



Chapter 03

SQL 기초

MySQL로 배우는 데이터베이스 개론과 실습

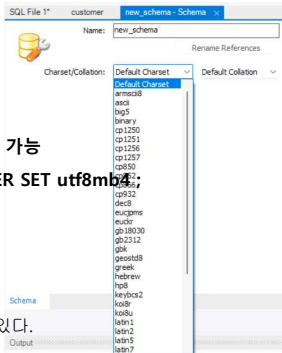
한글 입력 오류 문제 해결

- 테이블에 한글 입력시 Incorrect string value: 'WxE3Wx84WxB4WxE3Wx85Wx87...' for column ... 이라고 나올때 해결방법
- `SELECT schema_name , default_character_set_name FROM information_schema.schemata ;` 로 데이터베이스의 캐릭터셋 확인
- `create table` 테이블명
(`text varchar(100)`) `charset = utf8mb4 ;`
로 명확히 지정하고 작업

* 이미 만들어진 테이블의 경우에는 아래의 명령어로 변경 가능

`ALTER DATABASE database_name DEFAULT CHARACTER SET utf8mb4 ;`

* `collate` 는 문자를 정렬하는 방식으로
이 방식에 따라 'a'가 'B' 앞에 올 수도 있고 뒤에 올 수도 있다.



한글 입력 오류 문제 해결

- utf8mb4 는 emoji 문자(☺)를 입력할 수 있는 캐릭터셋
 - utf8mb4는 각 문자가 UTF-8 인코딩 체계에서 MaxByte 4로 저장됨
- 기존의 utf8은 원래는 4바이트까지의 자료형을 저장할 수 있지만, 전세계 모든 언어가 21bit로 3바이트 내로 처리가 되어 MySQL에서 3바이트 가변 자료형으로 설계함 , 이러한 이유 때문에 4바이트로 처리되는 이모지는 처리하지 못하는 문제가 생겼고, utf8mb4라는 체계를 추가해서 4바이트까지의 문자열을 처리할 수 있게 됨

```
CREATE TABLE products
(
  `product_no` INT NOT NULL AUTO_INCREMENT COMMENT 'id',
  `created_at` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '최초 등록일시',
  `is_deleted` TINYINT NOT NULL DEFAULT FALSE COMMENT '삭제여부',
  PRIMARY KEY (product_no)
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci COMMENT '상품 번호';
```

03. 데이터 조작용어 - 검색

3. 두 개 이상 테이블에서 SQL 질의



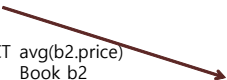
3. 두 개 이상 테이블에서 SQL 질의

❖ 부속질의 SQL 문 내에 또 다른 SQL 문을 작성해보자

- 상관 부속질의(correlated subquery)는 상위 부속질의의 튜플을 이용하여 하위 부속질을 계산함. 즉 상위 부속질의와 하위 부속질의가 독립적이지 않고 서로 관련을 맺고 있음.
- 상관쿼리는 서브쿼리 단독으로 실행할 수 없다.
- 실행되는 순서도 메인(Outer) 쿼리가 먼저 실행이 되서 서브(inner) 쿼리 쪽으로 들어간다. 이 때 메인쿼리는 서브쿼리로 한 행씩 넘긴다.

질의 3-31 각 출판사별로 출판사의 평균 도서 가격보다 비싼 도서를 구하시오.

```
SELECT  b1.bookname  
FROM    Book b1  
WHERE   b1.price > (SELECT avg(b2.price)  
                    FROM    Book b2  
                    WHERE   b2.publisher=b1.publisher);
```



bookname
골프 바이블
피겨 교본
야구의 추억

- 1) b1 테이블에서 한 행씩 읽어서 b2 테이블로 가져간다.
- 2) 출판사별 평균가격 구한다.
- 3) 조건에 맞는 것만 출력한다.

