04

SQL 고급

MySQL로 배우는 데이터베이스 SQL 프로그래밍

목차

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.

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

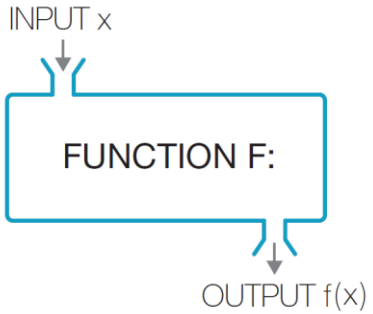


그림 4-1 함수의 원리

- 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_COUNR, 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     Dual;
```

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'
	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('ABCDEFG', 3, 4) => 'CDEF'
	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
	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	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	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 : 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	문자수	바이트수
축구의 역사	6	16
피겨 교본	5	13
역도 단계별기술	8	22

- **SUBSTR : 지정한 길이만큼의 문자열을 반환하는 함수**

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 날자형(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

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 웹페이지 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

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 대치하여 연산하거나 다른 값으로 출력**

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

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

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

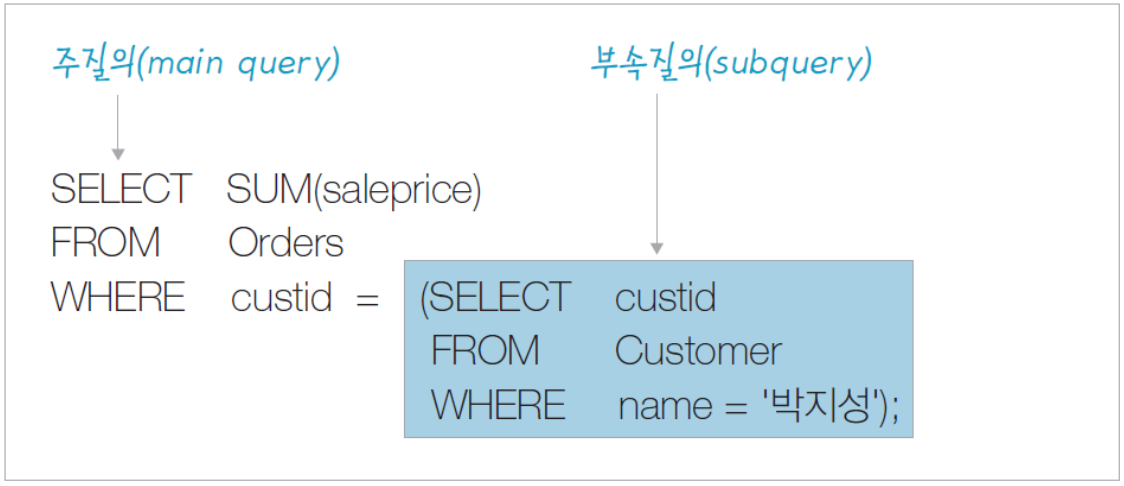


그림 4-2 **부속질의**

부속질의

표 4-6 부속질의의 종류

명칭	위치	영문 및 동의어	설명
스칼라 부속질의	SELECT 절	scalar subquery	SELECT 절에서 사용되며 단일 값을 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
- 원칙적으로 스칼라 값이 들어갈 수 있는 모든 곳에 사용 가능하며, 일반적으로 SELECT 문과 UPDATE SET 절에 사용됨
- 주질의와 부속질의와의 관계는 상관/비상관 모두 가능함

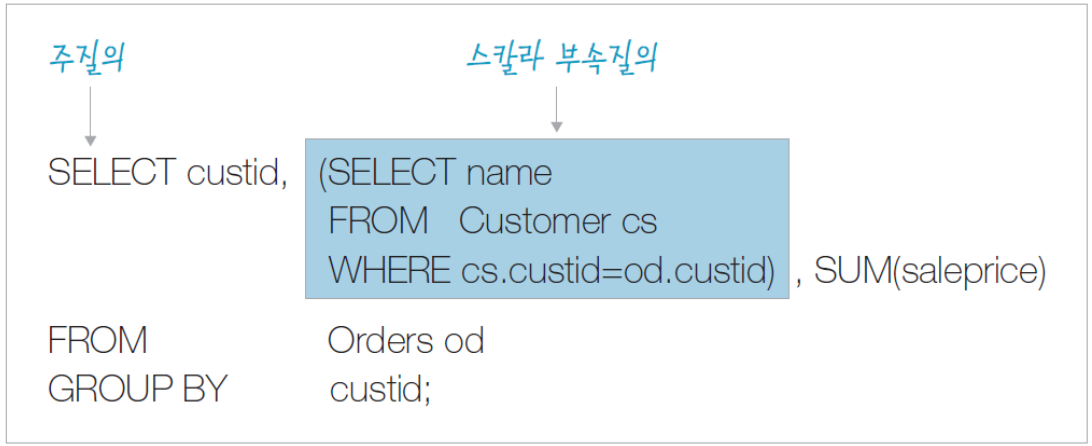


그림 4-3 스칼라 부속질의

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

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	축구아는 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

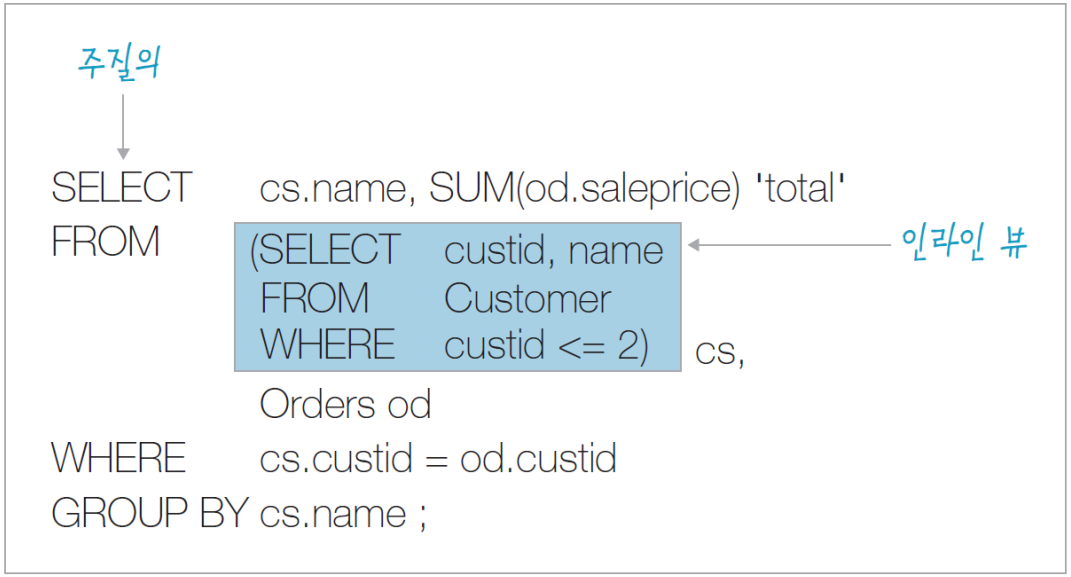


그림 4-5 인라인 뷰

3. 중첩질의 – WHERE 부속질의

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

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

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 EXIST의 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

03. 뷰

1. 뷰의 생성
2. 뷰의 수정
3. 뷰의 삭제

뷰

- **뷰(view) : 하나 이상의 테이블을 합하여 만든 가상의 테이블**

- **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** : 미리 정의된 뷰를 일반 테이블처럼 사용할 수 있기 때문에 편리하고, 사용자가 필요한 정보만 요구에 맞게 가공하여 뷰로 만들어 쓸 수 있음

    → 원본 테이블의 구조가 변해도 응용에 영향을 주지 않도록 하는 논리적 독립성 제공

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

뷰

뷰 Vorders

orderid	custid	name	bookid	bookname	saleprice	orderdate
1	1	박지성	1	축구의 역사	6000	2014-07-01
2	1	박지성	3	축구의 이해	21000	2014-07-03
3	2	김연아	5	피겨 교본	8000	2014-07-03
4	3	장미란	6	역도 단계별기술	6000	2014-07-04

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	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

C

custid	name	address	phone
1	박지성	영국 맨체스타	000-5000-0001
2	김연아	대한민국 서울	000-6000-0001
3	장미란	대한민국 강원도	000-7000-0001

B

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

그림 4-6 뷰

1. 뷰의 생성

- **기본 문법**

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

- **Book 테이블에서 '축구'라는 문구가 포함된 자료만 보여주는 뷰**

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

- **위 SELECT 문을 이용해 작성한 뷰 정의문**

