DB Index 1부

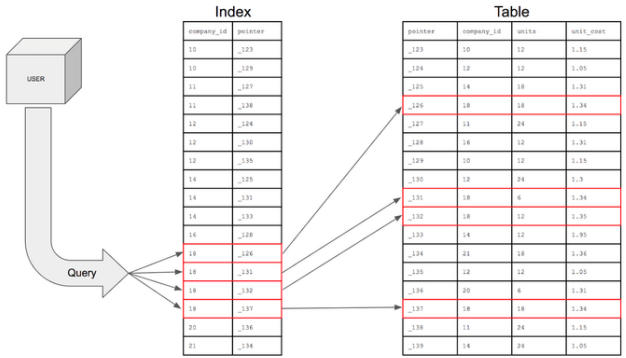
개념과 설계방법

박찬국 컨설턴트

Index란?

추가적인 쓰기 작업과 저장 공간을 활용하여 Database
테이블의 검색 속도를 향상시키기 위한 자료 구조

책의 색인



Index를 사용하면 좋은 점

- ➔ 성능의 향상
- ➔ 시스템 부하 감소

Index 사용시 주의점

➔ **Index 관리를 위한 추가작업**
- INSERT : 새로운 데이터에 대한 인덱스를 추가
- DELETE : 삭제하는 데이터의 인덱스를 사용하지 않는다는 작업 수행
- UPDATE : 기존의 인덱스를 사용하지 않음 처리, 갱신된 데이터에 대한 인덱스 추가

➔ **추가 저장 공간 필요**

➔ **잘못 사용하는 경우 성능 저하 가능성**

Index 언제 사용하는가?

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

Index 특징

- 하나 또는 여러 개의 컬럼에 설정 가능
- Where 절을 사용하지 않고 조회하는 것은 성능에 영향 없음

다중 컬럼 Index

➔ 두개 이상의 필드를 조합해서 생성한 Index

단일 Index

VS

다중 컬럼 Index

Table1(단일 인덱스)

```
CREATE TABLE table1(
    uid INT(11) NOT NULL auto_increment,
    id VARCHAR(20) NOT NULL,
    name VARCHAR(50) NOT NULL,
    address VARCHAR(100) NOT NULL,
    PRIMARY KEY('uid'),
    key idx_name(name),
    key idx_address(address)
)
```

'홍길동' VS '경기도'
더 빠른쪽부터 검색

Table2(다중 컬럼 인덱스)

```
CREATE TABLE table2(
    uid INT(11) NOT NULL auto_increment,
    id VARCHAR(20) NOT NULL,
    name VARCHAR(50) NOT NULL,
    address VARCHAR(100) NOT NULL,
    PRIMARY KEY('uid'),
    key idx_name(name, address)
)
```

'홍길동경기도'로 검색

QUREY문

```
SELECT * FROM table1 WHERE name='홍길동' AND address='경기도';
```

Table1(단일 인덱스)

```
CREATE TABLE table1(
    uid INT(11) NOT NULL auto_increment,
    id VARCHAR(20) NOT NULL,
    name VARCHAR(50) NOT NULL,
    address VARCHAR(100) NOT NULL,
    PRIMARY KEY('uid'),
    key idx_name(name),
    key idx_address(address)
)
```

Table2(다중 컬럼 인덱스)

```
CREATE TABLE table2(
    uid INT(11) NOT NULL auto_increment,
    id VARCHAR(20) NOT NULL,
    name VARCHAR(50) NOT NULL,
    address VARCHAR(100) NOT NULL,
    PRIMARY KEY('uid'),
    key idx_name(name, address)
)
```

QUREY문

```
SELECT * FROM table2 WHERE address='경기도';
```

Index 설계방법

- 무조건 많이 설정하지 않는다. (한 테이블당 3~5개가 적당 목적에 따라 상이)
- 조회시 자주 사용하는 컬럼
- 고유한 값 위주로 설계
- 카디널리티가 높을 수록 좋다 (= 한 컬럼이 갖고 있는 중복의 정도가 낮을 수록 좋다.)
- INDEX 키의 크기는 되도록 작게 설계
- PK, JOIN의 연결고리가 되는 컬럼
- 단일 인덱스 여러 개 보다 다중 컬럼 INDEX 생성 고려
- UPDATE가 빈번하지 않은 컬럼
- JOIN시 자주 사용하는 컬럼
- INDEX를 생성할 때 가장 효율적인 자료형은 정수형 자료(가변적 데이터는 비효율적)

Index 생성 방법

- 구글링…