3. 두 개 이상 테이블에서 SQL 질의

❖ 부속질의(subquery)_SQL 문 내에 또 다른 SQL 문을 작성해보자

Book 테이블 : b1으로 나타냄

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구하는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

Book 테이블 : b2로 나타냄

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구하는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

b1 테이블의
두들 t에
해당하는
출판사를
b2 테이블로
가져가서,
같은 출판사를
가진 두들들의
price 평균을
구한다.

avg(b2.price)

28500

b1.price > avg(b2.price)

그림 3-24 상관 부속질의의 데이터 예

참고> 상관 부속 질의(서브쿼리) 연습문제

- ❖ 매출테이블에서 각 브랜드별 제품 판매가 각 브랜드별
평균판매보다 높은 매출을 구하시오.

```
SELECT *  
FROM sales AS s1  
WHERE s1.amount > ( SELECT AVG(s2.amount)  
                     FROM sales AS s2  
                     WHERE s1.brand = s2.brand)
```

브랜드명	매출
나이키	160000
아디다스	110000
룰루레몬	150000
칸골	210000
MLB	250000
나이키	170000
MLB	120000
룰루레몬	160000
나이키	99000
MLB	330000
칸골	170000

3. 두 개 이상 테이블에서 SQL 질의

❖ 집합 연산_도서를 주문한 고객을 알고 싶다

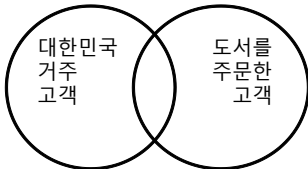
- 합집합 UNION, 차집합 MINUS, 교집합 INTERSECT - 테이블간의 집합연산

질의 3-32 대한민국에서 거주하는 고객의 이름과 도서를 주문한 고객의 이름을 보이시오.

```
SELECT name
FROM Customer
WHERE address LIKE '대한민국%'
UNION
SELECT name
FROM Customer
WHERE custid IN (SELECT custid FROM Orders);
```

name
김연아
장미란
박세리
박지성
추신수

{고객 이름} = {대한민국에 거주하는 고객 이름} ∪ {도서를 주문한 고객 이름}



두 개 이상 테이블에서 SQL 질의

❖ 집합 연산_도서를 주문한 고객을 알고 싶다

■ 합집합 UNION 과 UNION ALL

```
SELECT name
FROM Customer
WHERE address LIKE '대한민국%'
```

UNION ALL

```
SELECT name
FROM Customer
WHERE custid IN (SELECT custid FROM Orders);
```

정렬을 하려면 order by 는 어디에?

	name
▶	김연아
	장미란
	박세리
	박지성
	김연아
	장미란
	추신수

* 주의사항 - 두 개의 테이블의 컬럼의 수가 같아야 한다.
(타입이나 순서는 달라도 조회는 되나 첫번째 테이블의 컬럼명으로 조회됨)

```
SELECT name , 1
FROM Customer
WHERE address LIKE '대한민국%'
```

UNION

```
SELECT name
FROM Customer
WHERE custid IN (SELECT custid FROM Orders);
```

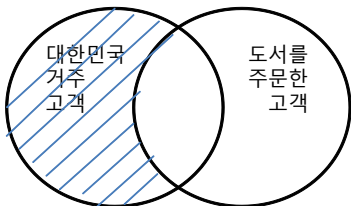
3. 두 개 이상 테이블에서 SQL 질의

■ <여기서 잠깐> MINUS, INTERSECT 연산자

MySQL에는 MINUS, INTERSECT 연산자가 없으므로 다음과 같이 표현한다.

[질의 3-32]에서 **MINUS** 연산을 수행한 “대한민국에서 거주하는 고객의 이름에서 도서를 주문한 고객의 이름 빼고 보이시오.” 질의를 NOT IN 연산자를 사용하면 다음과 같다.

```
SELECT name
FROM Customer
WHERE address LIKE '대한민국%' AND
      name NOT IN (SELECT name
                    FROM Customer
                    WHERE custid IN (SELECT custid FROM Orders));
```



(oracle SQL)

```
SELECT name
FROM Customer
WHERE address LIKE '대한민국%'
MINUS
SELECT name
FROM Customer
WHERE custid IN (SELECT custid FROM Orders);
```

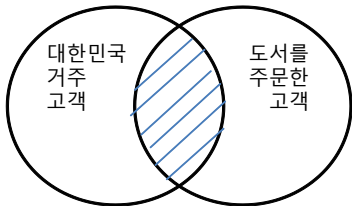
3. 두 개 이상 테이블에서 SQL 질의

■ <여기서 잠깐> MINUS, INTERSECT 연산자

MySQL에는 MINUS, INTERSECT 연산자가 없으므로 다음과 같이 표현한다.