```
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%';
```

<결과 확인>

```
SELECT      *
FROM        vw_Customer;
```

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

1. 뷰의 생성

질의 4-21 Orders 테이블에 고객이름과 도서이름을 바로 확인할 수 있는 뷰를 생성한 후, '김연아' 고객이 구입한 도서의 주문번호, 도서이름, 주문액을 보이시오.

```
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;
```

<결과 확인>

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

orderid	bookname	saleprice
3	피겨 교본	8000
9	Olympic Champions	7000

2. 뷰의 수정

❖ 기본 문법

```
CREATE OF 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 포함시키지 마시오.

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

<결과 확인>

```
SELECT      *
FROM        vw_Customer;
```

custid	name	address
1	박지성	영국 맨체스타

3. 뷰의 삭제

❖ 기본 문법

```
DROP VIEW 뷰이름 [ ,...n ];
```

질의 4-23　앞서 생성한 뷰 vw_Customer를 삭제하시오.

```
DROP VIEW vw_Customer;
```

<결과 확인>

```
SELECT    *
FROM      vw_Customer;
```

Message	Duration / Fetch
Error Code: 1146. Table 'madang.vw_customer' doesn't exist	0.000 sec

04. 인덱스

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

1. 데이터베이스의 물리적 저장

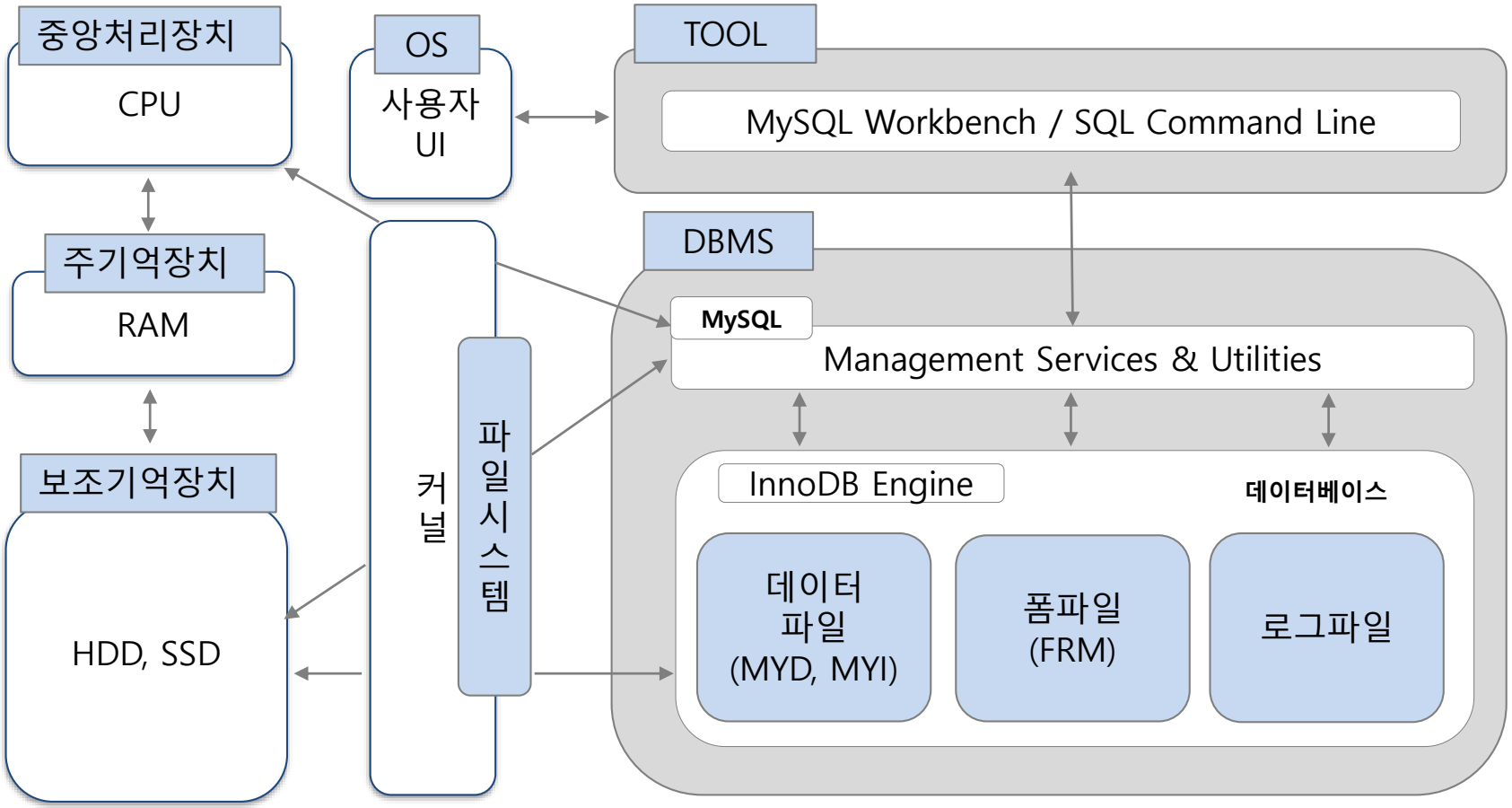


그림 4-7 **DBMS와 데이터 파일**

1. 데이터베이스의 물리적 저장

- **실제 데이터가 저장되는 곳은 보조기억장치**
  - 하드디스크, SSD, USB 메모리 등

- **가장 많이 사용되는 장치는 하드디스크**
  - 하드디스크는 원형의 플레이트(plate)로 구성되어 있고, 이 플레이트는 논리적으로 트랙으로 나뉘며 트랙은 다시 몇 개의 섹터로 나뉨
  - 원형의 플레이트는 초당 빠른 속도로 회전하고, 회전하는 플레이트를 하드디스크의 액세스 암(arm)과 헤더(header)가 접근하여 원하는 섹터에서 데이터를 가져옴
  - 하드디스크에 저장된 데이터를 읽어 오는 데 걸리는 시간은 모터(motor)에 의해서 분당 회전하는 속도(RPM, Revolutions Per Minute), 데이터를 읽을 때 액세스 암이 이동하는 시간(latency time), 주기억장치로 읽어오는 시간(transfer time)에 영향을 받음

1. 데이터베이스의 물리적 저장

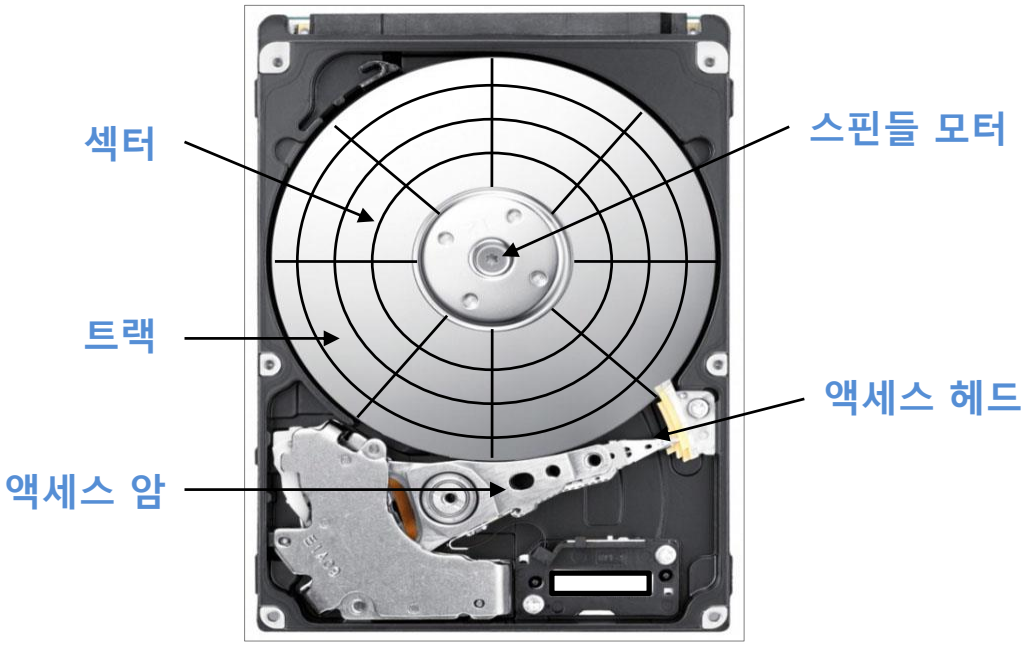


그림 4-8 하드디스크의 구조

1. 데이터베이스의 물리적 저장

❖ 액세스 시간(access time)

