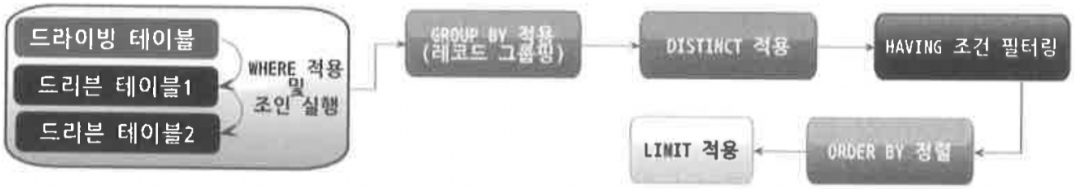
7.4 SELECT

생성일	@2021년 7월 3일 오후 11:02
태그	

1. SELECT 각 절의 처리 순서



[그림 7-4] 각 쿼리 절의 실행 순서

인덱스를 이용해 처리할때는 ORDER BY나 GROUP BY 절이 있다 하더라도 그 단계 자체가 불필요하므로 생략이 된다 (뭔 말이지)

예외의 경우

[그림 7-5] 쿼리 각 절의 실행 순서(예외적으로 ORDER BY가 조인보다 먼저 실행되는 경우)

첫번째 테이블만 읽어서 정렬 후, 나머지 테이블을 읽는 형태

→ GROUP BY 절이 없어 ORDER BY 만 사용된 쿼리에서 주로 사용되는 순서이다.

실행 순서가 위와 다르게 적용되는 쿼리가 필요하다면? Inline View를 사용

example) LIMIT를 먼저 적용하고 ORDER BY를 실행하고자 하면 서브쿼리로 작성된 인라인뷰로 먼저 LIMIT를 걸어줘야한다. 하지만 이렇게 인라인 뷰가 사용되면 임시테이블이 사용되기 때문에 주의해야함...

2. WHERE 절과 GROUP BY 절, 그리고 ORDER BY 절의 인덱스 사용

WHERE 절이나 ORDER BY 또는 GROUP BY가 인덱스를 사용하려면?

⇒ 인덱스된 칼럼의 값 자체를 변환하지 않고 그대로 사용한다는 조건을 만족해야함

```
SELECT * FROM salaries WHERE salary*10 > 150000;
```

```
SELECT * FROM salaries WHERE salary > 150000/10;
```

- 이렇게 인덱스의 컬럼 값을 가공을 시키면 옵티마이저가 인덱스를 최적으로 이용할 수 있게 표현식을 변환하지 못함
- 연산이 필요할 때에는 미리 계산된 값을 저장할 컬럼을 추가하고, 그 칼럼에 인덱스를 생성해야함

WHERE 절에서 사용되는 비교조건을 만들때 주의해야할 점

WHERE 절의 비교 조건에서 연산자 양쪽의 두 비교 대상 값은 데이터 타입이 일치해야함

```
CREATE TABLE tb_test (age VARCHAR(10), INDEX ix_age (age));
INSERT INTO tb_test VALUES ('1'), ('2'), ('3'), ('4'), ('5'), ('6'), ('7');
SELECT * FROM tb_test WHERE age=2;
```

id	select_type	table	type	key	key_len	ref	rows	Extra
1	SIMPLE	tb_test	index	ix_age	33		7	Using where; Using index

위 예제의 실행계획을 보면 type이 ref, range가 아니라 index로 되어있고, 이 말은 인덱스 풀 스캔을 탄다는것이다

- age 칼럼의 데이터타입(VARCHAR)과 비교되는 값인 2(INTEGER)의 데이터 타입이 다르기 때문
- 옵티마이저가 내부적으로 문자열타입→ 숫자 타입으로 데이터를 변환 후 비교를 하기 때문에 인덱스 레인지 스캔이 불가능하다.
- 이렇게 쿼리를 변경하면 레인지 스캔을 유도할 수 있다

```
SELECT * FROM tb_test WHERE age='2';
```

id	select_type	table	type	key	key_len	ref	rows	Extra
1	SIMPLE	tb_test	ref	ix_age	33	const	1	Using where; Using index

옵티마이저가 이런 선택을 하는 이유는?

