# Introduction

"쿼리의 검색속도를 높이기 위해서 반드시 알아야 하는 기술"

데이터 분석사 --> SQL

SQL 튜너 <--- 직업

SQL

|

3배차이

|

SQL 튜너 (월 1200만원부터 시작 ~ 2400만원 3.3%)

## SQL 튜닝 목차

1. 인덱스 튜닝
2. 조인 문장 튜닝
3. 서브쿼리 문장 튜닝

## 8 types of index access

1. Index range scan
2. Index unique scan
3. Index skip scan
4. Index full scan
5. Index fast full scan
6. Index merge scan
7. Index bitmap merge scan
8. Index join

# Index range scan

인덱스를 부분검색하는 스캔 방법

Select ename, sal

From emp

Where job='SAELSMAN';

Pb 1. 직업에 인덱스를 생성하고 직업의 인덱스의 구조가 어떻게 생겼는지 조회하시오!

CREATE INDEX emp\_job ON EMP(job);

SELECT job,ROWID

FROM EMP

WHERE job > ' ';

Machine generated alternative text:
8 
10 
11 
12 
13 
14 
JOB 
ANALYST AAAFKSAAEAAALDEAAJ 
ANALYST AAAFKSAAEAAALDEAAL 
CLERK 
AAAFKSAAEAAALDEAAH 
CLERK 
AAAFKSAAEAAALDEAAK 
CLERK 
AAAFKSAAEAAALDEAAM 
CLERK 
AAAFKSAAEAAALDEAAN 
MANAGEF AAAFKSAAEAAALDEAAE 
MANAGEF AAAFKSAAEAAALDEAAC 
MANAGEF AAAFKSAAEAAALDEAAD 
PRESIDE AAAFKSAAEAAALDEAAA 
SALESMA AAAFKSAAEAAALDEAAE 
SALESMA AAAFKSAAEAAALDEAAF 
SALESMA AAAFKSAAEAAALDEAAG 
SALESMA AAAFKSAAEAAALDEAAI 
8 
10 
11 
12 
13 
14 
ENAME 
AAAFKSAAEAAALDEAAA KING 
AAAFKSAAEAAALDSAAE 
aLAKE 
AAAFKSAAEAAALDEAAC 
CLARK 
AAAFKSAAEAAALDEAAD 
JONES 
AAAFKSAAEAAALDEAAE 
MARTIN 
AAAFKSAAEAAALDEAAF 
ALLEN 
AAAFKSAAEAAALDEAAG TURNER 
AAAFKSAAEAAALDEAAH 
JAMES 
AAAFKSAAEAAALDEAAI 
WARD 
AAAFKSAAEAAALDEAAJ 
FORD 
AAAFKSAAEAAALDEAAK 
SMITH 
AAAFKSAAEAAALDEAAL 
SCOTT 
AAAFKSAAEAAALDEAAM ADAMS 
AAAFKSAAEAAALDEAAN 
MILLER 
SAL 
JOB 
5000 PRESIDENT 
2850 MANAGER 
2450 MANAGER 
2975 MANAGER 
1250 SALESMAN 
1600 SALESMAN 
1500 SALESMAN 
950 CLERK 
1250 SALESMAN 
3000 ANALYST 
800 CLERK 
3000 ANALYST 
1100 CLERK 
1300 CLERK 

Pb 2. 사원이름에 인덱스를 걸고 아래의 SQL이 어느컬럼에 인덱스를 사용하는지 확인하세요.

SET autot traceonly EXPLAIN;

// 실행계획만 보겠다

SELECT ename,sal,job

FROM EMP

WHERE job='ANALYST' AND ename='ALLEN';

// 인덱스 빠른 쪽으로 엑세스함 (나는 아무것도 안탐)

Pb 3. 문제 2번에서 만약 emp\_ename 인덱스를 엑세스 하지 않고 emp\_job 인덱스를 했을때

인덱스 검색 범위가 더 작은 emp\_ename 인덱스를 타게 하려면 어떻게 해야하나?

Pb 4. 81년도에 입사했고 월급이 3000인 사원의 이름, 입사일, 월급을 출력하는데 월급과 입사일에 인덱스를 걸어서 쿼리를 작성하시오

SELECT hiredate,sal

FROM EMP

WHERE hiredate BETWEEN TO\_DATE('1981/01/01','RRRR/MM/DD') AND TO\_DATE('1981/12/31','RRRR/MM/DD') AND sal=3000;

Pb 5. 4번문제에서 월급의 인덱스를 엑세스 하지 못하고 입사일의 인덱스를 엑세스 했다면 월급의 인덱스를 엑세스를 할 수 있도록 힌트를 주시오.

