



Chapter 03

SQL 기초

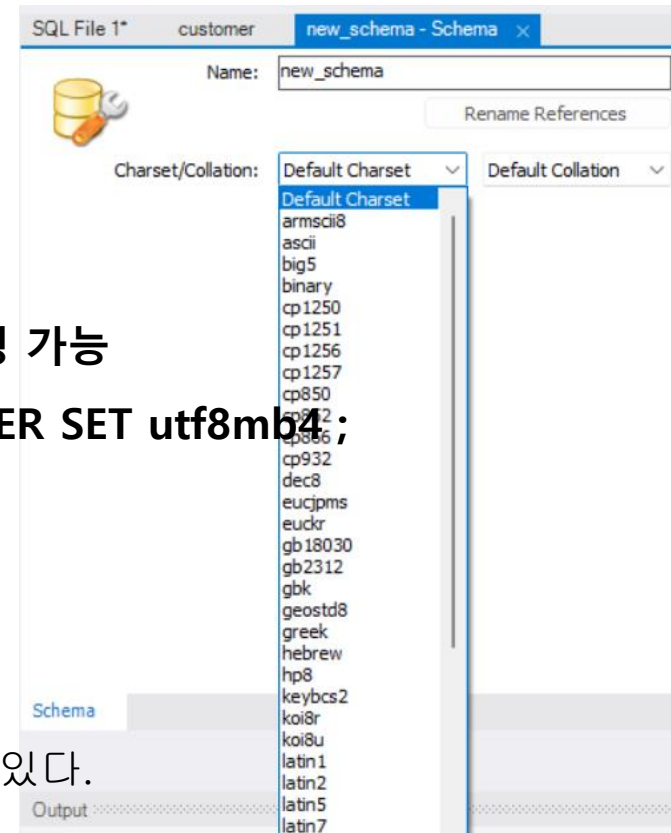
한글 입력 오류 문제 해결

- 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



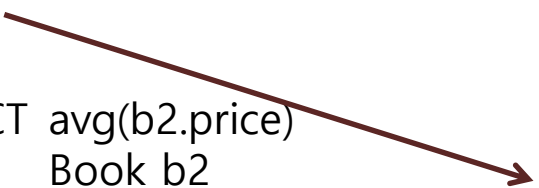
3. 두 개 이상 테이블에서 SQL 질의

❖ 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 : b1로 나타냄

| bookid | bookname | publisher | price |
|--------|-------------------|-----------|-------|
| 1 | 축구의 역사 | 굿스포츠 | 7000 |
| 2 | 축구아는 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
튜플 t에
해당하는
출판사를
b2 테이블로
가져가서,
같은 출판사를
가진 튜플들의
price 평균을
구한다.

Book 테이블 : b2로 나타냄

| bookid | bookname | publisher | price |
|--------|-------------------|-----------|-------|
| 1 | 축구의 역사 | 굿스포츠 | 7000 |
| 2 | 축구아는 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 |

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

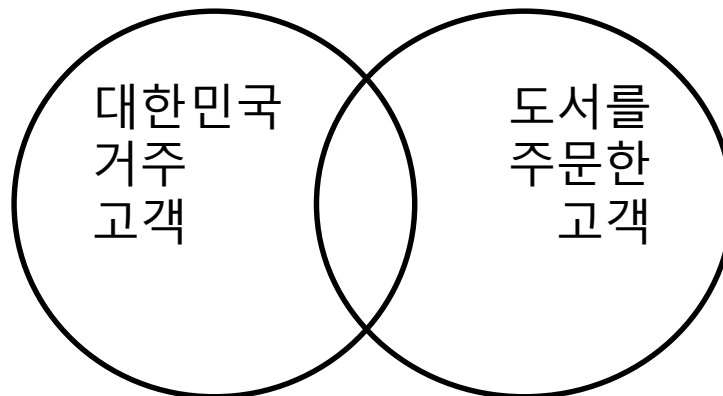
- 합집합 UNION, 차집합 MINUS, 교집합 INTERSECT - 테이블간의 집합연산

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);
```

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

| name |
|------|
| 김연아 |
| 장미란 |
| 박세리 |
| 박지성 |
| 추신수 |



두 개 이상 테이블에서 SQL 질의

정렬을 하려면 order by 는 어디에?

❖ 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);
```

| | 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);
```

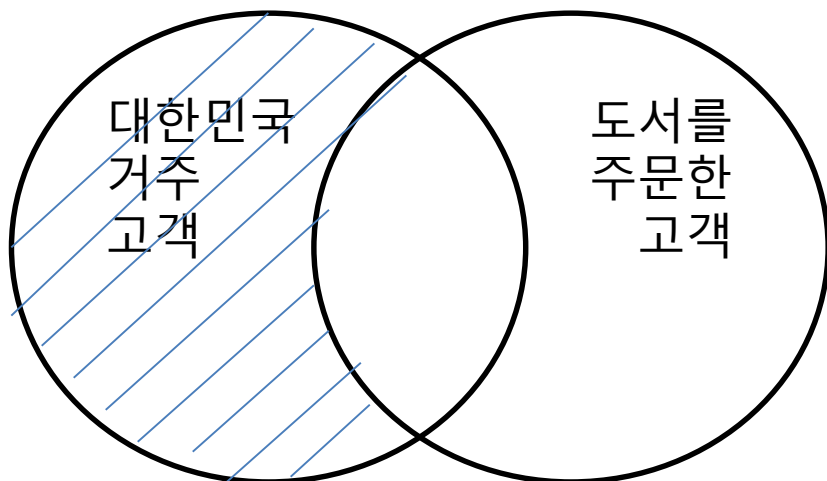
3. 두 개 이상 테이블에서 SQL 질의

■ <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);
```

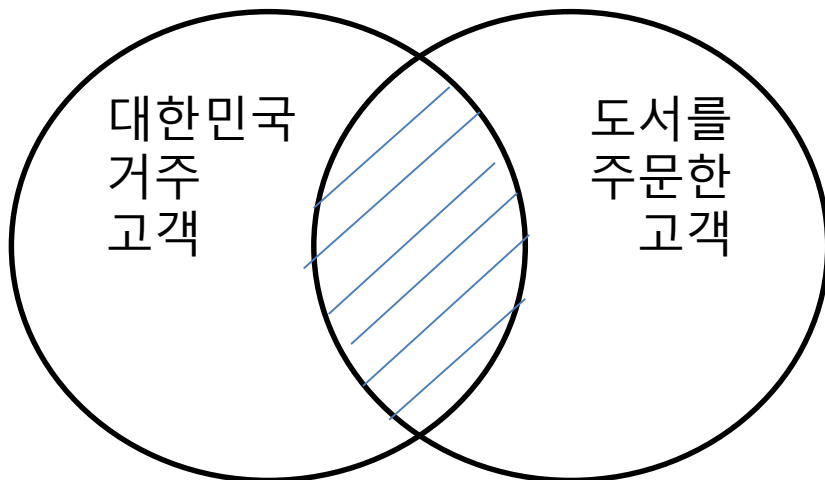
3. 두 개 이상 테이블에서 SQL 질의

■ <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 | true |
| 2 | 1 | 3 | 21000 | 2014-07-03 | |
| 3 | 2 | 5 | 8000 | 2014-07-03 | true |
| 4 | 3 | 6 | 6000 | 2014-07-04 | true |
| 5 | 4 | 7 | 20000 | 2014-07-05 | true |
| 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 | |