주의 문자열과 숫자를 비교할 때 나타나는 이러한 문제는 문자열 비교보다는 숫자 값의 비교가 빨라서 MySQL 옵티마이저가 숫자 타입에 우선권을 부여하기 때문에 발생하는 문제다. 이 예제와는 반대로 테이블의 칼럼은 숫자 타입인데, SQL의 비교 조건을 문자열 값과 비교하는 경우에는 이런 불합리한 현상이 발생하지 않는다.

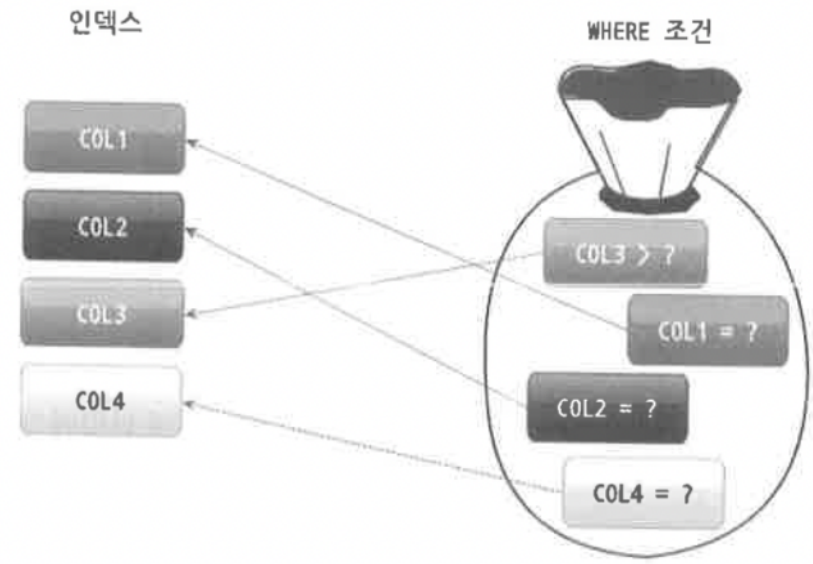
```
CREATE TABLE tb_test (age INT, INDEX ix_age (age));
INSERT INTO tb_test VALUES (1), (2), (3), (4), (5), (6), (7);
SELECT * FROM tb_test WHERE age='2';
```

위의 SQL 실행 계획을 확인해보면 type 칼럼에 index가 아니라 ref가 나타나는 것을 확인할 수 있을 것이다. 이 비교 조건에서는 상수값으로 지정한 문자열 '2'를 숫자 타입으로 먼저 변환한 후 tb_test 테이블의 age 칼럼과 비교하기 때문이다.

WHERE 절의 인덱스 사용

WHERE 절의 조건이 인덱스를 사용할 수 있는 기준

- 범위 제한 조건 : 인덱스를 구성하는 칼럼과 얼마나 좌측부터 일치하는가에 따라 달라짐
  - 동등 비교 조건
  - IN으로 구성된 조건
- 체크 조건



[그림 7-6] WHERE 조건의 인덱스 사용 규칙

4개의 컬럼이 인덱스로 구성되어있을때 col3은 크다 작다 조건으로 비교가 됨

→ 뒤의 col4의 조건은 범위 제한 조건으로 사용되지 못하고 체크 조건으로 사용됨

> 💡 WHERE 절에서 각 조건이 명시된 순서는 중요치 않고, 그 칼럼에 대한 조건이 있는지 없는지가 중요

(만약 크다 작다 조건으로 비교를 하는 칼럼이 있다면 해당 칼럼을 인덱스 맨 마지막에 추가해야함)

OR 연산자가 WHERE 절에 포함되어 있다면?