SELECT /\*+ index(emp emp\_sal)\*/ hiredate, sal

FROM EMP

WHERE hiredate BETWEEN TO\_DATE('1981/01/01','RRRR/MM/DD') AND TO\_DATE('1981/12/31','RRRR/MM/DD') AND sal=3000;

Pb 6. 부서 번호와 커미션에 각각 인덱스를 생성하고 부서번호가 30번이고 커미션이 300인 사원의 이름, 월급, 커미션, 부서번호를 출력 하시오

SELECT ename,sal,comm

FROM EMP

WHERE deptno=30 AND COMM=300;

Set autot on

// 이렇게 하면 항상 같이 나옴

# Unique index scan

ALTER TABLE EMP

ADD CONSTRAINT emp\_empno\_pk PRIMARY KEY(empno);

Pb 7. 사원 테이블에 걸린 인덱스 리스트를 조회 하시오

SELECT uniqueness,index\_name

FROM user\_indexes

WHERE table\_name='EMP';

UNIQUENESS 
NONUNI QUE 
NONUNIQUE 
NONUNIQUE 
NONUNIQUE 
NONUNIQUE 
NONUNIQUE 
UNIQUE 
INDEX NAME 
EMP JOE 
EMP 
EMP 
EMP 
EMP 
EMP 
EMP 
ENAME 
HIREDATE 
SAL 
COMM 
DEPTO 
EMPNO PK 

Pb 8. 사원번호에 empno에 인덱스의 구조를 확인 하시오.

Machine generated alternative text:
SELECT empno, rowid 
FROM EMP 
WHERE empno>0; 
EMPNO 
ROWID 
7369 AAAFKSAAEAAALDEAAK 
7499 AAAFKSAAEAAALDEAAF 
7521 AAAFKSAAEAAALDEAAI 
7566 AAAFKSAAEAAALDEAAD 
7654 AAAFKSAAEAAALDEAAE 
7698 AAAFKSAAEAAALDEAAE 
7782 AAAFKSAAEAAALDEAAC 
7788 AAAFKSAAEAAALDEAAL 
7839 AAAFKSAAEAAALDEAAA 
7844 AAAFKSAAEAAALDEAAG 
7876 AAAFKSAAEAAALDEAAM 
7900 AAAFKSAAEAAALDEAAH 
7902 AAAFKSAAEAAALDEAAJ 
7934 AAAFKSAAEAAALDEAAN 

Pb 9. 사원번호가 7654인 사원의 사원 번호와 이름을 조회하는데 인덱스를 통해서 조회될 수 있도록 힌트를 주시오.

SELECT /\*+ index\_desc(emp emp\_empno) \*/ empno,rowid

FROM EMP

WHERE empno=7654;

囗 EMPNO ROWID 
7934 AAAFKSAAEAAALDSAAN 
7 岂 2 AAAFKSAAEAAALDEAA 」 
7 岂 O AAAFKSAAEAAALDEAAH 
787E AAAFKSAAEAAALDEAAM 
78 上 4 AAAFKSAAEAAALDEAAG 
7839 AAAFKSAAEAAALDEAAA 
7788 AAAFKSAAEAAALDEAAL 
7782 AAAFKSAAEAAALDEAAC 
7E98 AAAFKSAAEAAALDEAAE 
7E54 AAAFKSAAEAAALDEAAE 
7566 AAAFKSAAEAAALDEAAD 
7521 AAAFKSAAEAAALDEAAI 
7499 AAAFKSAAEAAALDEAAF 
73E9 AAAFKSAAEAAALDEAAK 

Pb 10. 9번 문제의 SQL이 실행될 때의 인덱스 엑세스와 테이블 엑세스를 그림으로 그리시오.

