실무에서 바로 쓰는

SQL 튜닝 방법 20가지

이럴때는

방법1. SQL 의 실행계획을 확인하자 !

▣ 학습 내용

1. SQL의 실행계획이 무엇인지 학습합니다.

2. 예상 실행계획이 무엇인지 학습합니다.

3. 실제 실행계획이 무엇인지 학습합니다.

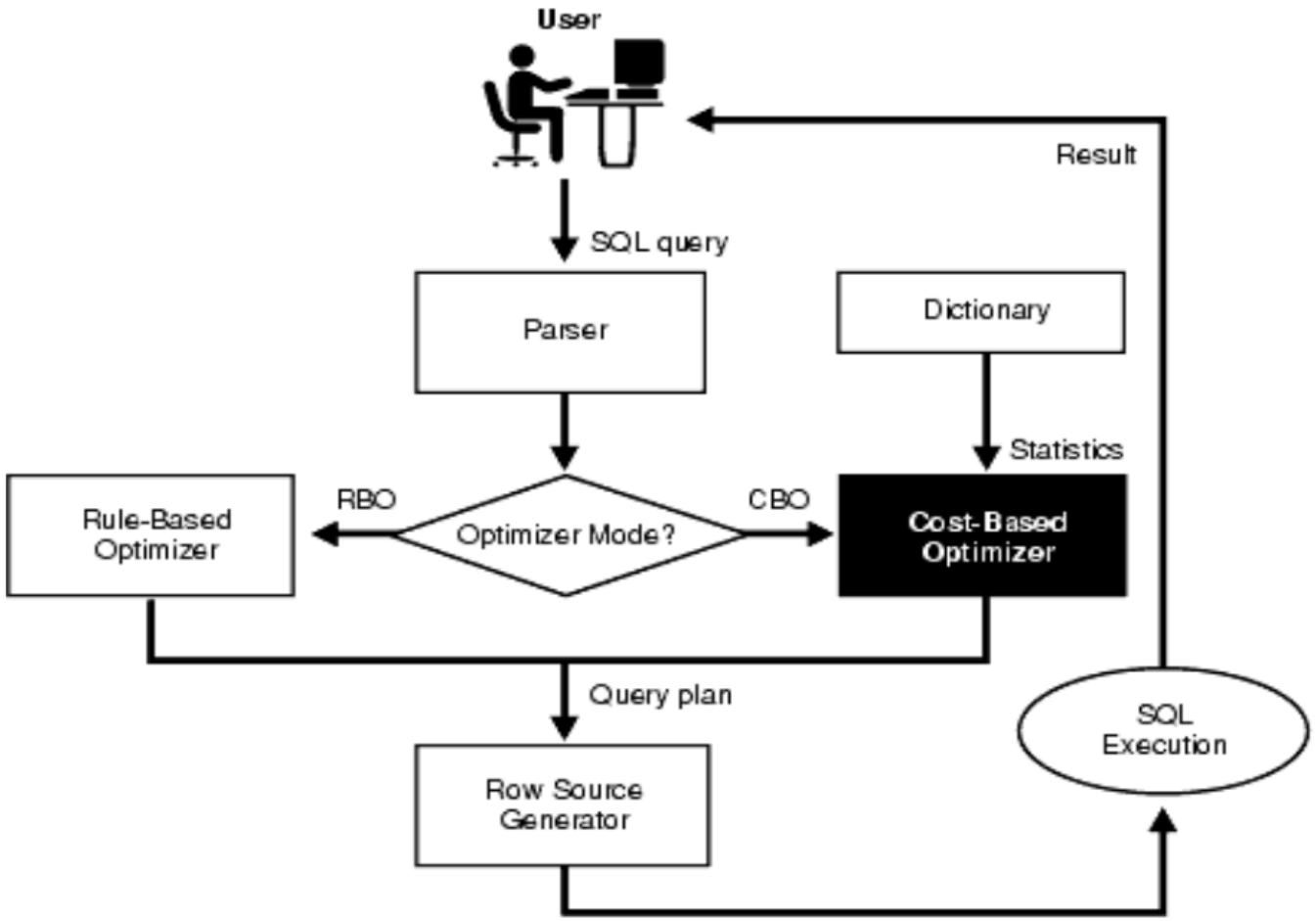
4. 실행계획을 제어하는 힌트가 무엇인지 학습합니다.

▣ 학습 목표

SQL의 실행계획을 보며 SQL을 튜닝할 수 있습니다.

1. SQL의 실행계획이란?

" SQL문을 실행하기전에 내부적으로 생성한 SQL실행 계획 "



2. 예상 실행계획이란?

" SQL문을 실행하기전에 만든 예상 계획 "

```
explain plan for
   select  ename, sal
     from emp
     where sal = 1300;

select * from table( dbms_xplan.display );
```

```
---------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |     1 |    20 |     3   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| EMP  |     1 |    20 |     3   (0)| 00:00:01 |
---------------------------------------------------------------------------
```

3. 실제 실행계획이란?

" SQL을 실행할 때 사용했던 실행계획 "

```
select /*+ gather_plan_statistics */ ename, sal
    from emp
    where sal = 1300;
```

```
SELECT *
 FROM TABLE(dbms_xplan.display_cursor(null,null,'ALLSTATS LAST'));
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		1	00:00:00.01	7
* 1	TABLE ACCESS FULL	EMP	1	1	1	00:00:00.01	7

4. 힌트(hint) 란 무엇인가?

" SQL을 실행할 때 옵티마이져로 하여금 힌트데로 실행계획을 생성해달라고 주문 "

```
select /*+ gather_plan_statistics full(emp) */ ename, sal
    from emp
    where sal = 1300;
```

힌트

```
SELECT *
 FROM TABLE(dbms_xplan.display_cursor(null,null,'ALLSTATS LAST'));
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		1	00:00:00.01	7
* 1	TABLE ACCESS FULL	EMP	1	1	1	00:00:00.01	7