→ 연산 처리 방법이 완전히 바뀐다

```
SELECT *
FROM employees
WHERE first_name='Kebin' OR last_name='Poly';
```

맨 첫번째 조건은 인덱스를 이용할 수 있지만 OR 뒤의 조건은 인덱스를 사용할 수 없다 (AND로 연결되어있으면 인덱스 사용 가능)

→ `풀 테이블 스캔 + 인덱스 레인지 스캔`의 작업량 보다 `풀 테이블 스캔` 한 번이 더 빠르기 때문에 옵티마이저가 `풀 테이블 스캔`을 선택

GROUP BY 절의 인덱스 사용

GROUP BY 절의 각 칼럼은 비교 연산자를 가지지 않으므로 범위 제한 조건이나 체크 조건과 같이 구분해서 생각하지 않아도 됨

[인덱스를 이용 가능한 경우] : GROUP BY 절에 명시된 칼럼의 순서가 인덱스 칼럼의 순서가 같을 경우

- GROUP BY 절에 명시된 칼럼이 인덱스 칼럼의 순서와 위치가 같아야 한다.
- 인덱스를 구성하는 칼럼 중에서 뒷쪽에 있는 칼럼은 GROUP BY 절에 명시되지 않아도 인덱스를 사용할 수 있지만 인덱스의 앞쪽에 있는 칼럼이 GROUP BY 절에 명시되지 않으면 인덱스를 사용할 수 없다.
- WHERE 조건절과는 달리, GROUP BY 절에 명시된 칼럼이 하나라도 인덱스에 없으면 GROUP BY 절은 전혀 인덱스를 이용하지 못한다.

```
... GROUP BY COL1
... GROUP BY COL1, COL2
... GROUP BY COL1, COL2, COL3
... GROUP BY COL1, COL2, COL3, COL4
```

만약 다음과 같이 col1, col2가 동등 비교 조건으로 사용된다면, GROUP BY 절에 col1, col2가 빠져도 인덱스를 이용한 GROUP BY 가 가능할 때도 있음

```
... WHERE COL1='상수' ... GROUP BY COL2, COL3
... WHERE COL1='상수' AND COL2='상수' ... GROUP BY COL3, COL4
... WHERE COL1='상수' AND COL2='상수' AND COL3='상수' ... GROUP BY COL4
```

ORDER BY 절의 인덱스 사용

MySQL에서 GROUP BY와 ORDER BY 는 처리 방법이 상당히 비슷함

→ 그래서 인덱스 사용 여부도 둘이 거의 흡사

[차이점]

- 정렬되는 각 컬럼의 ASC DESC 옵션이 인덱스와 같거나 정반대의 경우에만 사용가능

  → MySQL의 인덱스는 ASC (오름차순) 으로만 정렬되어있기 때문

- 인덱스의 모든 칼럼이 ORDER BY 절에 사용되어야 하진 않지만, 인덱스에 정의 된 칼럼의 왼쪽부터 일치해야함

[그림 7-8] ORDER BY 절의 인덱스 사용 규칙

만약 이렇게 되어있으면 ORDER BY에서 인덱스를 이용할 수 없다

```
... ORDER BY COL2, COL3
... ORDER BY COL1, COL3, COL2
... ORDER BY COL1, COL2 DESC, COL3
... ORDER BY COL1, COL3
... ORDER BY COL1, COL2, COL3, COL4, COL5
```

위의 각 예제가 인덱스를 사용하지 못하는 원인을 살펴보자.

- 첫 번째 예제는 인덱스의 제일 앞쪽 칼럼인 COL1이 ORDER BY 절에 명시되지 않았기 때문에 인덱스를 사용할 수 없다.
- 두 번째 예제는 인덱스와 ORDER BY 절의 칼럼 순서가 일치하지 않기 때문에 인덱스를 사용할 수 없다.
- 세 번째 예제는 ORDER BY 절의 다른 칼럼은 모두 오름차순인데, 두 번째 칼럼인 COL2의 정렬 순서가 내림차순이라서 인덱스를 사용할 수 없다.
- 네 번째 예제는 인덱스에는 COL1과 COL3 사이에 COL2 칼럼이 있지만 ORDER BY 절에는 COL2 칼럼이 명시되지 않았기 때문에 인덱스를 사용할 수 없다.
- 다섯 번째 예제는 인덱스에 존재하지 않는 COL5가 ORDER BY 절에 명시됐기 때문에 인덱스를 사용하지 못한다.