EMPNO 
ROWID 
7369 AAAFKSAAEAAALDEAAK 
7499 AAAFKSAAEAAALDEAAF 
7521 AAAFKSAAEAAALDEAAI 
7566 AAAFKSAAEAAALDEAAD 
7654 AAAFKSAAEAAALDEAAE 
7698 AAAFKSAAEAAALDEAAE 
7782 AAAFKSAAEAAALDEAAC 
7788 AAAFKSAAEAAALDEAAL 
7839 AAAFKSAAEAAALDEAAA 
7844 AAAFKSAAEAAALDEAAG 
7876 AAAFKSAAEAAALDEAAM 
7900 AAAFKSAAEAAALDEAAH 
7902 AAAFKSAAEAAALDEAAJ 
7934 AAAFKSAAEAAALDEAAN 
8 
10 
11 
12 
13 
14 
ROWID 
AAAFKSAAEAAALDSAAE 
AAAFKSAAEAAALDEAAC 
AAAFKSAAEAAALDEAAD 
AAAFKSAAEAAALDEAAE 
AAAFKSAAEAAALDEAAF 
AAAFKSAAEAAALDEAAG 
AAAFKSAAEAAALDEAAH 
AAAFKSAAEAAALDEAAI 
AAAFKSAAEAAALDEAAJ 
AAAFKSAAEAAALDEAAK 
AAAFKSAAEAAALDEAAL 
AAAFKSAAEAAALDEAAM 
AAAFKSAAEAAALDEAAN 
EMPNO 
ENAME 
7698 aLAKE 
7782 CLARK 
7566 JONES 
7654 MARTIN 
7499 ALLEN 
7844 TURNER 
7900 JAMES 
7521 WARD 
7902 FORD 
7369 SMITH 
7788 SCOTT 
7876 ADAMS 
7934 MILLER 
JOB 
MANAGER 
MANAGER 
MANAGER 
SALESMAN 
SALESMAN 
SALESMAN 
CLERK 
ANALYST 
ANALYST 
CLERK 
CLERK 

하나만 access 해서 빠르다

Pb 11. 사원 번호가 7654 이고 이름이 MARTIN인 사원의 사원 번호, 이름, 월급을 조회하고 사원 번호의 인덱스를 엑세스 했는지

사원 이름의 인덱스를 엑세스 했는지 확인 하시오.

SELECT empno,ename,sal

FROM EMP

WHERE empno=7654 AND ename='MARTIN';

// unique 인덱스를 먼저탄다

Primary key 제약이 걸렸다는것은 이미 데이터가 유니크 하다는게 보장이 된 상태라는 것이다.

Oper a bon 
SELECT STATEMENT optimizer Mode -ALL 
TABLE ACCESS BY INDEX ROWID 
INDEX UNIQUE SCAN 
Object Name 
ROWS 
SCOTT. EMP 
SCOTT.EMP EMPNO PK 

Pb 12. 문제 11번의 인덱스를 ename에 걸린 인덱스를 엑세스 하도록 힌트를 주고 실행하시오

SELECT /\*+ index\_desc(emp emp\_ename)\*/ empno,ename,sal

FROM EMP

WHERE empno=7654 AND ename='MARTIN';

Oper a bon 
SELECT STATEMENT optimizer Mode 
TABLE ACCESS ay INDEX ROWID 
INDEX RANGE SCAN DESCENDING 
Object Name 
SCOTT. EMP 
scon.EMP ENAME 

Pb 13. 위에꺼 그림

// unique scan 과 다르게 하나더 스캔한다

Machine generated alternative text:
ENAME 
ADAMS 
7839 KING 
ALLEN 
CLARK 
JAMES 
JONES 
MARTIN 
MILLER 
SCOTT 
TURNER 
WARD 
ROWID 
AAAFKSAAEAAALDEAAM 
AAAFKSAAEAAALDEAAF 
AAAFKSAAEAAALDEAAE 
AAAFKSAAEAAALDEAAC 
AAAFKSAAEAAALDEAAJ 
AAAFKSAAEAAALDEAAH 
AAAFKSAAEAAALDEAAD 
AAAFKSAAEAAALDEAAA 
AAAFKSAAEAAALDEAAE 
AAAFKSAAEAAALDEAAN 
AAAFKSAAEAAALDEAAL 
AAAFKSAAEAAALDEAAK 
AAAFKSAAEAAALDEAAG 
AAAFKSAAEAAALDEAAI 
8 
10 
11 
12 
13 
14 
AAAFKSAAEAAALDEAAA 
AAAFKSAAEAAALDEAAE 
AAAFKSAAEAAALDEAAC 
AAAFKSAAEAAALDEAAD 
AAAFKSAAEAAALDEAAE 
AAAFKSAAEAAALDEAAF 
AAAFKSAAEAAALDEAAG 
AAAFKSAAEAAALDEAAH 
AAAFKSAAEAAALDEAAI 
AAAFKSAAEAAALDEAAJ 
AAAFKSAAEAAALDEAAK 
AAAFKSAAEAAALDEAAL 
AAAFKSAAEAAALDEAAM 
AAAFKSAAEAAALDEAAN 
EMPNO 
ENAME 
7698 aLAKE 
7782 CLARK 
7566 JONES 
7654 MARTIN 
7499 ALLEN 
7844 TURNER 
7900 JAMES 
7521 WARD 
7902 FORD 
7369 SMITH 
7788 SCOTT 
7876 ADAMS 
7934 MILLER 
JOB 
MANAGER 
MANAGER 
MANAGER 
SALESMAN 
SALESMAN 
SALESMAN 
CLERK 
SALESMAN 
ANALYST 
ANALYST 
CLERK 
CLERK 

