

## 4장 SQL 고급

### 데이터베이스응용 한문석

#### 목차

1. 내장함수
2. 부속질의
3. 뷰
4. 인덱스

## 학습목표

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

## 내장함수(Built-in Function)

## 01. 내장함수

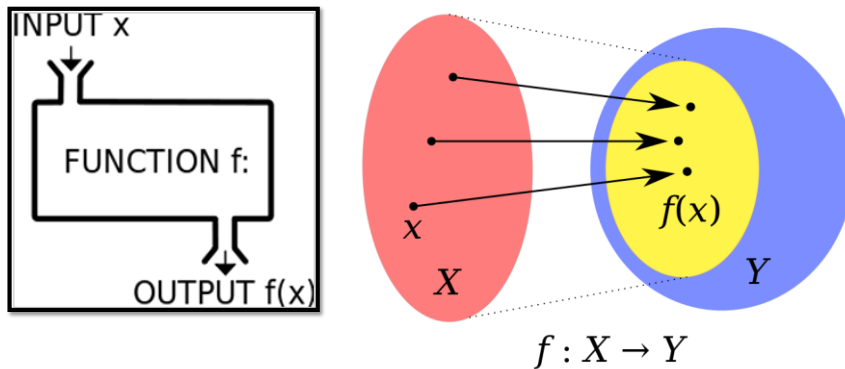
- SQL 내장 함수
- NULL 값 처리
- 행 번호 출력

## 01. 내장 함수

- SQL에서는 함수의 개념을 사용
- 입력
  - 수식의 함수와 마찬가지로 **특정 값이나 열의 값**
- 출력
  - 그 값을 계산하여 **결과 값**을 돌려줌
- SQL의 함수의 종류
  - DBMS가 제공하는 내장 함수(**built-in function**)
  - 사용자가 필요에 따라 직접 만드는 사용자 정의 함수(**user-defined function**)

## 01. 내장 함수

### 함수의 원리



2023-09-04

컴퓨터공학과

7

## 1. SQL 내장 함수(MySQL)

- 상수나 속성 이름을 입력 값으로 함
- 단일 값을 결과로 반환함
- 모든 내장 함수
  - 최초로 선언될 때 유효한 입력 값을 받아야 함
- 사용장소
  - SELECT, WHERE, UPDATE ~ SET 절

2023-09-04

컴퓨터공학과

8

# 1. SQL 내장 함수(MySQL)

## • 단일행 함수

### – 숫자함수

- ABS, CEIL, COS, EXP, FLOOR, LN=LOG, MOD, POWER, ROUND(number), SIGN, TRUNC(number)

### – 문자함수(문자반환)

- CHR, CONCAT, LOWER, LPAD
- LTRIM, STR, REPLACE, RPAD
- RTRIM, SUBSTR, TRIM, UPPER

# 1. SQL 내장 함수(MySQL)

## • 단일행 함수

### – 문자함수(숫자반환)

- ASCII, INSTR, LENGTH

### – 날짜/시간 함수

- ADDDATE, LAST\_DAY, DATE(date), SYSDATE
- DATE\_FORMAT(date, format),  
STR\_TO\_DATE(string, format)

# 1. SQL 내장 함수(MySQL)

- 단일행 함수
  - 변환 함수
    - CAST, CONVERT
    - DATE\_FORMAT, STR\_TO\_DATE
  - NULL 관련 함수
    - COALESCE, ISNULL, IFNULL, NULLIF

# 1. SQL 내장 함수(MySQL)

- 집계 함수
  - SUM, AVG, COUNT, MAX, MIN
  - CUME\_DIST, PERCENT\_RANK,
- 분석 함수
  - DENSE\_RANK, FIRST\_VALUE, LAST\_VALUE
  - LEAD, RANK

## 1.1 숫자 함수

함수	설명	예
ABS(숫자)	절대값 계산	$ABS(-4.5)=4.5$
CEILING(숫자)	숫자보다 크거나 같은 최소의 정수	$CEILING(4.1)=5$
FLOOR(숫자)	숫자보다 작거나 같은 최소의 정수	$FLOOR(4.1)=4$
ROUND(숫자, m)	숫자의 반올림, m은 반 올림 기준 자릿수	$ROUND(5.36, 1)=5.40$

2023-09-04

컴퓨터공학과

13

## 1.1 숫자 함수

함수	설명	예
LOG(숫자)	숫자의 자연로그 값을 반환	$LOG(10)=2.30259$
POWER(숫자, n)	숫자 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 (부호표시)	$SIGN(3.45)=1$

2023-09-04

컴퓨터공학과

14

## 1.1 수학 함수

Dual 테이블: 오라클에서 일시적 연산 작업에 사용하는 가상테이블  
MySQL/SQL서버는 From절 없이도 가능(Select문으로만 구성 가능)

- 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);
```

ROUND(4.875, 1)
4.9

2023-09-04

컴퓨터공학과

15

## 1.1 수학 함수

- 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;
```

$31000/3=10333$

고객번호	평균금액
1	13000
2	7500
4	16500
3	10300

2023-09-04

컴퓨터공학과

16



## 1.2 문자 함수

반환구분	함수	설명
<b>문자값 반환 함수</b>  <b>s</b> : 문자열 <b>c</b> : 문자 <b>n</b> : 정수 <b>k</b> : 정수	<b>CHR(k)</b>	정수 아스키 코드를 문자로 반환 CHR(68) = 'D'
	<b>CONCAT(s1,s2)</b>	두 문자열 연결 CONCAT( '마당' , '서점' ) = '마당 서점'
	<b>INITCAP(s)</b>	문자열의 첫 번째 알파벳을 대문자로 변환 INITCAP( 'the soap' ) = 'The Soap'
	<b>LOWER(s)</b>	대상 문자열을 모두 소문자로 변환 LOWER( 'MR. SCOTT' ) = 'mr. scott'
	<b>LPAD(s,n,c)</b>	대상 문자열 왼쪽부터 지정된 자리 수까지 지정된 문자 채움 LPAD( 'Page 1' , 10, '*' ) = '****Page 1'
	<b>LTRIM(s1,s2)</b>	대상 문자열의 왼쪽부터 지정된 문자들을 제거 LTRIM( '<=>BROWNING<=>' , '<>=' ) = 'BROWNING<=>'

2023-09-04

컴퓨터공학과

17

## 1.2 문자 함수

반환구분	함수	설명
<b>문자값 반환 함수</b>  <b>s</b> : 문자열 <b>c</b> : 문자 <b>n</b> : 정수 <b>k</b> : 정수	<b>REPLACE(s1,s2,s3)</b>	대상 문자열의 지정된 문자를 원하는 문자로 변경 (예) REPLACE( 'JACK and JUE' , 'J' , 'BL' ) = 'BLACK and BLUE'
	<b>RPAD(s,n,c)</b>	대상 문자열의 오른쪽부터 지정된 자리 수까지 지정된 문자로 채움 (예) RPAD( 'AbC' , 5, '*' ) = 'AbC**'
	<b>RTRIM(s1,s2)</b>	대상 문자열의 오른쪽부터 지정된 문자들을 제거 (예) RTRIM( '<=>BROWNING<=>' , '<>=' ) = '<=>BROWNING'
	<b>SUBSTR(s,n,k)</b>	대상 문자열의 지정된 자리에서부터 지정된 길이만큼 잘라서 반환 (예) SUBSTR( 'ABCDEF' , 3, 4 ) = 'CDEF'
	<b>TRIM(c FROM s)</b>	대상 문자열의 양쪽에서 지정된 문자를 삭제 (문자열만 넣으면 기본값으로 공백 제거) (예) TRIM(' FROM '==>BROWNING<==') = '>BROWNING<'
	<b>UPPER(s)</b>	대상 문자열을 모두 대문자로 변환 (예) UPPER( 'mr. scott' ) = 'MR. SCOTT'

2023-09-04

컴퓨터공학과

18

## 1.2 문자 함수

반환구분	함수	설명
숫자값 반환 함수	ASCII(c)	대상 알파벳 문자의 아스키 코드 값을 반환 (예) ASCII( 'D' ) = 68
	INSTR(s1,s2,n,k)	s1에서 n번째 문자부터 시작하여 찾고자 하는 문자열 s2가 k번째 나타나는 문자. 열 위치 반환, 예제에서 3번째부터 OR가 2번째 나타나는 자리 수 (예) INSTR( 'CORPORATE FLOOR' , 'OR' , 3, 2) = 14
	LENGTH(s)	대상 문자열의 글자 수를 반환 (예)LENGTH( 'CANDIDE' ) = 7

2023-09-04

컴퓨터공학과

19

## 1.2 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	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000

2023-09-04

컴퓨터공학과

20

## 1.2 문자 함수

- CHAR\_LENGTH : 글자의 수를 세어주는 함수 (단위가 바이트(byte)가 아닌 문자 단위)

질의 4-5 굿스포즈에서 출판한 도서의 제목과 제목의 글자 수를 확인하시오.

```
SELECT bookname "제목", CHAR_LENGTH(bookname)
"글자수", LENGTH(bookname) "바이트수"
FROM Book
WHERE publisher='굿스포즈';
```

Result Grid			Filter Rows:
제목	글자 수	바이트수	
▶ 축구역사	6	16	
피겨 교본	5	13	
역도 단계별기술	8	22	

2023-09-04

컴퓨터공학과

21

## 1.2 문자 함수

- SUBSTR:
  - 특정 위치에서 시작 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
추	1
장	1

2023-09-04

컴퓨터공학과

22

## 1.3 날짜 · 시간 함수

함수	반환형	설명
<b>STR_TO_DATE</b> (string, format)	DATE	문자형(CHAR) 데이터를 날짜형으로 반환 <b>STR_TO_DATE</b> ( '2020-09-07' , '%Y-%m-%d' ) = 2020-09-07
<b>DATE_FORMAT</b> (date, format)	STRING	날짜 데이터를 문자열(VARCHAR) 반환 <b>DATE_FORMAT</b> ( '2020-09-07' , '%Y-% m-%d' ) = 2020-09-07
<b>ADDDATE</b> (date, interval)	DATE	date형의 날짜에 interval 지정한 날만큼 더함 <b>ADDDATE</b> ( '2020-09-07' , INTERVAL 10 DAY) = 2020-09-17 동일 mysql> SELECT <b>DATE_ADD</b> ('2008-01-02', INTERVAL 31 DAY); -> '2008-02-02'

2023-09-04

컴퓨터공학과

23

## 1.3 날짜 · 시간 함수

함수	반환형	설명
<b>DATE</b> (date)	DATE	DATE형의 날짜 부분을 반환 <b>DATE</b> ( '2020-09-07 18:30:10' ); → 2020-09-07
<b>DATEDIFF</b> (date1, date2)	Integer	DATE형의 date1-date2 수행 날짜 차이 반환 <b>DATEDIFF</b> ( '2020-09-07' , '2020-09- 01' ) → 6
<b>LAST_DAY</b> (date)	DATE	date 형의 날짜에서 달의 마지막 날을 반환 <b>LAST_DAY</b> ('2020-09-07'); = 2020-09-30
<b>SYSDATE</b>	DATE	DBMS 시스템상의 오늘 날짜를 반환하는 함수 <b>SYSDATE</b> () = '2020-09-06 16:37:24'

2023-09-04

컴퓨터공학과

24

## 1.3 날짜 함수

### datetime의 주요 인자

인자	설명	인자	설명
%w	요일 순서(1~7, 월=1)	%i	분(0~59)
%W	요일(월요일~일요일)	%m	월 순서(01~12, January=01)
%a	요일의 약자(월~일)	%b	월 이름 약어(Jan~Dec)
%d	1달 중 날짜(1~31)	%M	월 이름(January~December)
%j	1년 중 날짜(1~366)	%s	초(0~59)
%h	12시간(1~12)	%Y	4자리 연도
%H	24시간(0~23)	%y	4자리 연도의 마지막 2자리