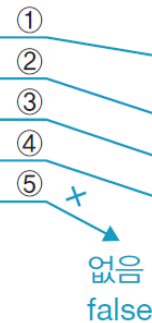


그림 3-25 EXISTS 상관 부속질의문 데이터 예

```
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

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;
```

1. MySQL
2. SQL
3. 데이터 정의어(DDL)
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

05. 데이터 조작용어 – 삽입, 수정, 삭제

1. INSERT 문
2. UPDATE 문
3. DELETE 문



1. INSERT 문

- INSERT 문은 테이블에 하나 또는 여러 행의 새로운 튜플을 삽입하는 명령임.
- INSERT 문의 기본 문법

```
INSERT INTO 테이블이름[(속성리스트)]  
VALUES (값리스트); /* 속성의 순서와 타입에 맞게 입력 */
```

질의 3-44 Book 테이블에 새로운 도서 '스포츠 의학'을 삽입하시오. 스포츠 의학은 한솔의학서적에서 출간했으며 가격은 90,000원이다.

```
INSERT INTO Book  
VALUES (11, '스포츠 의학', '한솔의학서적', 90000);
```

* 주의 : auto_increment 속성을 지정한 컬럼은 record 생성시 별도의 값을 입력하지 않는다.

```
INSERT INTO Book(price, bookname, publisher )  
VALUES ( 90000, '스포츠 의학', '한솔의학서적' );
```

| 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 |
| 11 | 스포츠 의학 | 한솔의학... | 90000 |

1. INSERT 문

질의 3-45 Book 테이블에 새로운 도서 '스포츠 의학'을 삽입하시오. 스포츠 의학은 한솔의학 서적에서 출간했으며 가격은 미정이다.

```
INSERT INTO Book(bookid, bookname, publisher)
VALUES (14, '스포츠 의학', '한솔의학서적');
```

| 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 |
| 11 | 스포츠 의학 | 한솔의학... | 90000 |
| 12 | 스포츠 의학 | 한솔의학... | 90000 |
| 13 | 스포츠 의학 | 한솔의학... | 90000 |
| 14 | 스포츠 의학 | 한솔의학... | NULL |

1. INSERT 문

- 대량 삽입(bulk insert)이란 한꺼번에 여러 개의 튜플을 삽입하는 방법임.
- 부속질의(서브쿼리)를 values 대신 사용할 수 있음

질의 3-46 수입도서 목록(Imported_book)을 Book 테이블에 모두 삽입하시오.
(Imported_book 테이블은 스크립트 Book 테이블과 같이 이미 만들어져 있음)

```
INSERT INTO Book(bookid, bookname, price, publisher)
SELECT bookid, bookname, price, publisher
FROM Imported_book;
```

| 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 |
| 11 | 스포츠 의학 | 한솔의학... | 90000 |
| 12 | 스포츠 의학 | 한솔의학... | 90000 |
| 13 | 스포츠 의학 | 한솔의학... | 90000 |
| 14 | 스포츠 의학 | 한솔의학... | NULL |
| 21 | Zen Golf | Pearson | 12000 |
| 22 | Soccer Skills | Human Kin... | 15000 |

2. UPDATE 문

- UPDATE 문은 특정 속성 값을 수정하는 명령이다.
- UPDATE 문의 기본 문법

```
UPDATE 테이블이름  
SET      속성이름1=값1[, 속성이름2=값2, ...]  
[WHERE <검색조건>]; /* 만일 where 절을 생략한다면 ? */
```

Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column. To disable safe mode, toggle the option in Preferences
-> SQL Editor and reconnect. 0.000 sec

Safe mode 를 항상 사용하다가 특정 경우에만 Safe mode 를 off 하고 싶을 경우,
set SQL_SAFE_UPDATES = 0; # disable safe mode

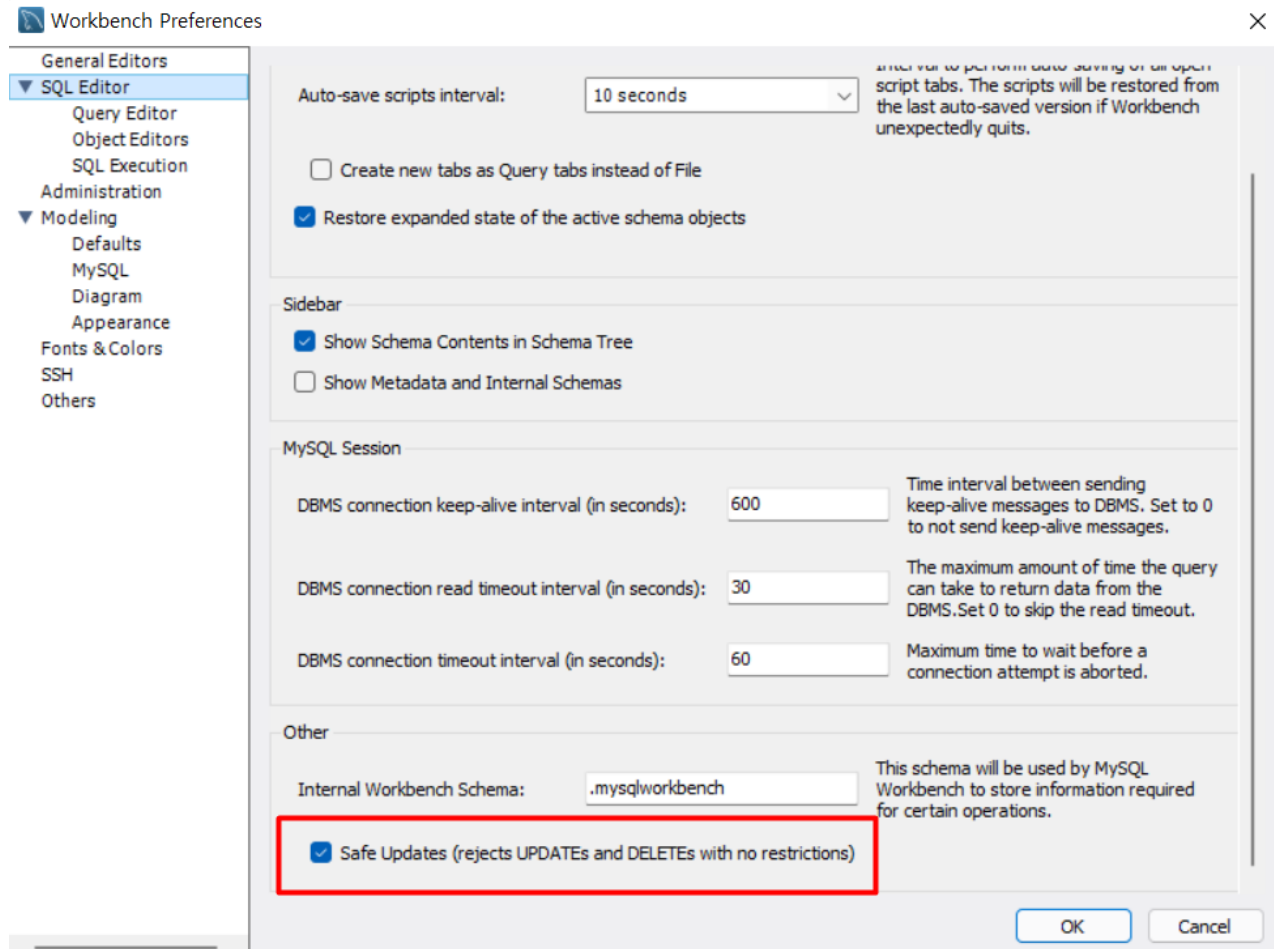
update 문 실행

set SQL_SAFE_UPDATES = 1; # enable safe mode

2. UPDATE 문

■ safe-mode 해지

- edit > preferences > SQL editor
- 변경 후 적용하려면 workbench 재시작해야 한다.