[질의 3-32]에서 **INTERSECT** 연산을 수행한 “대한민국에서 거주하는 고객 중 도서를 주문한 고객의 이름 보이시오.” 질의를 IN 연산자를 사용하면 다음과 같다.

```
SELECT name
FROM Customer
WHERE address LIKE '대한민국%' AND
      name IN (SELECT name
                FROM Customer
                WHERE custid IN (SELECT custid FROM Orders));
```



(oracle SQL)

```
SELECT name
FROM Customer
WHERE address LIKE '대한민국%'
```

INTERSECT

```
SELECT name
FROM Customer
WHERE custid IN (SELECT custid FROM Orders);
```

3. 두 개 이상 테이블에서 SQL 질의

❖ EXISTS_주문이 있는 고객을 알고 싶다

- EXISTS라는 원래 단어에서 의미하는 것과 같이 조건에 맞는 튜플이 존재하면 결과에 포함시킴. 즉 부속질의문의 어떤 행이 조건에 만족하면 참임.
- NOT EXISTS는 부속질의문의 모든 행이 조건에 만족하지 않을 때만 참임.

질의 3-33 주문이 있는 고객의 이름과 주소를 보이시오.

```
SELECT name, address
FROM Customer cs
WHERE EXISTS (SELECT *
               FROM Orders od
               WHERE cs.custid = od.custid);
```

name	address
박지성	영국 맨체스타
김연아	대한민국 서울
장미란	대한민국 강원도
추신수	미국 클리블랜드

* EXISTS는 상관 부속질의문 형식임.

3. 두 개 이상 테이블에서 SQL 질의

❖ EXISTS_주문이 있는 고객을 알고 싶다

Customer

custid	name	address	phone
1	박지성	영국 맨체스터	000-5000-0001
2	김연아	대한민국 서울	000-6000-0001
3	장미란	대한민국 강원도	000-7000-0001
4	추신수	미국 클리블랜드	000-8000-0001
5	박세리	대한민국 대전	NULL

Orders

orderid	custid	bookid	saleprice	orderdate
1	1	1	6000	2014-07-01
2	1	3	21000	2014-07-03
3	2	5	8000	2014-07-03
4	3	6	6000	2014-07-04
5	4	7	20000	2014-07-05
6	1	2	12000	2014-07-07
7	4	8	13000	2014-07-07
8	3	10	12000	2014-07-08
9	2	10	7000	2014-07-09
10	3	8	13000	2014-07-10

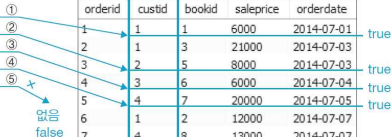


그림 3-25 EXISTS 상관 부속질의문 데이터 예

```
SELECT name, address
FROM Customer cs
WHERE EXISTS (SELECT *
              FROM Orders od
              WHERE cs.custid =od.custid);
```

각 행별 실행 순서

- 1) cs의 한 행을 가져온 후 서브쿼리의 cs 값으로 입력한다
- 2) 고객번호가 같은 것을 찾으면 exists 가 참이 되고
- 3) 해당 행에 대해 select name, address 가 반환된다.

