



Chap 04 데이터베이스_4.5 인덱스

4.5 인덱스

4.5.1 인덱스의 필요성

✓ 인덱스

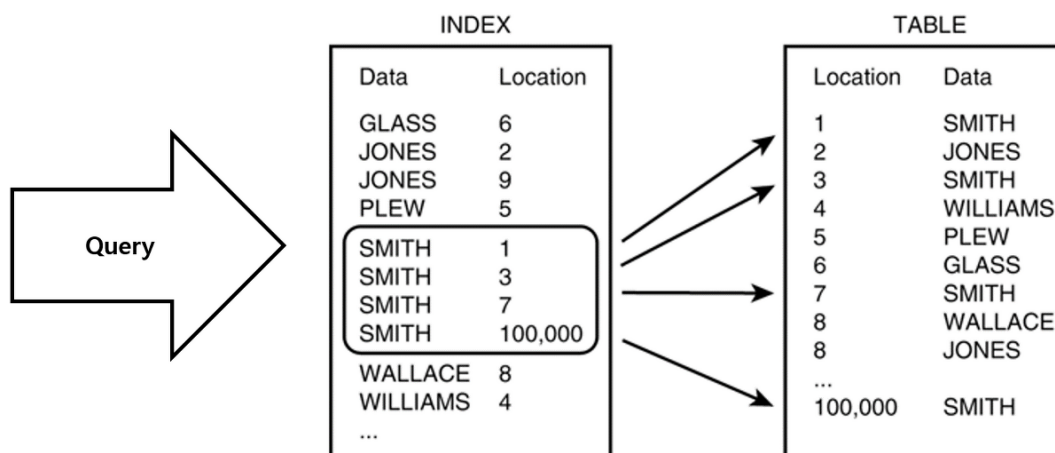
▪ 인덱스란?

데이터를 빠르게 찾을 수 있는 하나의 장치

→ 인덱스를 설정하면 테이블 안에 내가 찾고자 하는 데이터를 빠르게 찾을 수 있음

인덱스(Index) 란 데이터베이스 테이블의 검색 속도를 향상시키기 위한 자료구조

▪인덱스의 원리



인덱스 생성 → 데이터 = key, 데이터에 대한 위치 정보 = value

▪인덱스 관리

- DBMS는 인덱스를 항상 최신의 정렬된 상태로 유지해야 함
- CRUD가 빈번하게 일어나는 DB라면 그만큼 인덱스를 관리하는 동작이 필요하며 그에 따른 오버헤드 발생
 - insert : 새로운 데이터에 따른 인덱스를 추가

- update : 수정하고자 하는 데이터에 인덱스를 "사용하지 않음" 처리하고, 새로운 인덱스 추가
- delete : 삭제하는 데이터의 인덱스를 "사용하지 않음" 처리

*update, delete 연산은 해당 인덱스를 삭제하지 않고 "사용하지 않음"처리 → 완전 삭제 x

■인덱스 장점

- 검색 속도가 빨라짐
- 적은 처리량으로 결과 얻을 수 있음
 - 쿼리 부하 줄어듦
 - 다른 요청 처리 가능
 - 전체 시스템 성능 올라감

■인덱스 단점

- DB 공간이 추가로 필요함(10%)
- 처음 생성시 시간이 많이 소요될 수 있음
- 데이터 변경작업이 자주 일어나게 되면 성능 나빠짐

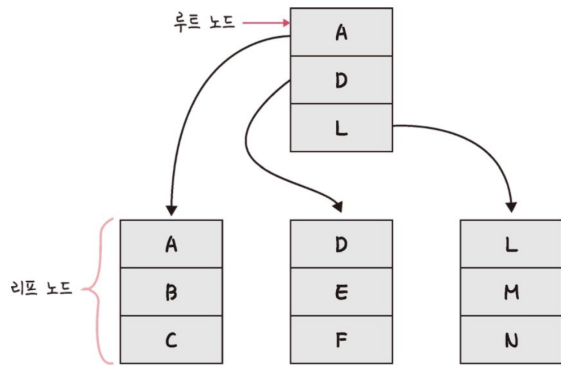
■ 인덱스 사용 예시

- 규모가 작지 않은 테이블
- CRUD 연산이 빈번하지 않은 테이블
- 데이터의 중복도가 낮은 테이블
- join 이나 where 또는 order by 연산이 자주 발생하는 테이블