2023-09-04

컴퓨터공학과

25

## 1.3 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;
```

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

2023-09-04

컴퓨터공학과

26

## 1.3 날짜 함수

질의 4-8 마당서점이 2014년 7월 7일에 주문 받은 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

2023-09-04

컴퓨터공학과

27

## 1.3 날짜 함수

질의 4-8 DBMS 서버에 설정된 현재 시간과 오늘 날짜를 확인하시오.

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

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
SYSDATE()	SYSDATE_1		
▶ 2020-09-06 17:26:41	2020/09/06 September 05:41		

2023-09-04

컴퓨터공학과

28

## 2. NULL 값 처리

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

## 2. NULL 값 처리

- 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 NULL
    - COUNT 함수의 결과는 0

## 2. NULL 값 처리

- NULL 값에 대한 연산과 집계 함수
  - Mybook 테이블 생성: 스크립트 참조

### Mybook

bookid	price
1	10000
2	20000
3	NULL

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

PRICE+100
{null}

2023-09-04

컴퓨터공학과

31

## 2. NULL 값 처리

- NULL 값에 대한 연산과 집계 함수

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

SUM(PRICE)	AVG(PRICE)	COUNT(*)	COUNT(PRICE)
30000	15000	3	2

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

SUM(PRICE)	AVG(PRICE)	COUNT(*)
{null}	{null}	0

### Mybook

bookid	price
1	10000
2	20000
3	NULL

2023-09-04

컴퓨터공학과

32



## 2. NULL 값 처리

- NULL 값을 확인하는 방법
  - IS NULL, IS NOT NULL
  - NULL 값을 찾을 때
    - ‘=’ 연산자가 아닌 ‘IS NULL’ 사용
  - NULL이 아닌 값을 찾을 때
    - ‘< >’ 연산자가 아닌 ‘IS NOT NULL’ 사용함

2023-09-04

컴퓨터공학과

33

## 2. NULL 값 처리

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

BOOKID	PRICE
3	(null)

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

BOOKID	PRICE



Mybook

bookid	price
1	10000
2	20000
3	NULL

2023-09-04

컴퓨터공학과

34

## 2. NULL 값 처리

- IFNULL

- NULL 값을 다른 값으로 대체하여 연산/출력

- IFNULL(속성, 값)

- 속성 값이 NULL이면 '값'으로 대체

- IFNULL(expr1,expr2)

- If expr1 is not NULL, IFNULL() returns expr1; otherwise it returns expr2.

2023-09-04

컴퓨터공학과

35

## 2. NULL 값 처리

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

```
mysql> SELECT IFNULL(1,0);
-> 1
mysql> SELECT IFNULL(NULL,10);
-> 10
mysql> SELECT IFNULL(1/0,10);
-> 10
mysql> SELECT IFNULL(1/0,'yes');
-> yes
```

2023-09-04

컴퓨터공학과

36

### 3. 행번호 출력

- 내장 함수는 아님
- 자주 사용되는 문법
- MySQL
  - 변수는 이름 앞에 @ 기호를 붙임
  - 치환문
    - SET과 := 기호를 사용함
- 자료를 일부분만 확인, 처리할 때 유용

2023-09-04

컴퓨터공학과

37

### 3. 행번호 출력

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

2023-09-04

컴퓨터공학과

38

## 부속질의(하위쿼리::Subquery)

### 부속질의 정의

- 하나의 SQL 문 안에 다른 SQL 문이 **중첩**  
– SELECT, INSERT, UPDATE 또는 DELETE 문  
이나 다른 하위 쿼리 내부에 중첩
- **multiple-part** questions
- 다른 테이블에서 가져온 데이터를 이용
- 현재 테이블에 있는 정보를 검색/가공

## 부속질의 정의

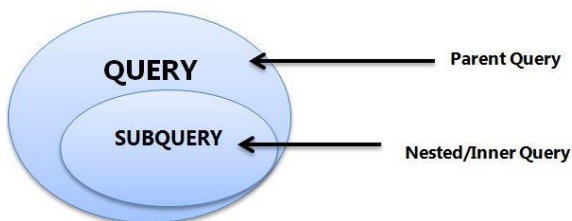
- 용도
  - 보통 데이터가 대량일 때
  - 데이터를 모두 압축해서 연산하는 조인보다
  - 필요한 데이터만 찾아서 공급해주는 부속질의가 성능 우수
- 구성
  - 주질의(main query, 외부질의)
  - 부속질의(sub query, 내부질의, 아워질의)

2023-09-04

컴퓨터공학과

41

## 부속질의 구성



주 질의 (main query)	부속질의 (sub query)
<pre> SELECT  SUM(saleprice) FROM    Orders WHERE   custid =           </pre>	<pre> (SELECT  custid FROM    Customer WHERE   name = '박지성')           </pre>

2023-09-04

컴퓨터공학과

42

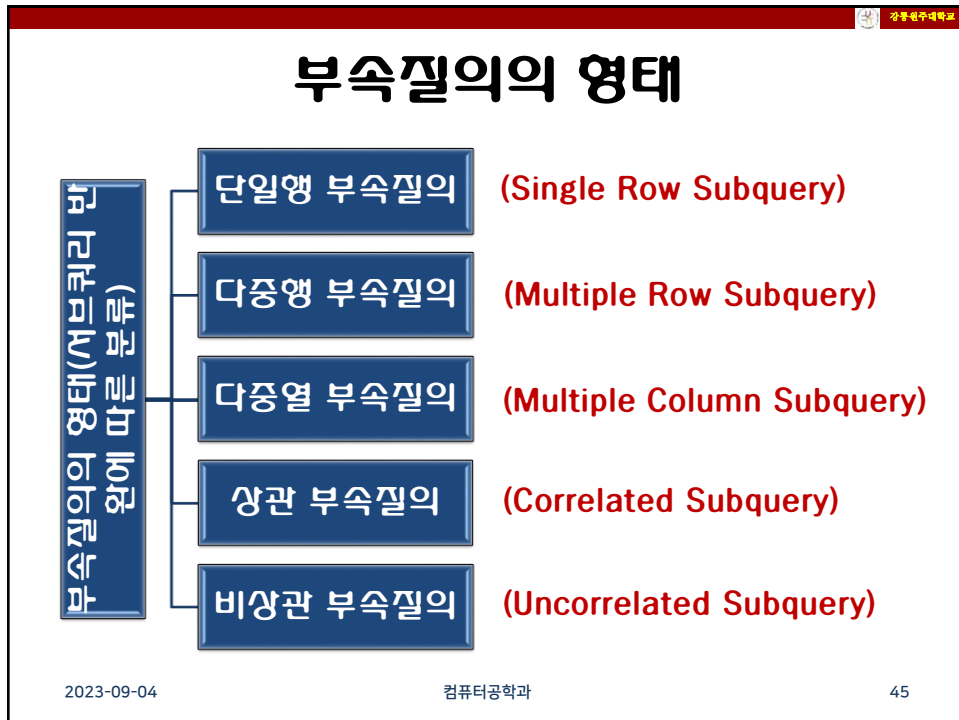
## 부속질의 예

```
-- 직업이 'MANAGER' 인 사원의 사원명, 부서명을 조회하는 예제
SELECT ename,
       (SELECT dname FROM dept d WHERE d.deptno = e.deptno) 'deptno'
FROM emp e
WHERE job ='MANAGER';
```

ENAME	DEPTNO
JONES	RESEARCH
BLAKE	SALES
CLARK	ACCOUNTING

## 부속질의(Subquery) 형태

- 스칼라 부속질의(**Scalar** Subquery)
  - SELECT 부속질의
- 인라인 뷰(**Inline View** or Table Subquery)
  - FROM 부속질의
- 중첩질의(**Nested** or Predicate Subquery)
  - WHERE 부속질의



## Single Row Subquery

- 하나의 컬럼으로 구성된 조외 결과
  - 행 하나를 outer 쿼리에 반환

```

SELECT agent_name, agent_code, phone_no
FROM agents
WHERE agent_code =
      (SELECT agent_code
      FROM agents
      WHERE agent_name = 'Alex');
          
```

2023-09-04
컴퓨터공학과
46

## Multiple Row Subquery

### • 서브쿼리 결과

- 여러 개의 행을 주쿼리에 반환
- IN, ANY, ALL, EXISTS 등의 연산자로 연음

```
SELECT ord_num, ord_amount, ord_date, cust_code, agent_code
FROM orders
WHERE agent_code IN (
    SELECT agent_code FROM agents
    WHERE working_area='Bangalore');
```

2023-09-04

컴퓨터공학과

47

## Multiple Column Subquery

### • 서브쿼리 결과

- 여러 개의 행을 outer 쿼리에 반환
- WHERE 또는 HAVING ~ IN 연산자 사이
  - 컬럼 리스트가 괄호로 묶여 있어야 함

```
SELECT ord_num, agent_code, ord_date, ord_amount
FROM orders
WHERE (agent_code, ord_amount) IN
    (SELECT agent_code, MIN(ord_amount)
    FROM orders
    GROUP BY agent_code);
```

2023-09-04

컴퓨터공학과

48



## Correlated Subquery

- 중첩 서브쿼리의 한 종류
- 주질의 속성을 내부질의 사용

```
SELECT ord_num, agent_code, ord_date, ord_amount
FROM orders
WHERE (agent_code, ord_amount) IN
      (SELECT agent_code, MIN(ord_amount)
       FROM orders
       GROUP BY agent_code);
```

2023-09-04

컴퓨터공학과

49

## Uncorrelated Subquery

- Simple Subquery
- 각 테이블에 대해서 한 번 평가되는 쿼리

```
SELECT SUM (Sales)
FROM Store_Information
WHERE Store_Name IN
      ( SELECT Store_Name
        FROM Geography
        WHERE Region_Name = 'West');
```

2023-09-04

컴퓨터공학과

50

## 02 부속질의 종류

명칭	위치	영문 및 동의어	설명
스칼라 부속질의	SELECT 절	scalar subquery	- SELECT 절에서 사용 - 단일 값을 반환 - 스칼라 부속질의
인라인 뷰	FROM 절	inline view, table subquery	- FROM 절에서 결과를 뷰(view) 형태로 반환 - 인라인 뷰
중첩질의	WHERE 절	nested subquery, predicate subquery	- WHERE 절에 술어와 같이 사용 - 결과를 안정시키기 위해 사용 - 상관 혹은 비상관 형태

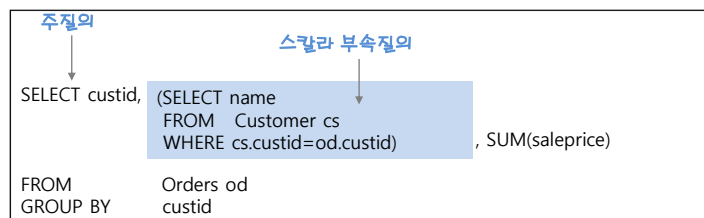
2023-09-04

컴퓨터공학과

51

### 2.1 스칼라 부속질의 – SELECT 부속질의

- **SELECT** 절에서 사용되는 부속질의
- 부속질의의 결과 값을 단일 행, 단일 열의 스칼라 값으로 반환함.
- 원칙적으로 스칼라 값이 들어갈 수 있는 모든 곳에 사용 가능
- 일반적으로 **SELECT** 문과 **UPDATE SET** 절에 사용됨.
- 주질의와 부속질의와의 관계는 상관/비상관 모두 가능함.



2023-09-04

컴퓨터공학과

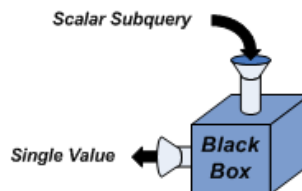
52

## 2.1 스칼라 부속질의 – SELECT 부속질의

질의 4-12 마당서점의 고객별 판매액을 보이시오  
(결과는 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
추신수	33000
장미란	31000



2023-09-04

컴퓨터공학과

53

## 2.1 스칼라 부속질의 – SELECT 부속질의

```
SELECT      custid "name", SUM(saleprice) "total"
FROM        Orders od
GROUP BY    od.custid;
```

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

```
SELECT      custid AS name, SUM(saleprice) AS total
FROM        Orders od
GROUP BY    od.custid;
```

2023-09-04

컴퓨터공학과

54

## 2.1 스칼라 부속질의 – SELECT 부속질의

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

CUSTID	total
1	39000
2	15000
4	33000
3	31000