Pb 14. 직업이 SALESMAN 인 사원들의 이름과 월급과 직업을 출력하는 쿼리의 실행계획 그림을 인덱스와 테이블로 나눠서 그리시오

SELECT /\*+ index\_desc(emp emp\_job) \*/ename,sal,job

FROM EMP

WHERE job='SALESMAN';

(주의사항! Index range scan 그림이어야함)

Machine generated alternative text:
JOB 
ROWID 
8 
10 
11 
12 
13 
14 
ANALYST 
ANALYST 
MANAGER 
MANAGER 
MANAGER 
PRESIDENT 
SALESMAN 
SALESMAN 
SALESMAN 
SALESMAN 
AAAFKSAAEAAALDEAAJ 
AAAFKSAAEAAALDEAAL 
AAAFKSAAEAAALDEAAH 
AAAFKSAAEAAALDEAAK 
AAAFKSAAEAAALDEAAM 
AAAFKSAAEAAALDEAAN 
AAAFKSAAEAAALDEAAE 
AAAFKSAAEAAALDEAAC 
AAAFKSAAEAAALDEAAD 
AAAFKSAAEAAALDEAAA 
AAAFKSAAEAAALDEAAE 
AAAFKSAAEAAALDEAAF 
AAAFKSAAEAAALDEAAG 
AAAFKSAAEAAALDEAAI 
8 
10 
11 
12 
13 
14 
AAAFKSAAEAAALDEAAA KING 
AAAFKSAAEAAALDEAAE 
AAAFKSAAEAAALDEAAC 
AAAFKSAAEAAALDEAAD 
AAAFKSAAEAAALDEAAE 
AAAFKSAAEAAALDEAAF 
AAAFKSAAEAAALDEAAG TURNER 
AAAFKSAAEAAALDEAAH 
AAAFKSAAEAAALDEAAI 
AAAFKSAAEAAALDEAAJ 
AAAFKSAAEAAALDEAAK 
AAAFKSAAEAAALDEAAL 
AAAFKSAAEAAALDEAAM ADAMS 
AAAFKSAAEAAALDEAAN 
ENAME 
aLAKE 
CLARK 
JONES 
MARTIN 
ALLEN 
JAMES 
WARD 
FORD 
SMITH 
SCOTT 
MILLER 
SAL 
JOB 
5000 PRESIDENT 
2850 MANAGER 
2450 MANAGER 
2975 MANAGER 
1250 SALESMAN 
1600 SALESMAN 
1500 SALESMAN 
950 CLERK 
1250 SALESMAN 
3000 ANALYST 
800 CLERK 
3000 ANALYST 
1100 CLERK 
1300 CLERK 

# Index skip scan

결합 인덱스에 두번째 칼럼에 인덱스를 쓰는방법이다. 첫번째 칼럼의 distinct 개수가 적을때 성능을 향상시킬수 있는 방법이다.

Index skip scan 을 이해하려면 결합 컬럼 인덱스에 대해서 이해가 먼저 되어야 한다.

Pb 14. Demobld.sql 수행후 결합컬럼 인덱스 생성

Pb 15. 방금만든 emp\_deptno\_sal의 인덱스의 구조가 어떻게 되었는지 확인하시오

SELECT deptno,sal,rowid

FROM EMP

WHERE deptno>0;

Pb 16. 부서번호가 20번인 사원들의 이름과 월급과 부서번호를 출력하는데 emp\_deptno\_Sal 인덱스를 엑세스 할 수 있도록 힌트를 주시오

SELECT /\*+ index\_desc(emp emp\_deptno\_sal) \*/ ename,sal,deptno

FROM EMP

WHERE deptno=20;

Pb 17. 월급이 3000인 사원의 이름, 월급을 출력하는데 실행 계획에 emp\_deptno\_sal인덱스를 엑세스 하는지 확인해보시오.