5. 힌트(hint) 의 종류

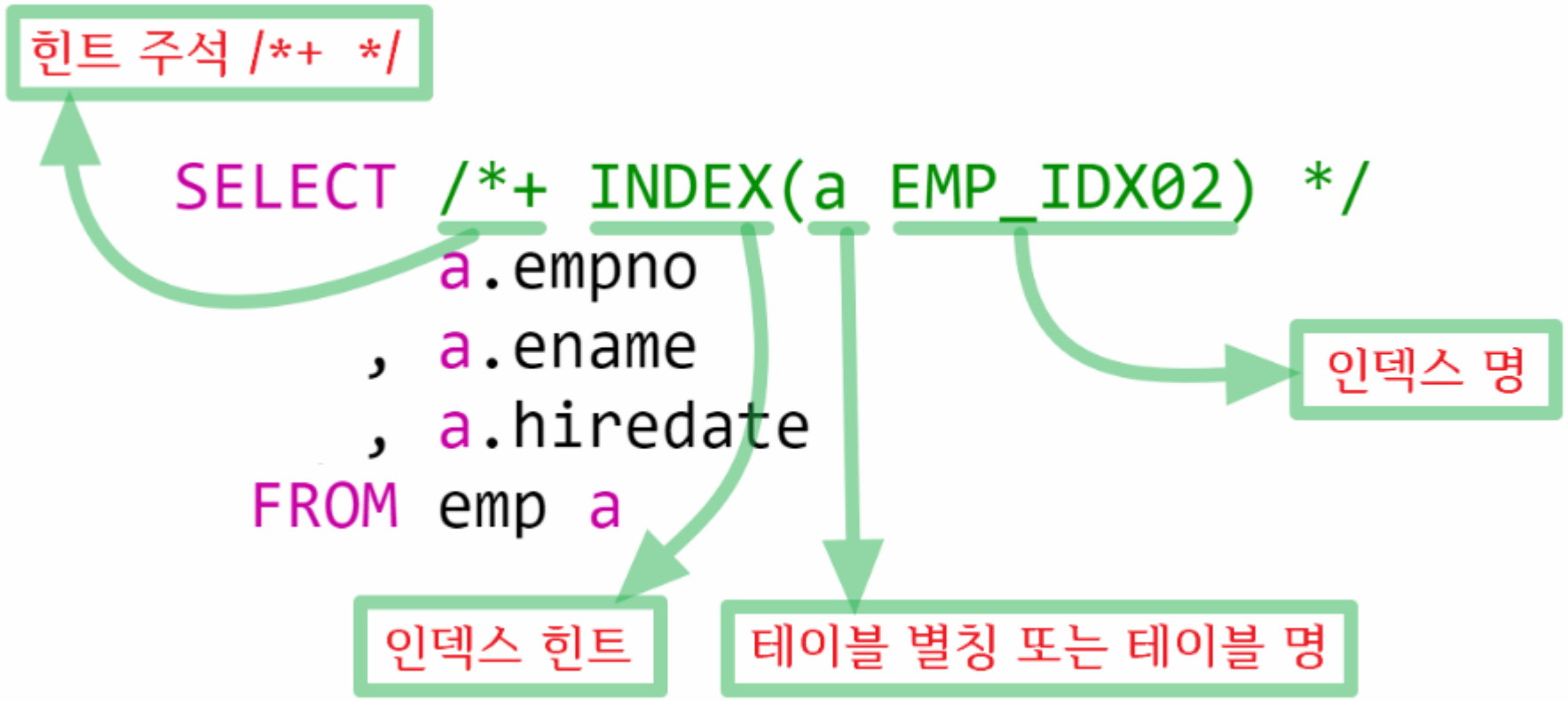


그림 출처: https://gent.tistory.com/306

먼저

방법2. 인덱스의 구조를 이해하자 !

▣ 학습 내용

1. ROWID 가 무엇인지 학습합니다.

2. 숫자형 컬럼의 인덱스의 구조를 살펴봅니다.

3. 문자형 컬럼의 인덱스의 구조를 살펴봅니다.

4. 날짜형 컬럼의 인덱스의 구조를 살펴봅니다.

▣ 학습 목표

인덱스의 구조를 이해하고 인덱스를 사용하는 방법을 학습합니다.

1. ROWID 란?

사원 테이블

rowid	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
AAAaXtAANAAAAC+AAA	7839	KING	PRESIDENT		1981-11-17	5000		10
AAAaXtAANAAAAC+AAB	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
AAAaXtAANAAAAC+AAC	7782	CLARK	MANAGER	7839	1981-05-09	2450		10
AAAaXtAANAAAAC+AAD	7566	JONES	MANAGER	7839	1981-04-01	2975		20
AAAaXtAANAAAAC+AAE	7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
AAAaXtAANAAAAC+AAF	7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
AAAaXtAANAAAAC+AAG	7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
AAAaXtAANAAAAC+AAH	7900	JAMES	CLERK	7698	1981-12-11	950		30
AAAaXtAANAAAAC+AAI	7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
AAAaXtAANAAAAC+AAJ	7902	FORD	ANALYST	7566	1981-12-11	3000		20
AAAaXtAANAAAAC+AAK	7369	SMITH	CLERK	7902	1980-12-09	800		20
AAAaXtAANAAAAC+AAL	7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
AAAaXtAANAAAAC+AAM	7876	ADAMS	CLERK	7788	1983-01-15	1100		20
AAAaXtAANAAAAC+AAN	7934	MILLER	CLERK	7782	1982-01-11	1300		10

2. 숫자형 컬럼의 인덱스는?

월급 데이터 검색을 위한 인덱스

⇩

SAL	ROWID
800	AAATc1AAHAAAAHeAAK
950	AAATc1AAHAAAAHeAAH
1100	AAATc1AAHAAAAHeAAM
1250	AAATc1AAHAAAAHeAAE
1250	AAATc1AAHAAAAHeAAI
1300	AAATc1AAHAAAAHeAAN
1500	AAATc1AAHAAAAHeAAG
1600	AAATc1AAHAAAAHeAAF
2450	AAATc1AAHAAAAHeAAC
2850	AAATc1AAHAAAAHeAAB
2975	AAATc1AAHAAAAHeAAD
3000	AAATc1AAHAAAAHeAAJ
3000	AAATc1AAHAAAAHeAAL
5000	AAATc1AAHAAAAHeAAA

사원 테이블

ROWID	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		1981-11-17	5000		10
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	1981-05-09	2450		10
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	1981-04-01	2975		20
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	1981-12-11	950		30
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	1981-12-11	3000		20
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	1980-12-09	800		20
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	1983-01-15	1100		20
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	1982-01-11	1300		10

3. 문자형 컬럼의 인덱스는 ?

이름 데이터 검색을 위한 인덱스

⇩

ENAME	ROWID
ADAMS	AAATc1AAHAAAAHeAAM
ALLEN	AAATc1AAHAAAAHeAAF
BLAKE	AAATc1AAHAAAAHeAAB
CLARK	AAATc1AAHAAAAHeAAC
FORD	AAATc1AAHAAAAHeAAJ
JAMES	AAATc1AAHAAAAHeAAH
JONES	AAATc1AAHAAAAHeAAD
KING	AAATc1AAHAAAAHeAAA
MARTIN	AAATc1AAHAAAAHeAAE
MILLER	AAATc1AAHAAAAHeAAN
SCOTT	AAATc1AAHAAAAHeAAL
SMITH	AAATc1AAHAAAAHeAAK
TURNER	AAATc1AAHAAAAHeAAG
WARD	AAATc1AAHAAAAHeAAI

사원 테이블

ROWID	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		1981-11-17	5000		10
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	1981-05-09	2450		10
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	1981-04-01	2975		20
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	1981-12-11	950		30
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	1981-12-11	3000		20
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	1980-12-09	800		20
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	1983-01-15	1100		20
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	1982-01-11	1300		10

4. 날짜형 컬럼의 인덱스는?

입사일 데이터 검색을 위한 인덱스

⇩

HIREDATE	ROWID
1981-11-17	AAAa3aAANAAAADOAAA
1981-05-01	AAAa3aAANAAAADOAAB
1981-05-09	AAAa3aAANAAAADOAAC
1981-04-01	AAAa3aAANAAAADOAAD
1981-09-10	AAAa3aAANAAAADOAAE
1981-02-11	AAAa3aAANAAAADOAAF
1981-08-21	AAAa3aAANAAAADOAAG
1981-12-11	AAAa3aAANAAAADOAAH
1981-02-23	AAAa3aAANAAAADOAAI
1981-12-11	AAAa3aAANAAAADOAAJ
1980-12-09	AAAa3aAANAAAADOAAK
1982-12-22	AAAa3aAANAAAADOAAL
1983-01-15	AAAa3aAANAAAADOAAM
1982-01-11	AAAa3aAANAAAADOAAN

사원 테이블

ROWID	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		1981-11-17	5000		10
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	1981-05-09	2450		10
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	1981-04-01	2975		20
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	1981-12-11	950		30
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	1981-12-11	3000		20
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	1980-12-09	800		20
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	1983-01-15	1100		20
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	1982-01-11	1300		10

이럴때는

방법3. index range scan 으로 유도하자 !

▣ 학습 내용

1. 숫자형 컬럼 인덱스 range scan 을 학습합니다.

2. 문자형 컬럼 인덱스 range scan 을 학습합니다.

3. 중복된 데이터가 있는 컬럼의 인덱스 range scan을 학습합니다.

▣ 학습 목표

인덱스 range scan 으로 SQL을 튜닝할 수 있습니다.

