인덱스 기본 사용법 = 인덱스 Range Scan

|  |  |
| --- | --- |
| Index Range Scan이 가능한 경우   * Index가 정렬되어 **순서대로 정렬**되어 있다. * 단어들이 모여있다. | Index Range Scan이 불가능 한 경우   * 시작점을 찾을 수 없다. * **가공값이나 중간값으로는 스캔 시작점을 찾을 수 없다.** * 가공값과 중간값으로 색인 사용시 색인 전체를 스캔해야 한다. |

인덱스 컬럼(선두 컬럼)을 가공하지 않아야 인덱스를 정상적으로 사용가능

인덱스를 정상적으로 사용한다. = 리프 블록 일부만 스캔하는 Index Range Scan의미

인덱스 Range Scan 할 수 없는 이유

인덱스 컬럼 가공시 > Range Scan 불가능

이유

? **인덱스 스캔 시작점을 찾을 수 없기 때문**

? **LIKE로 중간값 검색** : 포함하는 값은 전체 구간에 걸쳐 흩어져 있기 때문

OR 조건절

Union all 을 사용하여 IN, OR 대체

– Union all 을 이용하여 각 브랜치 별로 인덱스 스캔 시작점 찾기 가능

Index Range Scan이 불가능한 경우 – 90p sql문 박스

* OR, IN 조건은 옵티마이저에 의해 쿼리변환 기능으로 Index Range Scan으로 처리되기도 함

인덱스 사용조건

* 인덱스 선두 컬럼이 조건절에 있어야 한다. **가공하지 않은 상태로**

= 인덱스 선두 컬럼이 가공되지 않은 상태로 조건절에 있을 경우 Index Range Scan은 무조건 가능

인덱스를 잘 탄다. = 인덱스 리프 블록이 스캔하는 양 – 인덱스 스캔 효율화 3.3

인덱스를 이용한 소트 연산 생략 – 5.3 자세히

* 인덱스가 정렬되어 있다. = 데이터도 정렬되어 있음 = Range Scan 가능
* Order by 정렬 연산 생략 = 결과집합은 어차피 변경순번 순으로 정렬되기 때문.

ORDER BY 절에서 컬럼 가공 – 98~99p sql 참고

* **가공값 기준으로 정렬 요청시 생략 불가능**

SELECT-LIST 컬럼 가공

* 최소값 MIN – 옵티마이저는 정렬 연산 수행 안 함 / 가장 왼쪽 첫 번째 레코드가 최소값
* 최대값 MAX – 오른쪽 레코드 하나만 읽는다.
* 인덱스는 문자열 기준 정렬 = 숫자형으로 형변환시 생략 불가능

인덱스 컬럼 가공시 효과적인 SQL 작성하기 – Top N 알고리즘 (5.3.4 이력 조회)

자동 형변환  
옵티마이저가 sql을 자동으로 변환 -> 인덱스 컬럼이 가공됨

* 문자형 < 숫자형 (숫자형 컬럼 기준으로 문자형 컬럼을 변환)
* 날짜 포맷을 정확히 지정하여 코딩하자 (컴파일 오류나 결과집합이 달라질 수 있다.)
* 숫자형 컬럼 LIKE 검색시 **자동 형변환이 발생**하여 인덱스 액세스 조건으로 사용 불가
  + 문자형 컬럼이 숫자형으로 변환되는데 문자형 컬럼에 숫자로 변환할 수 없는 문자열이 입력되면 쿼리 수행 도중 **에러가 발생**한다.
* decode 형변환으로 인한 오류 조심! (데이터 타입 일치시 오류 해결)

자동 형변환에 의존하지 말고 인덱스 컬럼 기준으로 값을 **정확히 형변환**하자!!

**SQL 성능 - 블록 I/O를 줄일 수 있느냐 없느냐가 중요!!**

형변환 함수를 생략해도 옵티마이저가 자동으로 생성