<B Tree & B+ Tree>

이진 트리는 하나의 부모가 두 개의 자식밖에 가지질 못하고, 균형이 맞지 않으면 검색 효율이 선형검색 급으로 떨어진다. 하지만 이진 트리 구조의 간결함과 균형만 맞다면 검색, 삽입, 삭제 모두 O(logN)의 성능을 보이는 장점이 있기 때문에 계속 개선시키기 위한 노력이 이루어지고 있다.

1. B Tree

데이터베이스, 파일 시스템에서 널리 사용되는 트리 자료구조의 일종이다.

이진 트리를 확장해서, 더 많은 수의 자식을 가질 수 있게 일반화 시킨 것이다.

자식 수에 대한 일반화를 진행하면서, 하나의 레벨에 더 저장되는 것뿐 만 아니라 트리의 균형을 자동으로 맞춰주는 로직까지 갖추었다. 단순하고 효율적이며, 레벨로만 따지면 완전히 균형을 맞춘 트리다.

대량의 데이터를 처리해야 할 때, 검색 구조의 경우 하나의 노드에 많은 데이터를 가질 수 있다는 점은 상당히 큰 장점이다.

대량의 데이터는 메모리보다 블럭 단위로 입출력하는 하드디스크 or SSD 에 저장해야하기 때문!

ex) 한 블럭이 1024 바이트면, 2 바이트를 읽으나 1024 바이트를 읽으나 똑같은 입출력 비용 발생. 따라서 하나의 노드를 모두 1024 바이트로 꽉 채워서 조절할 수 있으면 입출력에 있어서 효율적인 구성을 갖출 수 있다.

→ B-Tree 는 이러한 장점을 토대로 많은 데이터베이스 시스템의 인덱스 저장 방법으로 애용하고 있음

2) 규칙

- 노드의 자료수가 N 이면, 자식 수는 N+1 이어야 함
- 각 노드의 자료는 정렬된 상태여야함
- 루트 노드는 적어도 2 개 이상의 자식을 가져야함
- 루트 노드를 제외한 모든 노드는 적어도 M/2 개의 자료를 가지고 있어야함
- 외부 노드로 가는 경로의 길이는 모두 같음.
- 입력 자료는 중복 될 수 없음

2. B+ Tree

데이터의 빠른 접근을 위한 인덱스 역할만 하는 비단말 노드(not Leaf)가 추가로 있음

(기존의 B-Tree 와 데이터의 연결리스트로 구현된 색인구조)

B-Tree 의 변형 구조로, index 부분과 leaf 노드로 구성된 순차 데이터 부분으로 이루어진다. 인덱스 부분의 key 값은 leaf 에 있는 key 값을 직접 찾아가는데 사용함.

1) 장점

블럭 사이즈를 더 많이 이용할 수 있음 (key 값에 대한 하드디스크 액세스 주소가 없기 때문)

leaf 노드끼리 연결 리스트로 연결되어 있어서 범위 탐색에 매우 유리함

2) 단점

B-tree 의 경우 최상 케이스에서는 루트에서 끝날 수 있지만, B+tree 는 무조건 leaf 노드까지 내려가봐야 함

3. B-Tree & B+ Tree

B-tree 는 각 노드에 데이터가 저장됨

B+tree 는 index 노드와 leaf 노드로 분리되어 저장됨

(또한, leaf 노드는 서로 연결되어 있어서 임의접근이나 순차접근 모두 성능이 우수함)

B-tree 는 각 노드에서 key 와 data 모두 들어갈 수 있고, data 는 disk block 으로 포인터가 될 수 있음

B+tree 는 각 노드에서 key 만 들어감. 따라서 data 는 모두 leaf 노드에만 존재

B+tree 는 add 와 delete 가 모두 leaf 노드에서만 이루어짐

<Reference>

1. https://wangin9.tistory.com/entry/B-tree-B-tree