WHERE 조건과 ORDER BY (또는 GROUP BY)절의 인덱스 사용

- WHERE 조건은 A인덱스 사용, ORDER BY는 B 인덱스 사용 → **이 경우는 인덱스 사용이 불가**
- WHERE절, GROUP BY절이 같이 사용된 경우와 GROUP BY, ORDER BY가 같이 사용된 쿼리도 마찬가지

[WHERE절과 ORDER BY 절이 인덱스를 이용할 수 있는 경우]

WHERE 절과 ORDER BY 절이 동시에 같은 인덱스를 이용

WHERE 절의 비교 조건에서 사용하는 칼럼과 ORDER BY 절의 정렬 대상 칼럼이 모두 하나의 인덱스에 연속해서 포함돼 있을 때 이 방식으로 인덱스를 사용할 수 있다. 이 방법은 나머지 2가지 방식보다 훨씬 빠른 성능을 보이기 때문에 가능하다면 이 방식으로 처리할 수 있게 쿼리를 튜닝하거나 인덱스를 생성하는 것이 좋다.

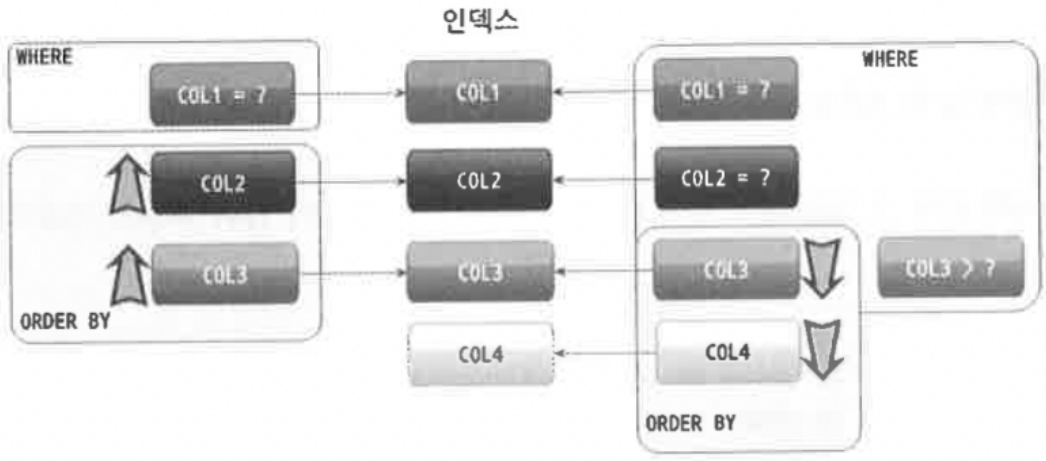
WHERE 절만 인덱스를 이용

ORDER BY 절은 인덱스를 이용한 정렬이 불가능하며, 인덱스를 통해 검색된 결과 레코드를 별도의 정렬 처리 과정(Filesort)을 거쳐서 정렬을 수행한다. 주로 이 방법은 WHERE 절의 조건에 일치하는 레코드의 건수가 많지 않을 때 효율적인 방식이다.

ORDER BY 절만 인덱스를 이용

ORDER BY 절은 인덱스를 이용해 처리하지만 WHERE 절은 인덱스를 이용하지 못한다. 이 방식은 ORDER BY 절의 순서대로 인덱스를 읽으면서, 레코드 한 건씩을 WHERE 절의 조건에 일치하는지 비교해 일치하지 않을 때는 버리는 형태로 처리한다. 주로 아주 많은 레코드를 조회해서 정렬해야 할 때는 이런 형태로 튜닝하기도 한다.

- WHERE 절에서 동등비교 조건으로 비교된 칼럼과 ORDER BY 절에 명시된 칼럼이 순서대로 빠짐없이 인덱스 칼럼의 왼쪽부터 일치해야함
  - 중간에 빠지는 칼럼이 있으면 두 절 모두 인덱스를 사용할 수 없음



[그림 7-9] WHERE 절과 ORDER BY 절의 인덱스 사용 규칙

  - 여기서 오른쪽 처럼 ORDER BY 절에 col3이 사용되고 있으면, WHERE절에 동등비교가 아닌 크다작다로 비교되어도 WHERE조건과 ORDER BY조건이 모두 인덱스를 이용할 수 있음

