PRESENTATION

22주차 주제 데이터베이스

데이터베이스 ~성능튜닝-인덱스~

by mun

데이터베이스 튜닝

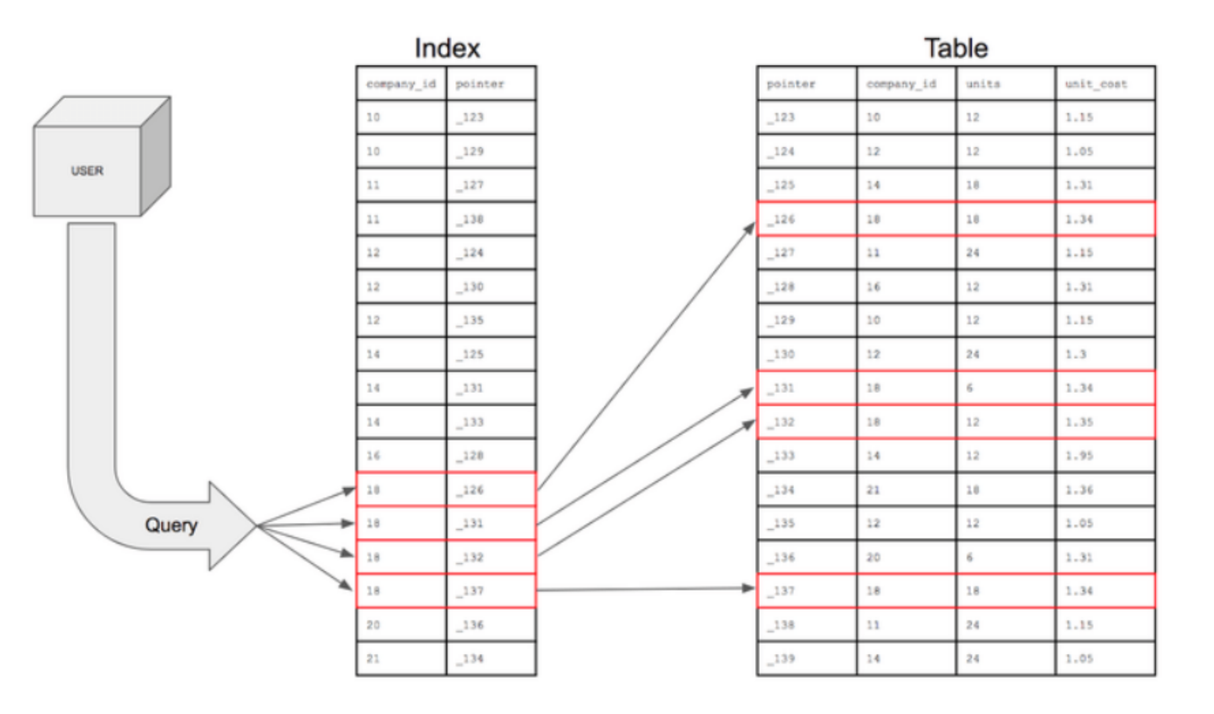
데이터베이스

여러 사람이 공유하여 사용할 목적으로 체계화해 통합, 관리하는 데이터의 집합

데이터베이스의 성능 향상을 위하여
운영체제나 DB자체의 구조를 이해하고, 필요한 요소를 변경하는 작업

인덱스

추가적인 쓰기 작업과 저장 공간을 활용하여 데이터베이스 테이블의 검색 속도를 향상시키기 위한 자료구조
저장한 컬럼을 기준으로 메모리 영역에 일종의 색인을 생성하는 것

