인덱스

데이터를 빠르게 찾을 수 있는 하나의 장치

ex) 책의 마지막 장에 있는 찾아보기

인덱스의 필요성

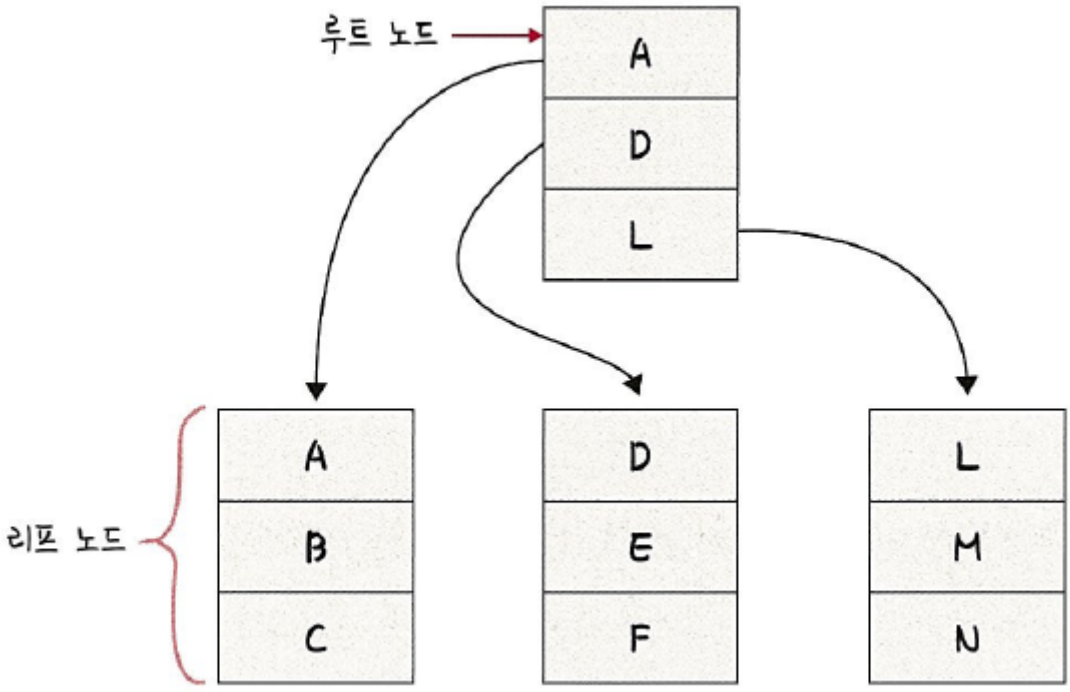
책의 본문이 있고, 그 본문 안에 내가 찾고자 하는 항목을 찾아보기를 통해 빠르게 찾을 수 있음

인덱스를 설정하면 테이블 내에 찾고자 하는 데이터를 빠르게 찾을 수 있음

B-트리

인덱스는 보통 B-트리로 이루어져 있음

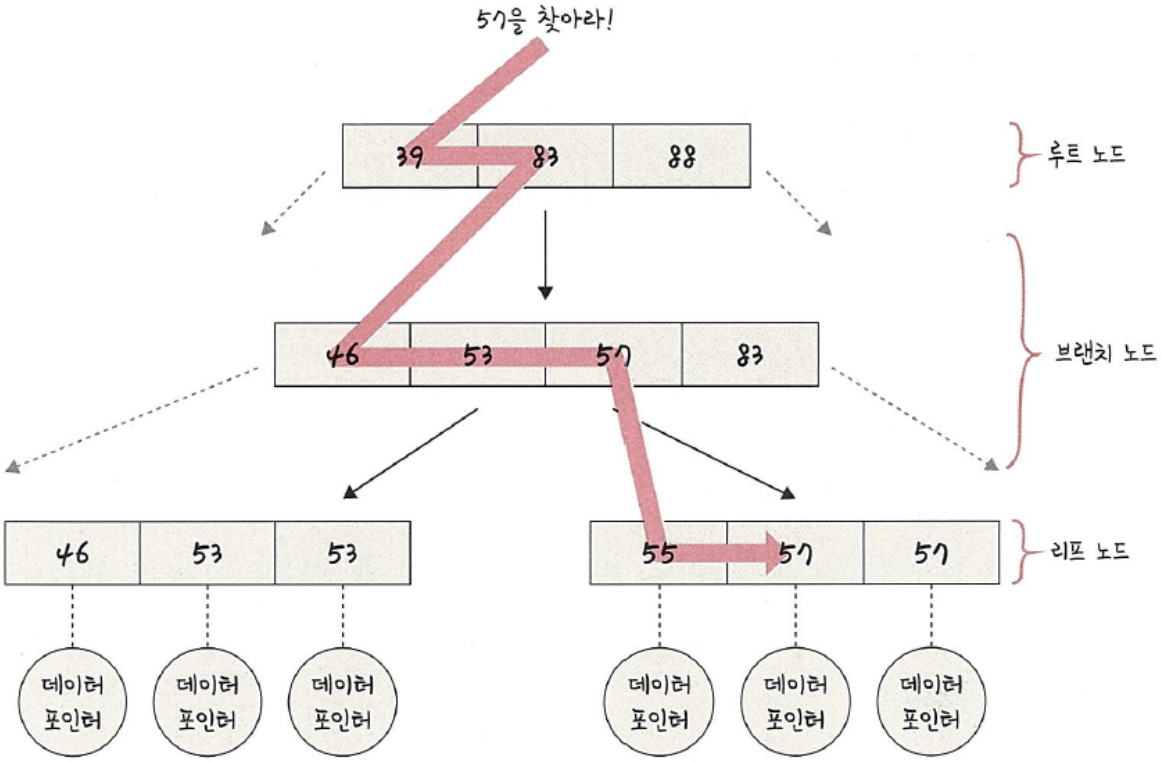
루트 노드, 리프노드, 루트노드와 리프노드 사이에 있는 브랜치 노드로 나뉨