2. UPDATE 문

- UPDATE 문은 특정 속성 값을 수정하는 명령이다.
- UPDATE 문의 기본 문법

```
UPDATE 테이블이름
SET      속성이름1=값1[, 속성이름2=값2, ...]
[WHERE <검색조건>];
```

질의 3-47 Customer 테이블에서 고객번호가 5인 고객의 주소를 '대한민국 부산'으로 변경하시오.

```
SET SQL_SAFE_UPDATES=0; /* Safe Updates 옵션 미 해제 시 실행 */
```

```
UPDATE Customer
SET address='대한민국 부산'
WHERE custid=5;
```

| 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 부산 | NULL |

2. UPDATE 문

- UPDATE 문은 다른 테이블의 속성 값을 이용할 수도 있음
- 부속질의(서브쿼리)를 SET 과 WHERE 절에 사용할 수 있음

질의 3-48 Book 테이블에서 14번 '스포츠 의학'의 출판사를 imported_book 테이블의 21번 책의 출판사와 동일하게 변경하시오.

```
UPDATE Book
SET publisher = (SELECT publisher
                  FROM imported_book
                  WHERE bookid = '21')
WHERE bookid = '14' ;
```

```
update book set publisher=(select publisher
                             from book where bookid='15')
where bookid = '10';
```

| 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 |
| 11 | 스포츠 의학 | 한솔의학... | 90000 |
| 12 | 스포츠 의학 | 한솔의학... | 90000 |
| 13 | 스포츠 의학 | 한솔의학... | 90000 |
| 14 | 스포츠 의학 | Pearson | NULL |
| 21 | Zen Golf | Pearson | 12000 |
| 22 | Soccer Skills | Human Kin... | 15000 |

* 주의 : 같은 테이블의 값을 직접 SELECT 해서 사용할 수 없음

3. DELETE 문

- DELETE 문은 테이블에 있는 기존 튜플을 삭제하는 명령
- 검색조건에 해당되는 튜플을 삭제함
- WHERE 절에 부속질의(서브쿼리)를 사용할 수 있음
- DELETE 문의 기본 문법

```
DELETE FROM   테이블이름  
[WHERE 검색조건]; /* where 절이 없다면 ?*/
```

3. DELETE 문

질의 3-49 Book 테이블에서 도서번호가 11인 도서를 삭제하시오.

```
DELETE FROM Book  
WHERE bookid = '11';
```

| 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 |
| 12 | 스포츠 의학 | 한솔의학... | 90000 |
| 13 | 스포츠 의학 | 한솔의학... | 90000 |
| 14 | 스포츠 의학 | Pearson | NULL |
| 21 | Zen Golf | Pearson | 12000 |
| 22 | Soccer Skills | Human Kin... | 15000 |

질의 3-50 모든 고객을 삭제하시오. /* 오류가 난다면 그 이유는? */

```
DELETE FROM Customer;
```

연습문제

1. 마당서점에서 다음의 심화된 질문에 대해 SQL 문을 작성하시오.

- (1) 박지성이 구매한 도서의 출판사와 같은 출판사에서 도서를 구매한 고객의 이름
- (2) 두 개 이상의 서로 다른 출판사에서 도서를 구매한 고객의 이름
- (3) (생략) 전체 고객의 30% 이상이 구매한 도서

2. 다음 질의에 대해 DML 문을 작성하시오.

- (1) 새로운 도서 ('스포츠 세계', '대한미디어', 10000원)이 마당서점에 입고되었다.
삽입이 안 될 경우 필요한 데이터가 더 있는지 찾아보자.
- (2) '삼성당'에서 출판한 도서를 삭제해야 한다.
- (3) '이상미디어'에서 출판한 도서를 삭제해야 한다. 삭제가 안 될 경우 원인을 생각해보자.
- (4) 출판사 '대한미디어'가 '대한출판사'로 이름을 바꾸었다.

1. 상관 부속질의
2. 튜플 변수
3. 집합 연산
4. EXISTS
5. CREATE
6. ALTER
7. DROP
8. INSERT
9. UPDATE
10. DELETE



Chapter 04

SQL 고급

목차

01

내장 함수

02

부속질의

03

뷰

04

인덱스

학습목표

- ❖ 내장 함수의 의미를 알아본다.
- ❖ 자주 사용되는 내장 함수 몇 가지를 직접 실습해본다.
- ❖ 부속질의의 의미와 종류를 알아보고 직접 실습해본다.
- ❖ 뷰의 의미를 알아보고, 뷰를 직접 생성, 수정, 삭제해본다.
- ❖ 데이터베이스의 저장 구조와 인덱스의 관계를 알아본다.
- ❖ 인덱스를 직접 생성, 수정, 삭제해본다. ㅏ ㅏ

01. 내장 함수

1. SQL 내장 함수
2. NULL 값 처리
3. 행번호 출력



1. SQL 내장 함수

- SQL에서는 함수의 개념을 사용
- 수학의 함수와 마찬가지로 특정 값이나 열의 값을 입력 받아 그 값을 계산하여 결과 값을 돌려줌

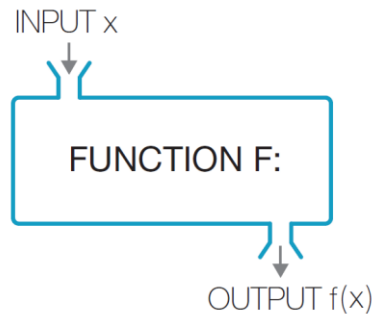


그림 4-1 함수의 원리

- SQL의 함수는 DBMS가 제공하는 내장 함수(built-in function), 사용자가 필요에 따라 직접 만드는 사용자 정의 함수(user-defined function)로 나뉨

1. SQL 내장 함수

- SQL 내장 함수는 상수나 속성 이름을 입력 값으로 받아 단일 값을 결과로 반환함
- 모든 내장 함수는 최초에 선언될 때 유효한 입력 값을 받아야 함

표 4-1 MySQL에서 제공하는 주요 내장 함수

| 구분 | | 함수 |
|------------------|--------------|--|
| 단일행 함수 | 숫자 함수 | ABS, CEIL, COS, EXP, FLOOR, LN, LOG, MOD, POWER, RAND, ROUND, SIGN, TRUNCATE |
| | 문자 함수(문자 반환) | CHAR, CONCAT, LEFT, RIGHT, LOWER, UPPER, LPAD, RPAD, LTRIM, RTRIM, REPLACE, REVERSE, RIGHT, SUBSTR, TRIM |
| | 문자 함수(숫자 반환) | ASCII, INSTR, LENGTH |
| | 날짜·시간 함수 | ADDDATE, CURRENT_DATE, DATE, DATEDIFF, DAYNAME, LAST_DAY, SYSDATE, TIME |
| | 변환 함수 | CAST, CONVERT, DATE_FORMAT, STR_TO_DATE |
| | 정보 함수 | DATABASE, SCHEMA, ROW_COUNT, USER, VERSION |
| | NULL 관련 함수 | COALESCE, ISNULL, IFNULL, NULLIF |
| 집계 함수 | | AVG, COUNT, MAX, MIN, STD, STDDEV, SUM |
| 윈도우 함수(혹은 분석 함수) | | CUME_DIST, DENSE_RANK, FIRST_VALUE, LAST_VALUE, LEAD, NTILE, RANK, ROW_NUMBER |