참고> 다중 행 부속 질의문에 사용 가능한 연산자

연산자	설명
IN	부속질의문의 결과 값 중 일치하는 것이 있으면 검색 조건이 참
NOT IN	부속질의문의 결과 값 중 일치하는 것이 없으면 검색 조건이 참
EXISTS	부속질의문의 결과 값이 하나라도 존재하면 검색 조건이 참
NOT EXISTS	부속질의문의 결과 값이 하나도 존재하지 않으면 검색 조건이 참
ALL	부속질의문의 결과 값 모두와 비교한 결과가 참이면 검색 조건을 만족(비교 연산자와 함께 사용)
ANY 또는 SOME	부속질의문의 결과 값 중 하나라도 비교한 결과가 참이면 검색 조건을 만족(비교 연산자와 함께 사용)

* in 의 경우 : 실제 존재하는 데이터들의 모든 값을 확인

- 1) subquery 먼저 실행
- 2) mainquery에서 하나의 행 가져옴
- 3) row의 값이 1에서 가져온 요소들에 포함되는지 체크하고 일치하면 row 출력
- 4) 1~3)을 반복

* exists 의 경우 : 해당 row 가 존재하는지만 확인하고 더 이상 수행하지 않음

- 1) mainquery를 실행하여 한 행 읽어옴
- 2) 해당 행에 대해 subquery 실행하여 그 결과가 존재하면 true
- 3) 1에서 가져온 행에 대해 select 실행 출력
- 4) 1~3)을 반복

<IN , EXISTS , JOIN 성능 비교>

- <IN> - 값 직접 비교, 테이블에 row 가 늘어날수록 느려짐, 직관적

```
SELECT custid  
FROM customer  
WHERE custid IN (SELECT custid FROM orders WHERE...)
```

- <EXISTS> - 값 True/False 비교 후 찾으면 바로 중지, IN 보다 빠름, 직관적

```
SELECT custid  
FROM customer as A  
WHERE custid EXISTS (SELECT custid FROM orders as B WHERE A.custid = B.custid and ...)
```

- <JOIN> - 가장 빠름, 비직관적, JOIN하는 테이블에 동일한 값이 있을 경우 여러행 반환

```
SELECT *  
FROM customer A INNER JOIN (SELECT * FROM orders) B  
ON A.custid = B.custid
```

- < 결론 >

조회하는 데이터가 많지 않을 경우(몇백~몇천) - IN

조회하는 데이터가 그보다 많을 경우(몇만건~) - EXISTS

매우 빠른 속도가 필요할 경우 - INNER JOIN

연습문제

1. 마당서점의 고객이 요구하는 다음 질문에 대해 SQL 문을 작성하시오.
 - (5) 박지성이 구매한 도서의 출판사 수
 - (6) 박지성이 구매한 도서의 이름, 가격, 정가와 판매가격의 차이
 - (7) 박지성이 구매하지 않은 도서의 이름

2. 마당서점의 운영자와 경영자가 요구하는 다음 질문에 대해 SQL 문을 작성하시오.
 - (8) 주문하지 않은 고객의 이름(부속질의 사용)
 - (9) 주문 금액의 총액과 주문의 평균 금액
 - (10) 고객의 이름과 고객별 구매액
 - (11) 고객의 이름과 고객이 구매한 도서 목록
 - (12) 도서의 가격(Book 테이블)과 판매가격(Orders 테이블)의 차이가 가장 많은 주문
 - (13) 도서의 판매액 평균보다 자신의 구매액 평균이 더 높은 고객의 이름

04. 데이터 정의어(DDL)

1. CREATE 문
2. ALTER 문
3. DROP 문



1. CREATE 문


■ 테이블 생성 규칙

- 테이블명
 - 객체를 의미할 수 있는 이름으로 단수형을 권장함(권장사항이기 때문에 복수형 사용도 가능함)
 - 다른 테이블의 이름과 중복되지 않아야 함
- 컬럼명
 - 한 테이블 내에서는 컬럼명이 중복되지 않아야 함
 - 테이블 생성시 각 컬럼들은 괄호 내에서 콤마로 구분됨
 - 컬럼 뒤에 데이터 유형이 반드시 지정되어야 함
- 테이블명 & 컬럼명
 - 사전에 정의된 예약어(Reserved words - SELECT, FROM, WHERE, SET 등)는 사용불가
 - 테이블과 컬럼명에는 문자,숫자,일부 기호(, \$, #)만 허용됨
 - 테이블명과 컬럼명은 반드시 문자로 시작해야 함(숫자, 기호 불가)
- 제약조건명 : 다른 제약조건의 이름과 중복되지 않아야 함