```
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
추신수	33000
장미란	31000

마당서점의 고객별 판매액

2023-09-04

컴퓨터공학과

55

## 2.1 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	BOOKNAME
1	1	1	6000	14/07/01	축구의 역사
2	1	3	21000	14/07/03	축구의 이해
3	2	5	8000	14/07/03	피겨 교본
4	3	6	6000	14/07/04	역도 단계별기술
5	4	7	20000	14/07/05	야구의 추억
6	1	2	12000	14/07/07	축구하는 여자
7	4	8	13000	14/07/07	야구를 부탁해
8	3	10	12000	14/07/08	Olympic Champions
9	2	10	7000	14/07/09	Olympic Champions
10	3	8	13000	14/07/10	야구를 부탁해

2023-09-04

컴퓨터공학과

56

## 2.2 인라인 뷰 – FROM 부속질의

- FROM 절에서 사용되는 부속질의.
- 인라인 뷰 부속질의 사용
  - 유도테이블(derived table)이라고 함
- 부속질의 결과 반환되는 데이터
  - 다중 행, 다중 열이어도 상관없음.
- **상관 부속질의**로 사용될 수는 없음
  - 가상의 테이블인 뷰 형태로 제공되기 때문

2023-09-04

컴퓨터공학과

57

## 2.2 인라인 뷰 – FROM 부속질의

**질의 4-14** 고객번호가 2 이하인 고객의 판매액을 보이시오  
(결과는 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;
```

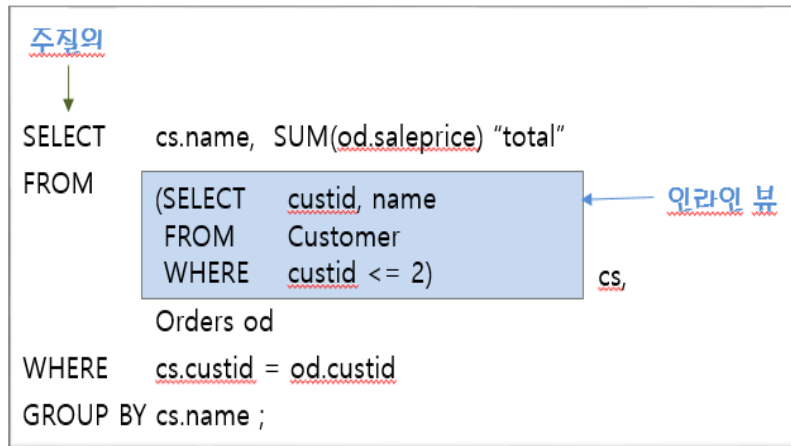
NAME	total
박지성	39000
김연아	15000

2023-09-04

컴퓨터공학과

58

## 2.2 인라인 뷰 – FROM 부속질의



2023-09-04

컴퓨터공학과

59

## 2.3 중첩질의 – WHERE 부속질의

- **WHERE 절**에서 사용되는 부속질의.
- WHERE 절은 보통 데이터를 선택하는 조건 혹은 술어 (predicate)와 같이 사용됨.
  - 중첩질의를 술어 부속질의(predicate subquery)라고 함.

### 중첩질의 연산자의 종류

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

2023-09-04

컴퓨터공학과

60

## 2.3 중첩질의 – WHERE 부속질의

- 부속질의가 반드시 단일 행, 단일 열 반환
- 아닐 경우 질의를 처리할 수 없음.
- 비교 연산자

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

2023-09-04

컴퓨터공학과

61

## 2.3 중첩질의 – WHERE 부속질의

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

2023-09-04

컴퓨터공학과

62

## 2.3 중첩질의 – WHERE 부속질의

### • IN 연산자

- 주질의 속성 값이 부속질의에서 제공한 결과 집합에 있는지 확인(check)하는 역할
- 부속질의의 결과 다중 행을 가질 수 있음.
- 주질의는 WHERE 절에 사용되는 속성 값을 부속질의의 결과 집합과 비교해 하나라도 있으면 참이 됨

### • NOT IN

- 이와 반대로 값이 존재하지 않으면 참이 됨.

2023-09-04

컴퓨터공학과

63

## 2.3 중첩질의 – WHERE 부속질의

질의 4-16 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
-------	-------

2023-09-04

컴퓨터공학과

64



## 2.3 중첩질의 – WHERE 부속질의

- ALL: 모두
  - 하나의 값 v가 집합 V내의 모든 값들과 같으면 참
  - 집합에서  $\forall$ 에 해당
- SOME(ANY): 어떠한(최소한 하나라도)
  - 하나의 값 v가 집합 V내의 어떤 하나의 값과 같으면 참
  - 집합에서  $\exists$ 에 해당
- 구문 구조
  - scalar\_expression { 비교연산자 (= | < | > | != | >= | <= | < > | < <= | > >=) }
  - {ALL | SOME | ANY} (부속질의)

2023-09-04

컴퓨터공학과

65

## 2.3 중첩질의 – WHERE 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
5	20000
2	21000

2023-09-04

컴퓨터공학과

66

## 2.3 중첩질의 – WHERE 부속질의

- EXISTS
  - 데이터의 존재 유무를 확인하는 연산자
  - 주질의에서 부속질의가 제공한 속성 값 사용
  - 부속질의에 조건을 만족하여 값이 존재하면
    - 참이 되고, 주질의는 해당 행의 데이터를 출력함.
- NOT EXISTS
  - EXISTS의 경우와 반대로 동작함.
- 구문 구조
  - WHERE [NOT] EXISTS (부속질의)

2023-09-04

컴퓨터공학과

67

## 2.3 중첩질의 – WHERE 부속질의

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

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

total
46000

2023-09-04

컴퓨터공학과

68


 강원대학교

# 03 뷰 VIEW

2023-09-04

컴퓨터공학과

69

 강원대학교

## 03. 뷰

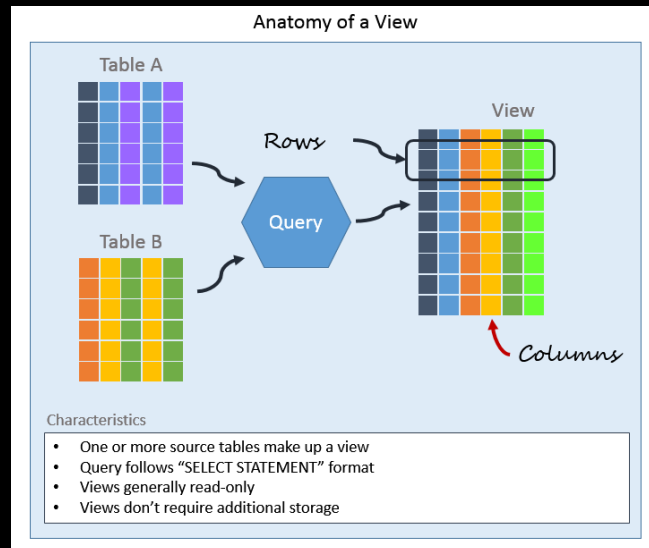
- 뷰의 생성
- 뷰의 수정
- 뷰의 삭제

2023-09-04

컴퓨터공학과

70

## 03 뷰



2023-09-04

컴퓨터공학과

71

## 03 뷰(view)

- 하나 이상의 테이블(**Defining Table**)로 만든 가상테이블(**Virtual Table**)
- 장점
  - 편리성
    - 미리 정의된 뷰를 일반 테이블처럼 사용
    - 사용자 요구 정보만 가공하여 뷰로 생성 사용
  - 재 사용성
    - 자주 사용되는 질의를 뷰로 미리 정의
  - 보안성
    - 각 사용자 별로 필요한 데이터만 선별하여 보여줄 수 있음
  - 독립성
    - 원본 테이블의 구조가 변해도 응용에 영향을 주지 않도록 하는 논리적 독립성 제공

2023-09-04

컴퓨터공학과

72

## 03 뷰(view) 특징

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

2023-09-04

컴퓨터공학과

73

## 03 뷰

뷰 Vorders

ORDERID	CUSTID	NAME	BOOKID	BOOKNAME	SALEPRICE	ORDERDATE
1	1 박지성		1 축구의 역사		6000	14/07/01
6	1 박지성		2 축구마는 여자		12000	14/07/07
2	1 박지성		3 축구의 이해		21000	14/07/03
3	2 김연아		5 피겨 교본		8000	14/07/03

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;
```

### Orders 테이블에 고객이름과 도서이름 추가하는 방법

베이스 릴레이션 Customer, Orders, Book

O				C				B			
ORDERID	CUSTID	BOOKID	SALEPRICE	ORDERDATE	CUSTID	NAME	ADDRESS	PHONE	BOOKID	BOOKNAME	PUBLISHER
1	1	1	6000	14/07/01	1	박지성	영국 맨체스터	000-5000-0001	1	축구역의 역사	굿스포츠
2	1	3	21000	14/07/03	2	김연아	대한민국 서울	000-6000-0001	2	축구마는 여자	나무수
3	2	5	8000	14/07/03	3	장미란	대한민국 강원도	000-7000-0001	3	축구역의 이해	대한미디어
4	3	6	6000	14/07/04							
5	4	7	20000	14/07/05							

2023-09-04

컴퓨터공학과

74

## 1. 뷰의 생성

### ■ 기본 문법

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

### ■ Book 테이블에서 '축구' 라는 문구가 포함된 자료만 보여주는 SELECT문

```
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%';
```

2023-09-04

컴퓨터공학과

75

## 1. 뷰의 생성

질의 4-20 주소에 '대한민국' 을 포함하는 고객들로 구성된 뷰를 만들고 조회하시오. 단, 뷰의 이름은 vw\_Customer로 한다.

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

### <결과 확인>

```
SELECT      *
FROM        vw_Customer;
```

CUSTID	NAME	ADDRESS	PHONE
2	김연마	대한민국 서울	000-6000-0001
3	장미란	대한민국 강원도	000-7000-0001
5	박세리	대한민국 대전	(null)

2023-09-04

컴퓨터공학과

76

## 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

2023-09-04

컴퓨터공학과

77

## 2. 뷰의 수정

### ■ 기본 문법

**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 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 맨체스터

2023-09-04

컴퓨터공학과

78

### 3. 뷰의 삭제

#### ■ 기본 문법

```
DROP VIEW 뷰이름 [ ,...n ];
```

필의 4-23 앞서 생성한 뷰 vw\_Customer를 삭제하십시오.

```
DROP VIEW vw_Customer;
```

<결과 확인>

```
SELECT      *
FROM        vw_Customer;
```

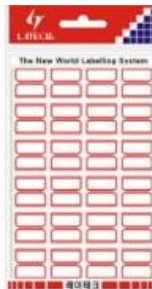
Message	Duration / Fetch
Error Code: 1146. Table 'madang.vw_customer' doesn't exist	0.000 sec

2023-09-04

컴퓨터공학과

79

## 04 인덱스(Index)





## 사 례

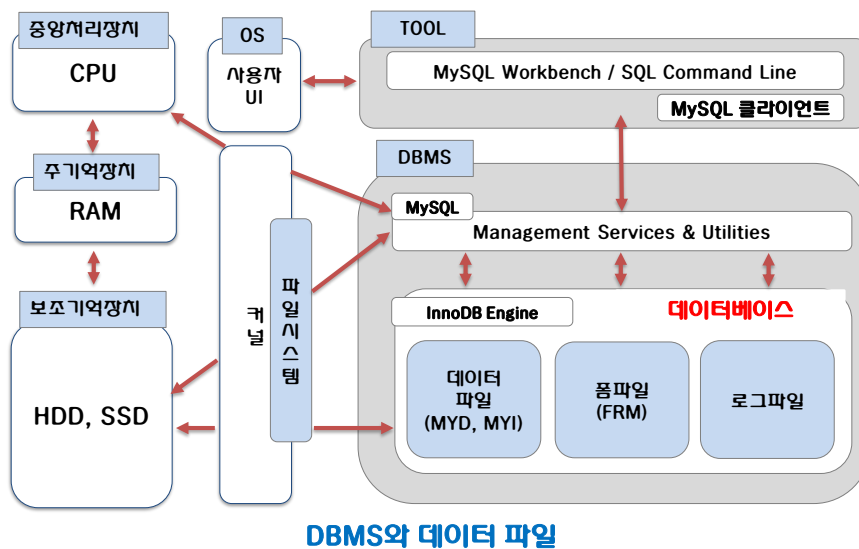
- 데이터베이스의 물리적 저장
- 인덱스와 B-tree
- MySQL 인덱스
- 인덱스의 생성
- 인덱스의 재구성과 삭제

2023-09-04

컴퓨터공학과

81

## 데이터베이스의 물리적 저장



2023-09-04

컴퓨터공학과

82

## 데이터베이스의 물리적 저장

- InnoDB:
  - a **storage engine** for the database management system MySQL and MariaDB
- .frm 파일
  - 테이블 구조 저장
- .MYD 파일
  - 실제 데이터
- .MYI 파일
  - Index 정보

2023-09-04

컴퓨터공학과

83

## 데이터베이스의 물리적 저장

- 실제 데이터가 저장되는 곳: 보조기억장치
  - 하드디스크, SSD, USB 메모리 등
- 가장 많이 사용되는 장치: 하드디스크
  - 하드디스크는 원형의 플레이트(plate)로 구성
  - 플레이트는 논리적으로 트랙과 섹터로 나뉨
- 원형의 플레이트는 초당 빠른 속도로 회전
  - 회전하는 플레이트를 하드디스크의 액세스 암(access arm)과 헤더(header)가 접근하여 원하는 섹터에서 데이터를 가져옴.

