Chapter 04

SQL 고급

목차

학습목표

- 내장 함수의 의미를 알아보고 자주 사용되는 내장 함수 몇 가지를 직접 실습해 본다.
- 부속질의의 의미와 종류를 알아보고 직접 실습해 본다.
- 뷰의 의미를 알아보고 뷰를 직접 생성·수정·삭제해 본다.
- 데이터베이스의 저장 구조와 인덱스 간 관계, 인덱스의 구조를 이해한다.
- MySQL 인덱스 종류를 알아보고 인덱스를 직접 생성·수정·삭제해 본다.

01 내장 함수, NULL

1. SQL 내장 함수
2. NULL 값 처리
3. 행번호 출력

1. SQL 내장 함수

❖ 함수의 원리

- 함수 y=f(x)는 값 x를 함수 f에 넣으면 y 값을 결과로 반환한다는 의미

❖ SQL 함수의 분류

- 내장 함수 : DBMS가 제공
- 사용자 정의 함수 : 사용자가 필요에 따라 직접 만듦

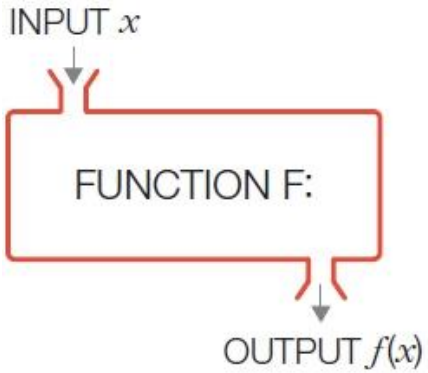


그림 4-1 함수의 원리

1. SQL 내장 함수

❖ SQL 내장 함수

- 상수나 속성 이름을 입력값으로 받아 단일 값을 결과로 반환함
- 모든 내장 함수는 사용될 때 유효한 입력값을 받아야 함
- SELECT 절과 WHERE 절, UPDATE 절 등에서 모두 사용 가능

```
SELECT   ... 함수명(인자1, 인자2, ...)
FROM     테이블이름
WHERE    ... 열이름=함수명(인자1, 인자2, ...);

UPDATE   테이블이름
SET      ... 열이름=함수명(인자1, 인자2, ...);
```

1. SQL 내장 함수

❖ MySQL에서 제공하는 주요 내장 함수

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_COUNR, USER, VERSION
	NULL 관련 함수	COALESCE, ISNULL, IFNULL, NULLIF
집계 함수		AVG, COUNT, MAX, MIN, STD, STDDEV, SUM
윈도 함수(혹은 분석 함수)		CUME_DIST, DENSE_RANK, FIRST_VALUE, LAST_VALUE, LEAD, NTILE, RANK, ROW_NUMBER

1. SQL 내장 함수

❖ 숫자 함수

- SQL 문에서는 수학의 기본적인 사칙 연산자(+, -, * , /)와 나머지(%) 연산자 기호를 그대로 사용
- MySQL은 이러한 연산자 중 사용 빈도가 높은 것을 내장 함수 형태로 제공

표 4-2 숫자 함수의 종류

함수	설명
ABS(숫자)	숫자의 절댓값을 계산함 예 ABS(-4.5) → 4.5
CEIL(숫자)	숫자보다 크거나 같은 최소의 정수 예 CEIL(4.1) → 5
FLOOR(숫자)	숫자보다 작거나 같은 최소의 정수 예 FLOOR(4.1) → 4
ROUND(숫자, m)	숫자의 반올림, m은 반올림 기준 자릿수 예 ROUND(5.36, 1) → 5.40
LOG(n, 숫자)	숫자의 자연로그 값을 반환함 예 LOG(10) → 2.30259
POWER(숫자, n)	숫자의 n 제곱 값을 계산함 예 POWER(2, 3) → 8
SQRT(숫자)	숫자의 제곱근 값을 계산함(숫자는 양수) 예 SQRT(9.0) → 3.0
SIGN(숫자)	숫자가 음수이면 -1, 0이면 0, 양수이면 1 예 SIGN(3.45) → 1

1. SQL 내장 함수 - 숫자 함수

❖ ABS 함수

- 절댓값을 구하는 함수
- 다음 질의는 SELECT 문에서 상수 값을 대상으로 ABS함수를 수행한 결과

질의 4-1

-78과 +78의 절댓값을 구하시오.

```
SELECT ABS(-78), ABS(+78);
```

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

❖ ROUND 함수

- 반올림한 값을 구하는 함수
- 다음 질의는 SELECT 문에서 상수 값을 대상으로 ROUND 함수를 수행한 결과

질의 4-2

4.875를 소수 첫째 자리까지 반올림한 값을 구하시오.

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

ROUND(4.875, 1)
4.9

1. SQL 내장 함수 - 숫자 함수

❖ 숫자 함수의 연산

- 숫자 함수에는 직접 숫자를 입력할 수도 있지만 열 이름을 사용할 수도 있음
- 여러 함수를 복합적으로 사용할 수도 있음
- 다음 질의는 ROUND 함수, SUM 함수와 COUNT 함수를 사용하여 연산을 수행

질의 4-3

고객별 평균 주문 금액을 100원 단위로 반올림한 값을 구하시오.

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

고객번호	평균금액
1	13000
2	7500
3	10300
4	16500

1. SQL 내장 함수

❖ 문자 함수

표 4-3 문자 함수의 종류(s: 문자열, c: 문자, n과 k: 정수)

문자값 반환 함수	설명
CONCAT(s1,s2)	두 문자열을 연결함 예 CONCAT('마당', '서점') → '마당서점'
LOWER(s)	대상 문자열을 모두 소문자로 변환함 예 LOWER('MR. SCOTT') → 'mr. scott'
LPAD(s,n,c)	대상 문자열의 왼쪽부터 지정한 자릿수까지 지정한 문자로 채움 예 LPAD('Page 1', 10, '*') → '****Page 1'
REPLACE(s1,s2,s3)	대상 문자열의 지정한 문자를 원하는 문자로 변경함 예 REPLACE('JACK & JUE', 'J', 'BL') → 'BLACK & BLUE'
RPAD(s,n,c)	대상 문자열의 오른쪽부터 지정한 자릿수까지 지정한 문자로 채움 예 RPAD('AbC', 5, '*') → 'AbC** '
SUBSTR(s,n,k)	대상 문자열의 지정된 자리에서부터 지정된 길이만큼 잘라서 반환함 예 SUBSTR('ABCDEFG', 3, 4) → 'CDEF'
TRIM(c FROM s)	대상 문자열의 양쪽에서 지정된 문자를 삭제함(문자열만 넣으면 기본값으로 공백 제거) 예 TRIM('=' FROM '==BROWNING==') → 'BROWNING'
UPPER(s)	대상 문자열을 모두 대문자로 변환함 예 UPPER('mr. scott') → 'MR. SCOTT'
숫자값 반환 함수	**설명**
ASCII(c)	대상 알파벳 문자의 아스키코드 값을 반환함 예 ASCII('D') → 68
LENGTH(s)	대상 문자열의 byte를 반환함(알파벳은 1byte, 한글은 3byte (UTF-8)) 예 LENGTH('CANDIDE') → 7
CHAR_LENGTH(s)	문자열의 문자 수를 반환함 예 CHAR_LENGTH('데이터') → 3

1. SQL 내장 함수 - 문자 함수

❖ REPALCE 함수

- 문자열을 치환하는 함수
- 예) 담당자의 실수로 도서의 제목을 잘못 입력한 경우 REPLACE 함수를 사용하면 일일이 변경하지 않고 한꺼번에 변경할 수 있음

질의 4-4

도서 제목에 야구가 포함된 도서를 농구로 변경한 후 도서 목록을 나타내시오.

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

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구 아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	배구 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 함수

- LENGTH( )는 바이트 수를 가져오는 함수
- CHAR_LENGTH 함수는 문자의 수를 가져오는 함수

질의 4-5

굿스포츠에서 출판한 도서의 제목과 제목의 문자 수, 바이트 수를 나타내시오.

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

제목	문자수	바이트수
축구의 역사	6	16
피겨 교본	5	13
배구 단계별기술	8	22

1. SQL 내장 함수 - 문자 함수

❖ SUBSTR 함수

- 문자열 중 특정 위치에서 시작하여 지정한 길이만큼의 문자열을 반환하는 함수
- 예) 마당서점의 고객 중 성(姓)이 같은 사람이 얼마나 있는지 알아보려면 이름 열에서 첫 번째 문자만 구하면 됨

질의 4-6

마당서점의 고객 중에서 성(姓)이 같은 사람이 몇 명이나 되는지 알고 싶다. 성별 인원수를 구하시오.

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

성	인원
박	2
김	2
추	1

TIP GROUP BY 절에는 열 이름뿐 아니라 [질의 4-6]과 같이 함수를 포함할 수도 있다.

1. SQL 내장 함수

❖ 날짜·시간 함수

- 날짜와 시간 부분을 나타내는 인수는 'format'으로 표기
- format은 날짜 형식 지정자로 날짜와 시간 부분을 표기하기 위해 특별한 규칙을 가짐

표 4-4 날짜 · 시간 함수의 종류

함수	반환형	설명
STR_TO_DATE(string, format)	DATE	문자열(STRING) 데이터를 날짜형(DATE)으로 반환함 예 STR_TO_DATE('2024-12-07', '%Y-%m-%d') → 2024-12-07
DATE_FORMAT(date, format)	STRING	날짜형(DATE) 데이터를 문자열(VARCHAR)로 반환함 예 DATE_FORMAT('2024-12-07', '%Y-%m-%d') → '2024-12-07'
ADDDATE(date, interval)	DATE	DATE형의 날짜에서 INTERVAL 지정한 시간만큼 더함 예 ADDDATE('2024-12-07', INTERVAL 10 DAY) → 2024-12-17
DATE(date)	DATE	DATE형의 날짜 부분을 반환함 예 DATE('2024-12-07 01:02:03') → 2024-12-07
DATEDIFF(date1, date2)	INTEGER	DATE형의 date1 - date2 날짜 차이를 반환함 예 DATEDIFF('2024-12-17', '2024-12-07') → 10
SYSDATE	DATE	DBMS 시스템상의 오늘 날짜를 반환함 예 SYSDATE( ) → 2024-12-07 21:47:01

1. SQL 내장 함수 – 날짜 · 시간 함수

- 날짜형 데이터는 '–'와 '+'를 사용하여 원하는 날짜로부터 이전(-)과 이후(+)를 계산할 수 있음
- 예) 날짜형 데이터 mydate 값이 '2024년 7월 1일'이라면 5일 전은 'INTERVAL -5 DAY', 5일 후는 'INTERVAL 5 DAY'와 같이 사용함

```
SELECT  ADDDATE('2024-07-01', INTERVAL -5 DAY) BEFORE5,
        ADDDATE('2024-07-01', INTERVAL 5 DAY) AFTER5;
```

BEFORE5	AFTER5
2024-06-26	2024-07-06

질의 4-7

마당서점은 주문일로부터 10일 후에 매출을 확정한다. 각 주문의 확정일자를 구하시오.