인덱스 스캔 방법 7가지

	인덱스 엑세스 방법	관련 힌트
1	index range scan	index
2	index unique scan	index
3	index skip scan	index_ss
4	index full scan	index_fs
5	index fast full scan	index_ffs
6	index merge scan	and_equal
7	index bitmap merge scan	index_combine

1. 숫자형 컬럼 인덱스 range scan

```
select /*+ index(emp emp_sal) */ ename, sal
 from emp
 where sal = 1600;
```

emp_sal 인덱스

SAL	ROWID
800	AAATc1AAHAAAAHeAAK
950	AAATc1AAHAAAAHeAAH
1100	AAATc1AAHAAAAHeAAM
1250	AAATc1AAHAAAAHeAAE
1250	AAATc1AAHAAAAHeAAI
1300	AAATc1AAHAAAAHeAAN
1500	AAATc1AAHAAAAHeAAG
1600	AAATc1AAHAAAAHeAAF
2450	AAATc1AAHAAAAHeAAC
2850	AAATc1AAHAAAAHeAAB
2975	AAATc1AAHAAAAHeAAD
3000	AAATc1AAHAAAAHeAAJ
3000	AAATc1AAHAAAAHeAAL
5000	AAATc1AAHAAAAHeAAA

emp 테이블

ROWID	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		1981-11-17	5000		10
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	1981-05-09	2450		10
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	1981-04-01	2975		20
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	1981-12-11	950		30
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	1981-12-11	3000		20
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	1980-12-09	800		20
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	1983-01-15	1100		20
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	1982-01-11	1300		10

2. 문자형 컬럼 인덱스 range scan

```
select /*+ index(emp emp_ename) */ ename, sal
 from emp
 where ename='SCOTT';
```

emp_ename 인덱스

ENAME	ROWID
ADAMS	AAATc1AAHAAAAHeAAM
ALLEN	AAATc1AAHAAAAHeAAF
BLAKE	AAATc1AAHAAAAHeAAB
CLARK	AAATc1AAHAAAAHeAAC
FORD	AAATc1AAHAAAAHeAAJ
JAMES	AAATc1AAHAAAAHeAAH
JONES	AAATc1AAHAAAAHeAAD
KING	AAATc1AAHAAAAHeAAA
MARTIN	AAATc1AAHAAAAHeAAE
MILLER	AAATc1AAHAAAAHeAAN
SCOTT	AAATc1AAHAAAAHeAAL
SMITH	AAATc1AAHAAAAHeAAK
TURNER	AAATc1AAHAAAAHeAAG
WARD	AAATc1AAHAAAAHeAAI

emp 테이블

ROWID	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		1981-11-17	5000		10
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	1981-05-09	2450		10
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	1981-04-01	2975		20
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	1981-12-11	950		30
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	1981-12-11	3000		20
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	1980-12-09	800		20
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	1983-01-15	1100		20
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	1982-01-11	1300		10

3. 중복된 데이터가 있는 컬럼 index range scan

```
select /*+ index(emp emp_job) */ ename, sal
 from emp
 where job='MANAGER';
```

emp_job 인덱스

JOB	ROWID
ANALYST	AAATc1AAHAAAAHeAAJ
ANALYST	AAATc1AAHAAAAHeAAL
CLERK	AAATc1AAHAAAAHeAAH
CLERK	AAATc1AAHAAAAHeAAK
CLERK	AAATc1AAHAAAAHeAAM
CLERK	AAATc1AAHAAAAHeAAN
MANAGER	AAATc1AAHAAAAHeAAB
MANAGER	AAATc1AAHAAAAHeAAC
MANAGER	AAATc1AAHAAAAHeAAD
PRESIDENT	AAATc1AAHAAAAHeAAA
SALESMAN	AAATc1AAHAAAAHeAAE
SALESMAN	AAATc1AAHAAAAHeAAF
SALESMAN	AAATc1AAHAAAAHeAAG
SALESMAN	AAATc1AAHAAAAHeAAI

emp 테이블

ROWID	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		1981-11-17	5000		10
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	1981-05-09	2450		10
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	1981-04-01	2975		20
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	1981-12-11	950		30
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	1981-12-11	3000		20
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	1980-12-09	800		20
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	1983-01-15	1100		20
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	1982-01-11	1300		10

이럴때는

방법4. where 절의 좌변을 가공하지 마라 !

▣ 학습 내용

1. 숫자형 컬럼 인덱스가 가공 되었을때 튜닝방법을 학습합니다.

2. 문자형 컬럼 인덱스가 가공 되었을때 튜닝방법을 학습합니다.

3. 날짜형 컬럼 인덱스가 가공 되었을때 튜닝방법을 학습합니다.

▣ 학습 목표

인덱스 컬럼이 가공된 SQL을 튜닝할 수 있습니다.

1. WHERE 절의 인덱스 컬럼이 가공되었다면?

튜닝전:

```
SQL> SELECT  ename, sal*12
             FROM  emp
             WHERE   sal * 12 = 36000;
```

인덱스 컬럼을 가공하지 말아라 !

2. 숫자형 컬럼에 인덱스를 생성하기

아래의 사원 테이블에 월급(sal) 에 인덱스를 생성하세요

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

3. 숫자형 컬럼 인덱스 컬럼이 가공되었다면?

연봉(sal * 12) 이 36000 인 사원들의 이름과 연봉을 출력하세요

튜닝전:

```
SQL> SELECT /*+ gather_plan_statistics */
            ename, sal*12
        FROM  emp
        WHERE   sal * 12 = 36000;
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		2	00:00:00.01	7
* 1	TABLE ACCESS FULL	EMP	1	2	2	00:00:00.01	7

4. 튜닝하려면?

연봉(sal * 12) 이 36000 인 사원들의 이름과 연봉을 출력하세요

튜닝후:

```
SQL> SELECT  ename, sal*12
             FROM  emp
             WHERE  sal  = 36000/12;
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		2	00:00:00.01	2
1	TABLE ACCESS BY INDEX ROWID BATCHED	EMP	1	2	2	00:00:00.01	2
* 2	INDEX RANGE SCAN	EMP SAL	1	2	2	00:00:00.01	1

5. 문자형 컬럼에 인덱스 생성하기

사원 테이블의 직업 컬럼에 인덱스를 생성하세요

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

6. 문자형 인덱스 컬럼이 가공이 되었다면?

직업의 첫번째 부터 5번째의 자리가 SALES인 사원들의 이름과 직업을 출력하는 아래의 SQL 을 튜닝하세요

튜닝전:

```
SQL> SELECT ename, job
        FROM emp
        WHERE substr(job,1,5) ='SALES';
```

```
-----------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |   A-Time    | Buffers |
-----------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |      1 |        |      4 |00:00:00.01 |       7 |
|*  1 |  TABLE ACCESS FULL| EMP  |      1 |      4 |      4 |00:00:00.01 |       7 |
-----------------------------------------------------------------------------------
```

7.튜닝하려면?

직업의 첫번째 부터 5번째의 자리가 SALES인 사원들의 이름과 직업을 출력하는 아래의 SQL 을 튜닝하세요

튜닝후:

```
SQL> SELECT ename, job
        FROM emp
        WHERE job like 'SALES%';
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		4	00:00:00.01	2
1	TABLE ACCESS BY INDEX ROWID BATCHED	EMP	1	4	4	00:00:00.01	2
* 2	INDEX RANGE SCAN	EMP JOB	1	4	4	00:00:00.01	1

8. 날짜형 컬럼 인덱스 range scan

1981년도에 입사한 사원들의 이름과 입사일을 출력하시오 !

튜닝전:

```
SQL> SELECT  ename, hiredate
            FROM  emp
            WHERE  to_char(hiredate, 'RRRR') ='1981';
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		10	00:00:00.01	7
* 1	TABLE ACCESS FULL	EMP	1	10	10	00:00:00.01	7

9. 튜닝하려면 ?