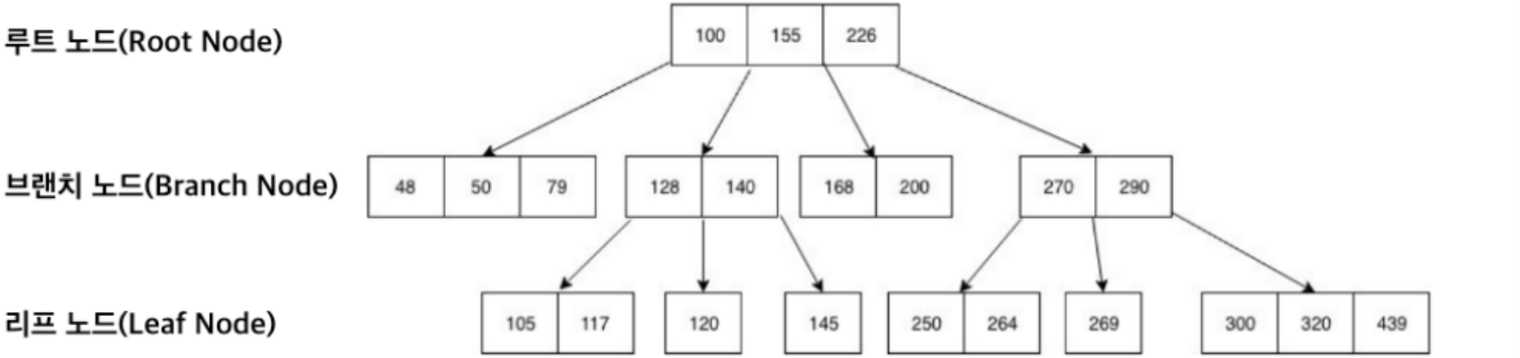


인덱스를 사용한 컬럼을 사용할 경우

```
SELECT USER WHERE NAME = 'minmunlee'
UPDATE USER SET NAME = 'mun' WHERE NAME = 'minmunlee';
DELETE USER WHERE NAME = 'mun'
```

SELECT 외에도 UPDATE, DELETE 성능도 함께 향상
index를 사용하지 않은 컬럼을 조회
→전체를 탐색하는 Full Scan을 수행
→처리 속도가 떨어진다