SELECT ename,sal

FROM EMP

WHERE sal=3000;

SELECT STATEMENT optimizer Mode=ALL ROWS 
TABLE ACCESS FULL 

왜 인덱스 emp\_deptno\_sal 인덱스를 엑세스 하지 못하고 full table scan 을 하는가?

결함컬럼 인덱스의 첫번째 컬럼이 where 절에 검색조건으로 존재해야 그 인덱스를 엑세스 할수 있다.

이런 경우에 emp\_deptno\_sal 인덱스를 엑세스 하게 하려면 index skip scan을 사용해야 한다.

두번째 index 칼럼에서 인덱스 사용한다.

//두번째 칼럼에만 인덱스 적용

SELECT /\*+index\_ss(emp emp\_deptno\_sal)\*/ ename,sal

FROM EMP

WHERE sal=1100;

인덱스 스킵 스캔의 효과를 보기위한 조건

Pb 18. 직업 + 월급으로 결합 컬럼 인덱스를 생성하고 아래의 SQL이 index skip scan 할 수 있도록 힌트를 주시오.

Select ename, sal, job, hiredate

From emp

Where sal=1250;

CREATE INDEX emp\_job\_sal ON EMP(job,sal);

SELECT /\*+ index\_ss(emp emp\_job\_sal) \*/ job,sal

FROM EMP

WHERE sal=1250;

### Index skip scan 효과를 보기위한 조건

결합 컬럼 인덱스의 첫번째 컬럼의 데이터의 종류가 몇가지 안되어야 효과를 볼 수 있다.

/ *+ index_ss (emp 든mp-c이1-c이2) 
SELECT 
F RC`1 EMP 
WHERE C012 = 30』 
결합 걸뤔 인택人이 젓번재 걸뤔의 데이터의 종류가 
몇가지 안되어야 효과를 몰 수 있다 
젓 번개 걸뤔의 데이터가 A-Z 까지 있는 경우와 A-C 
까지 있는 경우 들중에 어떤계 더 성능이 좋을 
A-C가 더 성능이 좋다 - 
1- COLI의 Ae 들어간뒤 COQ가 30 될때까지 검색을 하고 
견한뒤 그 다음 ROW도 확인한뒤 나감 
2- COLI의 8로 들어간뒤 COQ가 30을 틸견할때까지 검색。 
그 다음 걸뤔까지 30번이 있는지 확인하고 나감 
로 COLI의 들어간뒤 COQ가 30이 나을때까지 검색 
켠다면 그다뮴까지 검사를 하나 ROW가 끝나면 그냥 나음 
COLI 
A 
A 
A 
A 
8 
8 
8 
8 
8 
8 
C 
C 
COQ 
20 
30 
40 
20 
30 
40 
50 
60 
20

COL1 의 A~C가 더 성능이 좋은 이유는 A~Z면은 들어갔다가 나왔다 하는 숫자가 많아져서 느려지기 때문이다.

같은 ROW수의 A~Z가 COL1에 있다고 한다면 접근 횟수가 그만큼 늘어나기 때문에 느려진다.

Pb 21. 사원테이블의 사원번호에 인덱스를 생성하고 사원의 인원수가 몇명인지 카운트 하시오

# Index full scan

Index leaf block 전체를 순차적으로 access 하는 operation.

일반적으로 index full scan operation 이 발생을 하게 되면 SQL성능은 떨어지게 된다

!!Stopkey 활용 중요

SELECT \*

FROM (

SELECT /\*+ index\_desc(emp000 emp000\_sal)\*/ ename,sal,dense\_RANK()OVER (ORDER BY sal desc) AS 순위

FROM EMP000

WHERE sal>0

)

WHERE 순위=1;

Oper a bon 
SELECT STATEMENT optimizer Mode 
WINDOW NOSORT STOPKEY 
TABLE ACCESS BY INDEX ROWID 
INDEX RANGE SCAN DESCENDING 
Object Name 
scon.EMP000 
SCOTT.EMPOOO SAL 
E ytes 
57 M 
57 M 
17M 
17M 
Cost 
132861 
132861 
132861 
132861 

Machine generated alternative text:
Pr oper bes 
recursive calls 
db block gets 
consistent gets 
physical reads 

Stopkey 를 쓰도록 유도하면 인덱스에서 정렬을하고 위에것만 불러와서 대단히 빠를수있다.

