index 의 중요성과 이해 SQL 독학 강의#24편

sTricky 2020. 7. 10. 14:14

index의 중요성과 이해 SQL 독학 강의#24편

콘텐츠 index

5. 인덱스를 사용할 수 없는 SQL 조건절
6. 인덱스 생성 및 변경 SQL 예제

전편 강의 보러 가기

2020/07/08 - [Database/sql 강의] - mysql 제약조건 알아보기 SQL 독학 강의#23편

	mysql 제약조건 알아보기 SQL 독학 강의#23편 stricky.tistory.com

이번 포스팅에서 다룰 내용은 인덱스입니다.

인덱스를 데이터베이스에서 사용하는 가장 큰 이유는 쿼리의 성능을 높이기 위해서 일 것입니다. 인덱스는 테이블 내 데이터를 정렬한 뒤 필요로 하는 데이터만 빨리 가지고 오기 위해서 사용을 하게 됩니다.

인덱스는 데이터베이스 내에서도 성능과 관련해서 가장 중요한 내용입니다. 생각보다 간단하면서도 데이터베이스 내에서 어떤 역할을 하는지 천천히 알아보도록 하겠습니다.

인덱스란 무엇인가

인덱스란 우리 실생활에서도 접할 수 있는 것입니다.

간단하게 말하면 어떤 것이 어디 있는지를 알려주는 것이 인덱스의 역할입니다.

에버랜드 지도

위와 같이 에버랜드 지도가 있습니다.

에버랜드 놀이동산에 들어가서 정문에서부터 T 익스프레스를 타고 싶다면 어디로 가야 할지 이 지도를 보고 가면 되겠죠?

뭐 물론, 많이 가보신 분들이야 눈감고도 찾아가시겠지만..ㅎㅎ

이런 지도가 없이 에버랜드를 처음 간 사람은 T 익스프레스를 어떻게 찾아야 할까요?

물론 표지판을 따라서 갈 수 있겠지만 그게 없다면 왼쪽이나 오른쪽 어느 한쪽부터 반대편까지 다 찾아봐야 하는 거죠.

그러다가 T 익스프레스를 빨리 만날 수도 있겠지만, 그렇지 못하고 마지막까지 찾아 헤맬지도 모릅니다.

바로 그때, 이 지도의 역할이 인덱스의 역할을 하는 겁니다. 이 지도 한 장만 있다면 T 익스프레스로 가는 가장 빠른 길을 찾아서 단번에 찾을 수 있을 것입니다.

데이터베이스 내 테이블 데이터도 마찬가지입니다. 인덱스가 없다면 테이블 데이터 처음부터 끝까지 해당 데이터가 있는지 하나하나 찾아봐야 합니다. 그것이 바로 Full scan입니다.

만약 인덱스가 해당 테이블에 있다면 인덱스를 보고 해당 데이터의 위치로 바로 가서 데이터를 가지고 올 수 있겠죠? 그게 바로 index scan이라는 것 이죠!

인덱스가 무엇인지 감이 오시나요?

그럼 다음은 인덱스의 종료에 관해서 알아보도록 하겠습니다.

인덱스의 종류

우선 이 강의의 기준이 되는 DBMS인 Mysql을 기준으로 설명을 드리도록 하겠습니다.