- 범위조건의 비교가 WHERE절에서 사용되는 경우

```
SELECT * FROM tb_test WHERE col1 > 10 ORDER BY col1, col2, col3;
SELECT * FROM tb_test WHERE col1 > 10 ORDER BY col2, col3;
```

- 첫번째 쿼리에서, col1 > 10 조건을 만족하는 col1값은 여러개일 수 있지만 ORDER BY절에서 col1 ~3까지 순서대로 명시가 되었기때문에 인덱스를 사용해서 두 절 모두 처리 가능
- 두번째 쿼리에서는 ORDER BY절에 범위조건으로 비교하는 col1이 명시되지 않아서 정렬시 인덱스를 이용할 수 없음

[요약 - WHERE, ORDER BY 절 모두 인덱스를 이용해 처리 못하는 경우]

```
... WHERE COL1=10 ORDER BY COL3, COL4
... WHERE COL1>10 ORDER BY COL2, COL3
... WHERE COL1 IN (1,2,3,4) ORDER BY COL2
```

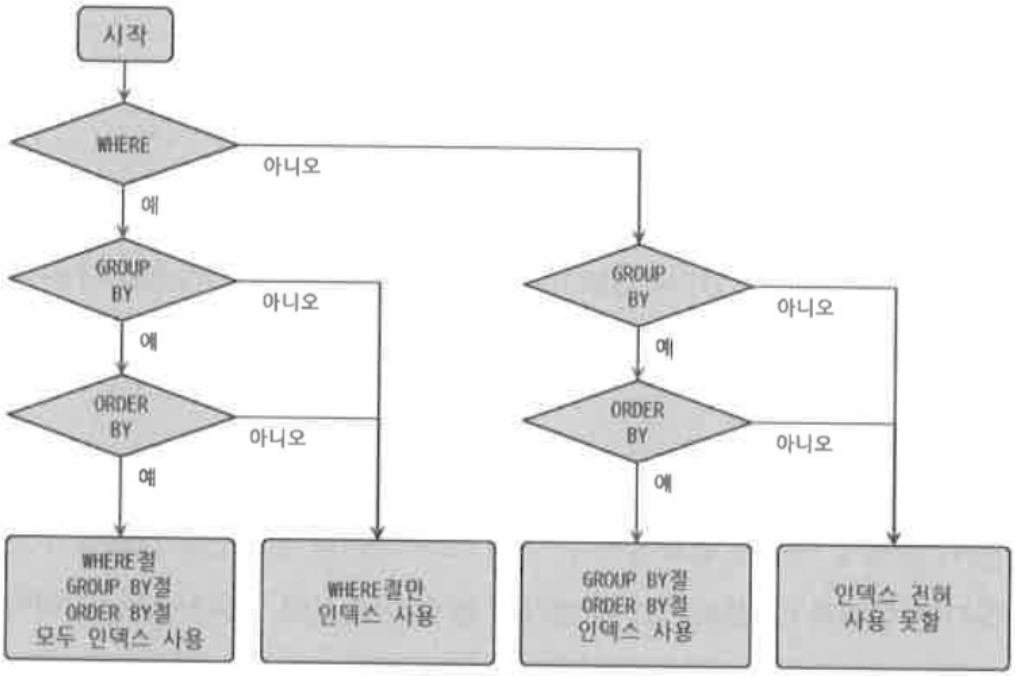
GROUP BY 절과 ORDER BY 절의 인덱스 사용

대부분의 로직은 WHERE과 ORDER BY절이 동시에 사용된 쿼리에 적용되는 기준과 비슷하다

- 두 절에 명시된 칼럼이 순서와 내용이 모두 같아야함
- 두 절 중 하나라도 인덱스를 이용할 수 없을 때 둘 다 인덱스를 사용하지 못한다

```
... GROUP BY COL1, COL2 ORDER BY COL2
... GROUP BY COL1, COL2 ORDER BY COL1, COL3
```

WHERE 조건과 ORDER BY 절, 그리고 GROUP BY 절의 인덱스 사용



[그림 7-10] WHERE 조건과 ORDER BY 절, 그리고 GROUP BY 절의 인덱스 사용 여부 판단

ㅇㅇ!

3. WHERE 절의 비교 조건 사용시 주의사항

막걸리 마시고 와서 다시,,,