1. CREATE 문

- 테이블 구성, 속성과 속성에 관한 제약 정의, 기본키 및 외래키를 정의하는 명령
- PRIMARY KEY : 기본키를 정할 때 사용
- FOREIGN KEY : 외래키를 지정할 때 사용
- ON UPDATE와 ON DELETE : 외래키 속성의 수정과 튜플 삭제 시 동작을 나타냄
- CREATE 문의 기본 문법

```
CREATE TABLE 테이블이름
( { ① 속성이름 데이터타입
  [NOT NULL | UNIQUE | DEFAULT 기본값 | CHECK 체크조건 | AUTO_INCREMENT]
}
  ② [CONSTRAINT 제약조건이름] [PRIMARY KEY 속성이름(들)]
{ ③ [CONSTRAINT 제약조건이름]
  [FOREIGN KEY 속성이름 REFERENCES 테이블이름(속성이름)]
  [ON DELETE { CASCADE | SET NULL } ]
}
)
```




테이블 이름은 먼저 생성되어 있어야 함

1. CREATE 문

● FK 제약 조건의 옵션 (cont'd)

▪ ON DELETE CASCADE



학과코드	학과명
aaa	경영정보
bbb	정보경영

학번	이름	학과코드
11	홍길동	aaa
22	강감찬	aaa
33	김유신	bbb

학과코드	학과명
aaa	경영정보

학번	이름	학과코드
11	홍길동	aaa
22	강감찬	aaa

▪ ON UPDATE CASCADE

학과코드	학과명
aaa	경영정보
ccc	정보경영

학번	이름	학과코드
11	홍길동	aaa
22	강감찬	aaa
33	김유신	ccc

▪ ON DELETE SET NULL

학과코드	학과명
aaa	경영정보

학번	이름	학과코드
11	홍길동	aaa
22	강감찬	aaa
33	김유신	NULL

▪ ON UPDATE SET NULL

학과코드	학과명
aaa	경영정보
ccc	정보경영

학번	이름	학과코드
11	홍길동	aaa
22	강감찬	aaa
33	김유신	NULL

1. CREATE 문

표 3-10 데이터 타입 종류

데이터 타입	설명	ANSI SQL 표준 타입
INTEGER INT	4바이트 정수형	INTEGER, INT SMALLINT
NUMERIC(m,d) DECIMAL(m,d)	전체자리수 m, 소수점이하 자리수 d를 가진 숫자 형	DECIMAL(p, s) NUMERIC[(p,s)]
CHAR(n)	문자형 고정길이, 문자를 저장하고 남은 공간은 공백으로 채운다. ('AA' = 'AA ')	CHARACTER(n) CHAR(n)
VARCHAR(n)	문자형 가변길이('AA' ≠ 'AA ')	CHARACTER VARYING(n) CHAR VARYING(n)
DATE	날짜형, 연도, 월, 날, 시간을 저장한다.	

1. CREATE 문

질의 3-34 다음과 같은 속성을 가진 NewBook 테이블을 생성하시오. 정수형은 INTEGER를 사용하며 문자형은 가변형 문자타입인 VARCHAR을 사용한다.

- bookid(도서번호)-INTEGER
- bookname(도서이름)-VARCHAR(20)
- publisher(출판사)-VARCHAR(20)
- price(가격)-INTEGER

```
CREATE TABLE      NewBook (  
  bookid           INTEGER,  
  bookname         VARCHAR(20),  
  publisher        VARCHAR(20),  
  price            INTEGER);
```

※ 기본키를 지정하고 싶다면 다음과 같이 생성한다.

```
CREATE TABLE NewBook (  
  bookid          INTEGER,  
  bookname        VARCHAR(20),  
  publisher       VARCHAR(20),  
  price           INTEGER,  
  PRIMARY KEY (bookid));
```