1. SQL 내장 함수

❖ 숫자 함수

표 4-2 숫자 함수의 종류

| 함수 | 설명 |
|---------------------|--|
| ABS(숫자) | 숫자의 절댓값을 계산 $ABS(-4.5) => 4.5$ |
| CEIL(숫자) | 숫자보다 크거나 같은 최소의 정수 $CEIL(4.1) => 5$ |
| FLOOR(숫자) | 숫자보다 작거나 같은 최소의 정수 $FLOOR(4.1) => 4$ |
| ROUND(숫자, m) | 숫자의 반올림, m은 반올림 기준 자릿수 $ROUND(5.36, 1) => 5.40$ |
| LOG(n, 숫자) | 숫자의 자연로그 값을 반환 $LOG(10) => 2.30259$ |
| POWER(숫자, n) | 숫자의 n제곱 값을 계산 $POWER(2, 3) => 8$ |
| SQRT(숫자) | 숫자의 제곱근 값을 계산(숫자는 양수) $SQRT(9.0) => 3.0$ |
| SIGN(숫자) | 숫자가 음수면 -1, 0이면 0, 양수면 1 $SIGN(3.45) => 1$ |

1. SQL 내장 함수

❖ 수학 함수

■ ABS 함수 : 절댓값을 구하는 함수

질의 4-1 -78과 +78의 절댓값을 구하시오.

```
SELECT ABS(-78), ABS(+78); /* FROM 이 없는 SELECT */  
FROM Dual; /* 가상의 테이블 */
```

| ABS(-78) | ABS(+78) |
|----------|----------|
| 78 | 78 |

■ ROUND 함수 : 반올림한 값을 구하는 함수

질의 4-2 4.875를 소수 첫째 자리까지 반올림한 값을 구하시오.

```
SELECT ROUND(4.875, 1);  
FROM Dual;
```

| ROUND(4.875, 1) |
|-----------------|
| 4.9 |

■ 숫자 함수의 연산

질의 4-3 고객별 평균 주문 금액을 백 원 단위로 반올림한 값을 구하시오.

```
SELECT custid '고객번호', ROUND(SUM(saleprice)/COUNT(*), -2) '평균금액'  
FROM Orders  
GROUP BY custid;
```

| 고객 번호 | 평균 금액 |
|----------|----------|
| 1 | 13000 |
| 2 | 7500 |
| 3 | 10300 |
| 4 | 16500 |

1. SQL 내장 함수

❖ 문자 함수

표 4-3 문자 함수의 종류

| 반환 구분 | 함수 | 설명 |
|---|--------------------------|--|
| 문자값 반환 함수 s : 문자열 c : 문자 n : 정수 k : 정수 | CONCAT(s1,s2) | 두 문자열을 연결, CONCAT('마당', ' 서점') => '마당 서점' |
| | LOWER(s) | 대상 문자열을 모두 소문자로 변환, LOWER('MR. SCOTT') => 'mr. scott' |
| | LPAD(s,n,c) | 대상 문자열의 왼쪽부터 지정한 자리수까지 지정한 문자로 채움 LPAD('Page 1', 10, '*') => '*****Page 1', SELECT LPAD('ABC',10,'0') FROM DUAL; => '0000000ABC' |
| | REPLACE(s1,s2,s3) | 대상 문자열의 지정한 문자를 원하는 문자로 변경 REPLACE('JACK & JUE', 'J', 'BL') => 'BLACK & BLUE' |
| | RPAD(s,n,c) | 대상 문자열의 오른쪽부터 지정한 자리수까지 지정한 문자로 채움 RPAD('AbC', 5, '*') => 'AbC**' |
| | SUBSTR(s,n,k) | 대상 문자열의 지정된 자리에서부터 지정된 길이만큼 잘라서 반환 SUBSTR('ABCDEFGH', 3, 4) => 'CDEFH' |
| | TRIM(c FROM s) | 대상 문자열의 양쪽에서 지정된 문자를 삭제(문자열만 넣으면 기본값으로 공백 제거) TRIM('= ' FROM '==BROWNING==') => 'BROWNING' |
| | UPPER(s) | 대상 문자열을 모두 대문자로 변환 UPPER('mr. scott') => 'MR. SCOTT' |
| 숫자값 반환 함수 | ASCII(c) | 대상 알파벳 문자의 아스키 코드 값을 반환, ASCII('D') => 68 |
| | LENGTH(s) | 대상 문자열의 Byte 반환, 알파벳 1byte, 한글 3byte (UTF8) LENGTH('CANDIDE') => 7 , select length('😊'); => 4 , select length('가'); => 3 |
| | CHAR_LENGTH(s) | 문자열의 문자 수를 반환, CHAR_LENGTH('데이터') => 3 |

1. SQL 내장 함수

❖ 문자 함수

■ REPLACE : 문자열을 치환하는 함수

질의 4-4 도서제목에 야구가 포함된 도서를 농구로 변경한 후 도서 목록을 보이시오.

```
SELECT    bookid, REPLACE(bookname, '야구', '농구') bookname, publisher, price
FROM      Book;
```

| 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 | 농구의 추억 | 이상미디어 | 20000 |
| 8 | 농구를 부탁해 | 이상미디어 | 13000 |
| 9 | 올림픽 이야기 | 삼성당 | 7500 |
| 10 | Olympic Champions | Pearson | 13000 |

1. SQL 내장 함수

❖ 문자 함수

- **LENGTH, CHAR_LENGTH** : 바이트수와 글자의 수를 세어주는 함수 (단위가 바이트 (byte)가 아닌 문자 단위)

질의 4-5 굿스포츠에서 출판한 도서의 제목과 제목의 글자 수를 확인하시오.
(한글은 2바이트 혹은 UNICODE 경우는 3바이트를 차지함)

```
SELECT    bookname '제목', CHAR_LENGTH(bookname) '문자수',  
          LENGTH(bookname) '바이트수'  
FROM      Book  
WHERE     publisher='굿스포츠';
```

| 제목 | 문 자 수 | 바이 트 수 |
|----------|-------------|--------------|
| 축구의 역사 | 6 | 16 |
| 피겨 교본 | 5 | 13 |
| 역도 단계별기술 | 8 | 22 |

- **질의 4-6** 마당서점의 고객 중에서 같은 성(姓)을 가진 사람이 몇 명이나 되는지 성별 인원수를 구하시오.

```
SELECT    SUBSTR(name, 1, 1) '성', COUNT(*) '인원'  
FROM      Customer  
GROUP BY SUBSTR(name, 1, 1);
```

| 성 | 인원 |
|---|----|
| 박 | 2 |
| 김 | 1 |
| 장 | 1 |
| 추 | 1 |

1. SQL 내장 함수

❖ 날짜·시간 함수

표 4-4 날짜·시간 함수의 종류

| 함수 | 반환형 | 설명 |
|--------------------------------------|---------|---|
| STR_TO_DATE(string, format)) | DATE | 문자열(String) 데이터를 날짜형(Date)으로 반환 STR_TO_DATE('2019-02-14', '%Y-%m-%d') => 2019-02-14 |
| DATE_FORMAT(date, format) | STRING | 날짜형(Date) 데이터를 문자열(VARCHAR)로 반환 DATE_FORMAT('2019-02-14', '%Y-%m-%d') => '2019-02-14' |
| ADDDATE(date, interval) | DATE | DATE 형의 날짜에서 INTERVAL 지정한 시간만큼 더함 ADDDATE('2019-02-14', INTERVAL 10 DAY) => 2019-02-24 |
| DATE(date) | DATE | DATE 형의 날짜 부분을 반환 SELECT DATE('2003-12-31 01:02:03'); => 2003-12-31 |
| DATEDIFF(date1, date2) | INTEGER | DATE 형의 date1 - date2 날짜 차이를 반환 SELECT DATEDIFF('2019-02-14', '2019-02-04') => 10 |
| SYSDATE | DATE | DBMS 시스템상의 오늘 날짜를 반환하는 함수 SYSDATE() => 2018-06-30 21:47:01 |