액세스 시간 = 탐색시간(seek time, 액세스 헤드를 트랙에 이동시키는 시간)
+ 회전지연시간(rotational latency time, 섹터가 액세스 헤드에 접근하는 시간)
+ 데이터 전송시간(data transfer time, 데이터를 주기억장치로 읽어오는 시간)

1. 데이터베이스의 물리적 저장

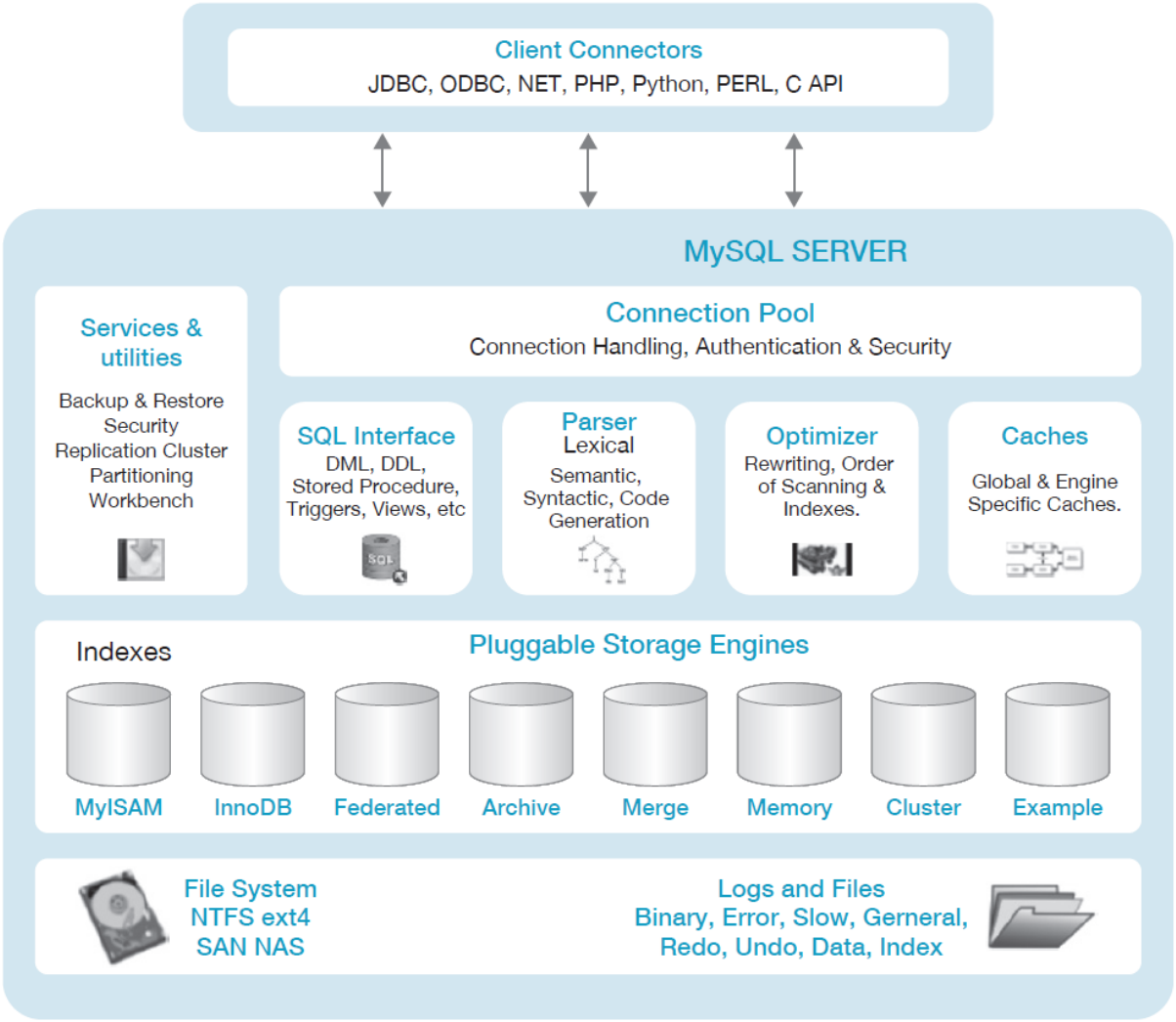


그림 4-9 **MySQL의 DBMS 구조**

1. 데이터베이스의 물리적 저장

표 4-8 **MySQL InnoDB 엔진 데이터베이스의 파일**

파일	설명
데이터 파일 (ibdata)	• 사용자 데이터와 개체를 저장 • 테이블과 인덱스로 구성 • 확장자는 *.ibd
폼파일 (frm File)	• 테이블에 대한 각종 정보와 테이블을 구성하는 필드, 데이터 타입에 대한 정보 저장 • 데이터베이스 구조 등의 변경사항이 있을 때 자동으로 업데이트됨