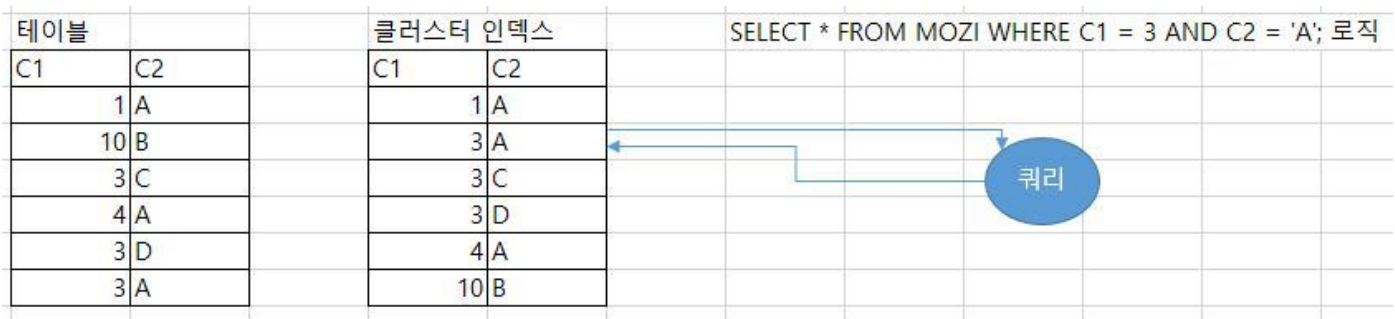
1. 클러스터 인덱스

Primary key 설정 시 자동 생성됩니다. 해당 Primary key 칼럼 데이터가 변경되더라도 항상 정렬을 유지합니다.

테이블당 1개의 클러스터 인덱스를 생성할 수 있습니다. 지정된 칼럼을 기준으로 테이블 내 데이터가 물리적으로 정렬됩니다. 테이블이 물리적으로 정렬되므로 인해 리프 노드가 필요 없으며, 추가적인 공간이 필요치 않습니다.

조회 성능은 빠르지만, insert, update, delete 작업처럼 데이터 변경이 있을 시에는 정렬 작업등으로 인하여 성능이 떨어지게 되는 단점을 가지고 있습니다.

또한 클러스터 인덱스가 생성 되게 되면 테이블 내 다른 보조 인덱스에 Primary key값을 포함하게 됩니다. 이로 인해 보조 인덱스들의 크기가 커질 수 있습니다.



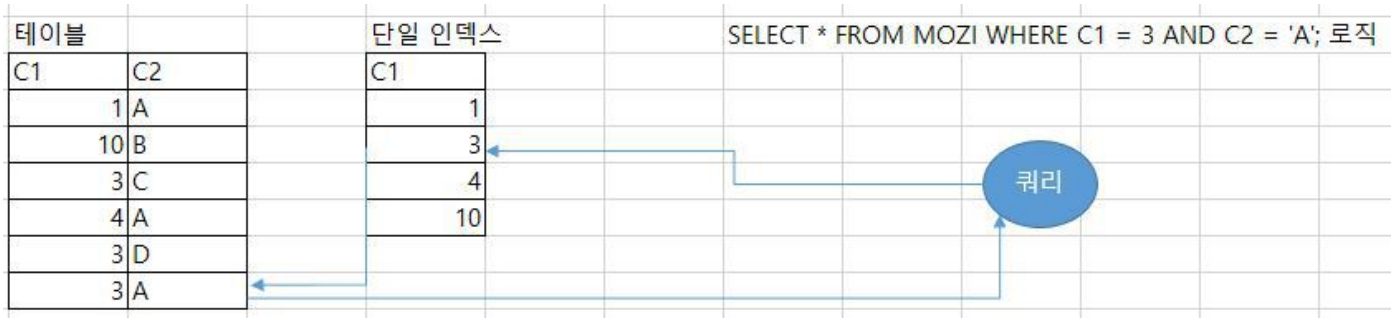
클러스터 인덱스

클러스터 인덱스를 제외한 기타 단일, 복합, 커버드 인덱스는 논 클러스터 인덱스라고 부르며, 테이블당 249개 까지 생성이 가능합니다.

2. 단일 인덱스

인덱스 생성 시 하나의 칼럼만 지정하는 경우를 말합니다.

주로 데이터가 많지 않은 경우 사용하게 됩니다. 주로 사용하는 SQL에 조건이 하나만 걸릴 때 생성하게 됩니다.