1981년도에 입사한 사원들의 이름과 입사일을 출력하시오 !

튜닝후:

```
SQL> SELECT  ename, hiredate
           FROM  emp
           WHERE   hiredate between to_date('1981/01/01', 'RRRR/MM/DD')
                                  and  to_date('1981/12/31', 'RRRR/MM/DD') +1;
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		10	00:00:00.01	2
* 1	FILTER		1		10	00:00:00.01	2
2	TABLE ACCESS BY INDEX ROWID BATCHED	EMP	1	10	10	00:00:00.01	2
* 3	INDEX RANGE SCAN	EMP HIREDATE	1	10	10	00:00:00.01	1

이럴때는

방법5. index unique scan 으로 유도하자 !

▣ 학습 내용

1. index unique scan 의 원리를 이해합니다.

2. index range scan 과 index unique scan 중에 옵티마이져가 어떤 인덱스를 사용하는지 확인합니다.

3. primary key 제약을 걸면 unique 인덱스가 생성됨을 학습합니다.

▣ 학습 목표

index unique scan 으로 SQL 을 튜닝할 수 있습니다.

인덱스 스캔 방법 7가지

	인덱스 엑세스 방법	관련 힌트
1	index range scan	index
2	index unique scan	index
3	index skip scan	index_ss
4	index full scan	index_fs
5	index fast full scan	index_ffs
6	index merge scan	and_equal
7	index bitmap merge scan	index_combine

1.index unique scan 이란 ?

```
select empno, ename
 from emp
 where  empno = 7788;
```

emp_empno 인덱스

empno	ROWID
7369	AAATdNAAHAAAAFeAAK
7499	AAATdNAAHAAAAFeAAF
7521	AAATdNAAHAAAAFeAAI
7566	AAATdNAAHAAAAFeAAD
7654	AAATdNAAHAAAAFeAAE
7698	AAATdNAAHAAAAFeAAB
7782	AAATdNAAHAAAAFeAAC
7788	AAATdNAAHAAAAFeAAL
7839	AAATdNAAHAAAAFeAAA
7844	AAATdNAAHAAAAFeAAG
7876	AAATdNAAHAAAAFeAAM
7900	AAATdNAAHAAAAFeAAH
7902	AAATdNAAHAAAAFeAAJ
7934	AAATdNAAHAAAAFeAAN

emp 테이블

ROWID	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		1981-11-17	5000		10
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	1981-05-09	2450		10
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	1981-04-01	2975		20
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	1981-12-11	950		30
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	1981-12-11	3000		20
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	1980-12-09	800		20
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	1983-01-15	1100		20
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	1982-01-11	1300		10

2. 두개중에 어느 인덱스를 사용하는게 좋은가?

```
select  /*+ gather_plan_statistics */ empno, ename, sal
 from emp
 where ename='SCOTT' and empno = 7788;
```

emp_empno 인덱스

empno	ROWID
7369	AAATdNAAHAAAAFeAAK
7499	AAATdNAAHAAAAFeAAF
7521	AAATdNAAHAAAAFeAAI
7566	AAATdNAAHAAAAFeAAD
7654	AAATdNAAHAAAAFeAAE
7698	AAATdNAAHAAAAFeAAB
7782	AAATdNAAHAAAAFeAAC
7788	AAATdNAAHAAAAFeAAL
7839	AAATdNAAHAAAAFeAAA
7844	AAATdNAAHAAAAFeAAG
7876	AAATdNAAHAAAAFeAAM
7900	AAATdNAAHAAAAFeAAH
7902	AAATdNAAHAAAAFeAAJ
7934	AAATdNAAHAAAAFeAAN

emp_ename 인덱스

ENAME	ROWID
ADAMS	AAATc1AAHAAAAHeAAM
ALLEN	AAATc1AAHAAAAHeAAF
BLAKE	AAATc1AAHAAAAHeAAB
CLARK	AAATc1AAHAAAAHeAAC
FORD	AAATc1AAHAAAAHeAAJ
JAMES	AAATc1AAHAAAAHeAAH
JONES	AAATc1AAHAAAAHeAAD
KING	AAATc1AAHAAAAHeAAA
MARTIN	AAATc1AAHAAAAHeAAE
MILLER	AAATc1AAHAAAAHeAAN
SCOTT	AAATc1AAHAAAAHeAAL
SMITH	AAATc1AAHAAAAHeAAK
TURNER	AAATc1AAHAAAAHeAAG
WARD	AAATc1AAHAAAAHeAAI

emp 테이블

ROWID	EMPNO	ENAME	JOB	...
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT	...
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	...
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	...
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	...
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	...
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	...
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	...
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	...
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	...
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	...
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	...
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	...
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	...
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	...

3.primary key 제약을 걸면 인덱스가 생성된다

```
alter table emp
 add constraint emp_empno_pk primary key(empno);
```

emp_empno_pk 인덱스

empno	ROWID
7369	AAATdNAAHAAAAFeAAK
7499	AAATdNAAHAAAAFeAAF
7521	AAATdNAAHAAAAFeAAI
7566	AAATdNAAHAAAAFeAAD
7654	AAATdNAAHAAAAFeAAE
7698	AAATdNAAHAAAAFeAAB
7782	AAATdNAAHAAAAFeAAC
7788	AAATdNAAHAAAAFeAAL
7839	AAATdNAAHAAAAFeAAA
7844	AAATdNAAHAAAAFeAAG
7876	AAATdNAAHAAAAFeAAM
7900	AAATdNAAHAAAAFeAAH
7902	AAATdNAAHAAAAFeAAJ
7934	AAATdNAAHAAAAFeAAN

emp 테이블

ROWID	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		1981-11-17	5000		10
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	1981-05-09	2450		10
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	1981-04-01	2975		20
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	1981-12-11	950		30
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	1981-12-11	3000		20
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	1980-12-09	800		20
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	1983-01-15	1100		20
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	1982-01-11	1300		10

이럴때는

방법6. index full scan 으로 유도하자 !

▣ 학습 내용

1. 테이블 전체 스캔의 원리를 학습합니다.

2. 결합 컬럼 인덱스를 이해하고 결합 컬럼 인덱스 사용시 주의사항을 학습합니다.

3. table full scan 과 index full scan 의 차이를 이해합니다.

▣ 학습 목표

1. index full scan 으로 실행계획을 생성하는 SQL을 작성할 수 있습니다.

2. 결합 컬럼인덱스를 검색 성능을 높일 수 있게 생성할 수 있습니다.

인덱스 스캔 방법 7가지

	인덱스 엑세스 방법	관련 힌트
1	index range scan	index
2	index unique scan	index
3	index full scan	index_fs
4	index skip scan	index_ss
5	index fast full scan	index_ffs
6	index merge scan	and_equal
7	index bitmap merge scan	index_combine

1. table full scan 이란?

테이블의 처음 부터 끝까지를 다 스캔하면서 원하는 데이터를 찾는 검색 방법입니다.

```
select ename, sal
 from emp
 where  sal = 3000;
```

emp 테이블

ROWID	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		1981-11-17	5000		10
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	1981-05-09	2450		10
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	1981-04-01	2975		20
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	1981-12-11	950		30
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	1981-12-11	3000		20
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	1980-12-09	800		20
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	1983-01-15	1100		20
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	1982-01-11	1300		10

2. 단일 컬럼 인덱스만 있었을때

인덱스를 통해서 테이블을 엑세스 합니다.