2. 인덱스와 B-tree

- 인덱스(index, 색인) : 도서의 색인이나 사전과 같이 데이터를 쉽고 빠르게 찾을 수 있도록 만든 데이터 구조

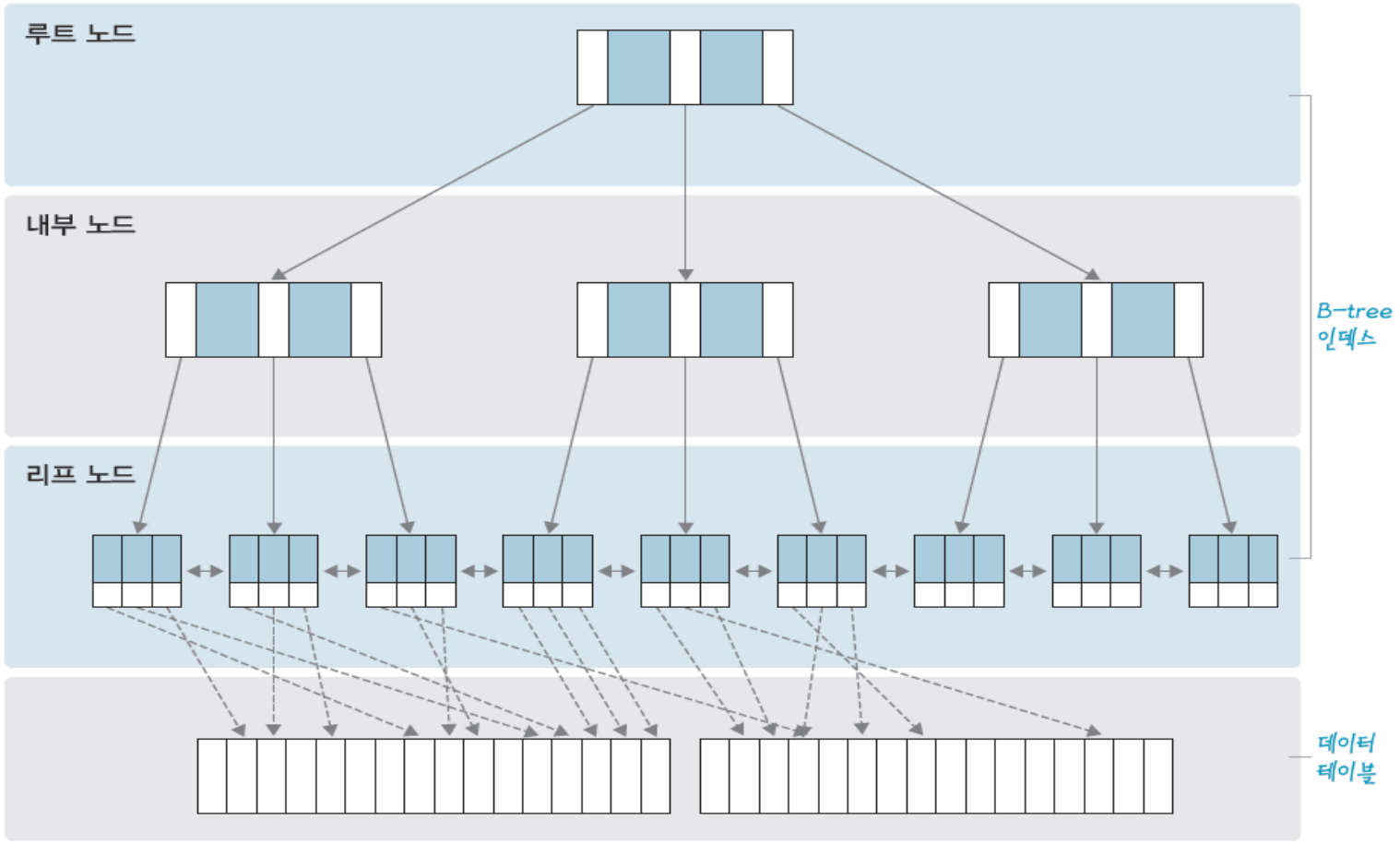


그림 4-10 B-tree의 구조

2. 인덱스와 B-tree

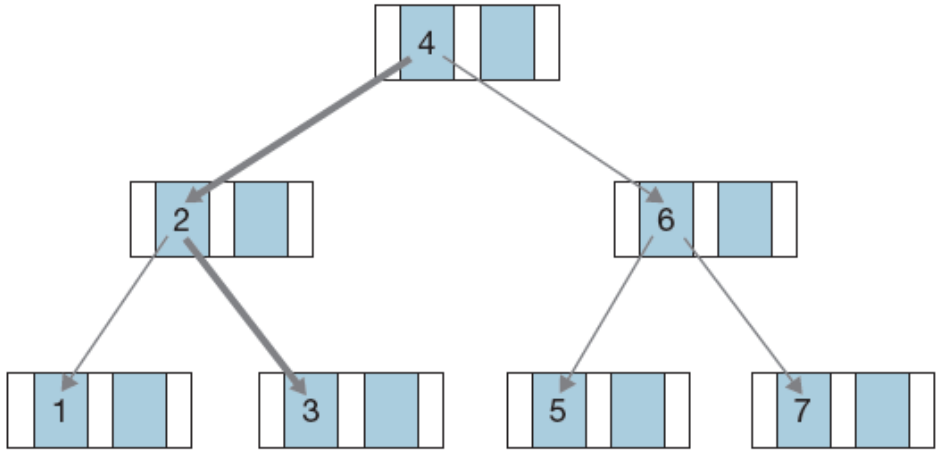


그림 4-11 **B-tree에서 검색 예**

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
- 데이터의 수정, 삭제 등의 변경이 발생하면 인덱스의 재구성이 필요함