1. SQL 내장 함수

❖ 날짜 함수

표 4-5 format의 주요 지정자

| 인자 | 설명 |
|----|-------------------------|
| %w | 요일 순서(0~6, Sunday=0) |
| %W | 요일(Sunday~Saturday) |
| %a | 요일의 약자(Sun~Sat) |
| %d | 1달 중 날짜(00~31) |
| %j | 1년 중 날짜(001~366) |
| %h | 12시간(01~12) |
| %H | 24시간(00~23) |
| %i | 분(0~59) |
| %m | 월 순서(01~12, January=01) |
| %b | 월 이름 약어(Jan~Dec) |
| %M | 월 이름(January~December) |
| %s | 초(0~59) |
| %Y | 4자리 연도 |
| %y | 4자리 연도의 마지막 2 자리 |

1. SQL 내장 함수

❖ 날짜 함수

질의 4-7 마당서점은 주문일로부터 10일 후 매출을 확정한다. 각 주문의 확정일자를 구하시오.

```
SELECT   orderid '주문번호', orderdate '주문일',  
          ADDDATE(orderdate, INTERVAL 10 DAY) '확정'  
FROM      Orders;
```

| 주문 번호 | 주문일 | 확정 |
|----------|------------|------------|
| 1 | 2014-07-01 | 2014-07-11 |
| 2 | 2014-07-03 | 2014-07-13 |
| 3 | 2014-07-03 | 2014-07-13 |
| 4 | 2014-07-04 | 2014-07-14 |
| 5 | 2014-07-05 | 2014-07-15 |
| 6 | 2014-07-07 | 2014-07-17 |
| 7 | 2014-07-07 | 2014-07-17 |
| 8 | 2014-07-08 | 2014-07-18 |
| 9 | 2014-07-09 | 2014-07-19 |
| 10 | 2014-07-10 | 2014-07-20 |

1. SQL 내장 함수

❖ 날짜 함수

- STR_TO_DATE : 문자형으로 저장된 날짜를 날짜형으로 변환하는 함수
- DATE_FORMAT : 날짜형을 문자형으로 변환하는 함수

질의 4-8 마당서점이 2014년 7월 7일에 주문받은 도서의 주문번호, 주문일, 고객번호, 도서번호를 모두 보이시오. 단, 주문일은 '%Y-%m-%d' 형태로 표시한다.

```
SELECT   orderid '주문번호', STR_TO_DATE(orderdate, '%Y-%m-%d') '주문일',  
          custid '고객번호', bookid '도서번호'  
FROM      Orders  
WHERE     orderdate=DATE_FORMAT('20140707', '%Y%m%d');
```

| 주문 번호 | 주문일 | 고객 번호 | 도서 번호 |
|----------|------------|----------|----------|
| 6 | 2014-07-07 | 1 | 2 |
| 7 | 2014-07-07 | 4 | 8 |

1. SQL 내장 함수

❖ 날짜 함수

- SYSDATE : MySQL의 현재 날짜와 시간을 반환하는 함수

질의 4-9 DBMS 서버에 설정된 현재 날짜와 시간, 요일을 확인하시오.

```
SELECT    SYSDATE(),  
          DATE_FORMAT(SYSDATE(), '%Y/%m/%d %M %h:%s') 'SYSDATE_1';
```

| SYSDATE() | SYSDATE_1 |
|---------------------|----------------------|
| 2019-05-30 13:33:46 | 2019/05/30 May 01:46 |

1. 다음 내장 함수의 결과를 적으시오.

ABS(-15)

CEIL(15.7)

COS(3.14159)

FLOOR(15.7)

LOG(10,100)

MOD(11,4)

POWER(3,2)

ROUND(15.7)

SIGN(-15)

TRUNC(15.7)

CHAR(67 USING utf8)

CONCAT('HAPPY', 'Birthday')

LOWER('Birthday')

LPAD('Page 1', 15, '*')

REPLACE('JACK', 'J', 'BL')

RPAD('Page 1', 15, '*')

SUBSTR('ABCDEFGH', 3, 4)

TRIM(LEADING 0 FROM '00AA00')

UPPER('Birthday')

ASCII('A')

LENGTH('Birthday')

ADDDATE('2019-02-14', INTERVAL 10 DAY)

LAST_DAY(SYSDATE())

NOW()

DATE_FORMAT(SYSDATE(), '%Y')

CONCAT(123)

STR_TO_DATE('12 05 2014', '%d %m %Y')

CAST('12.3' AS DECIMAL(3,1))

IF(1=1, 'aa', 'bb')

IFNULL(123, 345)

IFNULL(NULL, 123)

2. NULL 값 처리

■ NULL 값이란?

- 아직 지정되지 않은 값
- NULL 값은 '0', '' (빈 문자), '' (공백) 등과 다른 특별한 값
- NULL 값은 비교 연산자로 비교가 불가능함
- NULL 값의 연산을 수행하면 결과 역시 NULL 값으로 반환됨

■ 집계 함수를 사용할 때 주의할 점

- 'NULL+숫자' 연산의 결과는 NULL
- 집계 함수 계산 시 NULL이 포함된 행은 집계에서 빠짐
- 해당되는 행이 하나도 없을 경우 SUM, AVG 함수의 결과는 NULL이 되며, COUNT 함수의 결과는 0.

2. NULL 값 처리

■ NULL 값에 대한 연산과 집계 함수

(* Mybook 테이블 생성은 다음 페이지 스크립트 참조)

Mybook

| bookid | price |
|--------|-------|
| 1 | 10000 |
| 2 | 20000 |
| 3 | NULL |

```
SELECT price+100
FROM Mybook
WHERE bookid=3;
```

| |
|-----------|
| price+100 |
| NULL |

```
SELECT SUM(price), AVG(price), COUNT(*), COUNT(price)
FROM Mybook;
```

| SUM(price) | AVG(price) | COUNT(*) | COUNT(price) |
|------------|------------|----------|--------------|
| 30000 | 15000.0000 | 3 | 2 |

```
SELECT SUM(price), AVG(price), COUNT(*)
FROM Mybook
WHERE bookid >= 4;
```

| SUM(price) | AVG(price) | COUNT(*) |
|------------|------------|----------|
| NULL | NULL | 0 |

mybook.sql

■ -- Mybook 스키마 생성 -- MySQL

```
CREATE TABLE Mybook (  
    bookid    INTEGER,  
    price     INTEGER  
);
```

-- Mybook 데이터 생성

```
INSERT INTO Mybook VALUES(1, 10000);  
INSERT INTO Mybook VALUES(2, 20000);  
INSERT INTO Mybook VALUES(3, NULL);
```

```
COMMIT;
```


2. NULL 값 처리

■ NULL 값을 확인하는 방법 – IS NULL, IS NOT NULL

- NULL 값을 찾을 때는 '=' 연산자가 아닌 'IS NULL'을 사용,
- NULL이 아닌 값을 찾을 때는 '< >' 연산자가 아닌 'IS NOT NULL'을 사용함