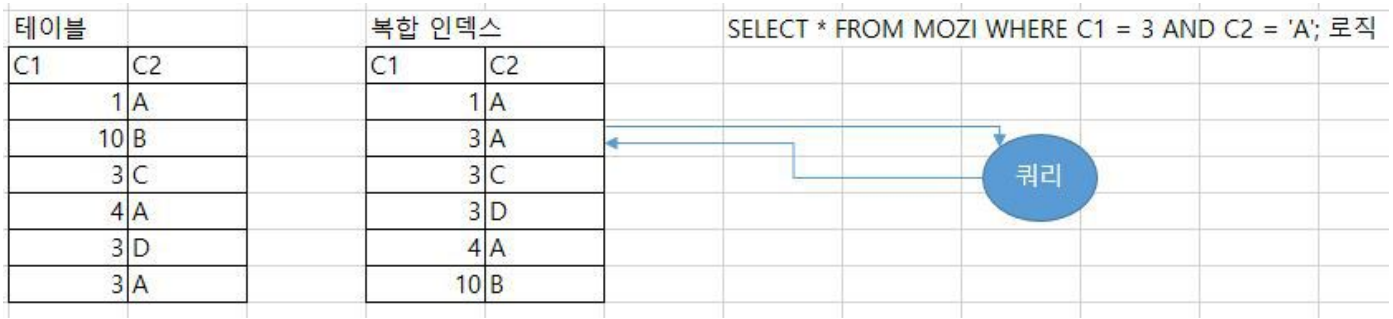


단일 인덱스

3. 복합 인덱스

인덱스를 생성할 때 칼럼을 두 개 이상 지정하는 경우를 말합니다.

주로 사용하는 조회 쿼리에 조건이 많이 걸린다면 고려해볼 만한 인덱스입니다.



복합 인덱스

복합 인덱스 역시 단일 인덱스와 마찬가지로 데이터베이스 내 테이블과 다른 공간에 해당 데이터를 가지고 와서 정렬하기 때문에 데이터베이스 내 용량을 점유하게 됩니다.

4. 커버드 인덱스

커버드 인덱스란, SQL내에서 출력하는 칼럼 및 조건에 삽입된 칼럼이 모두 인덱스에 정보가 있어서 실제 테이블을 조회하지 않고도 데이터를 가지고 올 수 있는 경우에 사용되는 인덱스를 말합니다.

커버드 SQL이라고도 하며, 성능적인 측면만 고려할 시 가장 좋은 방법일 수 있습니다.

```
Master-mysql> SELECT C1, C2 FROM MOZI WHERE C1 = 3 AND C2 = 'A';
+------+------+
| C1   | C2   |
+------+------+
|    3 | A    |
+------+------+
1 row in set (0.03 sec)
```

커버드 인덱스

인덱스 생성 기준

1. 테이블 내 데이터가 많을 때 만드는 걸 권장

테이블 내 데이터가 많지 않을 때는 Full scan이 Index scan보다 더 빠를 수 있습니다. 인덱스는 테이블 내 데이터가 많으면 많을수록 더 극명한 효과를 가지고 올 수 있습니다.

2. Primary key 칼럼에는 생성할 필요 없음

Primary key가 부여된 칼럼에는 위에서 언급했다시피 클러스터 인덱스가 생성됩니다. 또한 Unique제약 조건이 걸려있는 칼럼 또한 마찬가지입니다. 위 두 가지 제약조건이 적용된 칼럼에는 인덱스를 따로 추가할 필요가 없습니다.

3. Cardinality를 확인 후 생성

Cardinality란 어떤 칼럼 내 값의 분산도를 말합니다.

예를 들어서 주민등록번호 칼럼과 성별이 들어있는 칼럼을 생각해보면, 유일한 값인 주민등록 번호가 들어간 칼럼은 Cadinality가 높다라고 이야기할 수 있으며, 성별은 단 두 개의 데이터만 들어가 있으므로 Cardinality가 낮다고 표현할 수 있습니다.