Pb 20. 사원 테이블의 인원수가 몇명인지 카운트 하시오

SELECT COUNT(empno) FROM EMP;

SELECT COUNT(\*) FROM EMP;

SELECT COUNT(1) FROM EMP;

// three have equal performance (just automatically replaced with the first one)

Pb 21. 인덱스 걸고 스캔하시오

ALTER table EMP

ADD CONSTRAINT emp\_empno\_pk PRIMARY KEY(empno);

SELECT COUNT(\*)

FROM EMP;

Oper a bon 
SELECT STATEMENT opbmizer Mode 
SORT AGGREGATE 
INDEX FULL SCAN 
Object Name 
-ALL ROWS 
SCOTT.EMP EMPNO PK 

Index full scan 이 table full scan 보다 성능이 좋다

Pb 22. 문제 21번을 다시 수행하는데 테이블 full scan 으로 수행되게 하시오

Select /\*+ full(emp) \*/ count(empno)

From emp;

Pr oper bes 
I recursive calls 
2 db block gets 
3 consistent gets 

Pb 23. 다시 index full scan 힌트를 주고 실행한후에 몇개의 블럭을 읽었는지 확인하시오

Select /\*+ index full(emp) \*/ count(empno)

From emp;

Pr oper bes 
I recursive calk 
2 db block gets 
3 consistent gets 

# Index fast full scan

인덱스 full 스캔보다 더 성능이 좋은 스캔 -> **병렬처리**

Index fast full scan은 full table scan 과 마찬가지로 multi block I/O 를 한다. Multi block I/O 하기 때문에 데이터 출력 순서를 전혀 보장하지 않는다.

Index full scan은 Index 컬럼만 SQL에 포함되어 있을떄 발생할수있다.

--> 데이터 스캔 범위가 넓으면서 인덱스가 포함되있는 컬럼만 엑세스할때 유용하게 쓰일수 있다.

Pb 24.

Select /\*+ index\_ffs(emp emp\_empno\_pk) \*/ count(empno)

From emp;

Pr oper bes 
I recursive calls 
2 db block gets 
3 consistent gets 

Pb 25. 직업, 직업별 인원수를 출력하는데 ffs 인덱스를 이용해서 수행.

CREATE INDEX emp\_job\_empno ON EMP(job,empno);

SELECT /\*+ index\_ffs(emp emp\_job\_empno)\*/ job, COUNT(\*)

FROM EMP

GROUP BY job;

ALTER TABLE EMP

MODIFY job NOT NULL;

Pb 26. 부서번호와 부서번호별 인원수를 출력하는데 가장 빠르게 출력될 수 있도록 적절한 인덱스를 생성하고 힌트를 주고 수행.

CREATE INDEX emp\_deptno\_empno ON EMP(deptno,empno);

SELECT /\*+ index\_ffs(emp emp\_deptno\_empno) \*/ deptno,COUNT(empno)

FROM EMP

Group by deptno;

Pb 27. 26번SQL을 병렬처리 할 수 있도록 힌트를 주고 실행

CREATE INDEX emp\_deptno\_empno ON EMP(deptno,empno);

SELECT /\*+ index\_ffs(emp emp\_deptno\_empno) parallel\_index(emp, emp\_deptno\_empno,4) \*/

deptno, COUNT(\*)

FROM EMP

GROUP BY deptno;

Optimizer 
Default 
Operatio 
SELECT STATEMENT optimizer Mode=ALL 
PX COORDINATOR 
PX SEND QC (RANDOM) 
HASH GROUP ay 
PX RECENE 
PX SEND HASH 
HASH GROUP ay 
_—Cost_Obiect 
ROWS 
SYS.:TQIOOOI 
SYS.:TQIOOOO 
SCOTT.EMP DEPTNO 
EMPNO 
14 
182 
182 
182 
182 
182 
182 
182 
182 
6 
6 
6 
6 
6 
2 
2 
:QIOOI 
:QIOOI 
:QIOOI 
:QIOOO 
:QIOOO 
:QIOOO 
:QIOOO 
pcwp 
pcwp 
pcwp 
pcwc 
QC (RANDOM) 
HAS H 
PX BLOCK ITERATOR 
INDEX FAST FULL SCAN 

Show parameter cpu\_count

Pb 28. 이름에 EN또는 IN을 포함하고 있는 사원들의 이름, 월급, 직업을 출력

SELECT ename,sal,job

FROM EMP

WHERE ename LIKE '%EN%' OR ename LIKE '%IN%' ;