Mybook

| bookid | price |
|--------|-------|
| 1 | 10000 |
| 2 | 20000 |
| 3 | NULL |

```
SELECT *  
FROM Mybook  
WHERE price IS NULL;
```

| bookid | price |
|--------|-------|
| 3 | NULL |

```
SELECT *  
FROM Mybook  
WHERE price = '';
```

| bookid | price |
|--------|-------|
| | |

2. NULL 값 처리

- IFNULL : NULL 값을 다른 값으로 대체하여 연산하거나 다른 값으로 출력

IFNULL(속성, 값) /* 속성 값이 NULL이면 '값'으로 대체한다 */

질의 4-10 이름, 전화번호가 포함된 고객목록을 보이시오. 단, 전화번호가 없는 고객은 '연락처없음'으로 표시한다.

```
SELECT    name '이름', IFNULL(phone, '연락처없음') '전화번호'
FROM      Customer;
```

| 이름 | 전화번호 |
|-----|---------------|
| 박지성 | 000-5000-0001 |
| 김연아 | 000-6000-0001 |
| 장미란 | 000-7000-0001 |
| 추신수 | 000-8000-0001 |
| 박세리 | 연락처없음 |

3. 행번호 출력

- 내장 함수는 아니지만 자주 사용되는 문법
- MySQL에서 변수는 이름 앞에 @ 기호를 붙이며 치환문에는 SET과 := 기호를 사용함
- 자료를 일부분만 확인하여 처리할 때 유용함.

질의 4-11 고객 목록에서 고객번호, 이름, 전화번호를 앞의 두 명만 보이시오.

```
SET      @seq:=0;
SELECT   (@seq:=@seq+1) '순번', custid, name, phone
FROM     Customer
WHERE    @seq < 2;
```

| 순번 | custid | name | phone |
|----|--------|------|---------------|
| 1 | 1 | 박지성 | 000-5000-0001 |
| 2 | 2 | 김연아 | 000-6000-0001 |

2. Mybook 테이블을 생성하고 NULL에 관한 다음 SQL 문에 답하시오.
질의의 결과를 보면서 NULL에 대한 개념을 정리해보시오.

Mybook

| bookid | price |
|--------|-------|
| 1 | 10000 |
| 2 | 20000 |
| 3 | NULL |

- (1) SELECT *
FROM Mybook;
- (2) SELECT bookid, IFNULL(price, 0)
FROM Mybook;
- (3) SELECT *
FROM Mybook
WHERE price IS NULL;
- (4) SELECT *
FROM Mybook
WHERE price=' ';
- (5) SELECT bookid, price+100
FROM Mybook;
- (6) SELECT SUM(price), AVG(price), COUNT(*)
FROM Mybook
WHERE bookid >= 4;
- (7) SELECT COUNT(*), COUNT(price)
FROM Mybook;
- (8) SELECT SUM(price), AVG(price)
FROM Mybook;

3. MySQL의 행번호를 처리하는 다음의 SQL 문에 답하시요. 데이터는 마당서점 데이터베이스를 이용

- | | |
|------------|---|
| (1) SELECT | * |
| FROM | Book; |
| (2) SELECT | *, @RNUM := @RNUM + 1 AS ROWNUM |
| FROM | Book, (SELECT @RNUM := 0) R |
| WHERE | @RNUM < 5; |
| (3) SELECT | *, @RNUM := @RNUM + 1 AS ROWNUM |
| FROM | Book, (SELECT @RNUM := 1) R |
| WHERE | @RNUM <= 5; |
| ORDER BY | price; |
| (4) SELECT | *, @RNUM := @RNUM + 1 AS ROWNUM |
| FROM | (SELECT * FROM Book ORDER BY price) b, |
| | (SELECT @RNUM := 0) R |
| WHERE | @ROUM < 5; |
| (5) SELECT | *, @RNUM := @RNUM + 1 AS ROWNUM |
| FROM | (SELECT * FROM Book WHERE @RNUM<=5) b, |
| | (SELECT @RNUM := 0) R |
| ORDER BY | price; |
| (6) SELECT | *, @RNUM := @RNUM + 1 AS ROWNUM |
| FROM | (SELECT * FROM Book WHERE @RNUM <= 5 ORDER BY price) b, |
| | (SELECT @RNUM := 0) R; |

02. 부속질의

1. 스칼라 부속질의 – SELECT 부속질의
2. 인라인 뷰 – FROM 부속질의
3. 중첩질의 – WHERE 부속질의



부속질의

❖ 부속질의(subquery)란?

- 하나의 SQL 문 안에 다른 SQL 문이 중첩된(nested) 질의
- 다른 테이블에서 가져온 데이터로 현재 테이블에 있는 정보를 찾거나 가공할 때 사용
- 보통 데이터가 대량일 때 데이터를 모두 합쳐서 연산하는 조인보다 필요한 데이터만 찾아서 공급해주는 부속질의가 성능이 더 좋음
- 주질의(main query, 외부질의)와 부속질의(sub query, 내부질의)로 구성됨

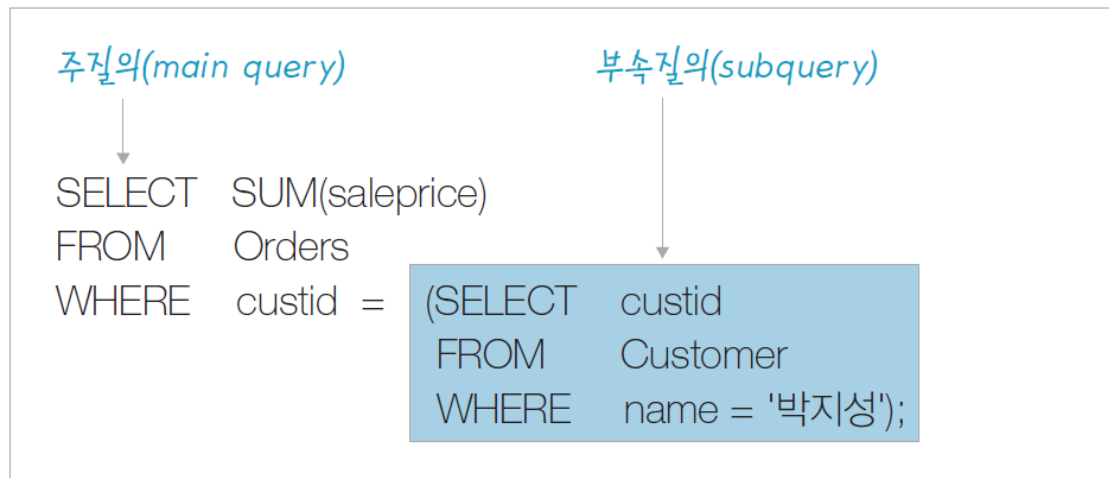


그림 4-2 부속질의

부속질의

표 4-6 부속질의의 종류

| 명칭 | 위치 | 영문 및 동의어 | 설명 |
|----------|----------|--|---|
| 스칼라 부속질의 | SELECT 절 | scalar subquery | SELECT 절에서 함수처럼 사용되며 단일 값, 한 행만을 반환하기 때문에 스칼라 부속질의라고 함. |
| 인라인 뷰 | FROM 절 | inline view, table subquery | FROM 절에서 결과를 뷰(view) 형태로 반환하기 때문에 인라인 뷰라고 함. |
| 중첩질의 | WHERE 절 | nested subquery, predicate subquery | WHERE 절에 술어와 같이 사용되며 결과를 한정시키기 위해 사용됨. 상관 혹은 비상관 형태. |

1. 스칼라 부속질의 – SELECT 부속질의

❖ 스칼라 부속질의(scalar subquery)란?