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

주문번호	주문일	확정
1	2024-07-01	2024-07-11
2	2024-07-03	2024-07-13
3	2024-07-03	2024-07-13
4	2024-07-04	2024-07-14
5	2024-07-05	2024-07-15
6	2024-07-07	2024-07-17
7	2024-07-07	2024-07-17
8	2024-07-08	2024-07-18
9	2024-07-09	2024-07-19
10	2024-07-10	2024-07-20

1. SQL 내장 함수 - 날짜 · 시간 함수

표 4-5 format의 주요 지정자(specifier)

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

- 예) '2024년 7월 1일'을 '%Y%m%d'로 쓰면 '20240701'이 됨
- 14시 20분 14초 같은 시각 정보도 포함하려면 '%Y%m%d%H%i%s' 형태로 쓰면 됨
- 결과는 '20240701142014'로 반환됨
- format 인자는 날짜·시간 함수를 필요에 따라 인자로 사용하기도 함

```
SELECT SYSDATE( ), DATE_FORMAT(SYSDATE( ), '%Y%m%d:%H%i%s');
```

1. SQL 내장 함수 - 날짜 · 시간 함수

❖ STR_TO_DATE 함수, DATE_FORMAT 함수

- STR_TO_DATE 함수는 CHAR형(문자열)으로 저장된 날짜를 DATE형으로 변환함
- DATE_FORMAT 함수는 STR_TO_DATE 함수와 반대로 날짜형을 문자형으로 변환함

질의 4-8

마당서점이 2024년 7월 7일에 주문받은 도서의 주문번호, 주문일, 고객번호, 도서번호를 모두 나타내시오. 단, 주문일은 '%Y-%m-%d' 형태로 표시한다.

```
SELECT  orderid '주문번호', DATE_FORMAT(orderdate, '%Y-%m-%d') '주문일',
        custid '고객번호', bookid '도서번호'
FROM    Orders
WHERE   orderdate= STR_TO_DATE('20240707', '%Y%m%d');
```

주문번호	주문일	고객번호	도서번호
6	2024-07-07	1	2
7	2024-07-07	4	8

1. SQL 내장 함수 - 날짜 · 시간 함수

❖ SYSDATE 함수

- MySQL 데이터베이스에 설정된 현재 날짜와 시간을 반환하는 함수

질의 4-9

DBMS 서버에 설정된 현재 날짜와 시간, 요일을 확인하시오.

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

SYSDATE( )	SYSDATE_1
2023-11-07 17:27:59	2023/11/07 Tue 05:29

연습문제 (Q4.1)

01 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)
- TRUNCATE(15.7, 0)
- CHAR(67 USING utf8)
- CONCAT('HAPPY', 'Birthday')
- LOWER('Birthday')
- LPAD('Page 1', 15, '*.')
- REPLACE('JACK', 'J', 'BL')
- RPAD('Page 1', 15, '*')
- SUBSTR('ABCDEFG', 3, 4)
- TRIM(LEADING 0 FROM '00AA00')
- UPPER('Birthday')
- ASCII('A')
- LENGTH('Birthday')
- ADDDATE('2024−02−14', INTERVAL 10 DAY)
- LAST_DAY(SYSDATE( ))
- NOW( )
- DATE_FORMAT(SYSDATE( ), '%Y')
- CONCAT(123)
- STR_TO_DATE('12 05 2024', '%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 값

- 아직 지정되지 않은 값 = 값을 알 수도 없고 적용할 수도 없다는 뜻
- NULL 값은 '0', ' '(빈 문자), ' '(공백) 등과 다른 특별한 값
- NULL 값은 =, < > 등과 같은 비교 연산자로 비교할 수 없음
- NULL 값의 연산을 수행하면 결과 역시 NULL 값으로 반환됨

❖ NULL 값에 대한 연산과 집계 함수

- 집계 함수를 사용할 때 NULL 값이 포함된 행에 대하여 다음과 같은 주의가 필요
  - 'NULL+숫자' 연산의 결과는 NULL
  - 집계 함수를 계산할 때 NULL이 포함된 행은 집계에서 빠짐
  - 해당되는 행이 하나도 없을 경우 SUM, AVG 함수의 결과는 NULL이 되고, COUNT 함수의 결과는 0이 됨

2. NULL 값 처리

- 다음 Mybook 테이블을 보면서 NULL 값이 어떻게 처리되는지 이해해보기

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

2. NULL 값 처리

❖ NULL 값을 확인하는 방법 - IS NULL, IS NOT NULL

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

Mybook

bookid	price
1	10000
2	20000
3	NULL

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

bookid	price
3	NULL

- 위의 SQL 문을 다음과 같이 작성하면 틀린 결과가 반환됨

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

bookid	price

```
SELECT *
FROM Mybook
WHERE price = NULL;
/* 마찬가지로 틀린 결과 */
```

2. NULL 값 처리

❖ IFNULL 함수

- NULL 값을 다른 값으로 대치하여 연산하거나 다른 값으로 출력하는 함수
- IFNULL 함수를 사용하면 NULL 값을 임의의 다른 값으로 변경 가능

```
IFNULL(속성, 값)  /* 속성값이 NULL이면 '값'으로 대치한다. */
```

질의 4-10

이름, 전화번호가 포함된 고객 목록을 나타내시오. 단, 전화번호가 없는 고객은 '연락처없음'으로 표시하시오.

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

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

3. 행번호 출력

- MySQL에서 변수는 이름 앞에 @ 기호를 붙이며 치환문에는 SET과 := 기호를 사용함

질의 4-11

고객 목록에서 고객번호, 이름, 전화번호를 앞의 2명만 나타내시오.

```
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

```
-- [참고] 순번 없이 2개의 투플만 출력하는 방법
SELECT custid, name, phone
FROM Customer
LIMIT 2;
```

연습문제 (Q4.2)

02 Mybook 테이블을 생성하고 NULL에 관한 다음 SQL 문에 답하시오. 또한 질의의 결과를 보면서 NULL에 대한 개념도 정리해 보시오.

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;
```

연습문제 (Q4.3)

04 마당서점 데이터베이스를 이용해 데이터 출력 개수 및 MySQL의 행번호를 처리하는 다음 SQL 문의 결과를 적으시오.

```
(1) SELECT      bookid, bookname, price
    FROM        Book; -- 모든 책 보기
(2) SELECT      bookid, bookname, price
    FROM        Book
    LIMIT       5; -- 임의로 5개만 보기
(3) SELECT      bookid, bookname, price
    FROM        Book
    ORDER BY    price
    LIMIT       5; -- price 순으로 5개만 보기
(4) SET         @RNUM:=0;
    SELECT      bookid, bookname, price, @RNUM:= @RNUM + 1 AS ROWNUM
    FROM        Book
    WHERE       @RNUM < 5;  -- 임의로 5개만 행번호와 함께 보기
(5) SELECT      bookid, bookname, price, @RNUM:= @RNUM + 1 AS ROWNUM
    FROM        Book, (SELECT @RNUM:= 0) R
    WHERE       @RNUM < 5; -- 임의로 5개만 행번호와 함께 보기
(6) SELECT      bookid, bookname, price, @RNUM:= @RNUM + 1 AS ROWNUM
    FROM        (SELECT * FROM Book ORDER BY price) b,
                (SELECT @RNUM:= 0) R
    WHERE       @RNUM < 5; -- 행번호와 함께 price 순으로 5개만 보기
(7) SELECT      bookid, bookname, price, @RNUM:= @RNUM + 1 AS ROWNUM
    FROM        (SELECT * FROM Book ORDER BY price) b,
                (SELECT @RNUM:= 0) R
    WHERE       @RNUM < 5
    LIMIT       5; -- 행번호와 함께 price 순으로 3개만 보기
```

02 부속질의

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

부속질의

❖ 부속질의

- 하나의 SQL 문 안에 다른 SQL 문이 중첩된 질의
- 주로 메인 쿼리의 조건에 따라 서브쿼리의 결과를 가져와서 메인 쿼리에서 사용하는 용도로 활용됨
- 다른 테이블에서 가져온 데이터로 현재 테이블에 있는 정보를 찾거나 가공하는 등의 작업을 수행할 수 있음

- 예) 마당서점 데이터베이스의 Orders 테이블에서 주문 내역은 알 수 있지만 도서를 주문한 고객의 이름은 알 수 없음. 고객 이름과 주문 내역을 같이 보려면 Orders 테이블과 Customer 테이블을 연관시켜야 함. 테이블의 관계를 기초로 박지성 고객의 주문 내역을 확인하려면 어떻게 해야 할까?
  - 조인을 사용할 경우 : Customer 테이블과 Orders 테이블의 고객번호로 조인한 후 필요한 데이터를 추출
  - 부속질의를 사용할 경우 : Customer 테이블에서 박지성 고객의 고객번호를 찾고, 찾은 고객번호를 바탕으로 Orders 테이블에서 확인