```
select  ename,sal
 from emp
where sal = 3000;
```

emp_sal 인덱스

SAL	ROWID
800	AAATdNAAHAAAAFeAAK
950	AAATdNAAHAAAAFeAAH
1100	AAATdNAAHAAAAFeAAM
1250	AAATdNAAHAAAAFeAAE
1250	AAATdNAAHAAAAFeAAI
1300	AAATdNAAHAAAAFeAAN
1500	AAATdNAAHAAAAFeAAG
1600	AAATdNAAHAAAAFeAAF
2450	AAATdNAAHAAAAFeAAC
2850	AAATdNAAHAAAAFeAAB
2975	AAATdNAAHAAAAFeAAD
3000	AAATdNAAHAAAAFeAAJ
3000	AAATdNAAHAAAAFeAAL
5000	AAATdNAAHAAAAFeAAA

emp 테이블

ROWID	EMPNO	ENAME	JOB	...
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT	...
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	...
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	...
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	...
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	...
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	...
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	...
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	...
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	...
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	...
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	...
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	...
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	...
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	...

3. 결합 컬럼 인덱스를 생성했다면?

검색하기를 원하는 데이터가 인덱스에 다 구성되어 있다면 테이블 엑세스를 하지 않습니다.

```
select  ename,sal
 from emp
where sal = 3000;
```

emp_sal_ename 인덱스

SAL	ENAME	ROWID
800	SMITH	AAATdNAAHAAAAFeAAK
950	JAMES	AAATdNAAHAAAAFeAAH
1100	ADAMS	AAATdNAAHAAAAFeAAM
1250	MARTIN	AAATdNAAHAAAAFeAAE
1250	WARD	AAATdNAAHAAAAFeAAI
1300	MILLER	AAATdNAAHAAAAFeAAN
1500	TURNER	AAATdNAAHAAAAFeAAG
1600	ALLEN	AAATdNAAHAAAAFeAAF
2450	CLARK	AAATdNAAHAAAAFeAAC
2850	BLAKE	AAATdNAAHAAAAFeAAB
2975	JONES	AAATdNAAHAAAAFeAAD
3000	FORD	AAATdNAAHAAAAFeAAJ
3000	SCOTT	AAATdNAAHAAAAFeAAL
5000	KING	AAATdNAAHAAAAFeAAA

emp 테이블

ROWID	EMPNO	ENAME	JOB	...
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT	...
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	...
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	...
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	...
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	...
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	...
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	...
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	...
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	...
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	...
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	...
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	...
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	...
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	...

4. index full scan 이란?

데이터를 검색할 때 인덱스 전체를 스캔 하면서 원하는 데이터를 검색하는 스캔 방법입니다.

```
select  ename,sal, job
 from emp
where ename='JONES';
```

emp_sal_ename 인덱스

SAL	ENAME	ROWID
800	SMITH	AAATdNAAHAAAAFeAAK
950	JAMES	AAATdNAAHAAAAFeAAH
1100	ADAMS	AAATdNAAHAAAAFeAAM
1250	MARTIN	AAATdNAAHAAAAFeAAE
1250	WARD	AAATdNAAHAAAAFeAAI
1300	MILLER	AAATdNAAHAAAAFeAAN
1500	TURNER	AAATdNAAHAAAAFeAAG
1600	ALLEN	AAATdNAAHAAAAFeAAF
2450	CLARK	AAATdNAAHAAAAFeAAC
2850	BLAKE	AAATdNAAHAAAAFeAAB
2975	JONES	AAATdNAAHAAAAFeAAD
3000	FORD	AAATdNAAHAAAAFeAAJ
3000	SCOTT	AAATdNAAHAAAAFeAAL
5000	KING	AAATdNAAHAAAAFeAAA

emp 테이블

ROWID	EMPNO	ENAME	JOB	...
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT	...
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	...
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	...
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	...
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	...
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	...
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	...
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	...
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	...
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	...
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	...
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	...
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	...
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	...

Quize

사원 테이블에 사원번호+사원이름+월급으로 결합 컬럼 인덱스를 생성하고 월급이 1250인 사원의 이름과 월급을 출력하는데 지금 생성한 인덱스를 사용하겠금 힌트를 주고 실행하세요.

실행하기전에 다음 링크의 스크립트를 통해 emp와 dept를 다시 생성하세요

https://cafe.daum.net/oracleoracle/Sdyr/846

배운 내용 정리

1. 테이블 엑세스를 피하고 좀더 빨리 데이터 검색을 하려면 인덱스에 해당 컬럼을 구성하여 결합 컬럼 인덱스를 생성하면 됩니다.

2. 결합 컬럼 인덱스의 첫번째 컬럼이 쿼리문의 where 절에 존재해야 인덱스를 엑세스 사용할 수 있게 됩니다.

3. 결합 컬럼 인덱스의 첫번째 컬럼이 검색하는 쿼리문의 where 절에 존재하지 않는다면 해당 인덱스를 사용하지 못하거나 index full scan 으로 수행 되게 됩니다.

4. index full scan 이 table full scan 보다 대체로 더 좋은 성능을 보입니다.

이럴때는

방법7. index skip scan 으로 유도하자 !

▣ 학습 내용

1. index skip scan 의 원리를 학습합니다.

2. 결합 컬럼 인덱스의 첫번째 컬럼이 where 절에 없을때 table full scan 과 index skip scan 의 차이를 학습합니다.

3. 결합 컬럼 인덱스의 첫번째 컬럼이 where 절에서 범위조건일 때 table full scan 과 index skip scan 의 차이를 학습합니다.

▣ 학습 목표

index skip scan 으로 SQL을 튜닝 할 수 있습니다.

1. index skip scan 이란 ?

인덱스 스킵 스캔은 결합 컬럼 인덱스의 첫 번째 컬럼이 WHERE 조건에 존재하지 않아도 인덱스를 이용할 수 있는 인덱스 액세스 방식

	인덱스 엑세스 방법	관련 힌트
1	index range scan	index
2	index unique scan	index
3	index full scan	index_fs
4	index skip scan	index_ss
5	index fast full scan	index_ffs
6	index merge scan	and_equal
7	index bitmap merge scan	index_combine

2. 결합 컬럼 인덱스의 첫번째 컬럼이 where 절에 없을 때

- 결합 컬럼 인덱스의 첫번째 컬럼이 where 절에 없다면 full table scan 으로 실행계획이 수행됩니다.

- 결합 컬럼 인덱스가 select 절에서 사용되기 위해서는 결합 컬럼 인덱스의 첫번째 컬럼이 where 절에 검색조건 으로 있어야 합니다.

```
select ename,deptno, job
 from emp
 where job ='MANAGER';
```

full table scan !

emp_deptno_job 인덱스

DEPTNO	JOB	ROWID
10	CLERK	AAAW+FAAHAAAAK1AAN
10	MANAGER	AAAW+FAAHAAAAK1AAC
10	PRESIDENT	AAAW+FAAHAAAAK1AAA
20	ANALYST	AAAW+FAAHAAAAK1AAJ
20	ANALYST	AAAW+FAAHAAAAK1AAL
20	CLERK	AAAW+FAAHAAAAK1AAK
20	CLERK	AAAW+FAAHAAAAK1AAM
20	MANAGER	AAAW+FAAHAAAAK1AAD
30	CLERK	AAAW+FAAHAAAAK1AAH
30	MANAGER	AAAW+FAAHAAAAK1AAB
30	SALESMAN	AAAW+FAAHAAAAK1AAE
30	SALESMAN	AAAW+FAAHAAAAK1AAF
30	SALESMAN	AAAW+FAAHAAAAK1AAG
30	SALESMAN	AAAW+FAAHAAAAK1AAI

emp 테이블

ROWID	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		1981-11-17	5000		10
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	1981-05-09	2450		10
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	1981-04-01	2975		20
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	1981-12-11	950		30
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	1981-12-11	3000		20
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	1980-12-09	800		20
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	1983-01-15	1100		20
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	1982-01-11	1300		10