3. MySQL 인덱스

❖ 클러스터 인덱스

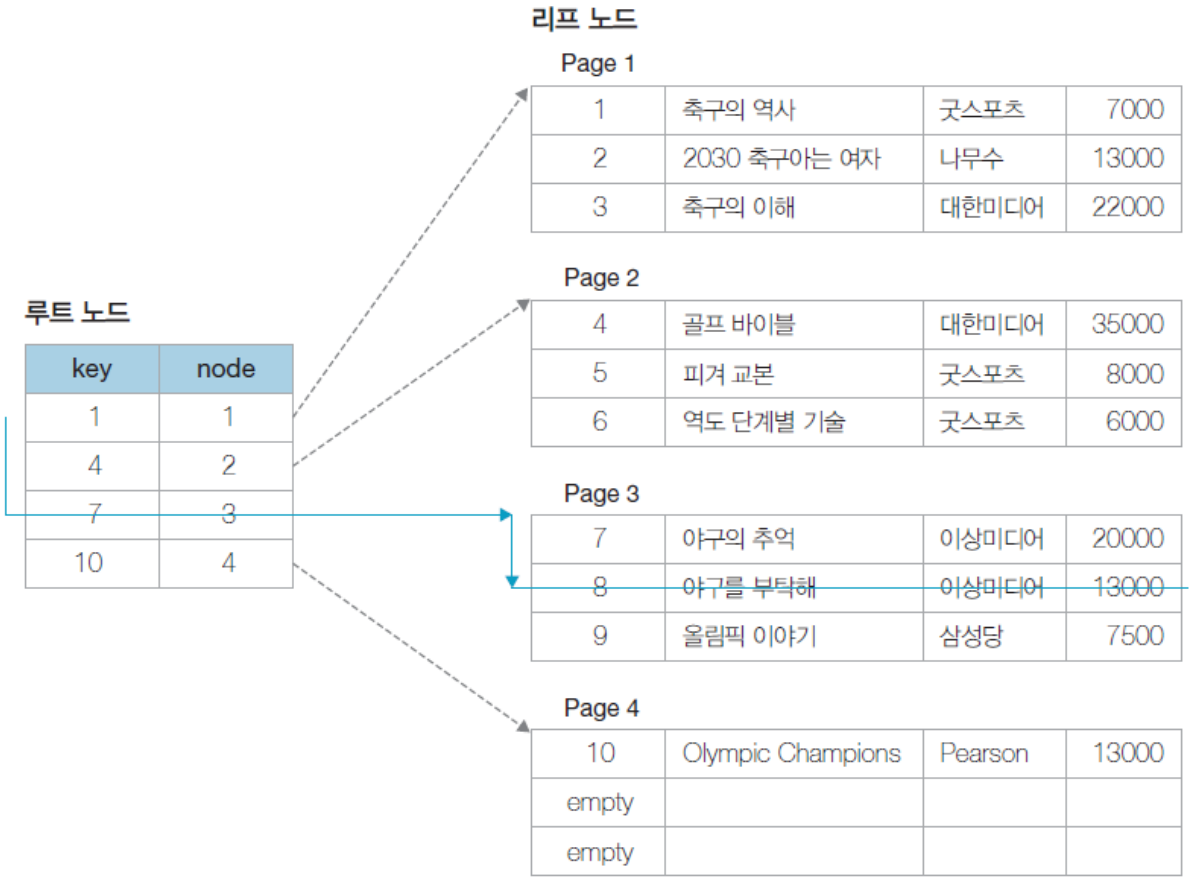


그림 4-12 클러스터 인덱스 예

3. MySQL 인덱스

❖ MySQL 인덱스 B-tree

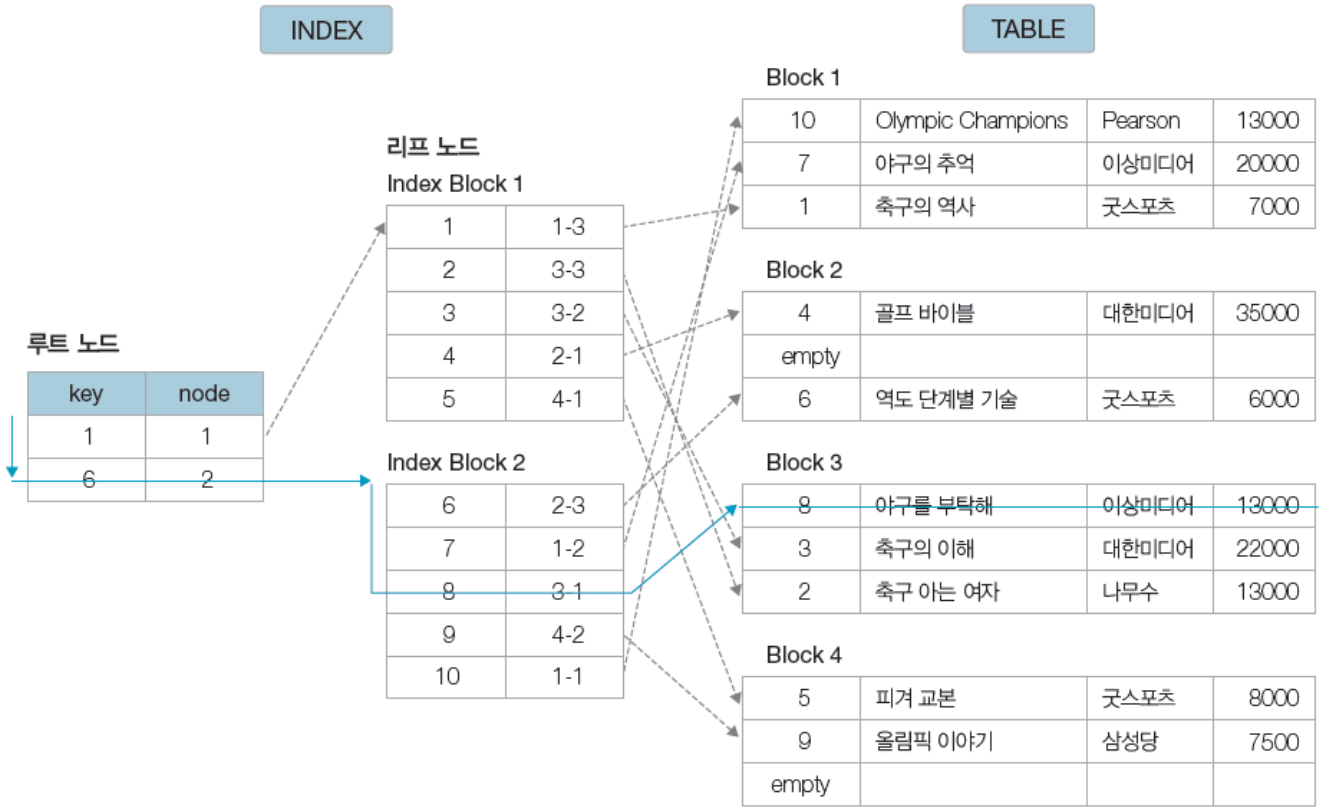


그림 4-13 **B-tree 인덱스의 예**

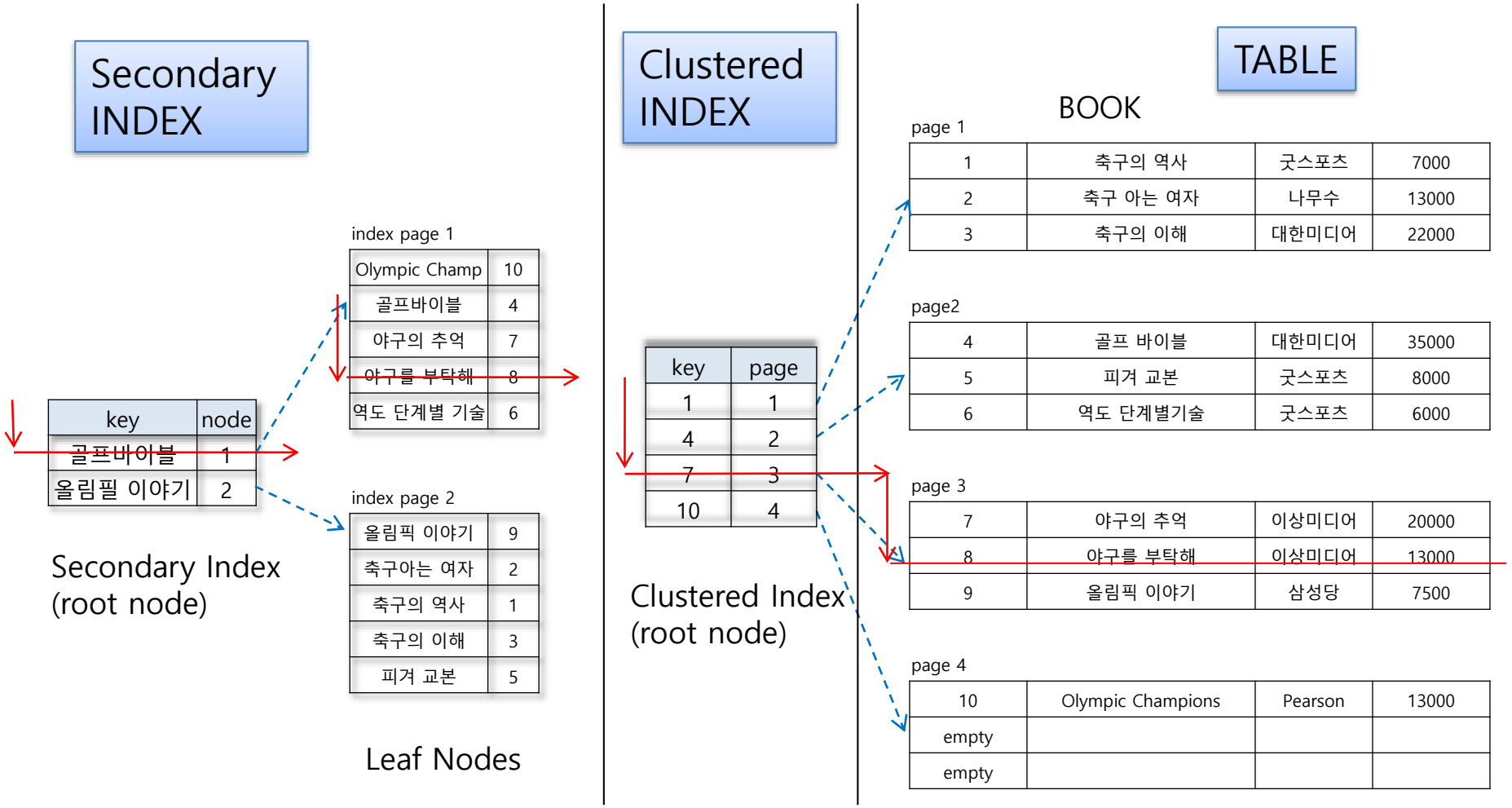
3. MySQL 인덱스

❖ MySQL 인덱스의 종류

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 MySQL 이 자체 생성한 행번호(Row ID)를 이용하여 클러스터 인덱스를 생성한다.
보조 인덱스	• 클러스터 인덱스가 아닌 모든 인덱스는 보조 인덱스이며 보조 인덱스의 각 레코드는 보조 인덱스 속성과 기본키 속성 값을 갖고 있다. • 보조 인덱스를 검색하여 기본키 속성 값을 찾은 다음 클러스터 인덱스로 가서 해당 레코드를 찾는다.

3. MySQL 인덱스