즉, Cardinality가 높은 칼럼에 인덱스를 생성하는 것이 유리합니다.

인덱스의 역효과

1. 오버헤드

테이블에 데이터가 insert, update, delete 될 때 인덱스 역시 갱신이 이루어집니다. 이때 인덱스를 갱신하는 것을 오버헤드라고 표현하는데, 그만큼 insert, update, delete 작업이 느려질 수 있습니다. 대신 그만큼 select는 빨리 지겠죠.

2. 오 사용

한 개의 테이블에 다수의 인덱스를 생성하게 되면 옵티마이저가 실행계획을 만들 때 의도하지 않은 인덱스를 사용하게 될 수 있습니다. 옵티마이저가 만능이 아니기 때문에 이런 문제가 발생할 수 있습니다.

그래서 사용자는 테이블에 인덱스를 생성할 때 무조건 생성하는 것이 아니라 필요한 곳에 적재적소에 생성을 하길 바랍니다.

B-Tree 인덱스

오라클이나 MSSQL, Postgres 등 다른 DBMS도 마찬가지지만 Mysql에서는 B-Tree 인덱스를 가장 많이 사용하고, 다른 인덱스 타입을 사용할 수 없게 되어 있습니다.

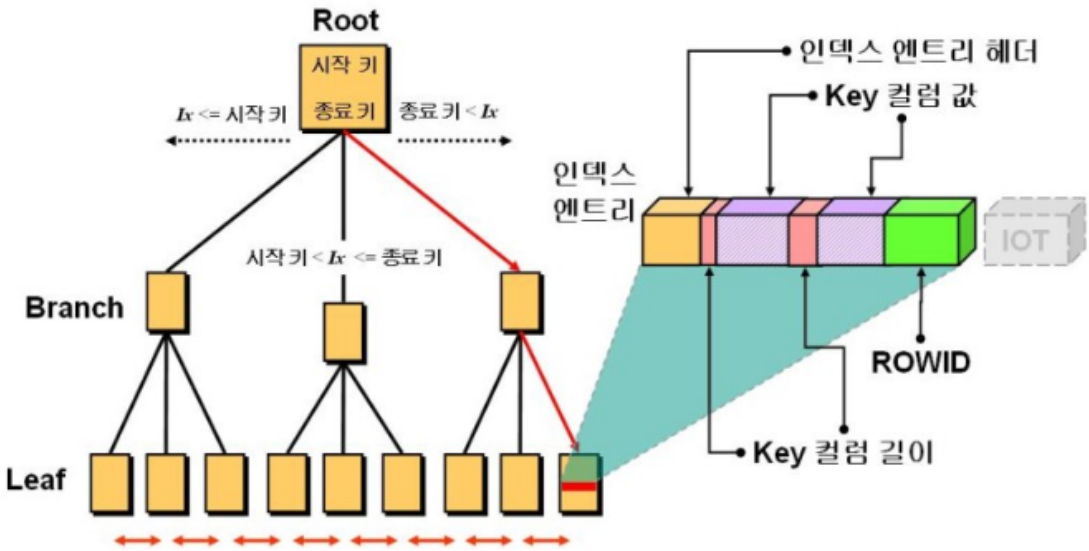
그래서, 다른 건 다 제외하고 B-Tree 인덱스에 관해서 설명을 드리도록 하겠습니다.

B-Tree 인덱스란 칼럼의 데이터는 변형시키지 않고 인덱스 내부에서 정렬된 상태로 유지를 하고 있습니다.

일반적으로 전문 검색 등의 특이한 상황을 배제하면 대다수 이 B-Tree인덱스를 사용합니다.

B-Tree 인덱스는 3개의 노드로 구분할 수 있습니다.