2023-09-04

컴퓨터공학과

84

## 데이터베이스의 물리적 저장

- 하드디스크 저장 데이터 읽어 오는 시간 요소
  - RPM, Revolutions Per Minute
    - 모터(motor)에 의해서 분당 회전하는 속도
  - latency time
    - 데이터를 읽을 때 액세스 암이 이동하는 시간
  - transfer time
    - 주기억장치로 읽어오는 시간

2023-09-04

컴퓨터공학과

85

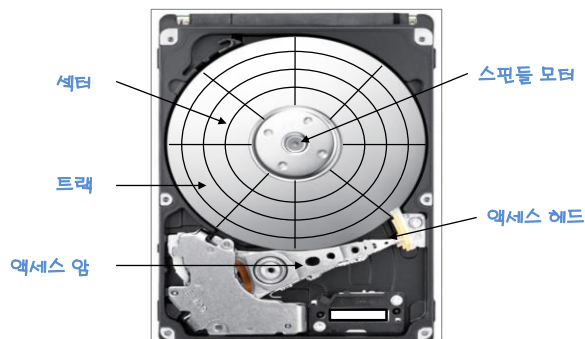
## 데이터베이스의 물리적 저장

### ● 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)

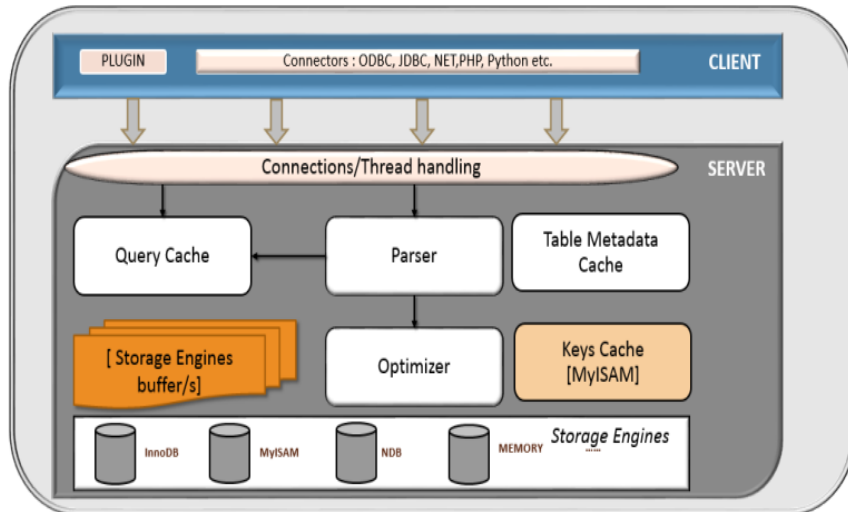


2023-09-04

컴퓨터공학과

86

## MySQL Logical Architecture:

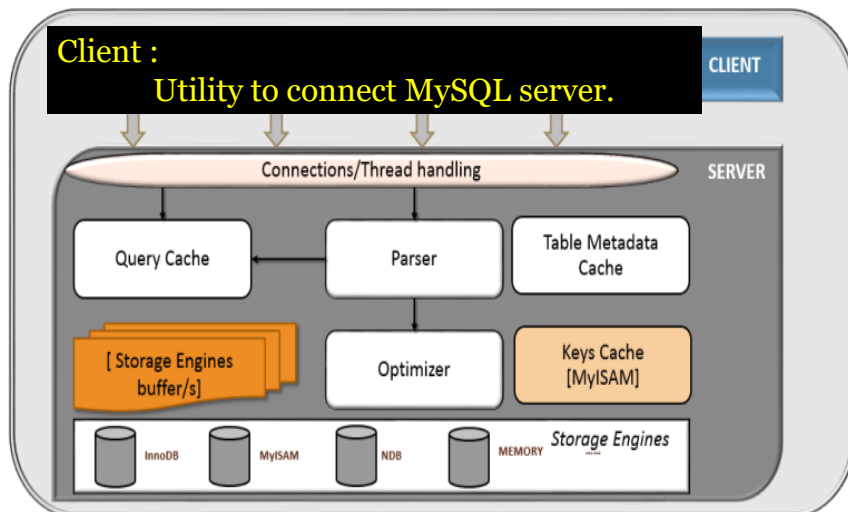


2023-09-04

컴퓨터공학과

87

## MySQL Logical Architecture:

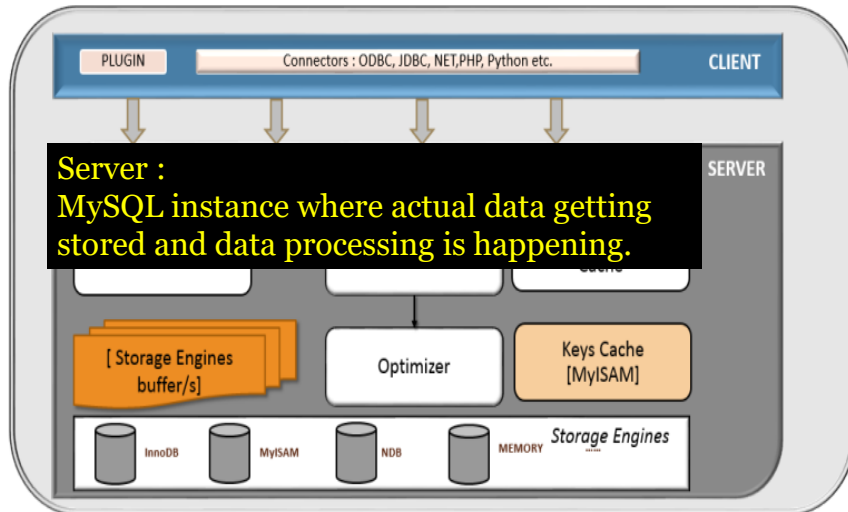


2023-09-04

컴퓨터공학과

88

## MySQL Logical Architecture:

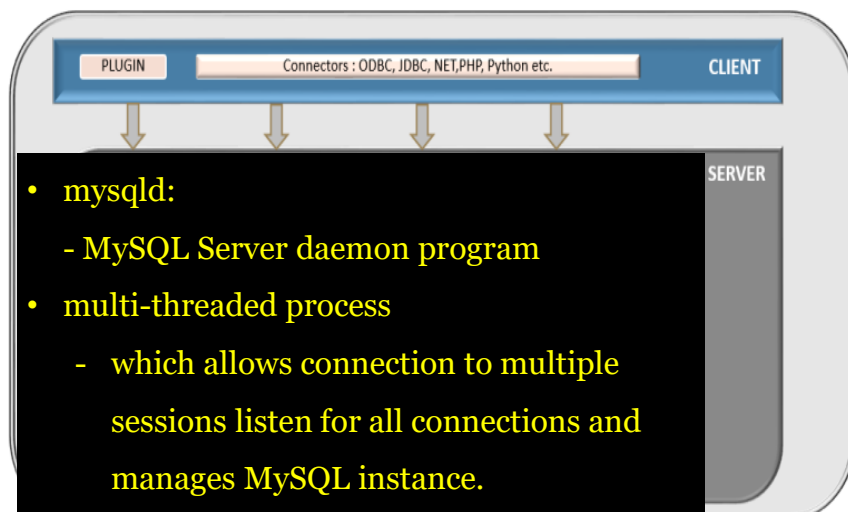


2023-09-04

컴퓨터공학과

89

## MySQL Logical Architecture:

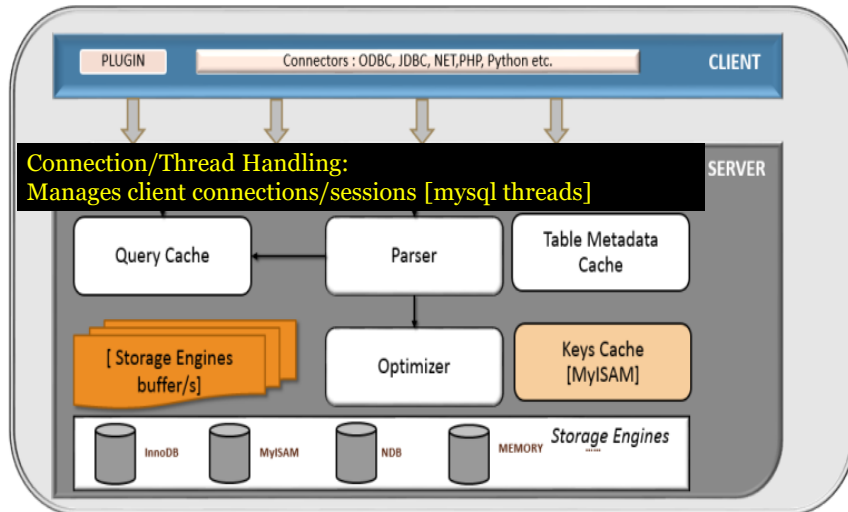


2023-09-04

컴퓨터공학과

90

## MySQL Logical Architecture:

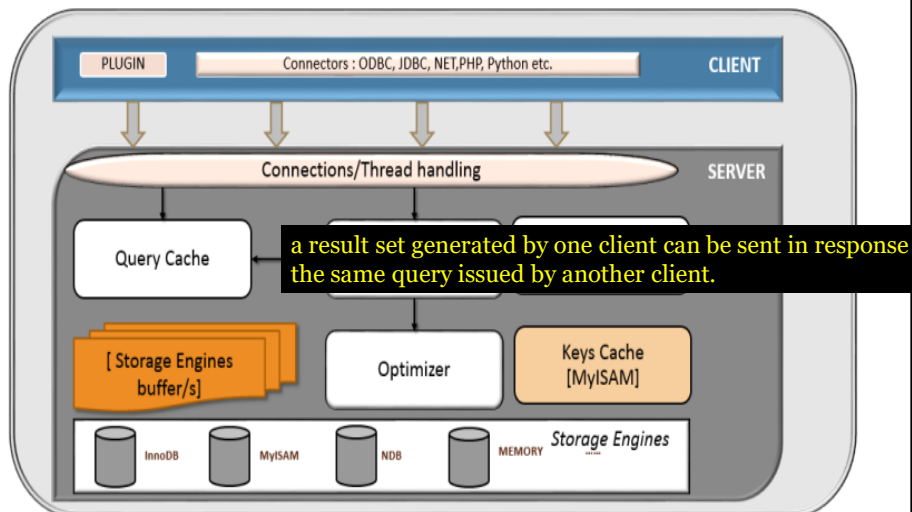


2023-09-04

컴퓨터공학과

91

## MySQL Logical Architecture:

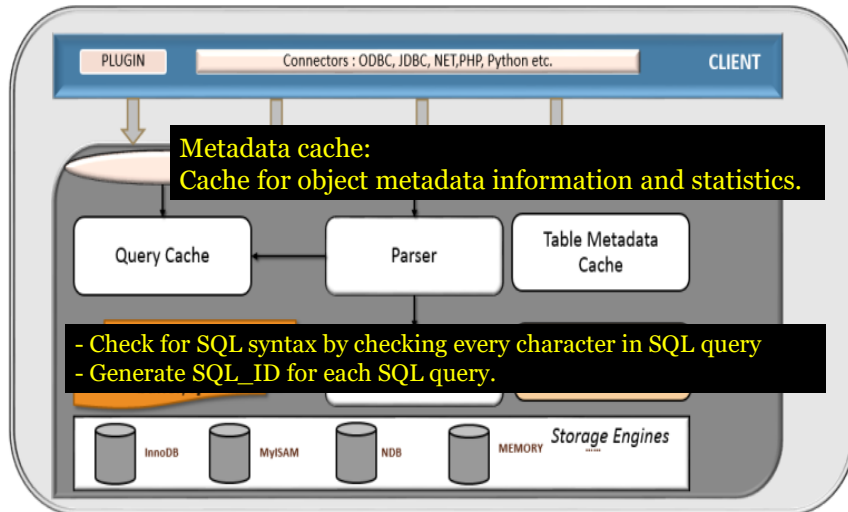


2023-09-04

컴퓨터공학과

92