부속질의

- 두 테이블을 연관시킬 때 조인을 선택할지 부속질의를 선택할지 여부는 데이터의 형태와 양에 따라 달라짐

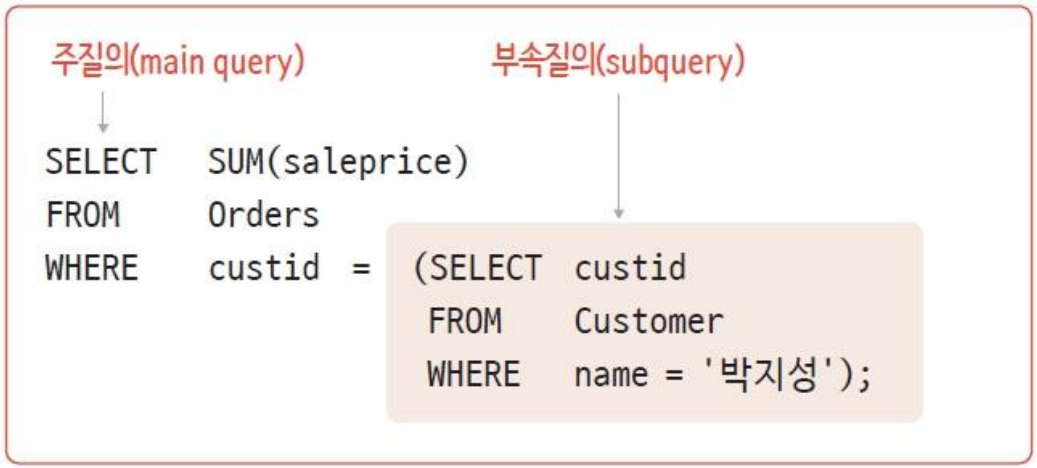


그림 4-2 부속질의

표 4-6 부속질의의 종류

부속질의	실무 용어	영문 및 동의어	설명
WHERE 부속질의	중첩질의	nested subquery, predicate subquery	WHERE 절에서 술어와 같이 사용되며 결과를 한정시킨다. 상관 혹은 비상관 형태다.
SELECT 부속질의	스칼라 부속질의	scalar subquery	SELECT 절에서 사용되며 단일 값을 반환한다.
FROM 부속질의	인라인 뷰	inline view, table subquery	FROM 절에서 결과를 뷰(view) 형태로 반환한다.

1. 중첩질의 - WHERE 부속질의

- WHERE 절은 보통 데이터를 선택하는 조건 혹은 술어와 같이 사용됨
- 중첩질의를 술어 부속질의라고도 부름

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	ALL, SOME (ANY)	다중	단일	가능
존재	EXISTS, NOT EXISTS	다중	다중	필수

1. 중첩질의 - WHERE 부속질의

❖ 비교 연산자

- 비교 연산자 사용 시 부속질의가 반드시 단일 행, 단일 열을 반환해야 하며, 아닐 경우 질의를 처리할 수 없음
- 주질의의 대상 열 값과 부속질의의 결과 값을 비교 연산자에 적용하여 참이면 주질의의 해당 열을 출력함

질의 4-12

평균 주문금액 이하의 주문에 대해서 주문번호와 금액을 나타내시오.

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

orderid	saleprice
1	6000
3	8000
4	6000
9	7000

1. 중첩질의 - WHERE 부속질의

질의 4-13

각 고객의 평균 주문금액보다 큰 금액의 주문 내역에 대해서 주문번호, 고객번호, 금액을 나타내시오.

```
SELECT orderid, custid, saleprice
FROM   Orders od1
WHERE  saleprice> (SELECT AVG(saleprice)
                   FROM   Orders od2
                   WHERE  od1.custid=od2.custid);
```

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

- [질의 4-13]은 상관 부속질의 형태로, 주질의는 행별로 고객번호를 부속질의에 공급
- 부속질의는 그 값으로 고객별 평균을 구하여 주질의의 해당 행과 비교 연산을 수행함

1. 중첩질의 - WHERE 부속질의

❖ IN, NOT IN(집합 연산자)

- IN 연산자는 주질의의 속성값이 부속질의에서 제공한 결과 집합에 있는지 확인하는 역할을 함
- 주질의는 WHERE 절에 사용되는 속성값을 부속질의의 결과 집합과 비교해 하나라도 있으면 참이 됨
- NOT IN 연산자에서는 이와 반대로 값이 존재하지 않으면 참이 됨

질의 4-14

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

1. 중첩질의 - WHERE 부속질의

❖ ALL, SOME/ANY(한정 연산자)

- ALL, SOME(ANY) 연산자의 구문 구조

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

질의 4-15

3번 고객이 주문한 도서의 최고 금액보다 더 비싼 도서를 구입한 주문의 주문번호와 판매금액을 보이시오.

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

orderid	saleprice
2	21000
5	20000

1. 중첩질의 - WHERE 부속질의

❖ EXISTS, NOT EXISTS(존재 연산자)

- 데이터의 존재 여부를 확인하는 연산자
- EXISTS 연산자의 구문 구조

```
WHERE [NOT] EXISTS (부속질의)
```

질의 4-16

EXISTS 연산자를 사용하여 대한민국에 거주하는 고객에게 판매한 도서의 총판매액을 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

2. 스칼라 부속질의 - SELECT 부속질의

- 부속질의의 결과 값을 단일 행, 단일 열의 스칼라 값으로 반환함
- 만약 결과 값이 다중 행이거나 다중 열이라면 DBMS는 그중 어떠한 행과 열을 출력해야 하는지 알 수 없어 에러를 출력함
- 결과가 없는 경우에는 NULL 값을 출력함

주질의(main query)　　부속질의(subquery)

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

그림 4-3 스칼라 부속질의

2. 스칼라 부속질의 - SELECT 부속질의

질의 4-17

마당서점의 고객별 판매액을 나타내시오(고객이름과 고객별 판매액 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
김연경	31000
추신수	33000

2. 스칼라 부속질의 - SELECT 부속질의

- [그림 4-4]는 스칼라 부속질의의 실행 과정을 나타낸 것

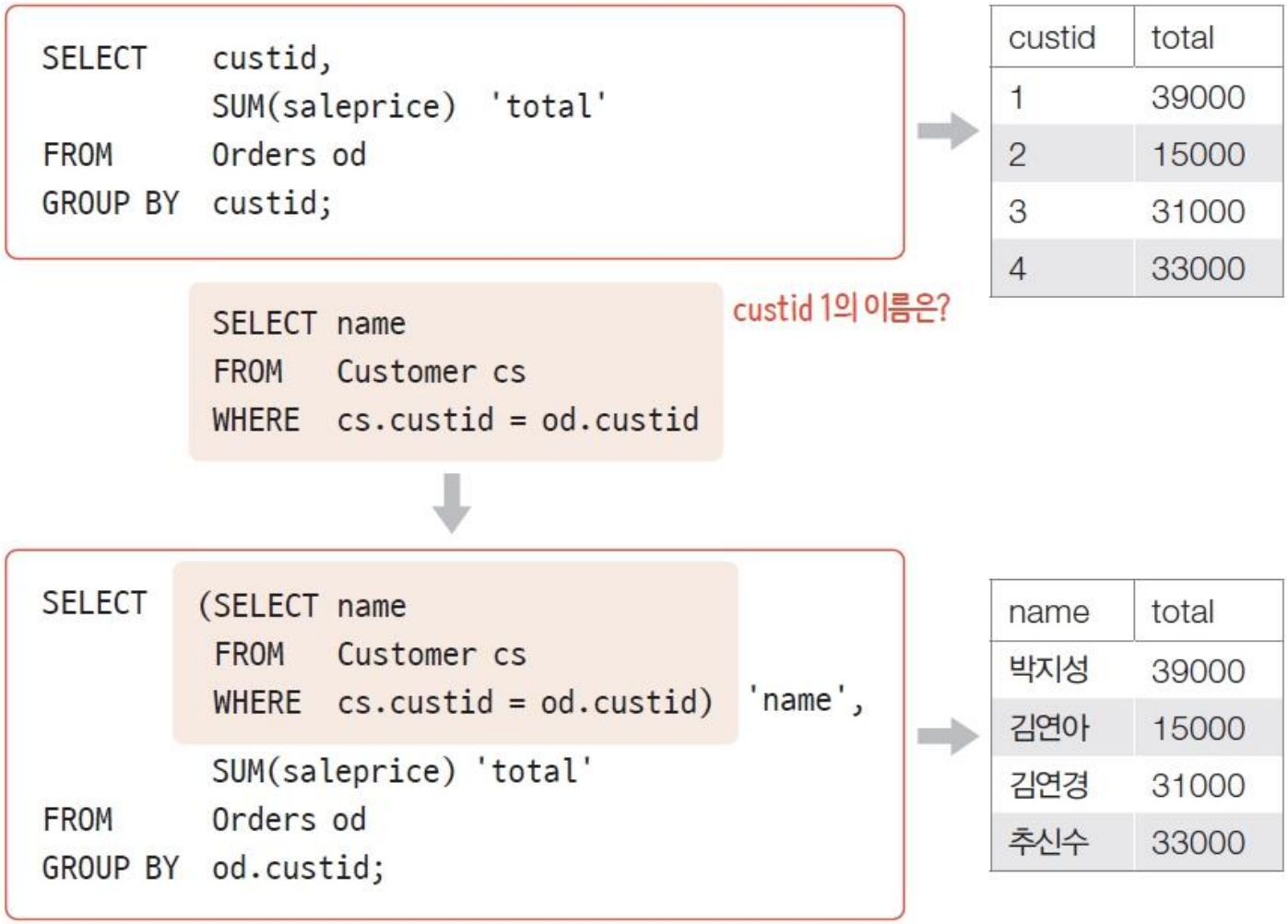


그림 4-4 마당서점의 고객별 판매액

2. 스칼라 부속질의 – SELECT 부속질의

- 스칼라 부속질의는 SELECT 문과 함께 UPDATE 문에서도 사용할 수 있음
- 실습을 위해 Orders 테이블에 다음 명령을 수행하여 새로운 속성인 도서이름 bname을 추가해보기

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

- 새로운 속성에는 NULL 값이 저장되어 있음
- 데이터를 입력하기 위해서는 모든 도서의 (bookid가 1, 2, 3, …, 10인 경우 각각) 도서이름을 수정해야 함
- 예) bookid가 1번인 도서의 이름을 수정

```
UPDATE   Orders
SET      bname ='피겨 교본'
WHERE    bookid=1;
```

2. 스칼라 부속질의 - SELECT 부속질의

질의 4-18

Orders 테이블에 각 주문에 맞는 도서이름을 입력하시오.