- SELECT 절에서 사용되는 부속질의로, 부속질의의 결과 값을 단일 행, 단일 열의 스칼라 값으로 반환함
- 결과값이 다중 행이거나 다중 열이라면 DBMS 는 그 중 어떤 행, 어떤 열을 출력해야 하는지 알 수 없어 에러를 출력함, 결과가 없는 경우에는 NULL값을 출력함.
- 원칙적으로 스칼라 값이 들어갈 수 있는 모든 곳에 사용 가능하며, 일반적으로 SELECT 문과 UPDATE SET 절에 사용됨
- 주질의와 부속질의와의 관계는 상관/비상관 모두 가능함

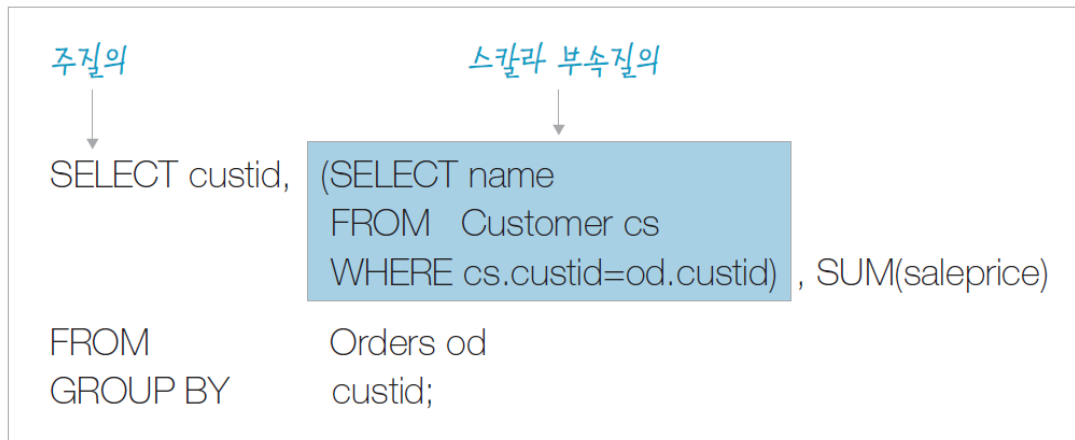


그림 4-3 스칼라 부속질의

* 스칼라 값이란 벡터 값에 대응되는 말로 단일 값을 의미함

1. 스칼라 부속질의 – SELECT 부속질의

질의 4-12 마당서점의 고객별 판매액을 보이시오(고객이름과 고객별 판매액을 출력).

```
SELECT      ( SELECT  name
              FROM    Customer cs
              WHERE   cs.custid=od.custid ) 'name', SUM(saleprice) 'total'
FROM        Orders od
GROUP BY    od.custid;
```

| name | total |
|------|-------|
| 박지성 | 39000 |
| 김연아 | 15000 |
| 장미란 | 31000 |
| 추신수 | 33000 |

1. 스칼라 부속질의 - SELECT 부속질의

```
SELECT    custid,  
          SUM(saleprice) 'total'  
FROM      Orders od  
GROUP BY custid ;
```



| custid | total |
|--------|-------|
| 1 | 39000 |
| 2 | 15000 |
| 3 | 31000 |
| 4 | 33000 |

```
SELECT    name  
FROM      Customer cs  
WHERE     cs.custid = od.custid
```

Custid 1의 이름은?



```
SELECT    (SELECT    name  
            FROM      Customer cs  
            WHERE     cs.custid = od.custid) name,  
          SUM(saleprice) 'total'  
FROM      Orders od  
GROUP BY od.custid ;
```



| name | total |
|------|-------|
| 박지성 | 39000 |
| 김연아 | 15000 |
| 장미란 | 31000 |
| 추신수 | 33000 |

그림 4-4 마당서점의 고객별 판매액

1. 스칼라 부속질의 – SELECT 부속질의

다음 실습을 위해 새로운 필드 도서이름(bname)을 추가해보자.

```
ALTER TABLE Orders ADD bname VARCHAR(40);
```

새로운 필드에는 NULL 값이 저장되어 있다.

bookid가 1,2,3,...10인 경우 각각의 도서이름을 수정해보자.

질의 4-12 Orders 테이블에 각 주문에 맞는 도서이름을 입력하시오.

```
UPDATE   Orders
SET       bookname = ( SELECT bookname
                        FROM Book
                        WHERE Book.bookid=Orders.bookid );
```

| orderid | custid | bookid | saleprice | orderdate | bname |
|---------|--------|--------|-----------|------------|-------------------|
| 1 | 1 | 1 | 6000 | 2014-07-01 | 축구의 역사 |
| 2 | 1 | 3 | 21000 | 2014-07-03 | 축구의 이해 |
| 3 | 2 | 5 | 8000 | 2014-07-03 | 피겨 교본 |
| 4 | 3 | 6 | 6000 | 2014-07-04 | 역도 단계별기술 |
| 5 | 4 | 7 | 20000 | 2014-07-05 | 야구의 추억 |
| 6 | 1 | 2 | 12000 | 2014-07-07 | 축구하는 여자 |
| 7 | 4 | 8 | 13000 | 2014-07-07 | 야구를 부탁해 |
| 8 | 3 | 10 | 12000 | 2014-07-08 | Olympic Champions |
| 9 | 2 | 10 | 7000 | 2014-07-09 | Olympic Champions |
| 10 | 3 | 8 | 13000 | 2014-07-10 | 야구를 부탁해 |

2. 인라인 뷰- FROM 부속질의

❖ 인라인 뷰(inline view)란?

- FROM 절에서 사용되는 부속질의
- 테이블 이름 대신 인라인 뷰 부속질의를 사용하면 보통의 테이블과 같은 형태로 사용할 수 있음
- 부속질의 결과 반환되는 데이터는 다중 행, 다중 열이어도 상관없음
- 다만 가상의 테이블인 뷰 형태로 제공되어 상관 부속질의로 사용될 수는 없음

질의 4-14 고객번호가 2 이하인 고객의 판매액을 보이시오(고객이름과 고객별 판매액 출력).

```
SELECT    cs.name, SUM(od.saleprice) 'total'
FROM      (SELECT custid, name
           FROM    Customer
           WHERE   custid <= 2) cs,
           Orders  od
WHERE     cs.custid=od.custid
GROUP BY  cs.name;
```