## MySQL Logical Architecture:

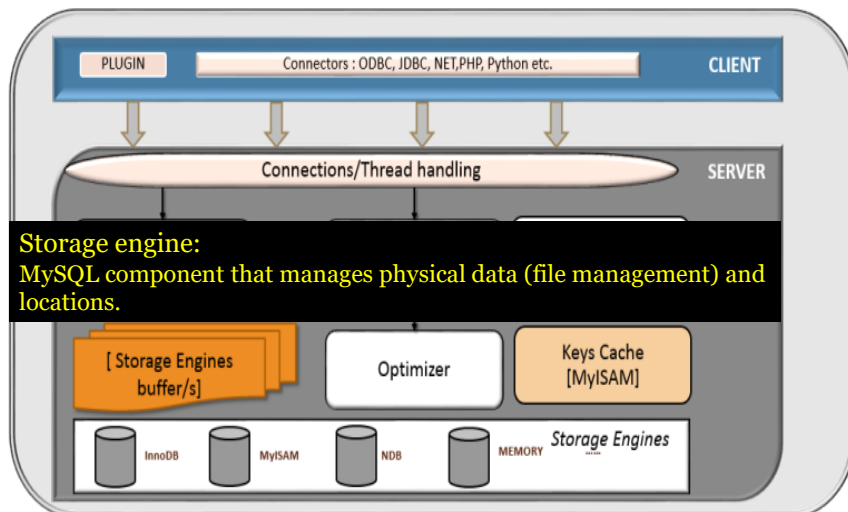


2023-09-04

컴퓨터공학과

93

## MySQL Logical Architecture:

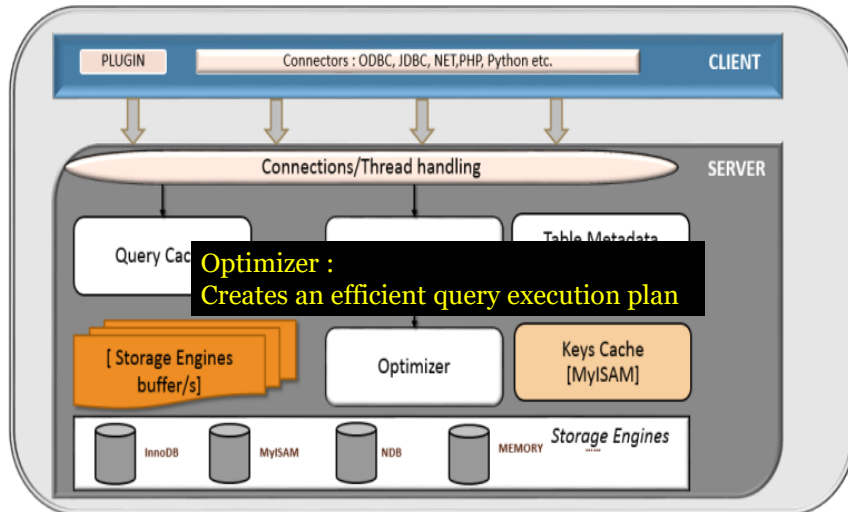


2023-09-04

컴퓨터공학과

94

## MySQL Logical Architecture:

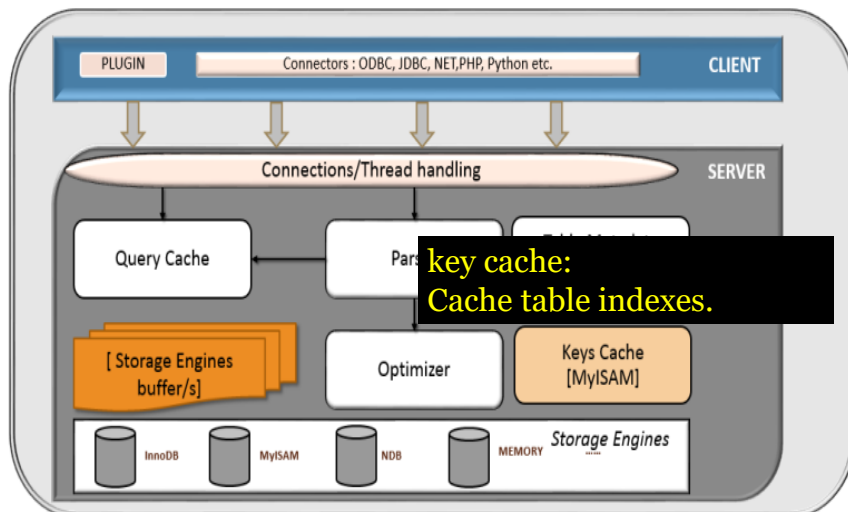


2023-09-04

컴퓨터공학과

95

## MySQL Logical Architecture:



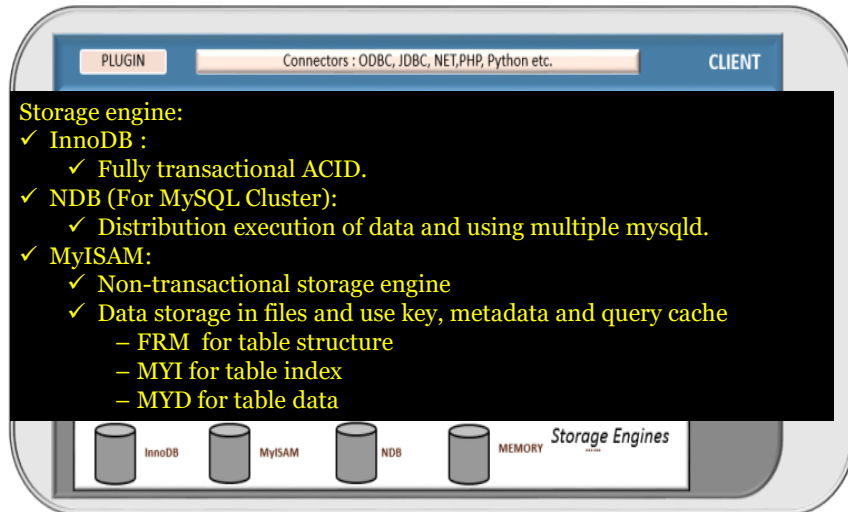
2023-09-04

컴퓨터공학과

96



## MySQL Logical Architecture:

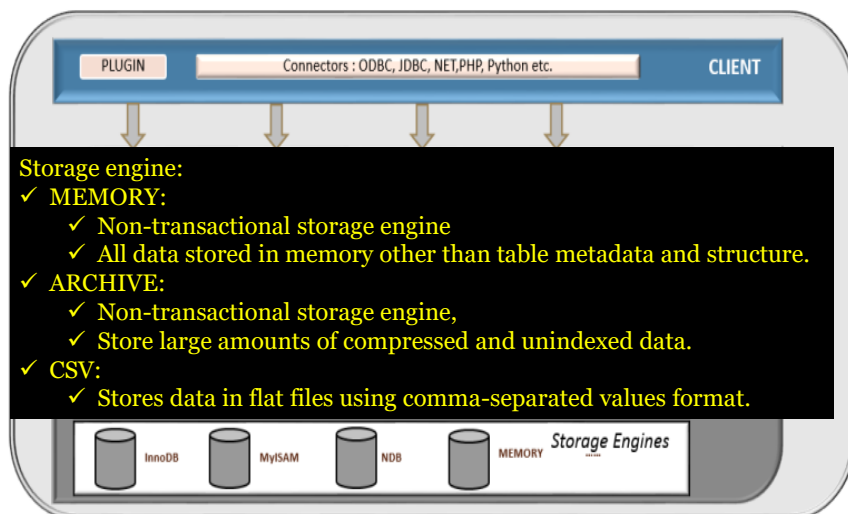


2023-09-04

컴퓨터공학과

97

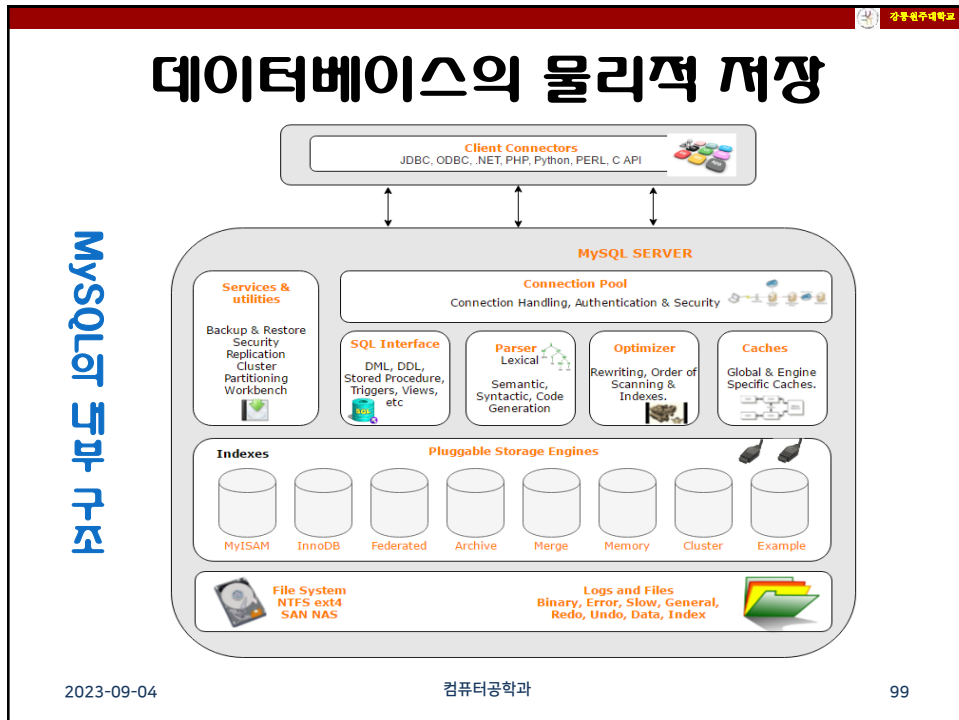
## MySQL Logical Architecture:



2023-09-04

컴퓨터공학과

98



## MySQL InnoDB 저장 DB 파일

MySQL 파일 저장위치: `show variables like 'datadir';`

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Variable_name	Value
datadir	C:\ProgramData\MySQL\MySQL Server 8.0\Data\

파일	설명
데이터 파일 (ibdata)	<ul style="list-style-type: none"> <li>• 사용자 데이터와 개체를 저장</li> <li>• 테이블과 인덱스로 구성</li> <li>• 확장자는 *.ibd</li> </ul>
폼파일 (frm File)	<ul style="list-style-type: none"> <li>• 테이블에 대한 각종 정보와 테이블을 구성하는 필드, 데이터 타입에 대한 정보 저장</li> <li>• 데이터베이스 구조 등의 변경사항이 있을 때 자동으로 업데이트 됨</li> </ul>