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

```
ALTER TABLE Orders ADD bname VARCHAR(40);
UPDATE  Orders
SET     bname = (SELECT  bookname
                 FROM    Book
                 WHERE   Book.bookid=Orders.bookid);
```

orderid	custid	bookid	saleprice	orderdate	bname
1	1	1	6000	2024-07-01	축구의 역사
2	1	3	21000	2024-07-03	축구의 이해
3	2	5	8000	2024-07-03	피겨 교본
4	3	6	6000	2024-07-04	배구 단계별기술
5	4	7	20000	2024-07-05	야구의 추억
6	1	2	12000	2024-07-07	축구 아는 여자
7	4	8	13000	2024-07-07	야구를 부탁해
8	3	10	12000	2024-07-08	Olympic Champions
9	2	10	7000	2024-07-09	Olympic Champions
10	3	8	13000	2024-07-10	야구를 부탁해

3. 인라인 뷰 - FROM 부속질의

질의 4-19

고객번호가 2 이하인 고객의 판매액을 나타내시오(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

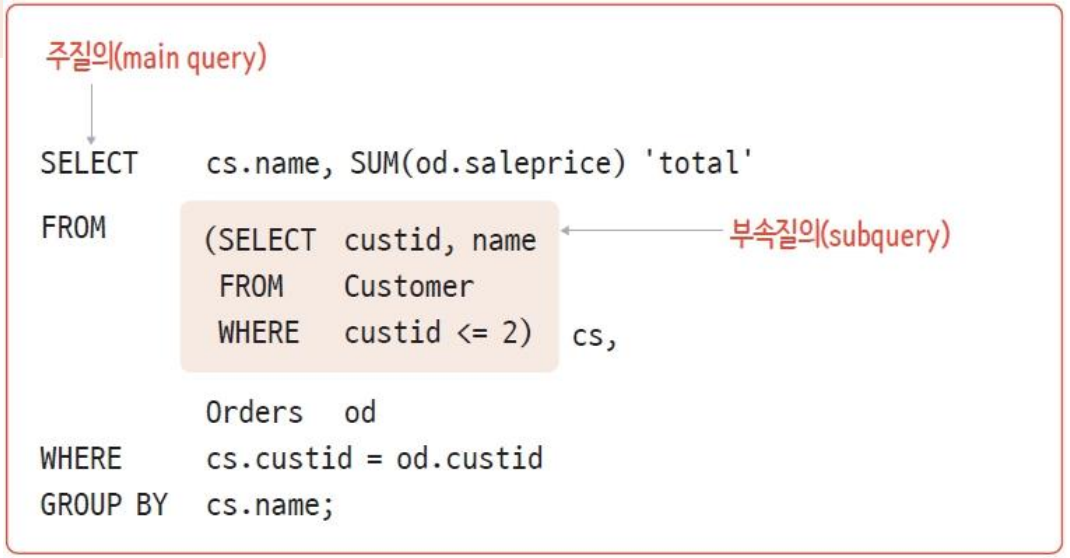


그림 4-5 인라인 뷰

연습문제 (Q4.4)

05 부속질의에 관한 다음 SQL 문을 수행해 보고, 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);
```

03 뷰

1. 뷰의 생성
2. 뷰의 수정
3. 뷰의 삭제

1. 뷰의 생성

- 마당서점을 운영하려면 서점의 매출 정보나 고객의 취향 등을 분석하기 위해 다양한 보고서가 필요
- 마당서점의 프로그래머가 보고서를 만들기 위해 Orders 테이블과 Customer 테이블, Book 테이블을 조인하거나 부속질의를 한다고 가정
- Orders 테이블에 고객이름과 도서이름을 추가하는 방법은 두 가지
  - ① 실제 물리적인 테이블에 열을 추가하여 데이터를 넣는 방법
  - ② [그림 4-6]과 같이 Orders 테이블, Customer 테이블, Book 테이블을 조인한 후 가상의 테이블인 뷰 Vorders를 생성하는 방법

뷰 Vorders

orderid	custid	name	bookid	bookname	saleprice	orderdate
1	1	박지성	1	축구의 역사	6000	2024-07-01
2	1	박지성	3	축구의 이해	21000	2024-07-03
3	2	김연아	5	피겨 교본	8000	2024-07-03
4	3	김연경	6	배구 단계별기술	6000	2024-07-04

1. 뷰의 생성

뷰 생성문

```
CREATE VIEW  Vorders
AS SELECT    orderid, O.custid, name, O.bookid, bookname, saleprice, orderdate
   FROM      Customer C, Orders O, Book B
   WHERE     C.custid=O.custid and B.bookid=O.bookid;
```

베이스 릴레이션 Customer, Orders, Book

O

orderid	custid	bookid	saleprice	orderdate
1	1	1	6000	2024-07-01
2	1	3	21000	2024-07-03
3	2	5	8000	2024-07-03
4	3	6	6000	2024-07-04
5	4	7	20000	2024-07-05

C

custid	name	address	phone
1	박지성	영국 맨체스터	000-5000-0001
2	김연아	대한민국 서울	000-6000-0001
3	김연경	대한민국 경기도	000-7000-0001

B

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구 아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000

그림 4-6 뷰

1. 뷰의 생성

❖ 뷰

- 실제 SQL 문에서 테이블과 동일하게 사용할 수 있는 데이터베이스 개체
- 뷰의 정의를 뷰의 생성이라고도 함
- 뷰를 생성하는 문법

```
CREATE VIEW 뷰이름 [(열이름 [,... n])]
AS SELECT 문
```

- Book 테이블에서 '축구'라는 문구가 포함된 자료만 보여주는 뷰를 만들어보기
  - 뷰에 사용할 SELECT 문

```
SELECT   *
FROM     Book
WHERE    bookname LIKE '%축구%';
```

  - 위 SELECT 문을 이용해 뷰 정의문을 작성하면 다음과 같음

```
CREATE VIEW  vw_Book
AS SELECT    *
   FROM      Book
   WHERE     bookname LIKE '%축구%';
```

1. 뷰의 생성

질의 4-20

주소에 '대한민국'을 포함하는 고객들로 구성된 뷰를 만들고 조회하시오. 뷰의 이름은 vw_Customer로 설정하시오.

```
CREATE VIEW  vw_Customer
AS SELECT    *
   FROM      Customer
   WHERE     address LIKE '%대한민국%';
```

뷰를 사용한 SELECT 문

```
SELECT *
FROM   vw_Customer;
```

custid	name	address	phone
2	김연아	대한민국 서울	000-6000-0001
3	김연경	대한민국 경기도	000-7000-0001
5	박세리	대한민국 대전	NULL

1. 뷰의 생성

질의 4-21

Orders 테이블에서 고객이름과 도서이름을 바로 확인할 수 있는 뷰를 생성한 후, '김연아' 고객이 구입한 도서의 주문번호, 도서이름, 주문액을 나타내시오.

```
CREATE VIEW   vw_Orders (orderid, custid, name, bookid, bookname,
              saleprice, orderdate)
AS SELECT     od.orderid, od.custid, cs.name,
              od.bookid, bk.bookname, od.saleprice, od.orderdate
    FROM      Orders od, Customer cs, Book bk
    WHERE     od.custid=cs.custid AND od.bookid=bk.bookid;
```

뷰를 사용한 SELECT 문

```
SELECT  orderid, bookname, saleprice
FROM    vw_Orders
WHERE   name ='김연아';
```

orderid	bookname	saleprice
3	피겨 교본	8000
9	Olympic Champions	7000

1. 뷰의 생성

❖ 뷰의 장점

- 편리성(및 재사용성) : 자주 사용되는 복잡한 질의를 뷰로 미리 정의해 놓을 수 있음

  → 복잡한 질의를 간단히 작성

- 보안성 : 사용자별로 필요한 데이터만 선별하여 보여줄 수 있고, 중요한 질의의 경우 질의 내용을 암호화할 수 있음

  → 개인정보(주민번호)나 급여, 건강 같은 민감한 정보를 제외한 테이블을 만들어 사용

- 독립성 : 원본 테이블의 구조가 변해도 응용에 영향을 주지 않도록 함

  → 논리적 데이터 독립성 제공 방법

❖ 뷰의 특징

- 원본 데이터 값에 따라 같이 변함
- 독립적인 인덱스 생성이 어려움
- 삽입, 삭제, 갱신 연산에 많은 제약이 따름

2. 뷰의 수정

❖ 뷰를 수정하는 문장

- CREATE VIEW 문에 OR REPLACE 명령을 더하여 작성

```
CREATE OR REPLACE VIEW 뷰이름 [(열이름 [,... n])]
AS SELECT 문
```

질의 4-22

[질의 4-20]에서 생성한 뷰 vw_Customer는 주소가 대한민국인 고객을 보여준다. 이 뷰를 영국을 주소로 가진 고객으로 변경하시오. phone 속성은 필요 없으므로 포함하지 마시오.

```
CREATE OR REPLACE VIEW vw_Customer (custid, name, address)
AS SELECT     custid, name, address
   FROM       Customer
   WHERE      address LIKE '%영국%';
```

뷰를 사용한 SELECT 문

```
SELECT  *
FROM    vw_Customer;
```

custid	name	address
1	박지성	영국 맨체스터

3. 뷰의 삭제

- 뷰가 필요 없어졌다면 DROP 문을 사용하여 뷰를 삭제함

```
DROP VIEW 뷰이름 [,... n];
```

질의 4-23

앞서 생성한 뷰 vw_Customer를 삭제하시오.