Pb 29. 위에꺼 튜닝

SELECT /\*+ index\_desc(emp emp\_ename) \*/ename,sal,job

FROM EMP

WHERE ename LIKE '%EN%' OR ename LIKE '%IN%' or ename> ' ';

SELECT /\*+ index\_ffs(emp emp\_ename) \*/ rowid

FROM EMP

WHERE ename LIKE '%EN%' OR ename LIKE '%IN%';

Machine generated alternative text:
Oper a bon 
SELECT STATEMENT optimizer Mode 
INDEX FAST FULL SCAN 
Object Name 
-ALL ROWS 
SCOTT. EMP 
B ytes 
ENAME 
Cost 

위에서 구한 rowid 3건을 가지고 해당 rowid의 테이블의 이름, 월급, 직업을 출력

SELECT ename,sal,job

FROM EMP

WHERE ROWID IN (

SELECT /\*+ index\_ffs(emp emp\_ename) \*/ rowid

FROM EMP

WHERE ename LIKE '%EN%' OR ename LIKE '%IN%');

Machine generated alternative text:
SELECT STATEMENT optimizer Mode 
TABLE ACCESS FULL 
SCOTT. EMP 

Machine generated alternative text:
ENAME 
MARTIN 
SAL 
JOE 
5000 PRESIDENT 
1 D5Q 
SALESMAN 
1600 SALESMAN 

SELECT /\*+ leading (e3 e1) use\_nl(e1) rowid(e1)\*/

e1.ename, e1.sal, e1.job

FROM EMP e1,

(SELECT /\*+ index\_ffs(e2 emp\_ename) no\_merge\*/ ROWID rr

FROM EMP e2

WHERE ename LIKE '%EN%'

OR ename LIKE '%IN%')e3

WHERE E1.ROWID = e3.rr;

Pb 30. 이름에 EN또는 IN을 포함하고 있는 사원들의 이름, 월급, 직업을 출력하는데 정규식을 사용해서 출력하면 성능이 더 좋아지는지 확인 해보시오

SELECT ename,sal,job

FROM EMP

WHERE regexp\_like(ename,'(EN|IN)');

# Index merge scan

Where 에 조건이 여러개가 있고 그 조건에 사용되는 컬럼이 다 단일컬럼 인덱스로 구성되어 있을때 하나의 인덱스를 사용하는게 아니라 여러개의 인덱스를 동시에 사용해서 시너지 효과를 보는 스캔방법.

Pb 31. demobld.sql을 돌리고 직업이 salesman 이고 부서번호가 30번인 사원들의 이름과 월급과 직업, 부서번호를 출력하시오

SELECT ename,sal,job,deptno

FROM EMP

WHERE job='SALESMAN' AND DEPTNO=30;

Machine generated alternative text:
Pr oper bes 
I recursive calls 
2 db block gets 
3 consistent gets 

Pb 32. 직업과 부서번호에 각각 단일 컬럼 인덱스를 생성하고 문제 31번 쿼리의 실행계획을 보면 직업과 부서번호 2개중에 어느 컬럼에 인덱스를 엑세스 하겠는가?

Mostaly job

Pb 33. 그렇다면 왜 deptno에 인덱스를 타지 않고 job의 인덱스를 탔을까?

Job 이 6개라서 deptno 4 라서? I don't agree a bit.

Pb 34. 그러면 emp\_dept 인덱스를 엑세스 하겠금 힌트를 주고 실행

SELECT /\*+ index(emp000 emp\_deptno)\*/ename,sal,job,deptno

FROM EMP

WHERE job='SALESMAN' AND DEPTNO=30;

Machine generated alternative text:
Pr oper bes 
I recursive calls 
2 db block gets 
3 consistent gets 

Pb 35. 아래의 SQL이 index merge scan 이 될수 있도록 힌트를 주시오

Select /\*+ and\_equal (emp emp\_job emp\_deptno) \*/ ename, sal, job, deptno

from emp

Where job='SALESMAN' and deptno=30;

Machine generated alternative text:
Operation 
SELECT STATEMENT optimizer 
TABLE ACCESS ay INDEX Row 
AND-EQUAL 
INDEX RANGE SCAN 
INDEX RANGE SCAN 
Object 
mod 
ID 
ROWS 
Name 
DEPTNO 
Joa 
Rows 
4 
4 
Byte s 
156 
156 
st 
SCOTT 
SCOTT 
SCOTT 
. EMP 
. EMP 
. EMP 

Pr oper bes 
recursive calls 
db block gets 
consistent gets 
physical reads 

# Index bitmap merge( combination)

Index merge scan 의 진화된 기술로 여러개의 인덱스를 각각 bitmap으로 변환해서 하나로 합쳐서 구한 rowid로 테이블을 엑세스 하는 스캔방법.