| name | total |
|------|-------|
| 박지성 | 39000 |
| 김연아 | 15000 |

2. 인라인 뷰- FROM 부속질의

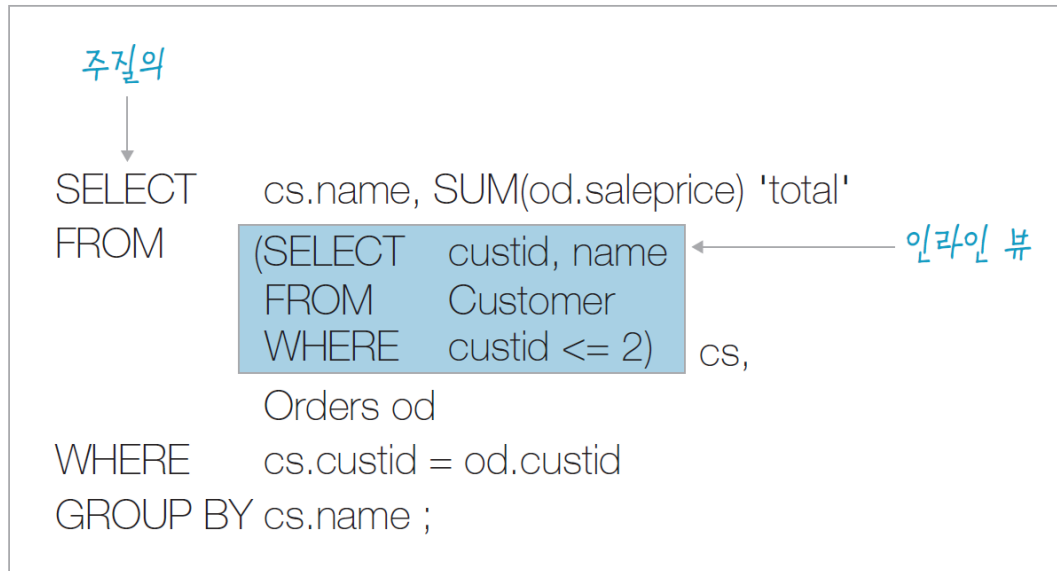


그림 4-5 인라인 뷰

아래 SQL과 비교해보자.

```
SELECT  cs.name, SUM(od.saleprice) 'total'
FROM    customer cs,
        Orders  od
WHERE   cs.custid=od.custid and od.custid <=2

GROUP BY cs.name;
```

3. 중첩질의 - WHERE 부속질의

- 중첩질의(nested subquery) : WHERE 절에서 사용되는 부속질의
- WHERE 절은 보통 데이터를 선택하는 조건 혹은 술어(predicate)와 같이 사용됨
→ 중첩질을 술어 부속질의(predicate subquery)라고도 함

표 4-7 중첩질의 연산자의 종류

| 술어 | 연산자 | 반환 행 | 반환 열 | 상관 |
|----------------|----------------------|------|------|----|
| 비교 | =, >, <, >=, <=, < > | 단일 | 단일 | 가능 |
| 집합 | IN, NOT IN | 다중 | 다중 | 가능 |
| 한정(quantified) | ALL, SOME(ANY) | 다중 | 단일 | 가능 |
| 존재 | EXISTS, NOT EXISTS | 다중 | 다중 | 필수 |

3. 중첩질의 - WHERE 부속질의

❖ 비교 연산자

부속질의가 반드시 단일 행, 단일 열을 반환해야 하며, 아닐 경우 질의를 처리할 수 없음

- ❖ 처리과정 : 주질의의 대상 열 값과 부속질의의 결과 값을 비교 연산자에 적용하여 참이면 주질의의 해당열을 출력한다.

질의 4-15 평균 주문금액 이하의 주문에 대해서 주문번호와 금액을 보이시오.

```
SELECT orderid, saleprice
FROM    Orders
WHERE   saleprice <= (SELECT AVG(saleprice)
                     FROM Orders);
```

| orderid | saleprice |
|---------|-----------|
| 1 | 6000 |
| 3 | 8000 |
| 4 | 6000 |
| 9 | 7000 |

질의 4-16 각 고객의 평균 주문금액보다 큰 금액의 주문 내역에 대해서 주문번호, 고객번호, 금액을 보이시오.

```
SELECT orderid, custid, saleprice
FROM    Orders od
WHERE   saleprice > (SELECT AVG(saleprice)
                     FROM Orders so
                     WHERE od.custid=so.custid);
```

| orderid | custid | saleprice |
|---------|--------|-----------|
| 2 | 1 | 21000 |
| 3 | 2 | 8000 |
| 5 | 4 | 20000 |
| 8 | 3 | 12000 |
| 10 | 3 | 13000 |

3. 중첩질의 - WHERE 부속질의

❖ IN, NOT IN

- IN 연산자는 주질의 속성 값이 부속질의에서 제공한 결과 집합에 있는지 확인하는 역할을 함
- IN 연산자는 부속질의의 결과 다중 행을 가질 수 있음
- 주질의는 WHERE 절에 사용되는 속성 값을 부속질의의 결과 집합과 비교해 하나라도 있으면 참이 됨
- NOT IN은 이와 반대로 값이 존재하지 않으면 참이 됨

질의 4-17 대한민국에 거주하는 고객에게 판매한 도서의 총판매액을 구하시오.

```
SELECT SUM(saleprice) 'total'
FROM Orders
WHERE custid IN (SELECT custid
                  FROM Customer
                  WHERE address LIKE '%대한민국%');
```

| |
|-------|
| total |
| 46000 |

* 다중열 반환하는 경우

```
SELECT 컬럼, 컬럼 ...
FROM 테이블
WHERE (컬럼1, 컬럼2, ...) IN (SELECT 컬럼1, 컬럼2, ...
                              FROM 테이블);
```

```
SELECT *
FROM EMP
WHERE (EMPNO, JOB) IN (SELECT EMPNO, JOB
                        FROM EMP
                        WHERE DEPTNO = 20);
```

3. 중첩질의 – WHERE 부속질의

❖ ALL, SOME(ANY)

- ALL은 모두, SOME(ANY)은 어떠한(최소한 하나라도)이라는 의미
- 구문 구조

```
scalar_expression { 비교연산자 ( =, <>, !=, >, >=, !=, <, <=, != ) }  
                { ALL | SOME | ANY } (부속질의)
```

질의 4-18 3번 고객이 주문한 도서의 최고 금액보다 더 비싼 도서를 구입한 주문의 주문번호와 금액을 보이시오.

```
SELECT orderid, saleprice  
FROM    Orders  
WHERE   saleprice > ALL (SELECT saleprice  
                        FROM    Orders  
                        WHERE custid='3');
```

| orderid | saleprice |
|---------|-----------|
| 2 | 21000 |
| 5 | 20000 |

3. 중첩질의 - WHERE 부속질의

❖ EXISTS, NOT EXISTS

- 데이터의 존재 유무를 확인하는 연산자
- 주질의에서 부속질의로 제공된 속성의 값을 가지고 부속질의에 조건을 만족하여 값이 존재하면 참이 되고, 주질의는 해당 행의 데이터를 출력함
- NOT EXISTS의 경우 이와 반대로 동작함
- 구문 구조

WHERE [NOT] EXISTS (부속질의)

질의 4-19 EXISTS 연산자로 대한민국에 거주하는 고객에게 판매한 도서의 총 판매액을 구하시오.

```
SELECT SUM(saleprice) 'total'
FROM Orders od
WHERE EXISTS (SELECT *
               FROM Customer cs
               WHERE address LIKE '%대한민국%' AND cs.custid=od.custid);
```

| |
|-------|
| total |
| 46000 |

5. 부속질의에 관한 다음 SQL 문을 수행해보고 어떤 질의에 대한 답인지 설명하시오.

```
(1) SELECT      custid, (SELECT      address
                                FROM      Customer cs
                                WHERE     cs.custid = od.custid) "address",
                                SUM(saleprice) "total"
FROM            Orders od
GROUP BY        od.custid;
```

```
(2) SELECT      cs.name, s
      FROM      (SELECT  custid, AVG(saleprice) s
                  FROM    Orders
                  GROUP BY custid) od, Customer cs
      WHERE     cs.custid = od.custid;
```

```
(3) SELECT      SUM(saleprice) "total"
FROM            Orders od
WHERE EXISTS    (SELECT      *
FROM            Customer cs
WHERE           custid <= 3 AND cs.custid = od.custid);
```