Index

장정훈

목적

- DB 테이블의 검색 속도를 향상시키기 위한 자료구조

- 테이블의 칼럼을 색인화.
  - Ex) 목차

- DB 안의 레코드를 처음부터 풀스캔 하는 것이 아님.
  - B+ Tree로 구성된 구조에서 Index 파일 검색으로 속도를 향상시키는 기술
    - B+ Tree : 데이터의 빠른 접근을 위한 인덱스 역할만 하는 비단말 노드

파일 구성

- 테이블 생성 시, FRM, MYD, MYI 3가지 파일이 생성.
- FRM : 테이블 구조가 저장된 파일
- MYD : 실제 데이터 파일
- MYI : Index 정보 파일(Index 사용 시 생성)

원리

1. MYI에 해당 컬럼을 색인화 하여 저장
    1. Index를 사용 안 할 시, MYI 파일은 비어있음.

2. Index를 해당 컬럼에 만들게 되면 해당 컬럼을 따로 인덱싱하여 MYI 파일에 입력

3. 사용자가 SELECT쿼리로 Index가 사용하는 쿼리를 사용 시, 해당 테이블을 검색하는 것이 아니라 B+ Tree로 정리해둔 MYI 파일의 내용을 검색.

4. Index를 사용하지 않는 Select쿼리라면 해당 테이블을 풀스캔

장점

- 키 값을 기초로 하여 테이블에서 검색, 정렬 속도 향상
- 그룹화 작업의 속도를 향상
- 테이블 행의 고유성을 강화
- 테이블의 기본 키는 자동으로 인덱스 됨.
- 필드 중에는 데이터 형식 때문에 인덱스 될 수 없는 필드가 있다.
  - Text 필드

단점

- .mdb 파일 크기가 증가
- 한 페이지를 동시에 수정할 수 있는 병행성이 줄어듦
- Index 된 필드에서 데이터를 업데이트, 레코드 추가 삭제 시 성능이 떨어짐.
- 데이터 변경이 자주 일어나는 경우, Index를 재작성 해야 함.

언제 쓰면 좋은데?

사용하면 좋은 경우	피해야 하는 경우
Where 절에서 자주 사용되는 컬럼	Data 중복도가 높은 컬럼
외래키가 사용되는 컬럼	DML 작업이 자주 일어나는 컬럼
Join에 자주 사용되는 컬럼	