Node 명	설명
Root Node	최상위 노드를 칭함
Leaf Node	최하위 노드를 칭함
Branch Node	Root 와 Leaf 노드를 연결 하는 노드를 칭함



B-Tree 인덱스

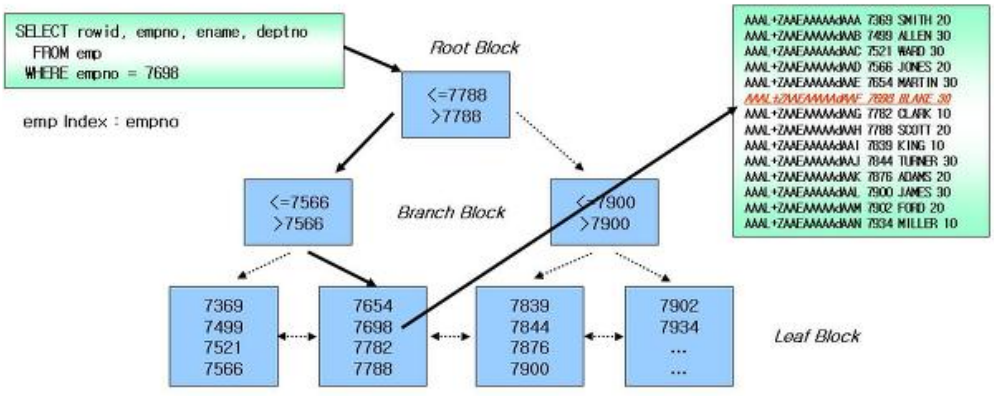
B-Tree 인덱스에 값이 새로 들어오면 저장되는 키값을 이용해서 인덱스 내 적당한 위치를 찾습니다.

저장될 위치가 결정되고 나면 레코드의 키값과 주소 정보를 B-Tree 인덱스의 Leaf Node에 저장합니다.

이때 Leaf Node가 꽉 차서 분리가 되기도 합니다.

일반적으로 테이블에 데이터를 저장하는 것보다 인덱스에 추가할 때 1~1.5배 더 많은 비용이 발생하게 됩니다.

B-Tree 인덱스의 구조상 인덱스 내 키가 변경될 때는 delete and insert가 수행됩니다. 단순하게 Leaf Node의 키값과 주소 값만 변경할 수는 없습니다.



index range scan 그림

인덱스를 사용 할 수 없는 SQL 조건절

1. Equal로 조건절이 작성되지 않는 경우

col_name <> 'Y'
col_name NOT IN (1, 2, 3)
col_name NOT BETWEEN 1 AND 10
col_name IS NOT NULL (Mysql 에서는 가능)

2. like를 사용하는 경우

col_name like '%TEST'

3. function을 사용하는 경우

substring(col_name , 1) = 'A'

4. 데이터 타입이 서로 다른 경우

char_col = 10 (char형 컬럼에 숫자형을 조건으로 입력 하는 경우)

인덱스 생성 및 변경 SQL 예제

1. 테이블에 인덱스를 생성하는 SQL 예제

```
CREATE INDEX <Index name>
ON <Table name> ( column 1, column 2, ... );

또는

-- 단일 인덱스
ALTER TABLE books ADD INDEX idx_test ( writer );

-- 복합 인덱스
ALTER TABLE books ADD INDEX idx_test ( writer, company, ... );
```

2. 인덱스 삭제 SQL 예제

```
-- 인덱스 삭제 SQL 예제
ALTER TABLE books DROP INDEX idx_test;
```

3. 유니크 인덱스 생성 SQL 예제

```
-- 유니크 인덱스 생성 SQL 예제
ALTER TABLE tablename ADD UNIQUE INDEX indexname (column1, column2);
```

4. 생성된 인덱스 확인 SQL 예제

```
-- 생성된 인덱스 확인 SQL 예제
SHOW INDEX FROM <Table name>;
```

자, 오늘 이렇게 인덱스에 관해서 정리를 좀 해봤습니다.

좀 더 자세하게 쓸 수도 있으나, 이 정도만 알아도 충분할 것으로 생각이 됩니다.