2023-09-04      컴퓨터공학과      100

# MySQL 파일 저장위치

## • 명령어 사용 확인

– show variables like 'datadir';

The screenshot shows the MySQL command line interface with the command `show variables like 'datadir';` executed. The result is displayed in a table:

Variable_name	Value
datadir	C:\ProgramData\MySQL\MySQL Server 8.0\Data\

Below the command line, a file explorer window shows the contents of the `C:\ProgramData\MySQL\MySQL Server 8.0\Data\` directory. The files listed are:

이름	수정된 날짜	유형	크기
book.ibd	2020-06-11 오전 11:08	IBD 파일	112KB
customer.ibd	2020-06-11 오전 11:08	IBD 파일	112KB
Imported_book.ibd	2020-06-11 오전 11:08	IBD 파일	112KB
orders.ibd	2020-06-11 오전 11:08	IBD 파일	144KB

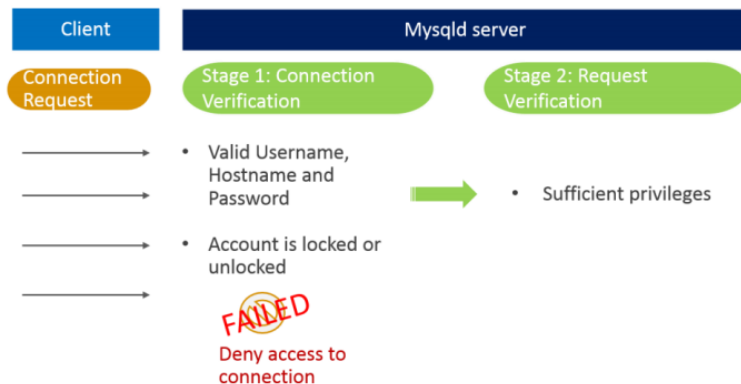
2023-09-04

컴퓨터공학과

101

# MySQL Connection

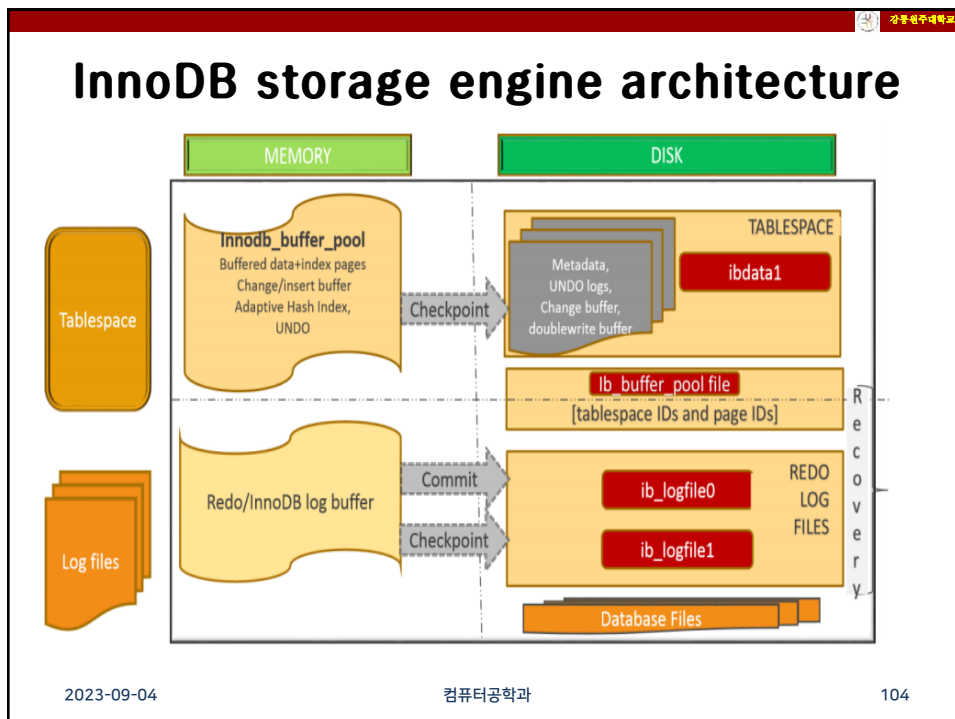
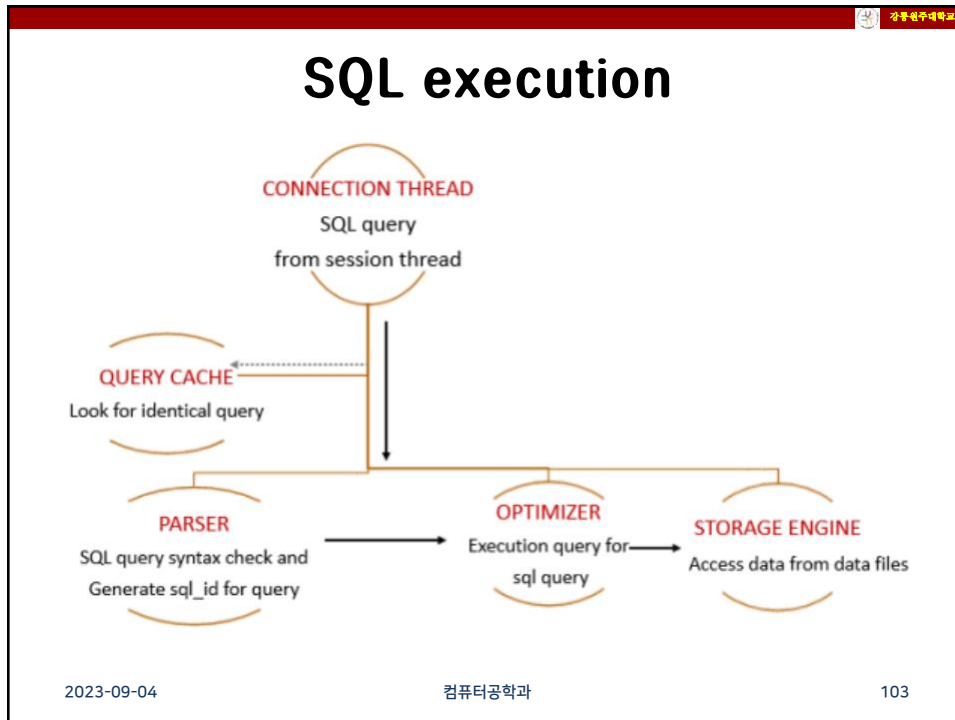
## Client Connection



2023-09-04

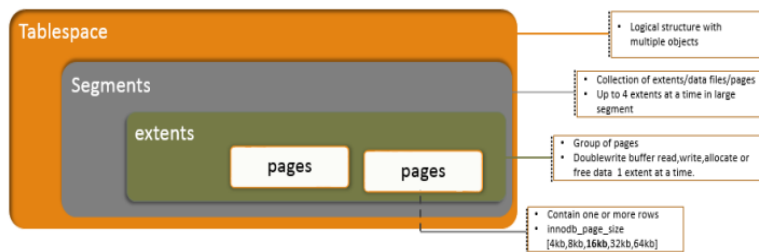
컴퓨터공학과

102



## Storage for InnoDB: Tablespace

- A logical structure associated with multiple data files (objects).
- Each tablespace contains
  - **pages (blocks), extents and segments.**



2023-09-04

컴퓨터공학과

105

## InnoDB components: In Memory

- **InnoDB buffer pool:**
  - Central buffer for InnoDB storage engine.
  - In this buffer, mysql load blocks and cache table index and data.
- **Change buffer:**
  - In a memory change buffer is a part of InnoDB buffer pool and on disk, it is a part of system tablespace.
- **Adaptive Hash Index:**
  - If a table fits almost entirely in main memory, a hash index can speed up queries.
- **Redo log buffer:**
  - Buffer for redo logs, it hold data to be written to the redo log.

2023-09-04

컴퓨터공학과

106

## InnoDB components: On Disk

- **System tablespace:**
  - contains several types of information for InnoDB objects.
- **General tablespace:**
  - Shared tablespace to store multiple table data.
- **File-Per-Table Tablespaces:**
  - A single-table tablespace that is created in its own data file in database directory.
- **InnoDB data dictionary:**
  - Storage area with metadata information for objects[tables, index, columns etc.].

2023-09-04

컴퓨터공학과

107

## InnoDB components: On Disk

- **Double write buffer:**
  - InnoDB writes pages flushed from InnoDB buffer pool, before writing to their proper location in the data files.
- **REDO logs:**
  - Used during crash recovery.
- **Temporary tablespace:**
  - Storage to keep and retrieve modified uncommitted data for temporary tables and related objects.

2023-09-04

컴퓨터공학과

108

# Database and Instance

- Database

- A set of files, located on disk, that store data.

- Database instance

- a set of memory structures that manage database files.

## 데이터의 저장과 관리

- 데이터 읽고 저장 시 속도문제 발생

- 컴퓨터시스템 처리속도 > 디스크 액세스 속도
  - 하드디스크보다 주기억장치 1000배 빠름

- 해결

- 버퍼 풀(buffer pool) 메모리
    - 주기억장치에 DBMS가 사용하는 공간
  - InnoDB buffer pool 일부를 DB Buffer Cache로 사용

- DB Buffer Cache

- 자주 사용하는 데이터 저장
  - LRU(Least Recently Used) 알고리즘 사용
    - 사용빈도가 높은 데이터 저장 관리

## InnoDB Buffer Pool

- InnoDB의 메모리 속 데이터 구조를 저장하는 메모리 공간
- 저장 데이터 구조
  - 액세스 할 테이블 및 인덱스 데이터를 캐시
- 버퍼 풀 LRU 알고리즘
  - LRU 알고리즘 변형을 사용하여 목록으로 관리
  - 새 페이지 추가
    - 공간이 필요한 경우 가장 최근에 사용된 페이지가 제거 되고 새 페이지가 목록 중간에 추가됨

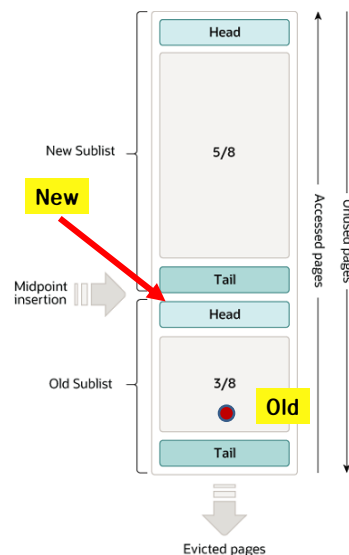
2023-09-04

컴퓨터공학과

111

## InnoDB Buffer Pool List

- 중간점 삽입 전략 사용
  - midpoint insertion strategy
- 두 개 아워 목록
  - New Sublist
    - 최근 액세스한 새로운 페이지 목록
  - Old Sublist
    - 최근에 액세스하지 않은 이전 페이지 목록