=

```
CREATE TABLE NewBook (  
  bookid          INTEGER PRIMARY KEY,  
  bookname        VARCHAR(20),  
  publisher       VARCHAR(20),  
  price           INTEGER);
```

1. CREATE 문

※ **bookid** 속성이 없어서 두 개의 속성 **bookname**, **publisher**가 기본키가 된다면 괄호를 사용하여 복합키를 지정한다.

```
CREATE TABLE      NewBook (  
    bookname        VARCHAR(20),  
    publisher        VARCHAR(20),  
    price            INTEGER,  
  
    PRIMARY KEY      (bookname, publisher));
```

bookname은 NULL 값을 가질 수 없고, publisher는 같은 값이 있으면 안 된다. price에 값이 입력되지 않을 경우 기본 값 10000을 저장한다. 또 가격은 최소 1,000원 이상으로 한다.

```
CREATE TABLE      NewBook (  
    bookname        VARCHAR(20)      NOT NULL,  
    publisher        VARCHAR(20)      UNIQUE,  
    price            INTEGER DEFAULT 10000  CHECK(price > 1000),  
    PRIMARY KEY      (bookname, publisher));
```

1. CREATE 문

질의 3-35 다음과 같은 속성을 가진 **NewCustomer** 테이블을 생성하시오.

- custid(고객번호) - INTEGER, 기본키, 자동증가
- name(이름) - VARCHAR(40)
- address(주소) - VARCHAR(40)
- phone(전화번호) - VARCHAR(30)

```
CREATE TABLE      NewCustomer (  
    custid          INTEGER AUTO_INCREMENT PRIMARY KEY,  
    name            VARCHAR(40),  
    address         VARCHAR(40),  
    phone           VARCHAR(30) );
```


1. CREATE 문

질의 3-36 다음과 같은 속성을 가진 **NewOrders** 테이블을 생성하시오.

- orderid(주문번호) - INTEGER, 기본키, 자동증가
- custid(고객번호) - INTEGER, NOT NULL 제약조건, 외래키(NewCustomer.custid, 연쇄삭제)
- bookid(도서번호) - INTEGER, NOT NULL 제약조건
- saleprice(판매가격) - INTEGER
- orderdate(판매일자) - DATE

```
CREATE TABLE      NewOrders (  
    orderid  INTEGER  AUTO_INCREMENT,  
    custid   INTEGER  NOT NULL,  
    bookid   INTEGER  NOT NULL,  
    saleprice      INTEGER,  
    orderdate      DATE,  
    PRIMARY KEY (orderid),  
    FOREIGN KEY (custid) REFERENCES NewCustomer(custid) ON DELETE CASCADE );
```

1. CREATE 문

- 외래키 제약조건을 명시할 때는 반드시 참조되는 테이블(부모 릴레이션)이 존재해야 하며 참조되는 테이블의 기본키여야 함
- 외래키 지정 시 ON DELETE 또는 ON UPDATE 옵션은 참조되는 테이블의 튜플이 삭제되거나 수정될 때 취할 수 있는 동작을 지정함
- NO ACTION은 어떠한 동작도 취하지 않음.

- 참고 > 다른 테이블을 가지고 테이블 만드는 법

CREATE TABLE aaa AS SELECT * FROM bbb;

단, 이렇게 해서 생성하는 경우 not null 제약조건만 복제되고
나머지 제약조건은 수동으로 추가해야 함.