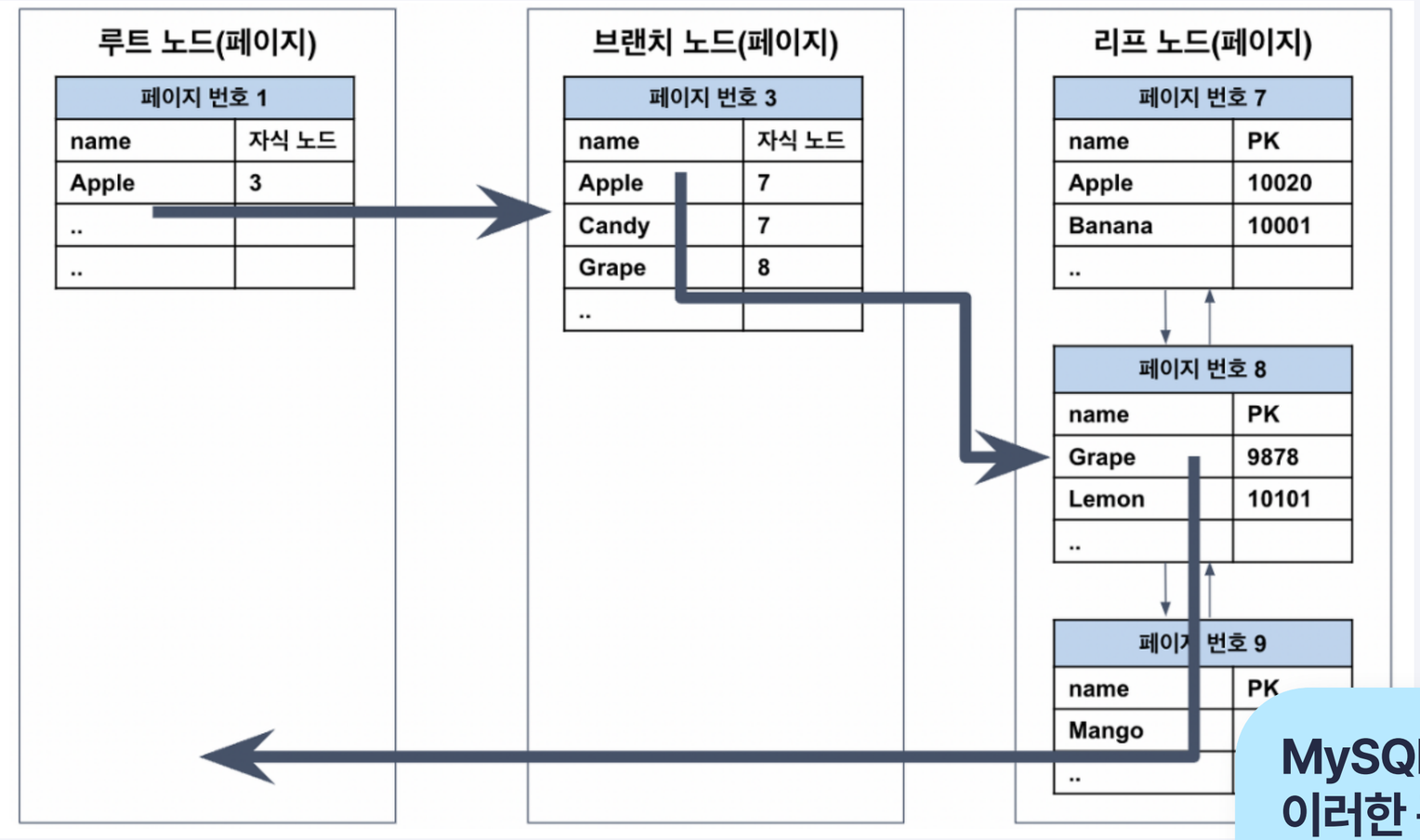
B-Tree(Balanced Tree) 구조



루트 노드(Root Node) 최상위 노드
브랜치 노드(Branch Node) 중간 노드
리프 노드(Leaf Node) 최하위 노드

- B-자식 2개 만을 갖는 이진 트리를 확장하여 N개의 자식을 가질 수 있도록 고안된 구조
- 좌우 자식 간의 균형이 맞지 않을 경우에는 매우 비효율적
  → 항상 균형을 맞춘다는 의미로 균형 트리(Balanced Tree)라고 불림

B-Tree(Balanced Tree) 구조



MySQL의 InnoDB가 클러스터 테이블이기 때문에
이러한 구조를 갖는 것
다른 DBMS의 경우에는 다를 수 있다

인덱스

추가적인 쓰기 작업과 저장 공간을 활용하여 데이터베이스 테이블의 검색 속도를 향상시키기 위한 자료구조
저장한 컬럼을 기준으로 메모리 영역에 일종의 색인을 생성하는 것

인덱스를 항상 최신의 정렬된 상태로 유지해야 원하는 값을 빠르게 탐색 가능
→인덱스가 적용된 컬럼에 INSERT, UPDATE, DELETE가 수행된다면 연산을 추가적으로 해줘야 함
→오버헤드가 발생

- **INSERT** 새로운 데이터에 대한 인덱스를 추가함
- **DELETE** 삭제하는 데이터의 인덱스를 사용하지 않는다는 작업을 진행함
- **UPDATE** 기존의 인덱스를 사용하지 않음 처리하고, 갱신된 데이터에 대해 인덱스를 추가함

새로운 데이터 삽입, 수정, 삭제 시
시간이 걸리게 된다

인덱스

추가적인 쓰기 작업과 저장 공간을 활용하여 데이터베이스 테이블의 검색 속도를 향상시키기 위한 자료구조
저장한 컬럼을 기준으로 메모리 영역에 일종의 색인을 생성하는 것

장점

- 테이블을 조회하는 속도와 그에 따른 성능을 향상 가능
- 전반적인 시스템의 부하를 줄일 수 있음

단점

- 인덱스를 관리하기 위해 DB의 약 10%에 해당하는 저장공간이 필요
- 인덱스를 관리하기 위해 추가 작업이 필요
- 인덱스를 잘못 사용할 경우 오히려 성능이 저하되는 역효과가 발생할 수 있음

- 규모가 작지 않은 테이블
- INSERT, UPDATE, DELETE가 자주 발생하지 않는 컬럼
- JOIN이나 WHERE 또는 ORDER BY에 자주 사용되는 컬럼

등, 조회 작업이 빈번한 컬럼에 활용하면 좋다

JPA 에서 인덱스 설정

```java
@Entity
@Table(indexes = @Index(name = "idx_post_title", columnList = "post_title"))
public class Post{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "post_title", nullable = false)
    private String postTitle;

    @Column(name = "post_content", nullable = false)
    private String postContent;

}
```

@Table(indexes = @Index(name = "[인덱스이름]", columnList="[컬럼명]"))

JPA 에서 인덱스 설정

두개의 컬럼으로 인덱스를 만들 경우

```java
@Entity
@Table(indexes = @Index(name = "idx_post_title", columnList = "post_title, post_content"))
public class Post{
    @Id
    @Ge
    private Long id;

    @Column(name = "post_title", nullable = false)
    private String postTitle;

    @Column(name = "post_content", nullable = false)
    private String postContent;

}
```

@Table(indexes = @Index(name = "[인덱스이름]", columnList="[컬럼명],[컬럼명],..."))

데이터의 사용시점을 파악하고
인덱스를 적절하게 적용해보아요

감사합니다

참고 :

https://devlog-wjdrbs96.tistory.com/351 , https://kghworks.tistory.com/106 ,
https://cheese10yun.github.io/checked-exception/ , https://kghworks.tistory.com/89 ,
http://wiki.hash.kr/index.php/%ED%8A%B8%EB%9E%9C%EC%9E%AD%EC%85%98