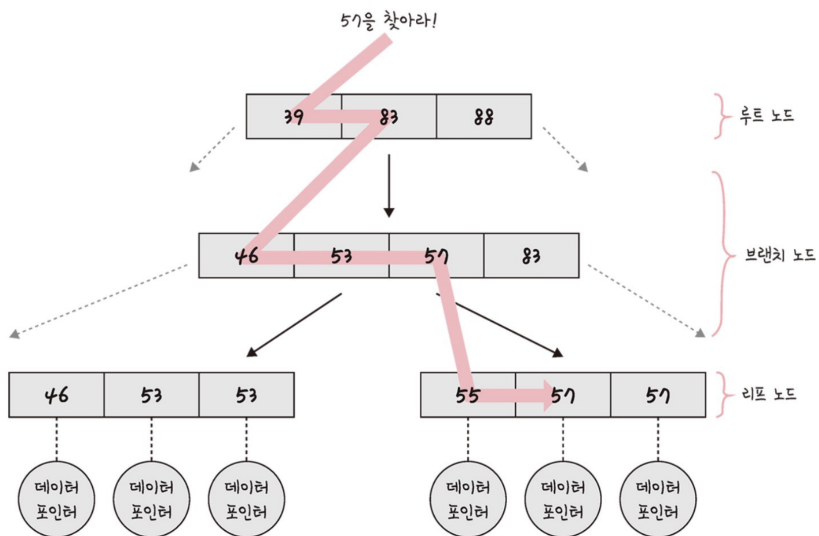
4.5.2 B-트리

<https://www.youtube.com/watch?v=ywYdEIs88Sw>

인덱스는 보통 B-트리 라는 자료구조로 이루어짐



▲ 그림 4-36 B-트리 예제 1



✅ 인덱스가 효율적인 이유, 대수 확장성

- 균형 잡힌 트리 구조 + 트리 깊이의 대수 확장성
- 대수 확장성 = 트리 깊이가 리프 노드 수에 비해 매우 느리게 성장하는 것
→ 인덱스가 한 깊이 씩 증가할 때마다 최대 인덱스 항목의 수는 4배씩 증가

▼ 표 4-3 트리의 대수확장성

트리 깊이	인덱스 항목의 수
3	64
4	256
5	1,024
6	4,096
7	16,384
8	65,536
9	262,144
10	1,048,576

*검색 특화

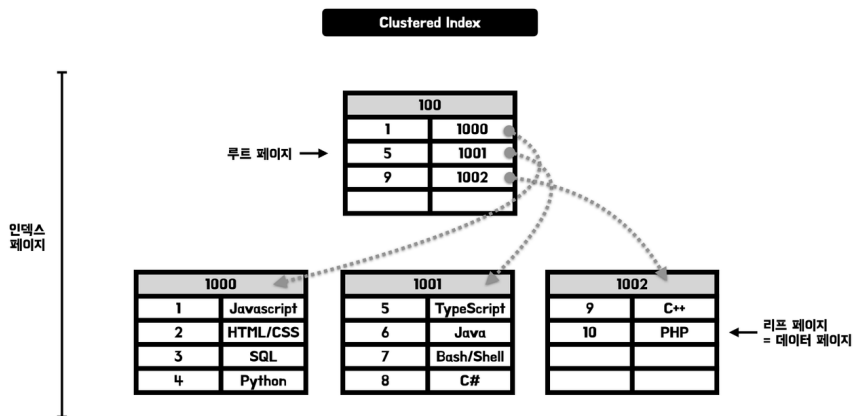
4.5.3 인덱스를 만드는 방법

✓ MySQL

■인덱스 종류

클러스터형 인덱스 / 세컨더리 인덱스

■클러스터형 인덱스 (Cluster Index)