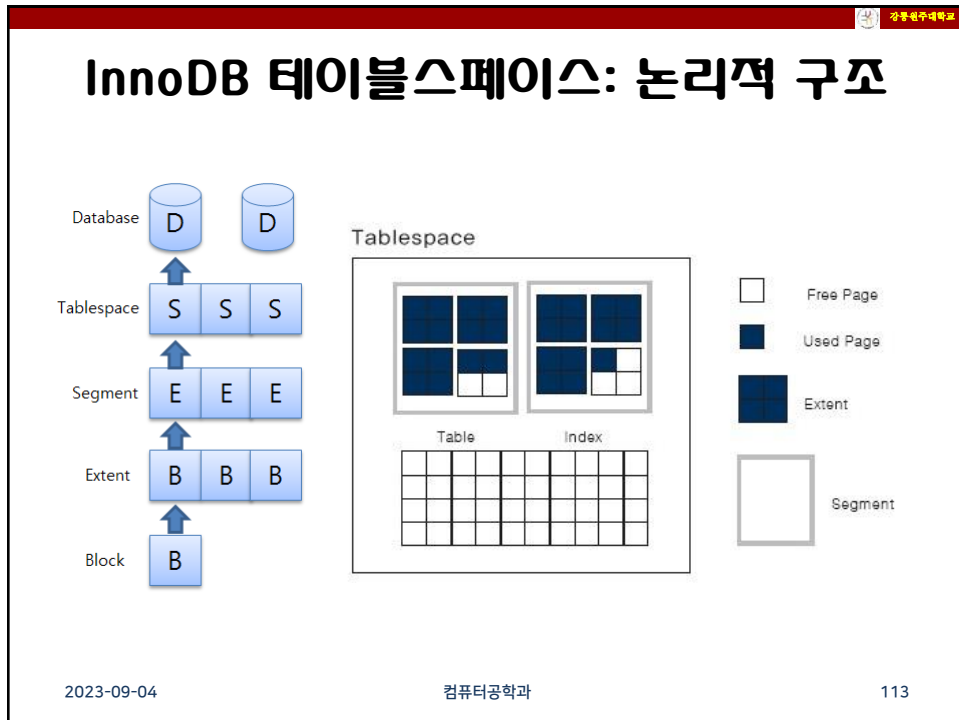


2023-09-04

컴퓨터공학과

112





## InnoDB 테이블스페이스: 접근 방법

- 물리적 블록 단위 접근: 보통 1블록 4KB
- 논리적 블록 단위
  - 4, 8, 16, 32, 64KByte
- 소프트웨어 단계: called Page
- 주기억장치에 Internal Index 저장

2023-09-04      컴퓨터공학과      114

## 저장구조 예

- 블록: 8KB
- 1익스텐트: 8개 블록 구성
- 테이블 스페이스 1MB
  - 16개 익스텐트 할당
  - $8KB \times 8 \times 16 = 1024KB(1MB)$

## 사 례

- 데이터베이스의 물리적 저장
- 인덱스와 B-tree
- MySQL 인덱스
- 인덱스의 생성
- 인덱스의 재구성과 삭제

# 인덱스란?

KING을 어떻게 찾을까요?

JAMES	ALLEN	JOHN	BLAKE
9	10	11	12
TURNER	KING	WORD	CLARK
5	6	7	8
SMITH	FORD	MILLER	JAMES
1	2	3	4

Black Board

INDEX

ALLEN - 10
BLAKE - 12
CLARK - 8
FORD - 2
HANES - 4
JAMES - 4
JAMES - 9
JOHN - 11
KING - 6
MILLER - 3
SMITH - 1
TURNER - 5
WORD - 7

2023-09-04
컴퓨터공학과
117

# 인덱스란?

USER

MySQL

InnoDB  
Buffer  
Cache

DISK

2023-09-04
컴퓨터공학과
118

# Index

- **Schema objects**
  - Speed up access to table rows.
- **Index-organized tables**
  - Tables stored in an index structure.
  - Ordered by indexing field (search key)
  - Attribute(s) used to look up records in a file
  - An index file consists of index entries
    - Records of the form

Search Key

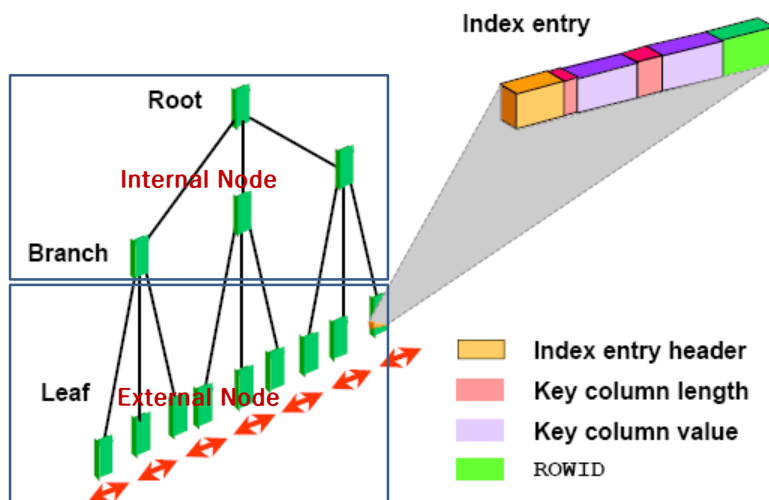
Record locations

2023-09-04

컴퓨터공학과

119

## B-Tree Index



2023-09-04

컴퓨터공학과

120

## 인덱스와 B-tree

- 인덱스(index, 색인)
  - 도서의 색인이나 사진과 같이 데이터를 쉽고 빠르게 찾을 수 있도록 만든 데이터 구조
  - 튜플의 키 값에 대한 물리적 위치 기록
- B-tree
  - RDBMS의 인덱스 구조
  - 데이터베이스와 파일 시스템에서 널리 사용되는 트리 자료구조의 일종

2023-09-04

컴퓨터공학과

121

## B 트리 정의

- Order m인 B tree
  - 모든 노드에는 최대 m개의 자식
  - 모든 내부 노드에는 최소  $\left\lceil \frac{m}{2} \right\rceil$ 개 자식
  - 모든 내부 노드에는 최소 두 개의 자식
  - 모든 리프 노드는 같은 깊이
  - k개의 자식이 있는 잎이 아닌 노드는 k-1개 키
- Order(최대 차수) 5인 B tree에서
  - 5-방향 탐색 트리(5-way search tree)
  - Order는 트리 자식 중 최대 차수
  - 키의 수는 4개, 자식의 최대 수는 5개

2023-09-04

컴퓨터공학과

122

## B 트리 정의

### • 문제1

- 원소 1~10까지를 위한 order 3인 B tree를 구성 하시오.

### • 문제2

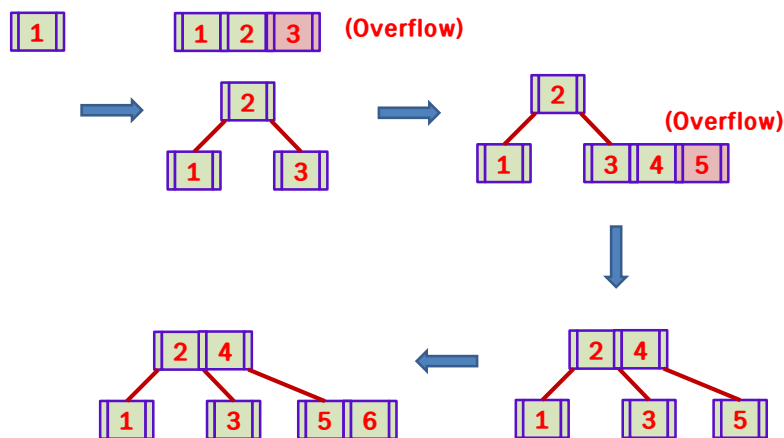
- Order가 4이고 높이가 3인 B tree가 가질 수 있는 최대 키의 수는?
- 키의 수는  $4-1=3$ 개
- $3+3*4+12*4+48*4 = 255$

2023-09-04

컴퓨터공학과

123

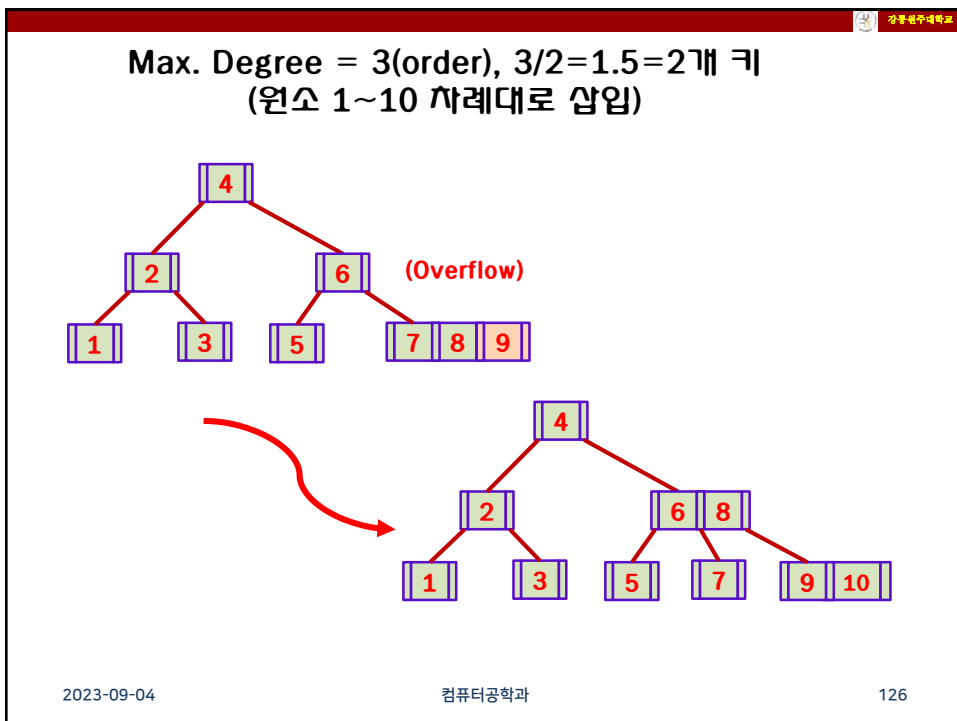
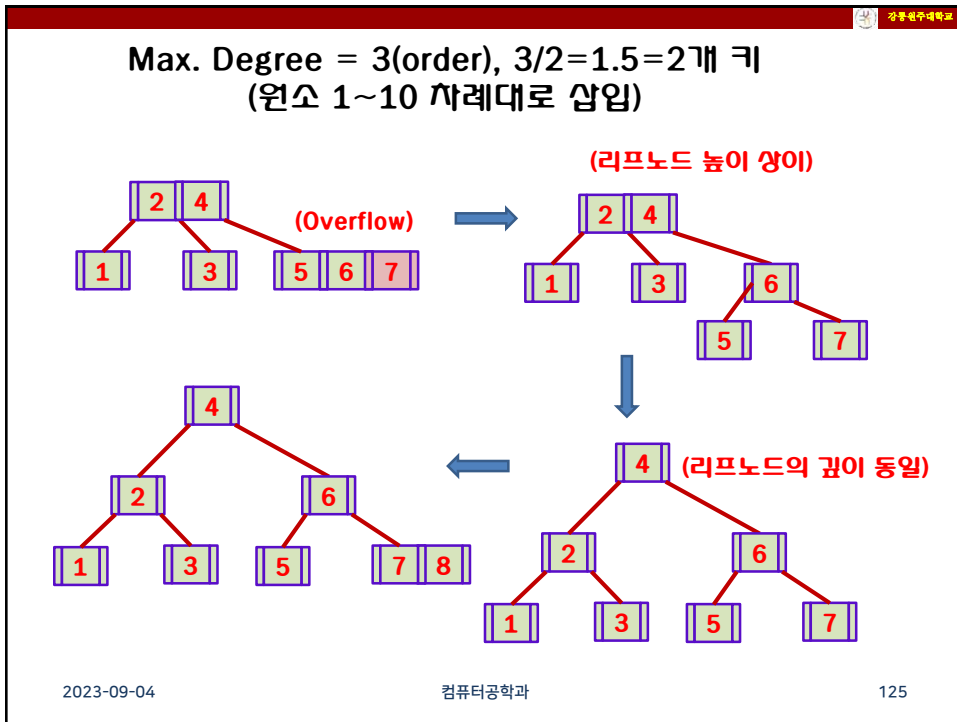
Max. Degree = 3(order),  $3/2=1.5=2$ 개 키  
(원소 1~10 차례대로 삽입)



2023-09-04

컴퓨터공학과

124



## 연습문제:

- 다음과 같은 키 값을 갖는 최대 차수가 3인 B-tree를 구성 하시오.

