4.5 인덱스

4.5.1 인덱스의 필요성

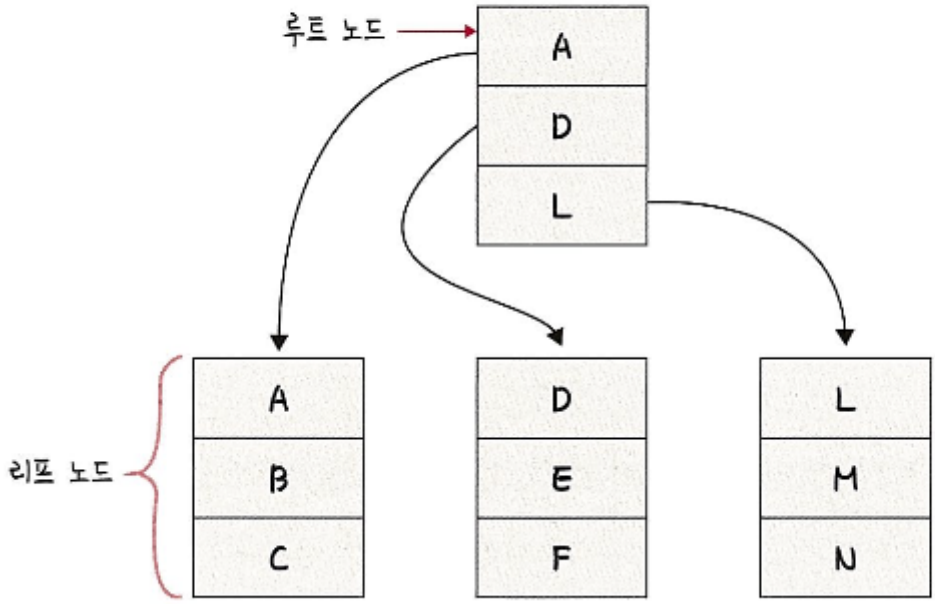
- 인덱스는 데이터를 빠르게 찾을 수 있는 하나의 장치이다.
- 예를 들어 책의 마지막 장에 있는 찾아보기를 생각하면 된다.

4.5.2 B-트리

- 인덱스는 보통 B-트리라는 자료구조로 이루어져 있다.
- 이는 루트 노드, 리프 노드, 루트노드와 리프노드 사이에있는 브랜치 노드로 나뉜다.

먼저 루트 노드와 리프 노드를 기반으로 설명하면 다음과 같습니다.

▼ 그림 4-36 B-트리 예제 1



- E를 찾는다고 하면 전체 테이블을 탐색하는 것이 아니라 E가 있을 법한 리프노드로 들어가서 E를 탐색하면 쉽게 찾을 수 있다.
- 이 자료 구조 없이E를탐색하고자 하면 A,B,C,D,E 다섯번 탐색해야 하지만, 이렇게 노드들로 나누면 두번만에 리프노드에서 찾을 수 있다.

인덱스가 효율적인 이유와 대수확장성

- 인덱스가 효율적인 이유는 효율적인 단계를 거쳐 모든 요소에 접근할 수 있는 균형잡힌 트리 구조와 트리 깊이의 대수확장성 때문이다
- 대수확장성이란 트리 깊이가 리프 노드 수에 비해 매우 느리게 성장하는 것을 의미한다.
- 기본적으로 인덱스가 한 깊이씩 증가할때마다 최대 인덱스 항목의 수는 4배씩 증가한다.

▼ 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

4.5.3 인덱스를 만드는 방법

MySQL

- 클러스트형 인덱스와 세컨더리 인덱스가 있으며 클러스터형 인덱스는 테이블당 하나를 설정할 수 있다.
- pk옵션으로 기본키를 만들면 클러스터형 인덱스를 생성할 수 있고, 기본키로 만들지 않고 unuque not null 옵션을 붙으면 클러스터형 인덱스를 만들 수 있다.
- create index .. 명령어를 기반으로 만들면 세컨더리 인덱스를 만들 수 있다.
- 하나의 인데스만 생성할 것이라면 클러스터형 인덱스를 만드는것이 세컨더리 인덱스를 만드는것보다 성능이 더좋다.

- 세컨더리 인데스는 보조 인덱스로 여러개의 필드 값을 기반으로 쿼리를 많이 보낼 때생성해야하는 인덱스이다.

MongoDB

MongoDB의 경우 도큐먼트를 만들면 자동으로 ObjectID가 형성되며, 해당 키가 기본키로 설정됩니다. 그리고 세컨더리키도 부가적으로 설정해서 기본키와 세컨더리키를 같이 쓰는 복합 인덱스를 설정할 수 있습니다.

4.5.4 인덱스 최적화 기법

1. 인덱스는 비용이다.

- 인덱스는 두 번 탐색하도록 강요된다. 인덱스 리스트, 그다음 컬렉션 순으로 탐색하기 때문에 관련 읽기 비용이 들게 된다.
- 또한 컬렉션이 수정되었을때 인덱스도 수정되어야 한다.
- B-트리의 높이를 균형있게 조절하는 비용도 들고, 데이터를 효율적으로 조회할 수 있도록 분산시키는 비용이 든다.
- 그렇기 때문에 쿼리에 있는 필드에 인덱스를 무작정 다 설정하는 것은 답이 아니다.
- 또한 컬렉션에서 가져와야하는 양이 많을수록 인데스를 사용하는 것은 비효율적이다.

2. 항상 테스팅 하라

- 인덱스 최적화 기법은 서비스 특징에 따라 잘라진다.
- 서비스에서 사용하는 객체의 깊이, 테이블의 양 등이 다르기 떄문이다.
- 그렇기 때문에 항상 테스팅을 해야한다
- explain()함수를 통해 인덱스를 만들고 쿼리를 보낸 이후에 테스팅을 하며 걸리는 시간을 최소화해야한다.
-

```sql
EXPLAIN
SELECT * FROM t1
JOIN t2 ON t1.c1 = t2.c1
```

3. 복합 인덱스는 같음, 정렬, 다중 값, 카디널리티 순이다

보통 여러 필드를 기반으로 조회를 할 때 복합 인덱스를 생성하는데, 이 인덱스를 생성할 때는 순서가 있고 생성 순서에 따라 인덱스 성능이 달라집니다. 같음, 정렬, 다중 값, 카디널리티 순으로 생성해야 합니다.

1. 어떠한 값과 같음을 비교하는 ==이나 equal이라는 쿼리가 있다면 제일 먼저 인덱스로 설정합니다.
2. 정렬에 쓰는 필드라면 그다음 인덱스로 설정합니다.
3. 다중 값을 출력해야 하는 필드, 즉 쿼리 자체가 >이거나 < 등 많은 값을 출력해야 하는 쿼리에 쓰는 필드라면 나중에 인덱스를 설정합니다.
4. 유니크한 값의 정도를 카디널리티라고 합니다. 이 카디널리티가 높은 순서를 기반으로 인덱스를 생성해야 합니다. 예를 들어 age와 email이 있다고 해봅시다. 어떤 것이 더 높죠? 당연히 email입니다. 즉, email이라는 필드에 대한 인덱스를 먼저 생성해야 하는 것입니다.