```
DROP VIEW vw_Customer;
```

```
SELECT  *
FROM    vw_Customer;
```

Message
Error Code: 1146. Table 'madangdb.vw_customer' doesn't exist

연습문제 (Q4.5)

12 마당서점 데이터베이스를 이용하여 다음에 해당하는 뷰를 작성하시오.

(1) 판매가격이 20,000원 이상인 도서의 도서번호, 도서이름, 고객이름, 출판사, 판매가격을 보여주는 highorders 뷰를 생성하시오.

(2) 생성한 뷰를 이용하여 판매된 도서의 이름과 고객의 이름을 출력하는 SQL 문을 작성하시오.

(3) highorders 뷰를 변경하고자 한다. 판매가격 속성을 삭제하는 명령을 수행하시오. 삭제 후 (2)번 SQL 문을 다시 수행하시오.

04 인덱스

1. 데이터베이스의 물리적 저장
2. 인덱스와 B-tree
3. MySQL 인덱스
4. 인덱스의 생성
5. 인덱스의 재구성과 삭제

1. 데이터베이스의 물리적 저장

- 우리가 테이블을 생성하고 테이블에 데이터를 저장할 때 DBMS는 데이터를 어디에 어떻게 저장할까?
- DBMS 역시 데이터를 DBMS만의 고유한 방식으로 저장하여 관리함

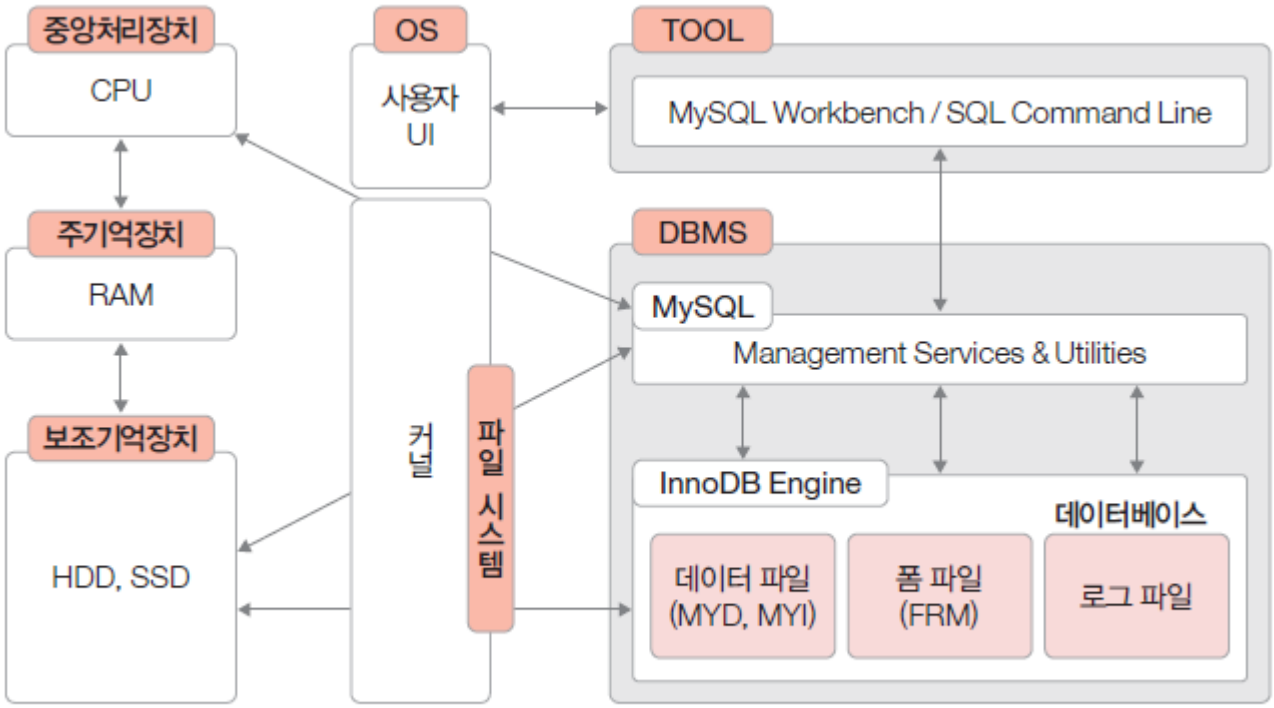
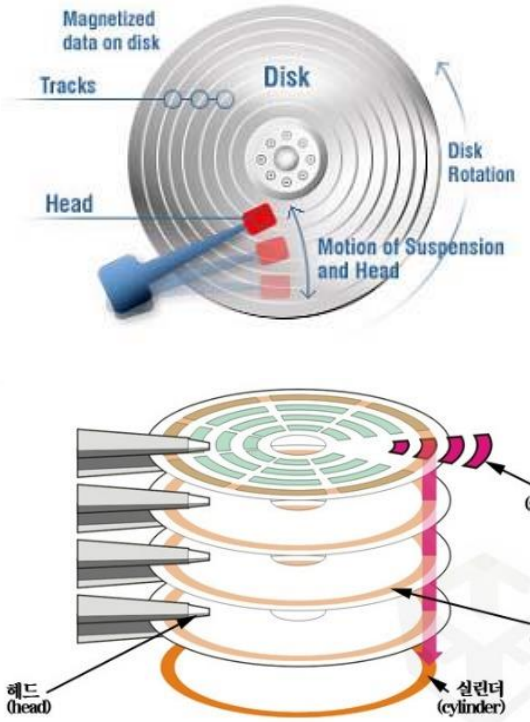


그림 4-7 DBMS와 데이터 파일

1. 데이터베이스의 물리적 저장

- 실제 데이터가 저장되는 곳이 보조기억장치
- 보조기억장치로는 일반적으로 하드디스크, SSD, USB 메모리 등이 있는데, 그중 하드디스크를 가장 많이 사용

그림 4-8 하드디스크의 구조



1. 데이터베이스의 물리적 저장

❖ 액세스 시간

- 디스크의 입출력 시간
- 액세스 시간은 데이터의 저장 및 읽기에 많은 영향을 끼침

- 액세스 시간은 다음과 같은 식으로 표현할 수 있음
  액세스 시간 = 탐색시간(액세스 헤드를 트랙에 이동시키는 시간)+회전 지연시간(섹터가 액세스 헤드에 접근하는 시간)+데이터 전송시간(데이터를 주기억장치로 읽어오는 시간)

- DBMS가 하드디스크에 데이터를 저장하고 읽어올 때, 속도 문제가 발생함
  - 컴퓨터 시스템에서 처리되는 연산 속도는 빠르지만, 디스크의 액세스 속도는 상대적으로 느리기 때문
  - 이러한 속도 문제를 줄이기 위해 DBMS는 주기억장치에 사용하는 공간 중 일부를 버퍼 풀로 만들어 사용

1. 데이터베이스의 물리적 저장

- 데이터 검색 시 DBMS는 버퍼 풀에 저장된 데이터를 우선 읽어들인 후 작업을 진행

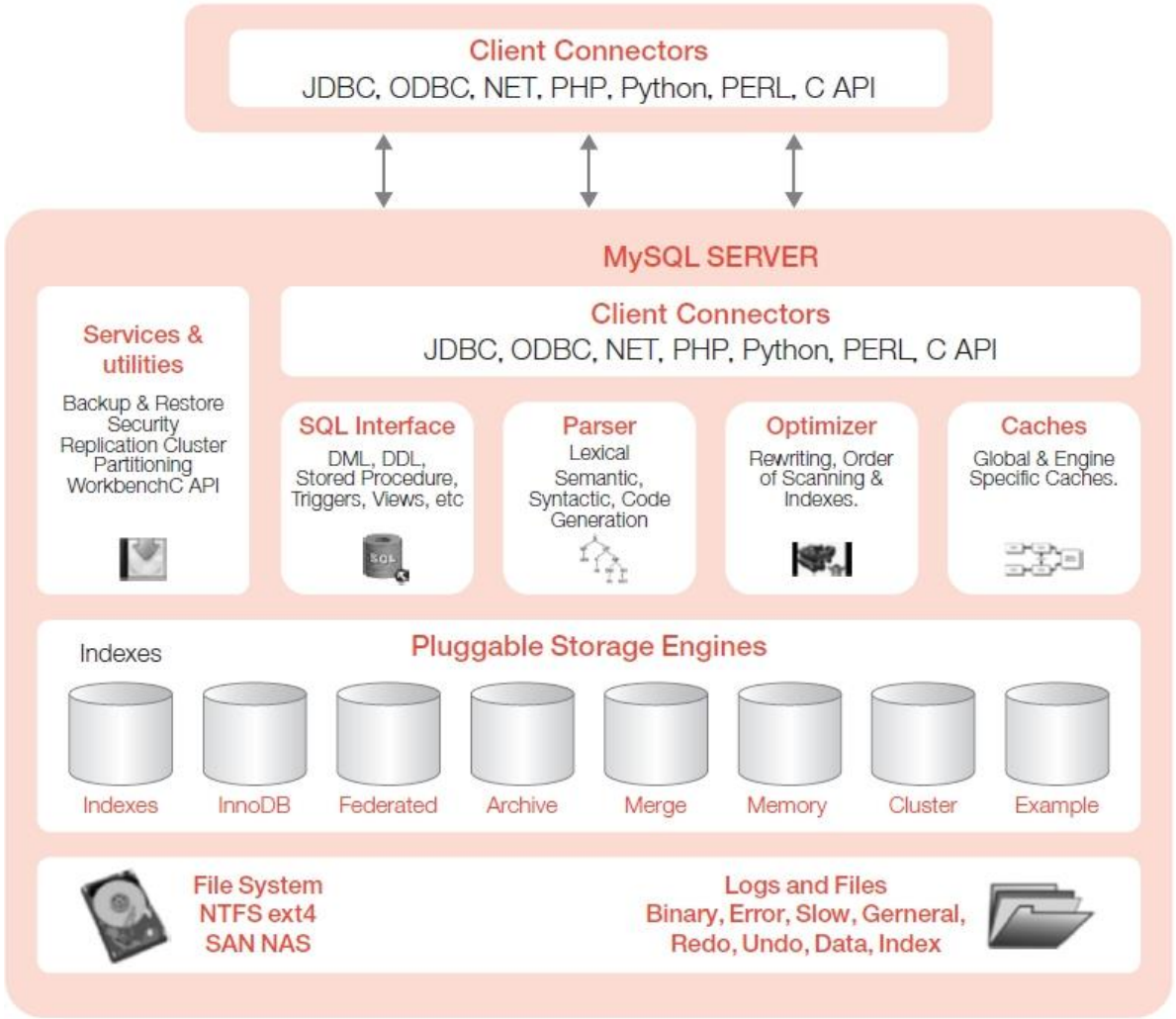


그림 4-9 MySQL DBMS 구조

1. 데이터베이스의 물리적 저장