Insert 절의 컬럼이 모두 인덱스에 등록되어 있지 않아도 됨

너무 많은 인덱스를 조합하면 오히려 성능이 떨어질수 있으니 주의

SELECT /\*+ index\_combin(emp) \*/

ename,sal,job,deptno

FROM EMP

WHERE job='SALESMAN' AND deptno=30;

//위와 같이 지정하게 되면 테이블내의 모든 인덱스를 조합한다

Pb 36. 우리반 테이블에 전공과 통신사에 각각 단일 컬럼 인덱스를 걸고 전공이 경제학이고 통신사가 SK인 학생들의 이름과 전공과 통신사를 출력하는 쿼리의 실행계획이 index bitmap merge scan이 되도록 하시오

SELECT /\*+ index\_combin(emp2) \*/ename,major,telecom

FROM EMP2

WHERE major='경제학' AND telecom='sk'

# Index join

테이블을 엑세스 하지 않으면서 인덱스만 가지고 조인을 해서 결과를 보여주는 스캔방법

Where 절과 insert 절에 컬럼이 모두 인덱스에 등록되어 있어야함

---> 조회하는 범위가 많지 않으면서 인덱스 안에있는것만 조합하여 조회할때

Select empno, ename

From emp

Where empno=7788;

'peration 
SELECT STATEMENT optimizer 
mod 
TABLE ACCESS ay INDEX ROWID 
INDEX RANGE SCAN 
ROWS 
Object 
Name 
SCOTT. EMP 
I SCOTT. EMPNO 
Rows 
Bytes C 

Pb 37. 아래의 SQL을 index join 엑세스 방법으로 힌트를 줘서 테이블 엑세스를 하지 말게 하시오

Select /\*+ index\_join(emp)\*/ empno, ename

From emp

Where empno=7788;

안됨

Pb 38. empno와 ename에 각각 not null 제약을 걸고 실행계획을 확인하시오

Select /\*+ index\_join(emp emp\_empno emp\_ename)\*/ empno, ename

From emp

Where empno=7788 AND empno IS NOT NULL AND ename IS NOT NULL;

Tree Vien Text Flow Chart 
Oper a bon 
SELECT STATEMENT optimizer Mode 
HASH JOIN 
INDEX RANGE SCAN 
INDEX FAST FULL SCAN 
E ytes 
Cost 
SCOTT. 
SCOTT. EMP EMPNO 
scon.EMP ENAME 

# Sum up

Pb 39. 전공이 통계학이고 나이가 20대인 학생의 학생번호, 이름과 전공과 나이를 출력하는 쿼리를 작성하는데 쿼리의 성능이 높아지도록 적절하게 인덱스를 생성하고 인덱스를 잘 탈 수 있도록 힌트를 사용해서 SQL을 작성하시오

여러칼럼 인덱스 걸어도됨

Create index emp2\_indx2 on emp2(major,age,empno, ename);

Consisent block 줄이는 방법 (별 도움안됨)

SELECT e.empno,e.EMAIL

FROM EMP2 e

WHERE e.major='통계학'

AND e.age LIKE '2%';

Pb 40. 아래의 sql을 튜닝하시오

CREATE INDEX emp2\_ename ON emp2(ename);

SELECT ename,age,major

FROM EMP2

WHERE SUBSTR(ename,1,1)='김';

---------------------------------------------------------

SELECT ename,age,major

FROM EMP2

WHERE ename LIKE '김%';

Pb 41. 아래의 SQL을 튜닝

SELECT ename,age,major

FROM EMP2

WHERE TO\_CHAR(birth,'RR/MM/DD') ='92/05/25';

----------------------------------------------------------------------------

SELECT ename,age,major

FROM EMP2

WHERE birth =TO\_DATE('1992/05/25','RRRR/MM/DD');

## Index combine 과 index join의 차이점

INDEX COMBINE은 SELECT 절의 컬럼과 WHERE 절의 컬럼이 모두 INDEX 에 존재하지 않아도됨

INDEX\_JOIN의 경우가 속도면에서 조금더 유리함