[그림 4-14] 클러스터 인덱스와 보조 인덱스를 동시에 사용하는 검색

4. 인덱스의 생성

❖ 인덱스 생성 시 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 경우 사용하지 않음
- 속성의 선택도가 낮을 때 유리함(속성의 모든 값이 다른 경우)

❖ 인덱스의 생성 문법

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

4. 인덱스의 생성

질의 4-24 Book 테이블의 bookname 열을 대상으로 비 클러스터 인덱스 ix_Book을 생성하라.

```
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
```

Table	Non_unique	Key_name	Seq_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

4. 인덱스의 생성

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

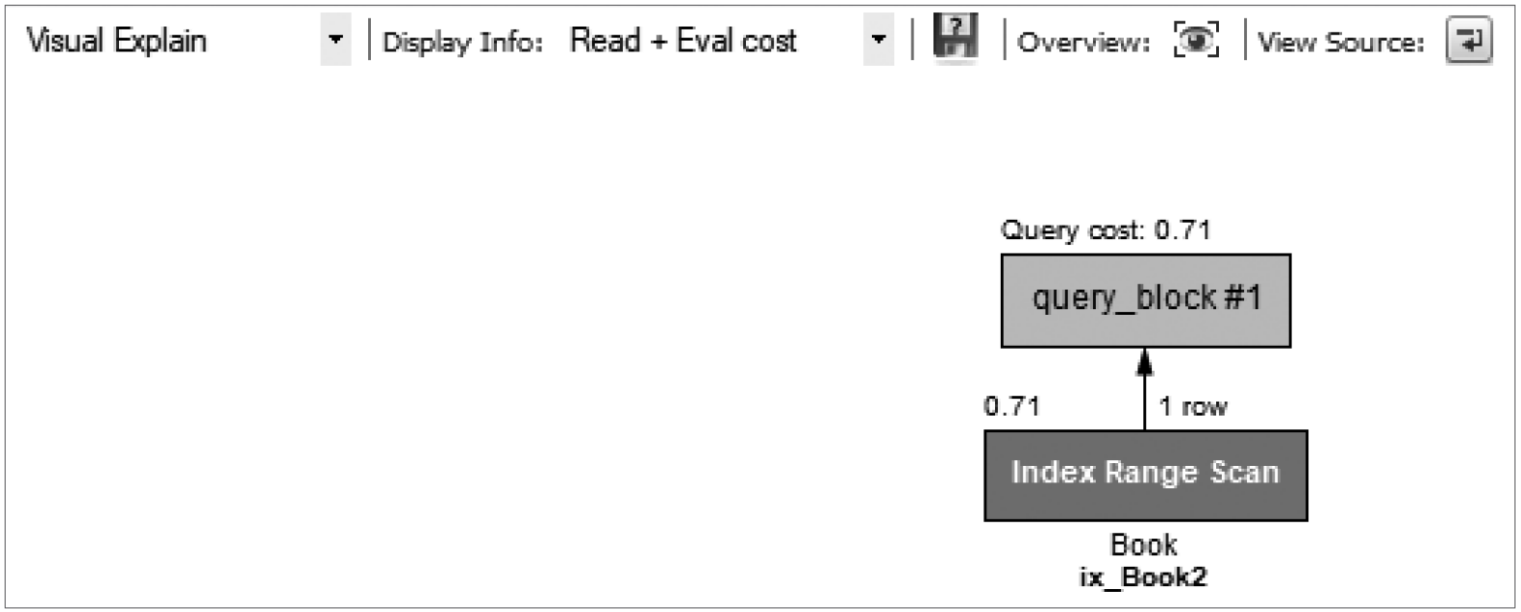


그림 4-15 실행 계획(Execution Plan)을 통한 인덱스 사용 확인

5. 인덱스의 재구성과 삭제

- **인덱스의 재구성은 ANALYZE TABLE 명령을 사용함.**
- **생성 문법**

```
ANALYZE TABLE 테이블이름;
```

질의 4-26　Book 테이블의 인덱스를 최적화하시오.

```
ALTER TABLE Book;
```

Table	Op	Msg_type	Msg_text
madang.book	analyze	status	OK

- **삭제 문법**

```
DROP INDEX 인덱스이름
```

질의 4-27　인덱스 ix_Book을 삭제하시오.

```
DROP INDEX ix_Book ON Bppk;
```

```
0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0          0.187 sec
```

요약

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