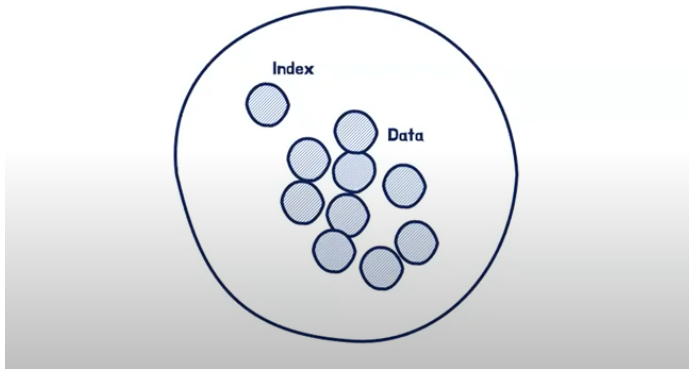


<https://hudi.blog>

Clustered Index

실제 데이터와 군집(강한 연관)을 이루는 인덱스

데이터가 테이블에 물리적으로 저장되는 순서를 정의




1. 정의 : 테이블 전체가 정렬된 인덱스가 되는 방식

⇒ 테이블 레코드들이 인덱스 컬럼의 정렬 순서대로 적재되어 있는 것

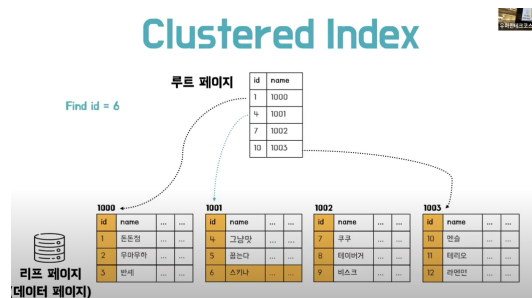
2. 특징

- 데이터와 함께 전체 테이블이 물리적으로 정렬됨
- 테이블당 하나만 생성 가능
- 레코드와 물리적 순서가 인덱스의 엔트리 순서와 일치하게끔 유지
- 데이터가 많고, select문 수행이 많은 경우 효과적
 - 데이터의 변경이 잦은 컬럼의 경우, 데이터 변경때마다 데이터 정렬 연산 이루어짐 → DB 과부하
- join 조건에 사용된느 경우 클러스터형 인덱스를 활용하면 효율적
- 테이블 생성시 primary key 조건 사용시 고유 인덱스 자동 생성
 - 기본적으로 인덱스가 클러스터형 인덱스

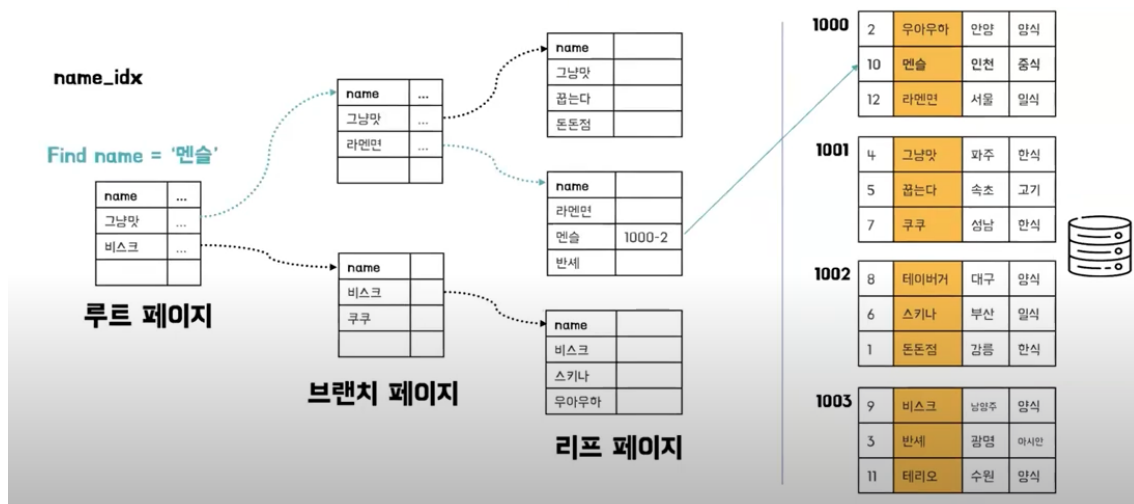
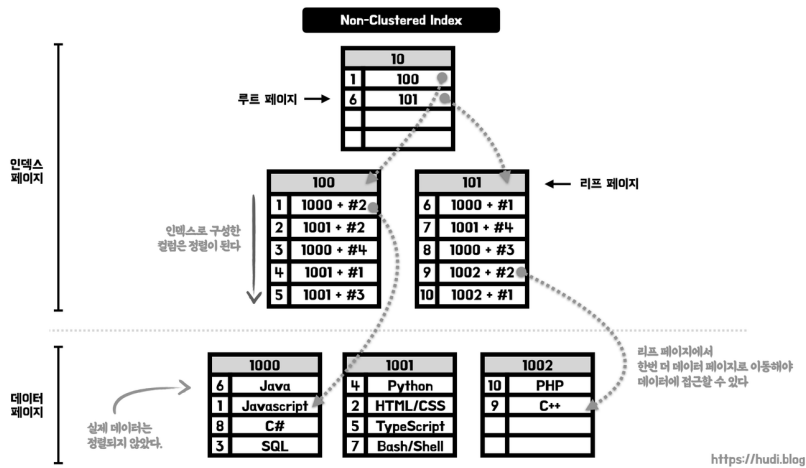
[예시]