```
select  /*+ index_ss(emp emp_deptno_job) */ ename,deptno, job
 from emp
 where job ='MANAGER';
```

index skip scan !

emp_deptno_job 인덱스

DEPTNO	JOB	ROWID
10	CLERK	AAAXAaAANAAAAE9AAN
10	MANAGER	AAAXAaAANAAAAE9AAC
10	PRESIDENT	AAAXAaAANAAAAE9AAA
20	ANALYST	AAAXAaAANAAAAE9AAJ
20	ANALYST	AAAXAaAANAAAAE9AAL
20	CLERK	AAAXAaAANAAAAE9AAK
20	CLERK	AAAXAaAANAAAAE9AAM
20	MANAGER	AAAXAaAANAAAAE9AAD
30	CLERK	AAAXAaAANAAAAE9AAH
30	MANAGER	AAAXAaAANAAAAE9AAB
30	SALESMAN	AAAXAaAANAAAAE9AAE
30	SALESMAN	AAAXAaAANAAAAE9AAF
30	SALESMAN	AAAXAaAANAAAAE9AAG
30	SALESMAN	AAAXAaAANAAAAE9AAI

emp 테이블

ROWID	EMPNO	ENAME	JOB	MGR	...
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		···
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	···
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	···
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	···
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	···
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	···
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	···
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	···
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	···
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	···
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	···
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	···
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	···
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	···

3. 결합 컬럼 인덱스의 첫번째 컬럼이 범위조건일때

- 이번에는 결합 컬럼 인덱스의 첫번째 조건이 where 절에 존재하긴 하는데 등치조건(=)으로 사용된게 아니라 범위 조건(between.. and) 으로 사용된 경우에 튜닝 방법입니다.

- 결합 컬럼 인덱스의 첫번째 컬럼이 범위 조건이면 결합 컬럼 인덱스를 사용하더라도 index range scan 검색 성능이 느릴 수 있습니다.

- 이럴때 인덱스 스킵 스캔으로 유도하면 좋은 결과를 보일 수 있습니다.

```
select  /*+ gather_plan_statistics index(emp emp_sal_job) */ ename, sal, job, deptno
 from emp
 where  sal between 950 and 4500  and  job='MANAGER';
```

index range scan !

emp_sal_job 인덱스

SAL	JOB	ROWID
800	CLERK	AAAXAaAANAAAAE9AAK
950	CLERK	AAAXAaAANAAAAE9AAH
1100	CLERK	AAAXAaAANAAAAE9AAM
1250	SALESMAN	AAAXAaAANAAAAE9AAE
1250	SALESMAN	AAAXAaAANAAAAE9AAI
1300	CLERK	AAAXAaAANAAAAE9AAN
1500	SALESMAN	AAAXAaAANAAAAE9AAG
1600	SALESMAN	AAAXAaAANAAAAE9AAF
2450	MANAGER	AAAXAaAANAAAAE9AAC
2850	MANAGER	AAAXAaAANAAAAE9AAB
2975	MANAGER	AAAXAaAANAAAAE9AAD
3000	ANALYST	AAAXAaAANAAAAE9AAJ
3000	ANALYST	AAAXAaAANAAAAE9AAL
5000	PRESIDENT	AAAXAaAANAAAAE9AAA

emp 테이블

ROWID	EMPNO	ENAME	JOB	MGR	...
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		...
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	...
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	...
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	...
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	...
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	...
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	...
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	...
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	...
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	...
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	...
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	...
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	...
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	...

```
select  /*+ gather_plan_statistics index_ss(emp emp_sal_job) */ ename, sal, job, deptno
 from emp
 where  sal between 950 and 4500  and  job='MANAGER';
```

index skip scan !

emp_sal_job 인덱스

SAL	JOB	ROWID
800	CLERK	AAAXAaAANAAAAE9AAK
950	CLERK	AAAXAaAANAAAAE9AAH
1100	CLERK	AAAXAaAANAAAAE9AAM
1250	SALESMAN	AAAXAaAANAAAAE9AAE
1250	SALESMAN	AAAXAaAANAAAAE9AAI
1300	CLERK	AAAXAaAANAAAAE9AAN
1500	SALESMAN	AAAXAaAANAAAAE9AAG
1600	SALESMAN	AAAXAaAANAAAAE9AAF
2450	MANAGER	AAAXAaAANAAAAE9AAC
2850	MANAGER	AAAXAaAANAAAAE9AAB
2975	MANAGER	AAAXAaAANAAAAE9AAD
3000	ANALYST	AAAXAaAANAAAAE9AAJ
3000	ANALYST	AAAXAaAANAAAAE9AAL
5000	PRESIDENT	AAAXAaAANAAAAE9AAA

emp 테이블

ROWID	EMPNO	ENAME	JOB	MGR	...
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		...
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	...
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	...
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	...
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	...
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	...
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	...
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	...
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	...
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	...
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	...
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	...
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	...
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	...

4. index skip scan의 효과 높이는 결합 컬럼 인덱스

```
select  ename, sal, job, deptno
 from emp
 where  sal = 3000 :
```

첫번째 컬럼의 데이터의 종류가 적은 인덱스

DEPTNO	SAL
10	1000
10	2000
10	3000
10	4000
10	5000
10	6000
10	7000
20	1000
20	2000
20	3000
20	4000
20	5000
20	6000
20	7000

첫번째 컬럼의 데이터의 종류가 많은 인덱스

JOB	SAL
ANALYST	1000
ANALYST	2000
ANALYST	3000
CLERK	1000
CLERK	2000
CLERK	3000
CLERK	4000
MANAGER	1000
MANAGER	2000
MANAGER	3000
SALESMAN	1000
SALESMAN	2000
SALESMAN	3000
SALESMAN	4000

Quize

다음의 SQL을 튜닝하시오 !

문제를 풀기에 앞서서 먼저 도스창에서 c##scott 으로 접속하여 @demo.sql 와 @m.sql 을 수행합니다.

SQL> @demo.sql

hiredate + job 으로 결합 컬럼 인덱스를 생성합니다.

```
create index emp_hiredate_job  on  emp(hiredate, job);
```

튜닝전:

```
select /*+ gather_plan_statistics  index(emp) */ ename, sal
   from emp
   where  hiredate between to_date('1980/01/01','RRRR/MM/DD')
                                and  to_date('1981/12/31','RRRR/MM/DD')
  and job='MANAGER';
```

배운 내용 정리

1. 결합 컬럼 인덱스의 첫번째 컬럼이 where 절에 없으면 full table scan 으로 수행되거나 index full scan 으로 수행됩니다.

2. 결합 컬럼 인덱스의 첫번째 컬럼이 where 절에 없을때 index skip scan 으로 유도하면 좋은 결과를 얻을 수 있습니다.

3. index skip scan 의 검색 속도는 결합 컬럼 인덱스의 첫번째 컬럼의 데이터의 종류가 적을 수록 빠릅니다.

4. index skip scan 의 힌트는 index_ss(테이블명 인덱스 이름) 입니다.

이럴때는

방법8. index fast full scan 으로 유도하자 !

▣ 학습 내용

1. index fast full scan 의 원리를 이해합니다.

2. table full scan 과 index fast full scan 의 차이를 이해합니다.

3. index fast full scan 의 성능을 올리는 방법을 학습 합니다.

▣ 학습 목표

index fast full scan 을 이용해서 SQL을 튜닝할 수 있습니다.

1. index fast full scan 이란 ?

index fast full scan은 인덱스 트리 구조를 무시하고 인덱스 세그먼트 전체를 multiblock i/o 방식으로 스캔하는 스캔방법 입니다.

	인덱스 엑세스 방법	관련 힌트
1	index range scan	index
2	index unique scan	index
3	index full scan	index_fs
4	index skip scan	index_ss
5	index fast full scan	index_ffs
6	index merge scan	and_equal
7	index bitmap merge scan	index_combine