1   5   7   11   13   15   17   19

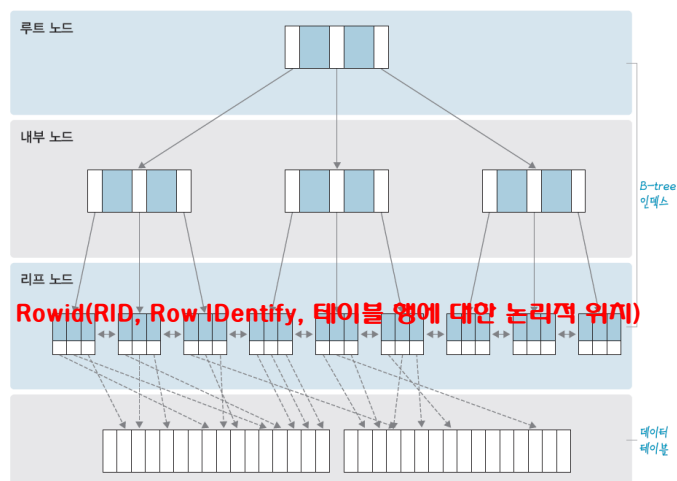
**Hidden Answer**

2023-09-04

컴퓨터공학과

127

## B-tree의 구조



2023-09-04

컴퓨터공학과

128



## B-tree의 구조

- 목적: 데이터 검색시간 단축 자료구조
- Rudolf Bayer 고안
- 구성
  - Root Node
  - Internal Node
  - Leaf Node
    - 균형트리(Balanced Tree)
- 각 노드는 키 값과 포인터 가짐
  - 키 값은 오름차순 저장

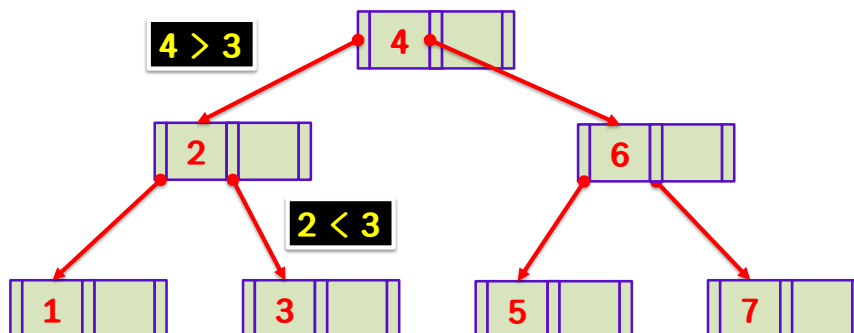
2023-09-04

컴퓨터공학과

129

## B-tree 검색 예

- 3이란 값 검색 시(2개의 키와 3개 자식)



2023-09-04

컴퓨터공학과

130

## 인덱스의 특징

- 테이블에서 한 개 이상의 속성을 이용, 생성
- 빠른 검색과 효율적인 레코드 접근이 가능
- 순서대로 정렬된 속성과 데이터의 위치만 보유하므로
  - 테이블보다 작은 공간을 차지
- 저장된 값들은 테이블의 부분집합
- 일반적으로 B-tree 형태의 구조
- 데이터의 수정, 삭제 등의 변경이 발생하면
  - 인덱스의 재구성이 필요

2023-09-04

컴퓨터공학과

131

## B Tree vs. B+ Tree

- B Tree
  - 내부 및 리프 노드 모두에 키와 레코드 저장
  - 탐색 키들이 반복 저장 안됨
  - 2종류 노드에 데이터가 있어 탐색속도 느림
- B<sup>+</sup> Tree
  - B 트리의 확장으로 중복 키들 가능
  - 내부노드는 키 저장/리프노드에 데이터 저장
    - 탐색속도 빠름
  - 리프노드는 연결리스트를 사용

2023-09-04

컴퓨터공학과

132

## B+ Tree 예

다음 B+트리에서 키 값 45를 탐색할 때

K=45 발견

2023-09-04
컴퓨터공학과
133

## 사 례

- 데이터베이스의 물리적 저장
- 인덱스와 B-tree
- MySQL 인덱스
- 인덱스의 생성
- 인덱스의 재구성과 삭제

2023-09-04
컴퓨터공학과
134

# MySQL 인덱스

- B-tree 인덱스 기본 사용
- B-trees, short for **balanced trees**
- two types of **blocks(=nodes)**
  - branch blocks for searching
  - leaf blocks that store values.
    - 연속된 키 값의 레코드에 대한 rowid 저장
    - rowid: row identity, 레코드 식별자
- RID에 의해 실제 데이터의 저장위치 찾을

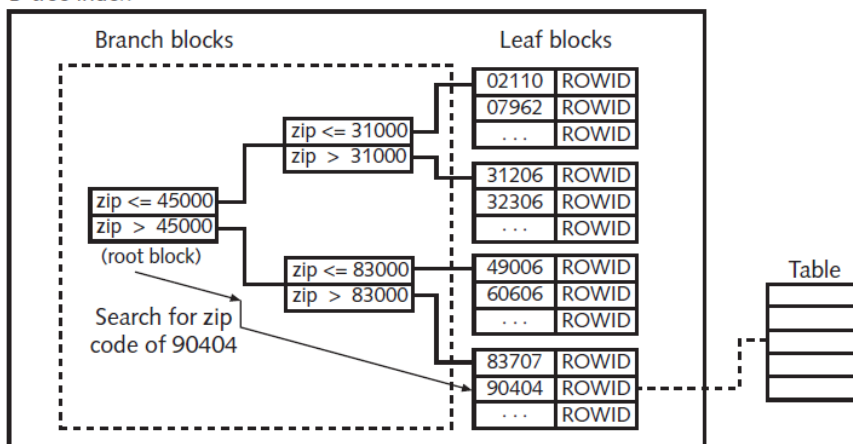
2023-09-04

컴퓨터공학과

135

# MySQL 인덱스

B-tree index



2023-09-04

컴퓨터공학과

136

## MySQL 인덱스의 종류

- 클러스터 인덱스
  - **PRIMARY\_KEY**(Clustered index)
  - 영어사전
- 보조인덱스
  - **UNIQUE**(Secondary Index)
  - 찾아보기

2023-09-04

컴퓨터공학과

137

## MySQL 인덱스의 종류: 클러스터 인덱스

- 기본적인 인덱스
- 테이블 생성 시 기본키를 지정하면,
  - 기본키에 대하여 클러스터 인덱스를 생성
- 기본키를 지정하지 않으면,
  - 먼저 나오는 UNIQUE 속성: 클러스터 인덱스 생성
- 기본키나 UNIQUE 속성이 없는 테이블
  - MySQL 이 자체 생성한 행번호(Row ID)를 이용
  - 클러스터 인덱스 생성

2023-09-04

컴퓨터공학과

138

# MySQL 인덱스: 클러스터 인덱스

## Book 테이블에 클러스터인덱스 생성한 경우

도서번호 8번을 찾는 경우

루트 노드

key	node
1	1
4	2
7	3
10	4

리프 노드

Page 1

1	축구의 역사	굿스포츠	7000
2	2030 축구하는 여자	나무수	13000
3	축구의 이해	대한미디어	22000

Page 2

4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별 기술	굿스포츠	6000

Page 3

7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500

Page 4

10	Olympic Champions	Pearson	13000
empty			
empty			

2023-09-04

컴퓨터공학과

139

## MySQL 인덱스의 종류: 보조 인덱스

- 클러스터 인덱스가 아닌 모든 인덱스
- 각 레코드는 보조 인덱스 속성과 기본키 속성 값을 갖고 있음
- 보조 인덱스를 검색하여 기본키 속성 값을 찾은 다음
  - 클러스터 인덱스로 가서 해당 레코드를 찾음

2023-09-04

컴퓨터공학과

140

# MySQL 인덱스: 보조인덱스

마당서점 Book 테이블  
Bookid 속성 인덱스

INDEX

TABLE

도서번호 8번을  
찾는 경우

루트 노드

key	node
1	1
6	2

리프 노드

Index Block 1

1	1-3
2	3-3
3	3-2
4	2-1
5	4-1

Index Block 2

6	2-3
7	1-2
8	3-1
9	4-2
10	1-1

Block 1

10	Olympic Champions	Pearson	13000
7	아구의 추억	이상미디어	20000
1	축구의 역사	굿스포츠	7000

Block 2

4	골프 바이블	대한미디어	35000
empty			
6	역도 단계별 기술	굿스포츠	6000

Block 3

8	아구를 부탁해	이상미디어	13000
3	축구의 이해	대한미디어	22000
2	축구 아는 여자	나무수	13000

Block 4

5	피겨 교본	굿스포츠	8000
9	올림픽 이야기	삼성당	7500
empty			

리프노드: rowid 저장  
rowid: 테이블 상의 데이터 위치 지정  
<Block번호-Block내의 행 순번>

2023-09-04

컴퓨터공학과

141

# MySQL 인덱스 사용 검색

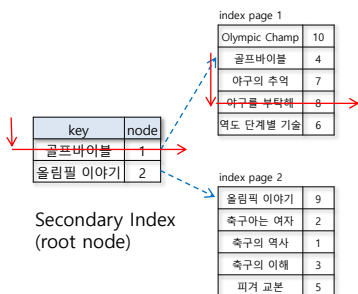
Bookname '아구를 부탁해' 찾는 과정

- 클러스터 인덱스/보조 인덱스 동시 사용

Secondary  
INDEX

Clustered  
INDEX

TABLE



Leaf Nodes

key	page
1	1
4	2
7	3
10	4

Clustered Index  
(root node)

BOOK

page 1

1	축구의 역사	굿스포츠	7000
2	축구 아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000

page 2

4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별 기술	굿스포츠	6000

page 3

7	아구의 추억	이상미디어	20000
8	아구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500

page 4

10	Olympic Champions	Pearson	13000
empty			
empty			

2023-09-04

컴퓨터공학과

142

## 차 례

- 데이터베이스의 물리적 저장
- 인덱스와 B-tree
- MySQL 인덱스
- 인덱스의 생성
- 인덱스의 재구성과 삭제

## 인덱스 생성 시 고려사항

- WHERE 절에 자주 사용되는 속성
- 조인에 자주 사용되는 속성
- 단일 테이블에 인덱스가 많으면
  - 속도가 느려질 수 있음
- 속성이 가공되는 경우 사용하지 않음
- 속성의 선택도가 낮을 때 유리함
  - 속성의 모든 값이 다른 경우



## 인덱스의 생성 문법 및 예

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

**질의 4-24** Book 테이블의 bookname 열을 대상으로 인덱스 ix\_Book을 생성하라.

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

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

2023-09-04

컴퓨터공학과

145

## 인덱스의 생성 문법 및 예

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

**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

2023-09-04

컴퓨터공학과

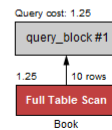
146

## 실행 계획(Execution Plan): 인덱스 사용 확인

**show index from book;**  
**계획 설명: [Query] – [Explain Current Statement]**

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

Visual Explain | Display Info: Read + Eval cost | Overview: | View Sources:



2023-09-04

컴퓨터공학과

147

## 사 례

- 데이터베이스의 물리적 저장
- 인덱스와 B-tree
- MySQL 인덱스
- 인덱스의 생성
- 인덱스의 재구성과 삭제

2023-09-04

컴퓨터공학과

148

## 인덱스의 재구성

- 인덱스 재구성
  - ANALYZE TABLE 명령 사용
- 생성 문법
 

**ANALYZE TABLE** 테이블이름;
- B-tree 인덱스 노드 갱신 빈번
  - 단편화(fragmentation) 발생
    - 삭제된 레코드의 인덱스 값 비어 있는 현상
    - 성능저하 야기
    - 인덱스 재구성하여 조각화 최소화

2023-09-04

컴퓨터공학과

149

## 인덱스의 재구성

- analyze table book;

필의 4-26 Book 테이블의 인덱스를 최적화 하시오.

**analyze table** book;

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
Table	Op	Msg_type	Msg_text
▶ madang.book	analyze	status	OK

2023-09-04

컴퓨터공학과

150

## 인덱스의 삭제

- 인덱스 삭제
  - DROP INDEX 명령 사용
- 생성 문법
  - DROP INDEX 인덱스이름;
- 예

질의 4-27 인덱스 ix\_Book을 삭제하십시오.

```
DROP INDEX ix_Book;
```

## 요약

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