id	name	address	category
2	무아무아	안암	양식
10	멘슬	인천	중식
12	라멘면	서울	일식
4	그냥맛	파주	한식
3	반세	광명	아시아
5	끓는다	속초	고기
7	쿠쿠	심남	한식
8	테이버거	대구	양식
6	스키나	부산	일식
1	돈돈점	강릉	한식
9	비스크	남양주	양식
11	테리오	수원	양식



■세컨더리 인덱스 = 비클러스터형 인덱스



1. 정의 : 클러스터형 인덱스와 다르게 물리적으로 테이블을 정렬하지 않음

2. 특징

- 여러 개의 필드 값을 기반으로 쿼리를 많이 보낼 때 생성
- 정렬된 별도의 인덱스 페이지를 생성하고 관리
= 실제 데이터를 함께 가지 있지 않음
- 테이블 당 여러개 생성 가능
- Primary key 이외에 필요한 정렬 기준이 있을 경우 사용
- 인덱스 페이지와 데이터 페이지가 구분 되어 있음
- Data Record가 정렬되어 있지 않음
- unique하지 않아도 가능

▪ 클러스터 인덱스 vs 세컨더리 인덱스

상황) 범위로 검색하는 경우 성능차이

- 클러스터형 인덱스
생성 시 정렬이 되어 있어 해당 범위의 리프 페이지만 읽으면 쉽게 데이터 조회 가능
- 세컨더리 클러스터 인덱스
범위에 해당하는 데이터가 서로 다른 데이터 페이지에 존재하여 원하는 데이터를 찾기 위해 더 많은 데이터 페이지를 읽어야 데이터 얻을 수 있음

상황) 데이터 저장의 경우 (insert)

- 클러스터형 인덱스
데이터 삽입을 위해 페이지 분할 일어남
- 세컨더리 클러스터 인덱스
데이터 페이지를 정렬할 필요는 없으므로 데이터 페이지 뒤쪽 빈 부분에 삽입

즉, age란 하나의 필드만으로 쿼리 보낼 때는 클러스터형 인덱스, age/name/email 등 다양한 필드 기반으로 쿼리 보낼 경우는 세컨더리 인덱스 사용

✅ MongoDB

DOCUMENT 만들면 자동으로 objectID 형성됨 → 해당 키가 기본키로 설정

4.5.4 인덱스 최적화 기법

- DB마다 조금씩 다르지만 기본 골조는 같음 → (MongoDB 기준으로 설명)

01. 인덱스는 비용이다

- 인덱스는 두번 탐색하도록 가용
인덱스 리스트, 컬렉션 순으로 탐색 진행 → 관련 읽기 비용 듬
- 컬렉션 수정시 인덱스도 수정되어야함
 - B-트리의 높이를 균형 있게 조절하는 비용
 - 데이터를 효율적으로 조회 할 수 있도록 분산시키는 비용

⇒ 쿼리에 있는 필드에 인덱스를 모두 설정하면 안됨.

⇒ 컬렉션에서 가져와야 하는 양이 많을 경우 인덱스 사용은 비효율적

02. 항상 테스트 하라

- 인덱스 최적화 기법은 서비스 특징에 따라 달라짐 → 항상 테스트 하는 것이 중요

- explain() 함수를 통해 인덱스를 만들고 쿼리를 보낸 이후에 테스트를 하며 걸리는 시간을 최소화

03. 복합 인덱스는 같음, 정렬, 다중 값, 카디널리티 순

- 보통 여러 필드를 기반으로 조회를 할 때 복합 인덱스 생성
- 같음, 정렬, 다중 값, 커널리티 순으로 생성 해야함