```
select /*+ gather_plan_statistics */ job, count(*)
    from  emp
    group  by  job;
```

full table scan !

인덱스가 없다면?

emp 테이블

ROWID	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		1981-11-17	5000		10
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	1981-05-09	2450		10
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	1981-04-01	2975		20
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	1981-12-11	950		30
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	1981-12-11	3000		20
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	1980-12-09	800		20
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	1983-01-15	1100		20
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	1982-01-11	1300		10

```
select /*+ index_ffs(emp emp_job) */ job, count(*)
 from emp
 group by job;
```

직업에 인덱스는 있으나 not null 이 보장되지 않는다면 ?

full table scan !

emp_job 인덱스

JOB	ROWID
ANALYST	AAAXDOAANAAAAHjAAJ
ANALYST	AAAXDOAANAAAAHjAAL
CLERK	AAAXDOAANAAAAHjAAH
CLERK	AAAXDOAANAAAAHjAAK
CLERK	AAAXDOAANAAAAHjAAM
CLERK	AAAXDOAANAAAAHjAAN
MANAGER	AAAXDOAANAAAAHjAAB
MANAGER	AAAXDOAANAAAAHjAAC
MANAGER	AAAXDOAANAAAAHjAAD
PRESIDENT	AAAXDOAANAAAAHjAAA
SALESMAN	AAAXDOAANAAAAHjAAE
SALESMAN	AAAXDOAANAAAAHjAAF
SALESMAN	AAAXDOAANAAAAHjAAG
SALESMAN	AAAXDOAANAAAAHjAAI

emp 테이블

ROWID	EMPNO	ENAME	JOB	MGR	...
AAATc1AAHAAAAHeAAA	7839	KING	PRESIDENT		···
AAATc1AAHAAAAHeAAB	7698	BLAKE	MANAGER	7839	···
AAATc1AAHAAAAHeAAC	7782	CLARK	MANAGER	7839	···
AAATc1AAHAAAAHeAAD	7566	JONES	MANAGER	7839	···
AAATc1AAHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	···
AAATc1AAHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	···
AAATc1AAHAAAAHeAAG	7844	TURNER	SALESMAN	7698	···
AAATc1AAHAAAAHeAAH	7900	JAMES	CLERK	7698	···
AAATc1AAHAAAAHeAAI	7521	WARD	SALESMAN	7698	···
AAATc1AAHAAAAHeAAJ	7902	FORD	ANALYST	7566	···
AAATc1AAHAAAAHeAAK	7369	SMITH	CLERK	7902	···
AAATc1AAHAAAAHeAAL	7788	SCOTT	ANALYST	7566	···
AAATc1AAHAAAAHeAAM	7876	ADAMS	CLERK	7788	···
AAATc1AAHAAAAHeAAN	7934	MILLER	CLERK	7782	···

2. 직업 컬럼에 not null 을 보장하는 방법

- 테이블에 직접 not null 제약을 걸어준다.

```
alter table emp
  modify job not null;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

- where 절에 is not null 을 사용한다.

```
select /*+ index_ffs(emp emp_job) */ job, count(*)
 from emp
 where job is not null
 group by job;
```

```
select /*+ index_ffs(emp emp_job) */ job, count(*)
 from emp
 group by job;
```

직업 컬럼에 not null 이 보장 된다면 ?

index fast full scan !

emp_job 인덱스

JOB	ROWID
ANALYST	AAAXDOAANAAAAHjAAJ
ANALYST	AAAXDOAANAAAAHjAAL
CLERK	AAAXDOAANAAAAHjAAH
CLERK	AAAXDOAANAAAAHjAAK
CLERK	AAAXDOAANAAAAHjAAM
CLERK	AAAXDOAANAAAAHjAAN
MANAGER	AAAXDOAANAAAAHjAAB
MANAGER	AAAXDOAANAAAAHjAAC
MANAGER	AAAXDOAANAAAAHjAAD
PRESIDENT	AAAXDOAANAAAAHjAAA
SALESMAN	AAAXDOAANAAAAHjAAE
SALESMAN	AAAXDOAANAAAAHjAAF
SALESMAN	AAAXDOAANAAAAHjAAG
SALESMAN	AAAXDOAANAAAAHjAAI