- DBMS는 데이터베이스별로 하나 이상의 데이터 파일을 생성함
- 테이블은 생성 시 정의된 내용에 따라 논리적으로 구분 지어져 각각의 데이터 파일에 저장됨

- MySQL의 저장장치 엔진 Engines은 플러그인 방식으로 선택할 수 있으며 InnoDB 엔진이 기본으로 설치되어 있음

표 4-8 MySQL InnoDB 엔진 데이터베이스의 파일

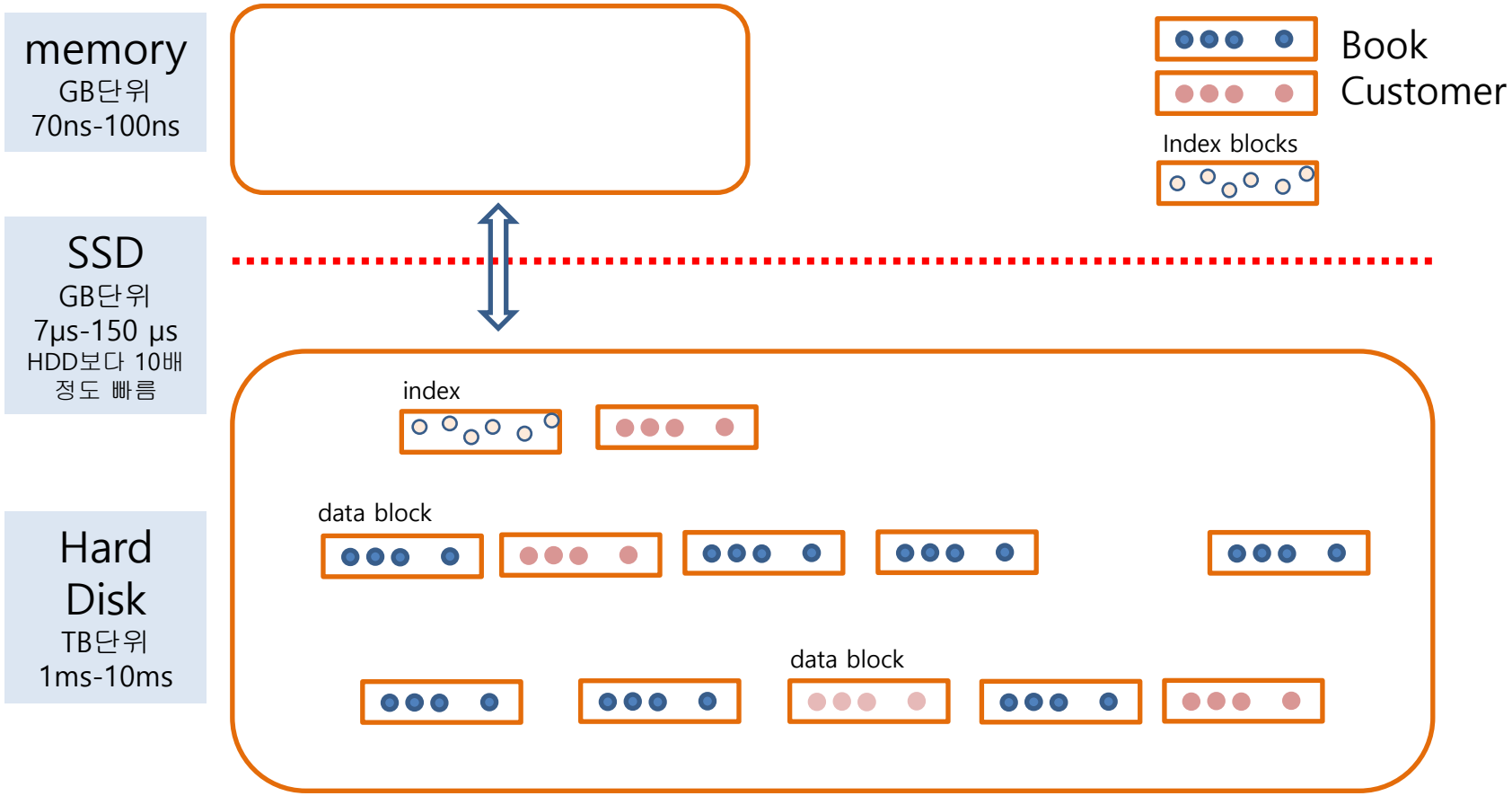
파일	설명
데이터 파일(ibdata)	• 사용자 데이터와 개체를 저장함 • 테이블과 인덱스로 구성함 • 확장자는 *.ibd
폼 파일(frm File)	• 테이블에 대한 각종 정보와 테이블을 구성하는 필드, 데이터 타입에 대한 정보를 저장함 • 데이터베이스 구조 등의 변경 사항이 있을 때 자동으로 업데이트됨

2. 인덱스와 B-tree

- **인덱스의 필요성(저장장치간의 속도차이가 있으므로 필요한 데이터만 접근해야함)**

  질의 검색시 data block(여러 개의 record를 저장하는 저장단위, 2KB 4KB ...) 읽는 횟수를 최소화 필요

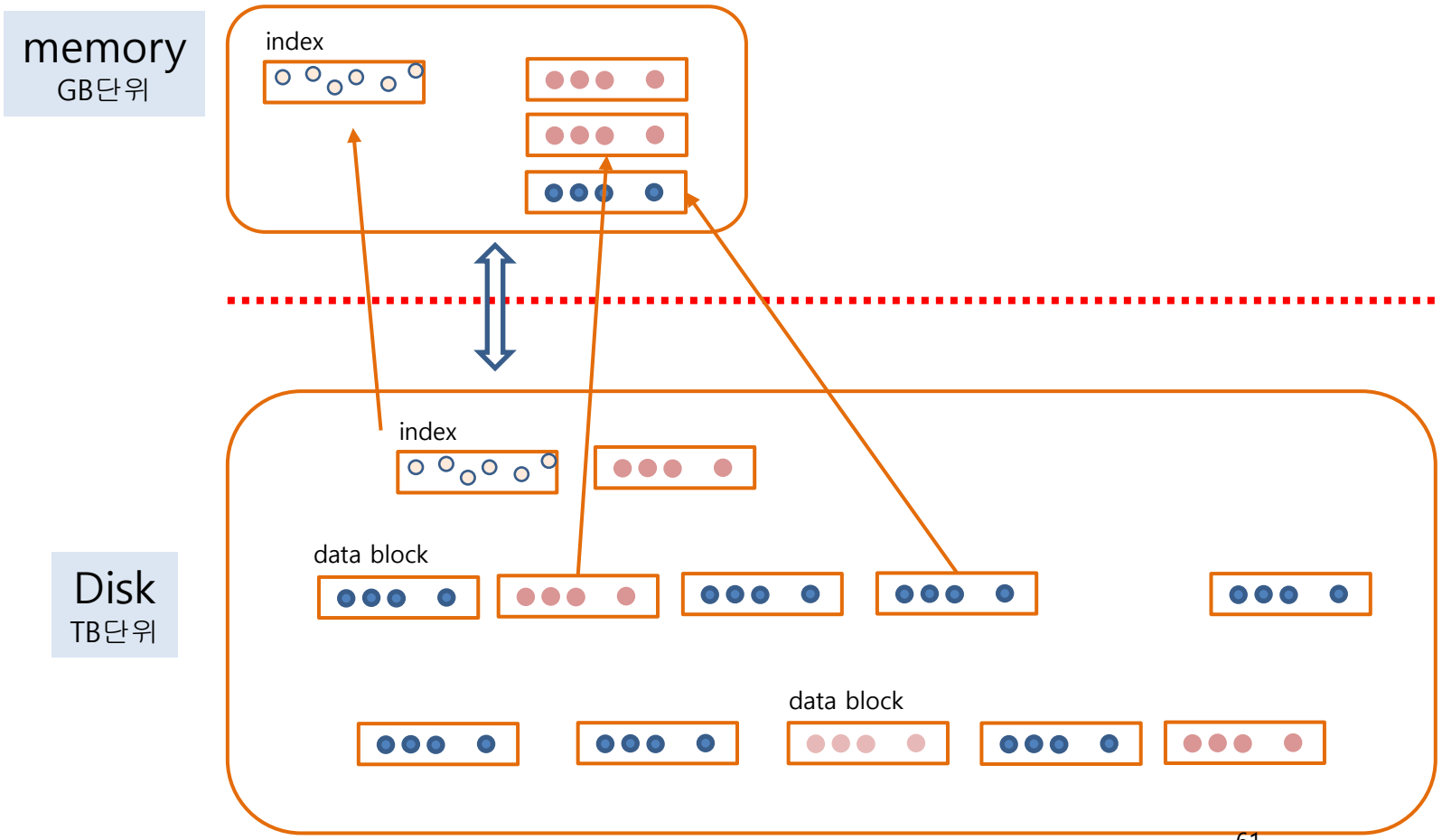
  disk에 있는 데이터는 memory에 있는 데이터에 비하여 읽어들이는 속도가 10000배 정도 소요된다.



㈜ https://www.formulusblack.com/blog/compute-performance-distance-of-data-as-a-measure-of-latency/

2. 인덱스와 B-tree

- 인덱스의 필요성 memory와 disk



2. 인덱스와 B-tree

❖ 인덱스(색인)

- 데이터를 쉽고 빠르게 찾을 수 있도록 만든 데이터 구조
- 일반적인 RDBMS의 인덱스는 대부분 B-tree 구조로 되어 있음

❖ B-tree

- 데이터의 검색 시간을 단축하기 위한 자료구조

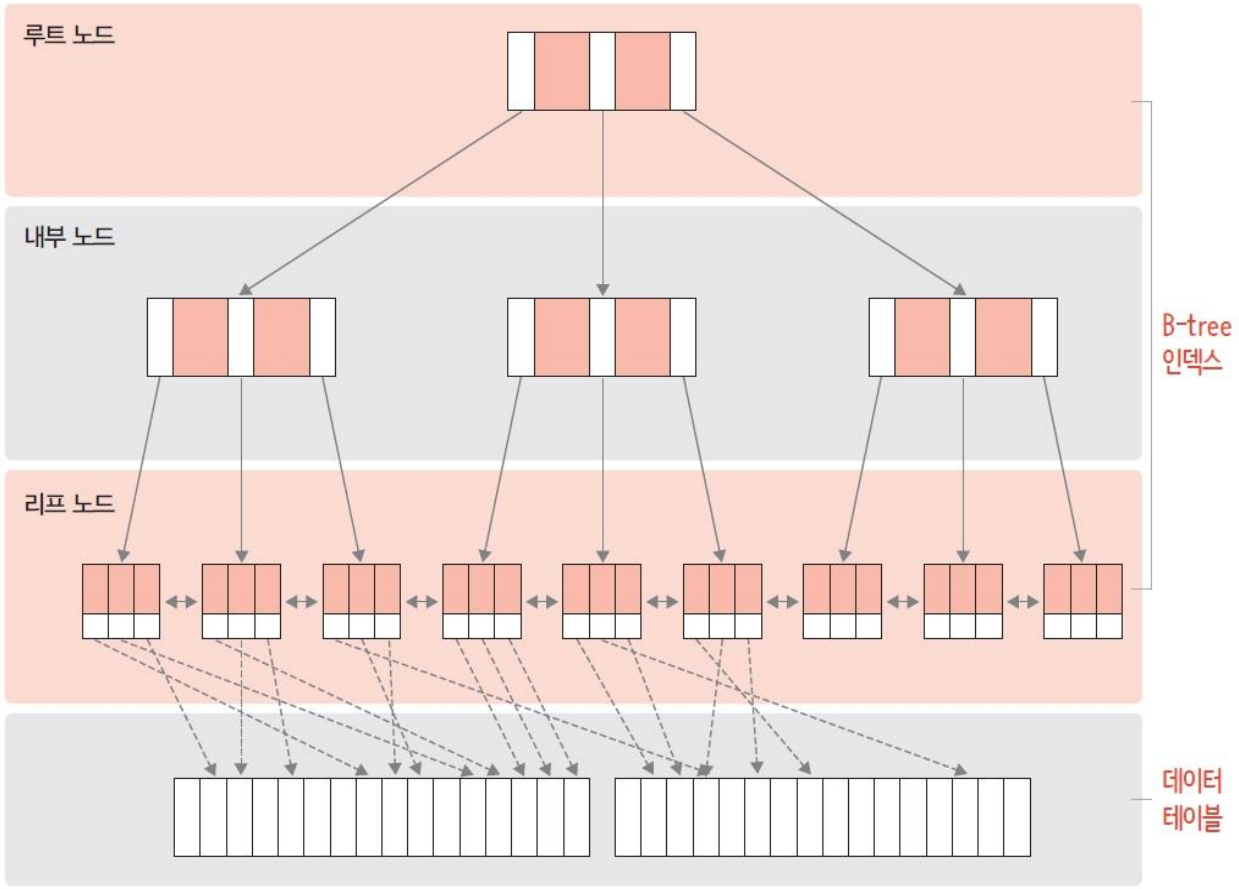


그림 4-10 B-tree의 구조

2. 인덱스와 B-tree

- B-tree의 각 노드는 키 값과 포인터를 가짐
- [그림 4-11]과 같이 최대 세 개의 자식을 가지는 B-tree에서 3이라는 값을 찾는다고 가정

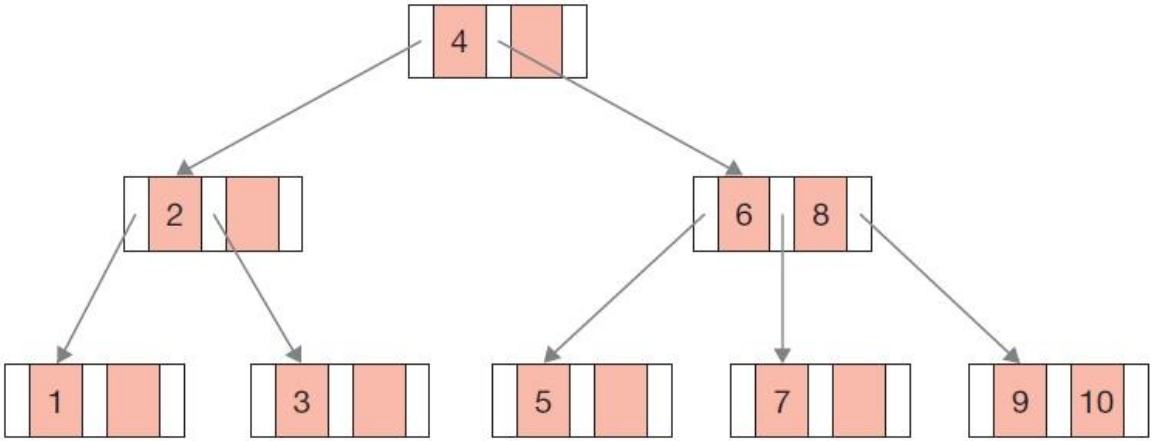


그림 4-11 B-tree에서의 검색 예

- B-tree에서 검색은 루트 노드에서부터 값을 비교하여 중간 단계인 내부 노드에서 해당 노드를 찾고, 이런 단계를 거쳐 최종적으로 마지막 레벨인 리프 노드에 도달함
- B-tree는 데이터를 검색할 때 특유의 트리 구조를 이용하기 때문에 한 번 검색할 때마다 검색 대상이 1/m(m은 자식의 개수)로 줄어 접근 시간이 적게 걸림
