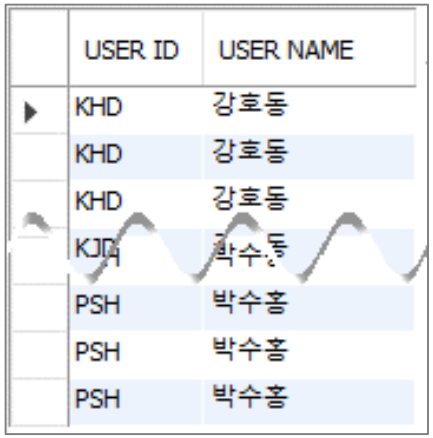
1-2 열린 쿼리 창을 모두 닫고 새 쿼리 창 열기

2 뷰 생성하기

2-1 기본적인 뷰 생성

```
USE cookDB;
CREATE VIEW v_userbuyTBL
AS
SELECT U.userID AS 'USER ID', U.userName AS 'USER NAME', B.prodName AS 'PRODUCT NAME',
          U.addr, CONCAT(U.mobile1, U.mobile2) AS 'MOBILE PHONE'
   FROM userTBL U
      INNER JOIN buyTBL B
         ON U.userID = B.userID;

SELECT `USER ID`, `USER NAME` FROM v_userbuyTBL; -- 주의! 백틱(키보드 1의 왼쪽 키) 사용
```



	USER ID	USER NAME
▶	KHD	강호동
	KHD	강호동
	KHD	강호동
	KJD	박수동
	PSH	박수홍
	PSH	박수홍
	PSH	박수홍

[실습 9-5] 뷰 활용하기

교재 330~336p 참고

3 뷰 수정, 삭제하기

3-1 ALTER VIEW 문 사용하여 뷰 수정

```
ALTER VIEW v_userbuyTBL
AS
SELECT U.userID AS '사용자 아이디', U.userName AS '이름', B.prodName AS '제품 이름',
          U.addr, CONCAT(U.mobile1, U.mobile2) AS '전화 번호'
   FROM userTBL U
      INNER JOIN buyTBL B
         ON U.userID = B.userID;

SELECT `이름`, `전화 번호` FROM v_userbuyTBL; -- 주의! 백틱 사용
```

3-2 DROP VIEW 문 사용하여 뷰 삭제

```
DROP VIEW v_userbuyTBL;
```

4 뷰의 정보 확인하기

4-1 CREATE OR REPLACE VIEW 문을 사용하여 기존의 뷰를 덮어써 새 뷰 만들기

```
USE cookDB;
CREATE OR REPLACE VIEW v_userTBL
AS
      SELECT userID, userName, addr FROM userTBL;
```

[실습 9-5] 뷰 활용하기

4-2 sys.sql_modules에 들어 있는 뷰의 정보 확인

```
DESCRIBE v_userTBL;
```

	Field	Type	Null	Key	Default	Extra
▶	userID	char(8)	NO		NULL	
	userName	varchar(10)	NO		NULL	
	addr	char(2)	NO		NULL	

4-3 뷰의 소스코드 확인

```
SHOW CREATE VIEW v_userTBL;
```

[실습 9-5] 뷰 활용하기

교재 330~336p 참고

5 뷰의데이터 수정하기

5-1 아이디가 LKK인 회원의 주소를 부산으로 변경

```
UPDATE v_userTBL SET addr = '부산' WHERE userID='LKK';
```

5-2 뷰에 새로운 데이터 입력

```
INSERT INTO v_userTBL (userID, userName, addr) VALUES ('KBM', '김병만', '충북');
```

실행 결과

Error Code: 1423. Field of view 'cookdb.v_usertbl' underlying table doesn't have a default value

6 그룹 함수를 포함하는 뷰의 데이터 수정하기

6-1 SUM() 함수를 사용하는 뷰를 간단히 정의

```
CREATE OR REPLACE VIEW v_sum
AS
      SELECT userID AS 'userID', SUM(price * amount) AS 'total'
         FROM buyTBL GROUP BY userID;

SELECT * FROM v_sum;
```

userID	total
KHD	1210
KJD	75
KYM	200
LHJ	95
PSH	1920

[실습 9-5] 뷰 활용하기

교재 330~336p 참고

6-2 INFORMATION_SCHEMA의 VIEWS 테이블에서 전체 시스템에 저장된 뷰의 정보 확인(뷰를 통해서는 데이터를 삽입(INSERT), 수정 (UPDATE), 삭제(DELETE)할 수 없음)

```
SELECT * FROM INFORMATION_SCHEMA.VIEWS
    WHERE TABLE_SCHEMA = 'cookDB' AND TABLE_NAME = 'v_sum';
```