예를 들어 E를 찾는다고 하면 전체 테이블을 탐색하는 것이 아니라, E가 있을법한 리프노드로 들어가서 E를 탐색하면 쉽게 찾을 수 있음

B-트리 구조를 사용하지 않고 E를 탐색하면 A,B,C,D,E 총 다섯 번을 탐색해야함

B-트리 구조를 사용하면 두번 만에 리프노드에서 찾을 수 있음



트리 탐색은 맨 위의 루트 노드부터 탐색이 일어나며, 브랜치 노드를 거쳐 리프 노드까지 내려옴

57보다 크거나 같을때까지를 기반으로 처음 루트 노드에서는 39, 83 이후 아래 노드로 내려와 46, 53, 57 등 정렬된 값을 기반으로 탐색함

루트 노드부터 시작해서 마지막 리프 노드에 도달해서 57이 가리키는 데이터 포인터를 통해 결과값을 반환함

인덱스가 효율적인 이유와 대수확장성

인덱스가 효율적인 이유

효율적인 단계를 거쳐 모든 요소에 접근할 수 있는 균형 잡힌 트리구조와 트리 깊이의 대수 확장성

대수확장성

트리 깊이가 리프 노드 수에 비해 매우 느리게 성장한다는 것

기본적으로 인덱스가 한 깊이씩 증가할때마다 최대 인덱스 항목의 수는 4배씩 증가함

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

실제 인덱스는 위보다 훨씬 더 효율적임

인덱스 생성

MySQL

클러스터형 인덱스와 세컨더리 인덱스가 있음

클러스터형 인덱스

테이블당 하나 설정 가능

primary key 옵션으로 기본키를 만들면 자동으로 생성됨

unique not null 옵션을 사용해도 생ㅅ엉됨

세컨더리 인덱스

create index 명령어를 기반으로 만들면 세컨더리 인덱스를 만들 수 있음

보조 인덱스로 여러개의 필드 값을 기반으로 쿼리를 많이 보낼 때 생성해야함

ex) age 라는 하나의 필드만으로 쿼리를 보낼 경우

클러스터형 인덱스만 필요함

age, name, email 등 다양한 필드를 기반으로 쿼리를 보낼 때는 세컨더리 인덱스 필요

하나의 인덱스만 생성할 것이라면, 클러스터형 인덱스를 만드는 것이 세컨더리 인덱스를 만드는 것보다 성능이 좋음

MongoDB

도큐먼트를 만들면 자동으로 Object ID 가 형성됨

해당 키가 기본키로 설정됨

세컨더리키도 부가적으로 설정해서 기본키와 세컨더리키를 같이 쓰는 복합 인덱스 설정이 가능

인덱스 최적화 기법

DB 마다 조금씩 다르나, 기본적인 골조는 동일함

MongoDB 기준 설명

1. 인덱스는 비용이다

인덱스 리스트, 그다음 컬렉션 순으로 탐색하므로, 인덱스는 두번 탐색하도록 강요함 → 관련 읽기 비용이 들어감

컬렉션이 수정되었을 때 인덱스도 수정되어야함. (책의 본문이 수정 되었을 때, 목차나 찾아보기도 수정해야함)

이때, B-트리의 높이를 균형 있게 조절하는 비용도 들고, 데이터를 효율적으로 조회할 수 있도록 분산시키는 비용도 들게됨

그러므로, 쿼리에 있는 필드에 인덱스를 무작정 다 설정하는 것은 답이 아님

또한, 컬렉션에서 가져와야 하는 양이 많을수록 인덱스를 사용하는 것은 비효율적임

2. 항상 테스팅하라

서비스에서 사용하는 객체의 깊이, 테이블의 양 등이 다르므로, 인덱스 최적화 기법은 서비스 특징에 따라 달라짐 → 항상 테스팅하는 것이 중요함

explain() 함수를 통해 인덱스를 만들고, 쿼리를 보낸 이후에 테스팅을 하며 걸리는 시간을 최소화해야함

3. 복합 인덱스는 같음, 정렬, 다중값, 카디널리티 순이다

보통 여러 필드를 기반으로 조회를 할 때 복합 인덱스를 생성하는데, 이 인덱스를 생성할 때는 순서가 있고 생성 순서에 따라 인덱스 성능이 달라짐

같음, 정렬, 다중 값, 카디널리티 순으로 생성해야함

1. 어떠한 값과 같음을 비교하는 == 이나 equal 이라는 쿼리가 있다면 제일 먼저 인덱스로 설정함
2. 정렬에 쓰는 필드라면 그 다음 인덱스로 설정
3. 다중 값을 출력해야 하는 필드, 즉 쿼리 자체가 > 이거나 < 등 많은 값을 출력해야 하는 쿼리에 쓰는 필드라면 나중에 인덱스를 설정함
4. 유니크한 값의 정도를 카디널리티라고함. 카디널리티가 높은 순서를 기반으로 인덱스를 생성해야함
   ex) age, email이 있을 경우, email 의 카디널리티가 더 높으므로, email 에 대한 인덱스를 먼저 생성해야함