- 100만 개의 투플을 가진 데이터도 디스크 블록을 서너 번 읽으면 찾을 수 있음

2. 인덱스와 B-tree

❖ 인덱스의 특징

- 인덱스는 테이블에서 한 개 이상의 속성을 이용하여 생성함
- 빠른 검색과 함께 효율적인 레코드 접근이 가능함
- 순서대로 정렬된 속성과 데이터의 위치만 보유하므로 테이블보다 작은 공간을 차지함
- 저장된 값들은 테이블의 부분집합이 됨
- 일반적으로 B-tree 형태의 구조를 가짐
- 데이터에서 수정, 삭제 등의 변경이 발생하면 인덱스를 재구성해야 함

3. MySQL 인덱스

- MySQL의 인덱스는 클러스터 인덱스와 보조 인덱스로 나뉨

❖ 클러스터 인덱스

- Book 테이블에 클러스터 인덱스를 생성한 경우

그림 4-12 클러스터 인덱스

3. MySQL 인덱스

❖ 보조 인덱스

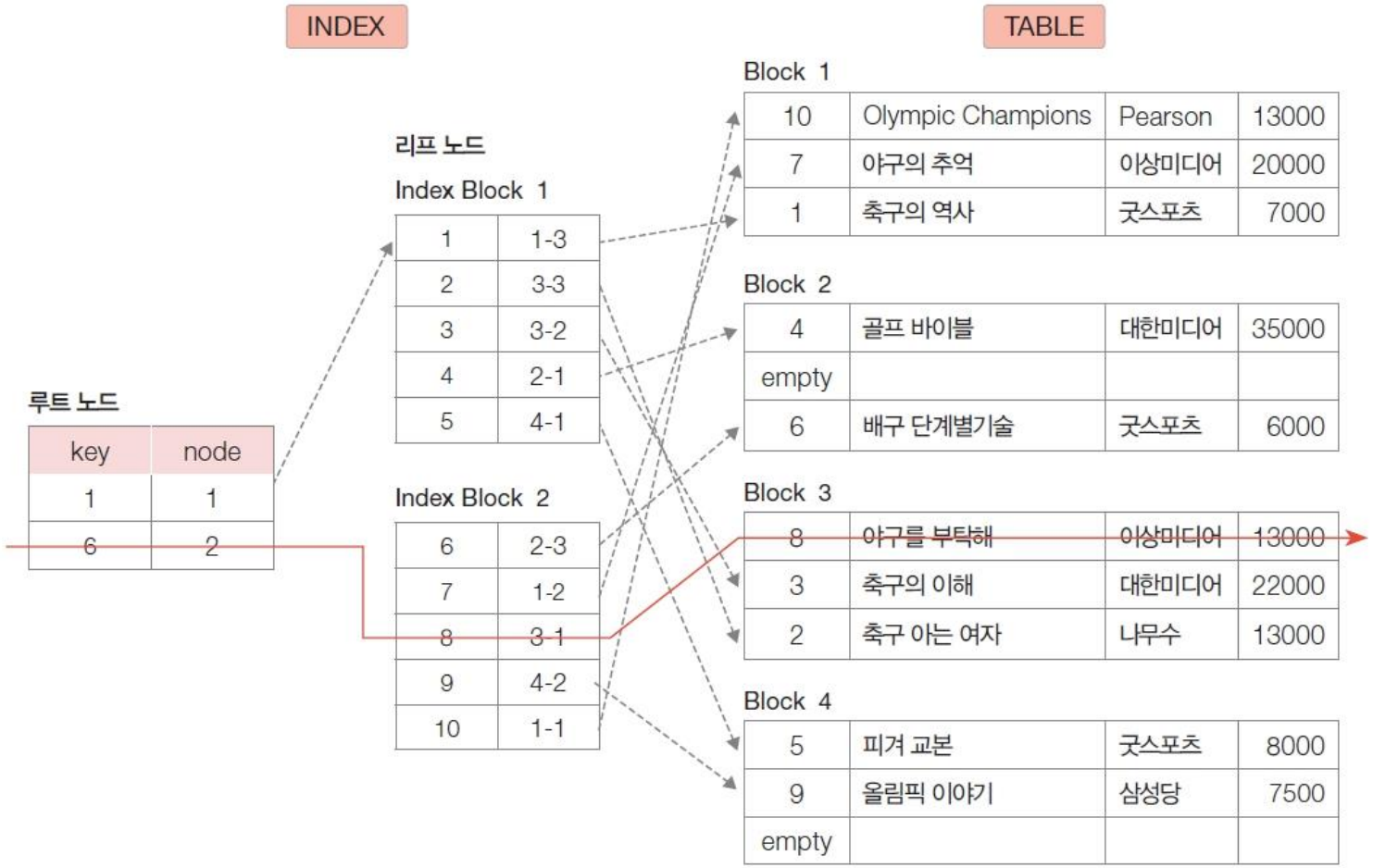


그림 4-13 B-tree 인덱스의 예

3. MySQL 인덱스

❖ MySQL 인덱스

- 예) Book 테이블에서 bookid를 클러스터 인덱스로, bookname을 보조 인덱스로 사용하여 bookid와 bookname 모두 빠른 검색을 필요로 하는 경우
  - [그림 4-14]는 bookname으로 '야구를 부탁해' 책을 찾는 과정을 나타냄

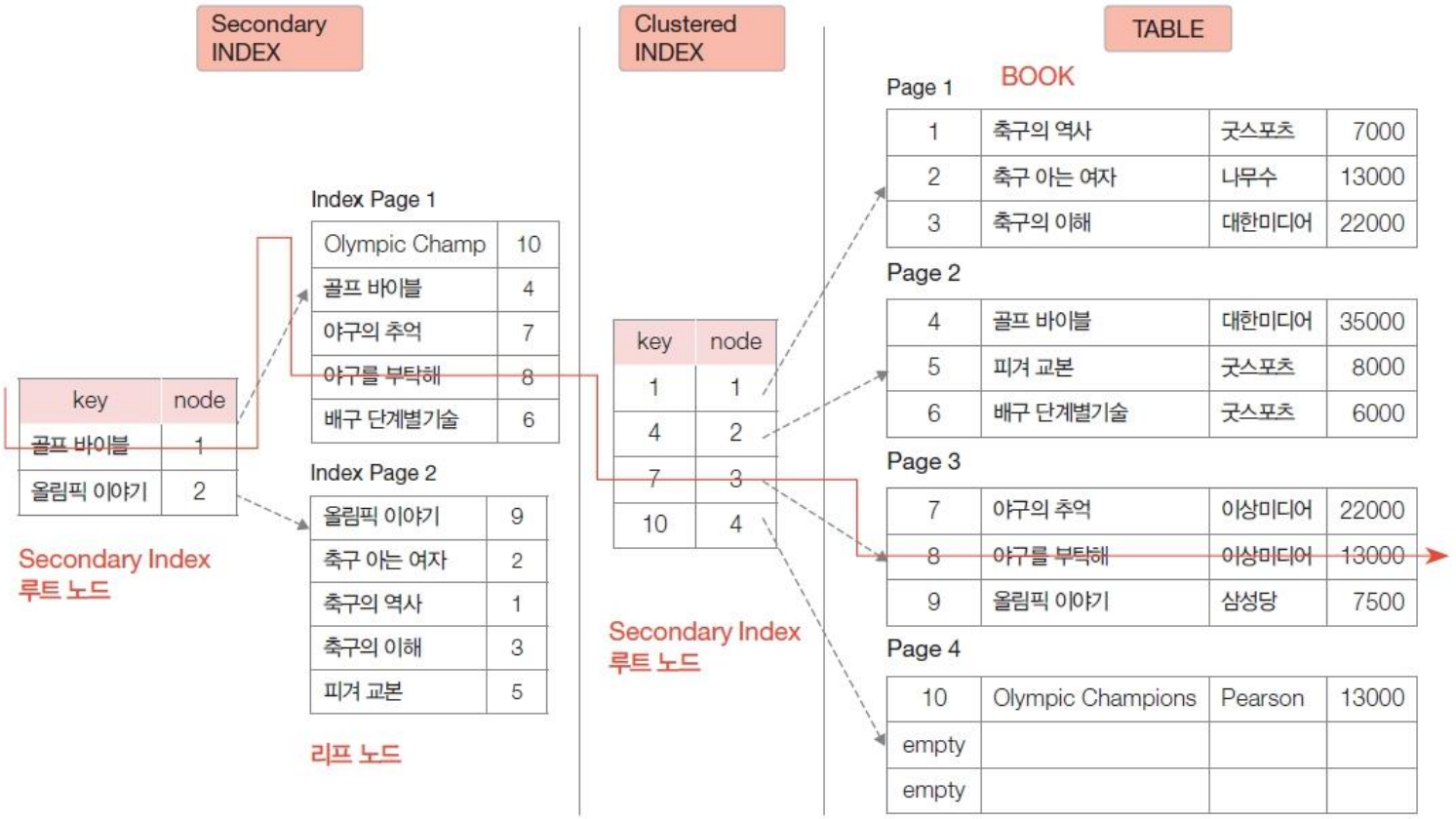


그림 4-14 클러스터 인덱스와 보조 인덱스를 동시에 사용하는 검색

3. MySQL 인덱스

❖ MySQL의 인덱스 정리

표 4-9 MySQL 인덱스의 종류

인덱스 명칭	설명 / 생성 예
클러스터 인덱스	• 기본적인 인덱스로 테이블 생성 시 기본키를 지정하면 기본키에 대하여 클러스터 인덱스를 생성한다. • 기본키를 지정하지 않으면 먼저 나오는 UNIQUE 속성에 대하여 클러스터 인덱스를 생성한다. • 기본키나 UNIQUE 속성이 없는 테이블은 MySQL이 자체 생성한 행번호row ID를 이용하여 클러스터 인덱스를 생성한다.
보조 인덱스	• 클러스터 인덱스가 아닌 모든 인덱스는 보조 인덱스이며 보조 인덱스의 각 레코드는 보조 인덱스 속성과 기본키 속성값을 갖고 있다. • 보조 인덱스를 검색하여 기본키 속성값을 찾은 다음 클러스터 인덱스로 가서 해당 레코드를 찾는다.

4. 인덱스의 생성

❖ 인덱스를 생성하기 전 고려사항

- 인덱스는 WHERE 절에 자주 사용되는 속성이어야 함
- 인덱스는 조인에 자주 사용되는 속성이어야 함
- 단일 테이블에 인덱스가 많으면 속도가 느려질 수 있음(테이블당 4~5개 정도 권장)
- 속성이 가공되는 경우에는 사용하지 않음
- 속성의 선택도가 낮을 때 유리함(속성의 모든 값이 다른 경우)

- 인덱스를 생성할 때는 CREATE INDEX 문을 이용하며, 문법은 다음과 같음