	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	VIEW_DEFINITION	CHECK_OPTION	IS_UPDATABLE	DEFINER
	def	cookdb	v_sum	select `cookdb`.`buvtbl`.`userID` AS `userID`...	NONE	NO	root@localhost

6-3 뷰를 통해 데이터를 삽입, 수정, 삭제할 수 없는 경우

- SUM() 등의 집계 함수를 사용한 뷰
- UNION ALL, JOIN 등을 사용한 뷰
- DISTINCT, GROUP BY 등을 사용한 뷰

[실습 9-5] 뷰 활용하기

교재 330~336p 참고

7 지정한 범위로 뷰 생성하고 데이터 입력하기

7-1 키가 180cm 이상인 사람만 보여주는 뷰 생성

```
CREATE OR REPLACE VIEW v_height180
AS
	SELECT * FROM userTBL WHERE height >= 180;

SELECT * FROM v_height180;
```

userID	userName	birthYear	addr	mobile1	mobile2	height	mDate
KHD	강호동	1970	경북	011	22222222	182	2007-07-07
LHJ	이휘재	1972	경기	011	88888888	180	2006-04-04
NHS	남희석	1971	충남	016	66666666	180	2017-04-04
PSH	박수홍	1970	서울	010	00000000	183	2012-05-05

7-2 v_height180 뷰에서 키가 180cm 미만인 사람의 데이터 삭제

```
DELETE FROM v_height180 WHERE height < 180;
```

실행 결과

```
0 row(s) affected
```

[실습 9-5] 뷰 활용하기

교재 330~336p 참고

7-3 v_height180 뷰에서 키가 180cm 미만인 사람의 데이터 입력

```
INSERT INTO v_height180 VALUES ('SJH', '서장훈', 1974, '경기', '010', '5555555', 158, '2019-01-01');
```

실행 결과

```
1 row(s) affected
```

7-4 SELECT * FROM v_height180; 문으로 뷰 확인

7-5 키가 180cm 이상인 사람만 보여주는 뷰에 키가 180cm 이상인 사람만 입력되게 하려면 WITH CHECK OPTION 문 사용

```
ALTER VIEW v_height180
AS
	SELECT * FROM userTBL WHERE height >= 180
		WITH CHECK OPTION;
INSERT INTO v_height180 VALUES('KBM', '김병만', 1977, '서울', '010', '3333333', 155, '2019-3-3');
```

실행 결과

```
Error Code: 1369. CHECK OPTION failed 'cookdb.v_height180'
```

[실습 9-5] 뷰 활용하기

교재 330~336p 참고

8 복합 뷰 생성하고 데이터 입력하기

8-1 2개 이상의 테이블이 관련된 복합 뷰를 생성하고 데이터 입력

```
CREATE OR REPLACE VIEW v_userbuyTBL
AS
SELECT U.userID, U.userName, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2) AS mobile
   FROM userTBL U
      INNER JOIN buyTBL B
         ON U.userID = B.userID;

INSERT INTO v_userbuyTBL VALUES ('PKL', '박경리', '운동화', '경기', '00000000000', '2020-2-2');
```

실행 결과

```
Error Code: 1394. Can not insert INTo join view 'cookdb.v_userbuytbl' without fields list
```

9 뷰가 참조하는 데이터 삭제하기

9-1 두 테이블 삭제

```
DROP TABLE IF EXISTS buyTBL, userTBL;
```

[실습 9-5] 뷰 활용하기

9-2 뷰 다시 조회

```
SELECT * FROM v_userbuyTBL;
```

실행 결과

Error Code: 1356. View 'cookdb.v_userbuytbl' references invalid table(s) or column(s) or function(s) or definer/invoker of view lack rights to use them

9-3 뷰의 상태 확인

```
CHECK TABLE v_userbuyTBL;
```

Table	Op	Msg_type	Msg_text
cookdb.v_userbuytbl	check	Error	Table 'cookdb.usertbl' doesn't exist
cookdb.v_userbuytbl	check	Error	View 'cookdb.v_userbuytbl' references invalid table(s) or column(s) or function(s) or definer/invoker of view lack rights to use them
cookdb.v_userbuytbl	check	error	Corrupt