2. ALTER 문

- ALTER 문은 생성된 **테이블의 속성**과 속성에 관한 **제약**을 변경하며,
기본키 및 외래키를 변경함
- ADD : 컬럼의 추가 - 새로 추가한 컬럼은 테이블의 맨 마지막에 추가됨
- DROP COLUMN : 컬럼을 삭제할 때 사용함, 삭제 후 최소 하나 이상의 컬럼이 테이블에 존재해야 함.
- RENAME COLUMN..TO : 해당 컬럼의 모든 정의가 그대로 유지되고 이름만 변경
- MODIFY는 속성의 기본값을 설정하거나 삭제할 때 사용함
 - 이미 입력되어 있는 값에 영향을 미치는 변경은 허용하지 않음
 - 데이터 타입 변경 - 테이블에 아무 행도 없거나, 해당 컬럼이 NULL 만 갖고 있을 때 가능
 - 컬럼의 크기 변경
 - 확대는 항상 가능하나, 축소는 테이블에 아무 행도 없거나, 컬럼이 null 만 갖고 있거나 현재 저장된 값을 수용할 수 있는 크기로의 축소만 가능
 - DEFAULT 값 추가 및 수정
 - 추가 및 수정 이후 삽입되는 행에만 영향을 미침
- ADD <제약이름>, DROP <제약이름>은 제약사항을 추가하거나 삭제할 때 사용함

2. ALTER 문

- ALTER 문의 기본 문법 : <https://dev.mysql.com/doc/refman/8.0/en/alter-table.html>

ALTER TABLE 테이블이름

[ADD 속성이름 데이터타입]

[DROP COLUMN 속성이름]

[MODIFY 속성이름 데이터타입]

[MODIFY 속성이름 [NULL | NOT NULL]]

[ADD PRIMARY KEY(속성이름)]

[[ADD | DROP] 제약이름]

2. ALTER 문

질의 3-37 NewBook 테이블에 VARCHAR(13)의 자료형을 가진 isbn 속성을 추가하시오.

질의 3-38 NewBook 테이블의 isbn 속성의 데이터 타입을 INTEGER형으로 변경하시오.

질의 3-39 NewBook 테이블의 isbn 속성을 삭제하시오.

질의 3-40 NewBook 테이블의 bookid 속성에 NOT NULL 제약조건을 적용하시오.

질의 3-41 NewBook 테이블의 bookid 속성을 기본키로 변경하시오.

2. ALTER 문

질의 3-37 NewBook 테이블에 VARCHAR(13)의 자료형을 가진 isbn 속성을 추가하시오.

```
ALTER TABLE NewBook ADD isbn VARCHAR(13);
```

질의 3-38 NewBook 테이블의 isbn 속성의 데이터 타입을 INTEGER형으로 변경하시오.

```
ALTER TABLE NewBook MODIFY isbn INTEGER;
```

질의 3-39 NewBook 테이블의 isbn 속성을 삭제하시오.

```
ALTER TABLE NewBook DROP COLUMN isbn;
```

질의 3-40 NewBook 테이블의 bookid 속성에 NOT NULL 제약조건을 적용하시오.

```
ALTER TABLE NewBook MODIFY bookid INTEGER NOT NULL;
```

질의 3-41 NewBook 테이블의 bookid 속성을 기본키로 변경하시오.

```
ALTER TABLE NewBook ADD PRIMARY KEY(bookid);
```

3. DROP 문

- DROP 문은 테이블을 삭제하는 명령
- DROP 문은 테이블의 **구조와 데이터를 모두 삭제**하므로 사용에 주의해야 함 (데이터만 삭제하려면 DELETE 문을 사용함).
삭제시 참조무결성에 위배가 될 수 있으므로 유의해야 한다.
- DROP문의 기본 문법

```
DROP TABLE 테이블이름
```

질의 3-42 NewBook 테이블을 삭제하시오.

```
DROP TABLE NewBook;
```

질의 3-43 NewCustomer 테이블을 삭제하시오. 만약 삭제가 거절된다면 원인을 파악하고 관련된 테이블을 같이 삭제하시오(NewOrders 테이블이 NewCustomer를 참조하고 있음).

```
DROP TABLE NewCustomer;
```

요약

1. MySQL
2. SQL
3. 데이터 정의어(DLL)
4. 데이터 조작어(DML)
5. WHERE 조건
6. 집계 함수
7. GROUP BY
8. HAVING
9. 조인
10. 동등조인(내부조인)
11. 부속질의
12. 상관 부속질의
13. 튜플 변수
14. 집합 연산
15. EXISTS
16. CREATE
17. ALTER
18. DROP