```
CREATE [UNIQUE] INDEX [인덱스이름]
ON 테이블이름 (컬럼 [ASC | DESC] [{, 컬럼 [ASC | DESC]} ...])[;]
```

  - [UNIQUE]는 테이블의 속성값에 대하여 중복이 없는 유일한 인덱스를 생성하는 것을 말함
  - [ASC | DESC]는 컬럼 값의 정렬 방식을 의미함

4. 인덱스의 생성

질의 4-24

Book 테이블의 bookname 열을 대상으로 인덱스 ix_Book을 생성하시오.

```
CREATE INDEX ix_Book ON Book(bookname);
```

```
0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
```

질의 4-25

Book 테이블의 publisher, price 열을 대상으로 인덱스 ix_Book2를 생성하시오.

```
CREATE INDEX ix_Book2 ON Book(publisher, price);
```

```
0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
```

4. 인덱스의 생성

- 생성된 인덱스는 SHOW INDEX 명령어로 확인 가능

```
SHOW INDEX FROM Book;
```

Table	Non_unique	Key_name	Seg_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
book	0	PRIMARY	1	bookid	A	10	NULL	NULL		BTREE
book	1	ix_Book	1	bookname	A	10	NULL	NULL	YES	BTREE
book	1	ix_Book2	1	publisher	A	6	NULL	NULL	YES	BTREE
book	1	ix_Book2	2	price	A	10	NULL	NULL	YES	BTREE

- MySQL이 생성된 인덱스를 활용하여 SQL 문을 처리하는지 확인하려면 MySQL Workbench에서 [Query] - [Explain Current Statement]를 누르면 됨

```
SELECT   *
FROM     Book
WHERE    publisher='대한미디어' AND price >= 30000;
```

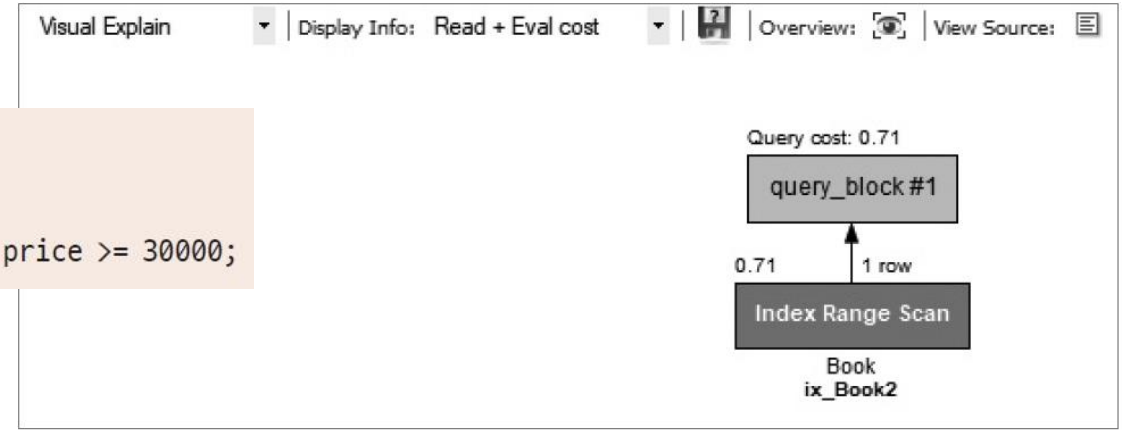


그림 4-15 실행 계획을 통한 인덱스 사용 확인

5. 인덱스의 재구성과 삭제

- 인덱스의 재구성은 ANALYZE TABLE 명령을 사용하여 수행함
- B-tree 인덱스는 데이터의 수정·삭제·삽입이 잦으면 노드의 갱신이 주기적으로 일어나 단편화 현상이 나타남
- 이럴 경우 ANALYZE 문법을 통해 인덱스를 다시 생성해줌

```
ANALYZE TABLE 테이블이름;
```

질의 4-26

Book 테이블의 인덱스를 최적화하시오.

```
ANALYZE TABLE Book;
```

Table	Op	Msg_type	Msg_text
madangdb.book	analyze	status	OK

5. 인덱스의 재구성과 삭제

- 하나의 테이블에 인덱스가 많으면 데이터베이스 성능에 좋지 않은 영향을 미침
- 사용하지 않는 인덱스는 삭제해야 함
- 인덱스의 삭제는 DROP INDEX 명령을 사용하여 수행

질의 4-27

인덱스 ix_Book을 삭제하시오.

```
DROP INDEX ix_Book ON Book;
```

```
0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
```

연습문제 (Q4.6)

[추가 연습 문제 – 실행계획 p261참조]

[마당서점 데이터베이스 인덱스] 마당서점 데이터베이스에서 다음 SQL 문을 수행하고 데이터베이스가 인덱스를 사용하는 과정을 확인하시오.

(1) 다음 SQL 문을 수행해본다.

```
SELECT name, address FROM Customer WHERE name LIKE '박세리';
```

(2) 실행 계획을 살펴본다. 실행 계획은 Workbench에서 [Query] → [Explain Current Statement]를 선택하면 표시된다.

(3) Customer 테이블에 name으로 인덱스를 생성하시오. 생성 후 (1)번의 SQL 문을 다시 수행하고 실행 계획을 살펴보시오.

(4) 같은 질의에 대한 두 가지 실행 계획을 비교해보시오.

(5) (3)번에서 생성한 인덱스를 삭제하시오.

요약

1. 내장 함수
2. 부속질의
3. 뷰
4. 인덱스
5. B-tree
6. MySQL 인덱스의 종류