결과

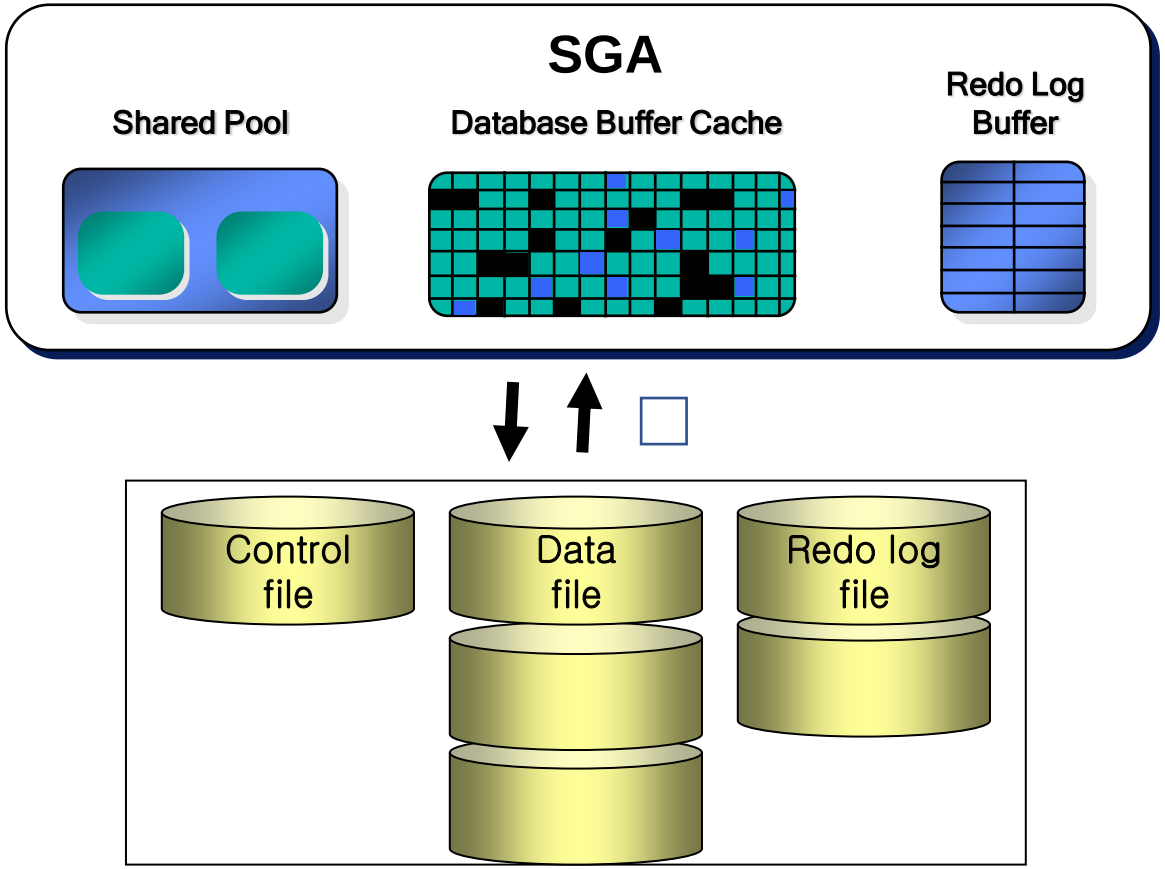
JOB	COUNT(*)
ANALYST	2
CLERK	4
SALESMAN	4
MANAGER	3
PRESIDENT	1

3. index full scan 과 index fast full scan 의 차이

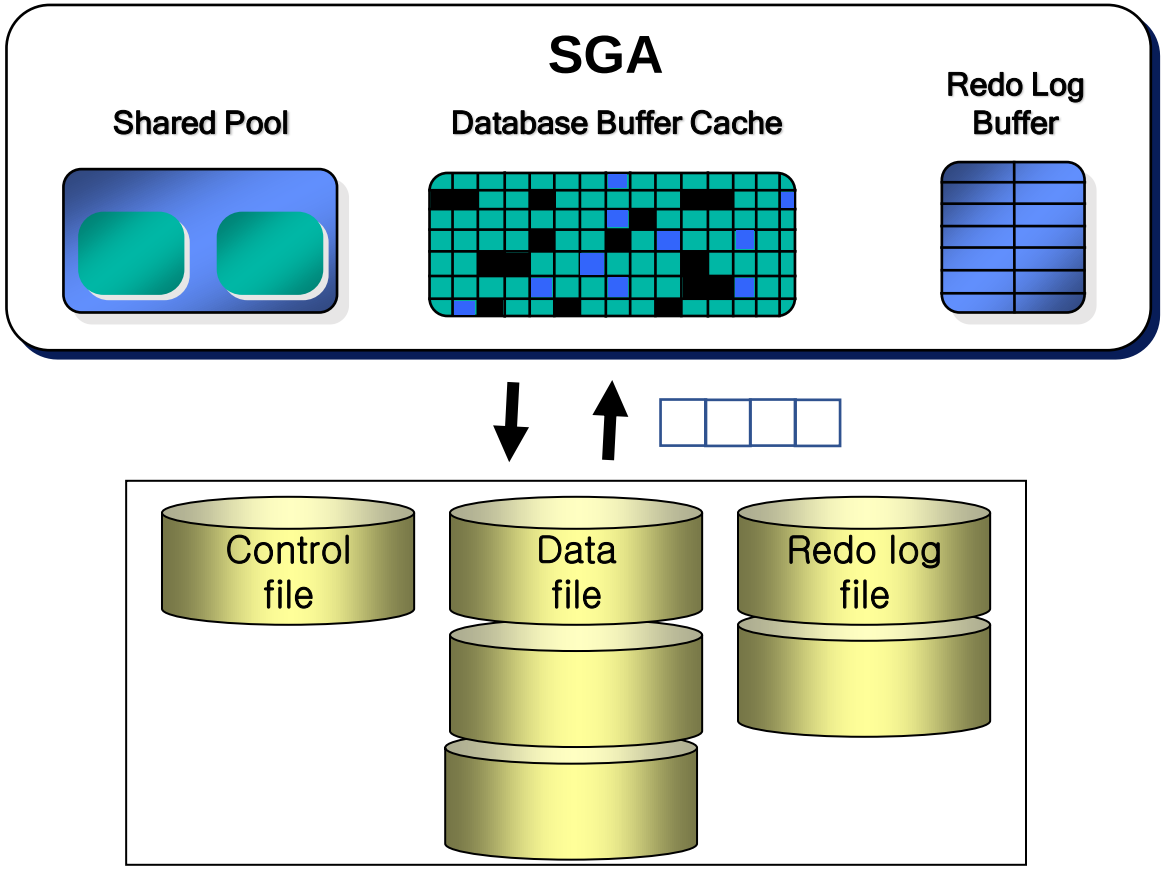
	index full scan	index fast full scan
I/O 방식	single block i/o	multi block i/o
정렬	정렬 보장	정렬 안됨
속도	느림	빠름
병렬읽기	지원 안됨	지원됨

4. single block i/o 와 multi block i/o 의 차이

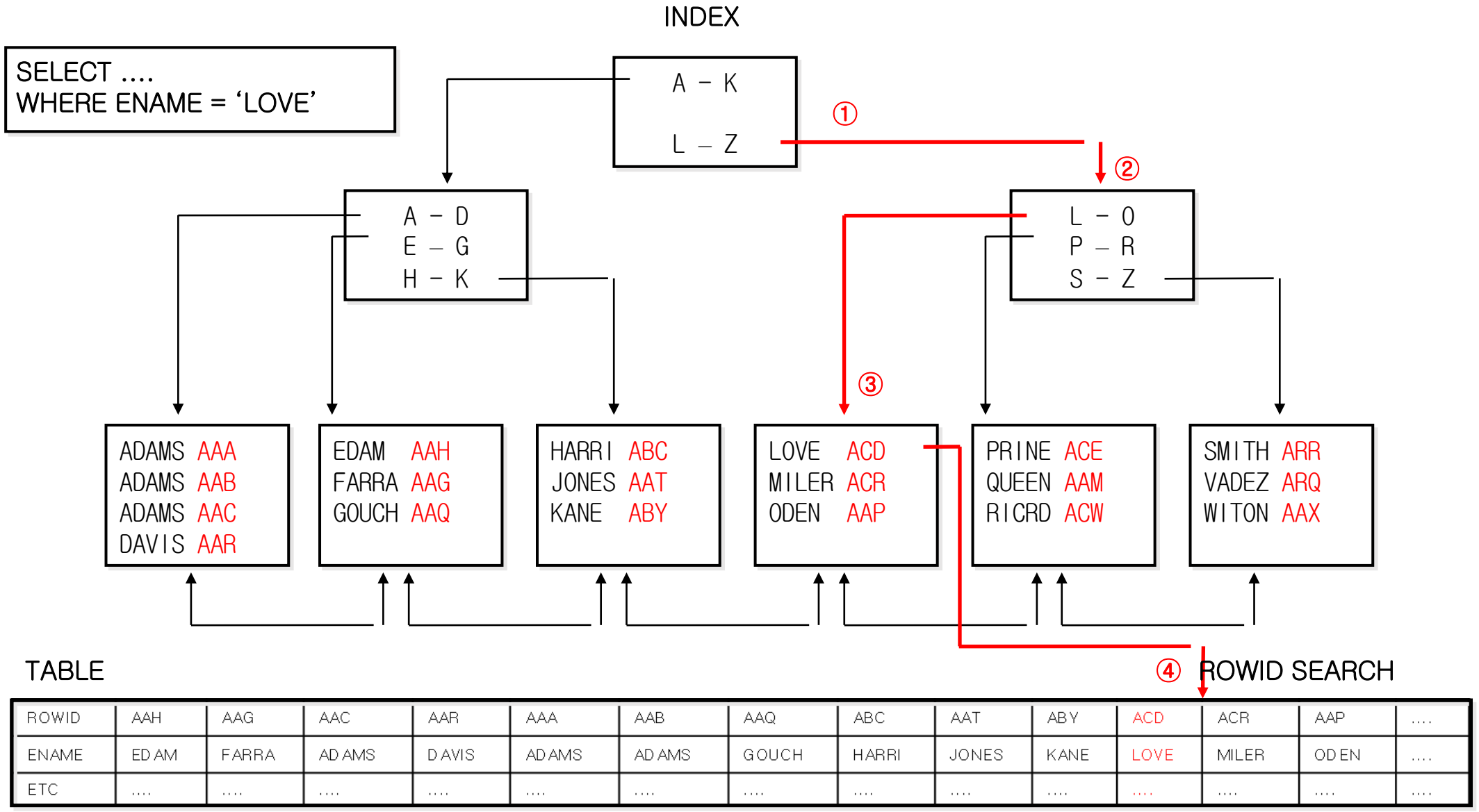
sigle block i/o



multi block i/o

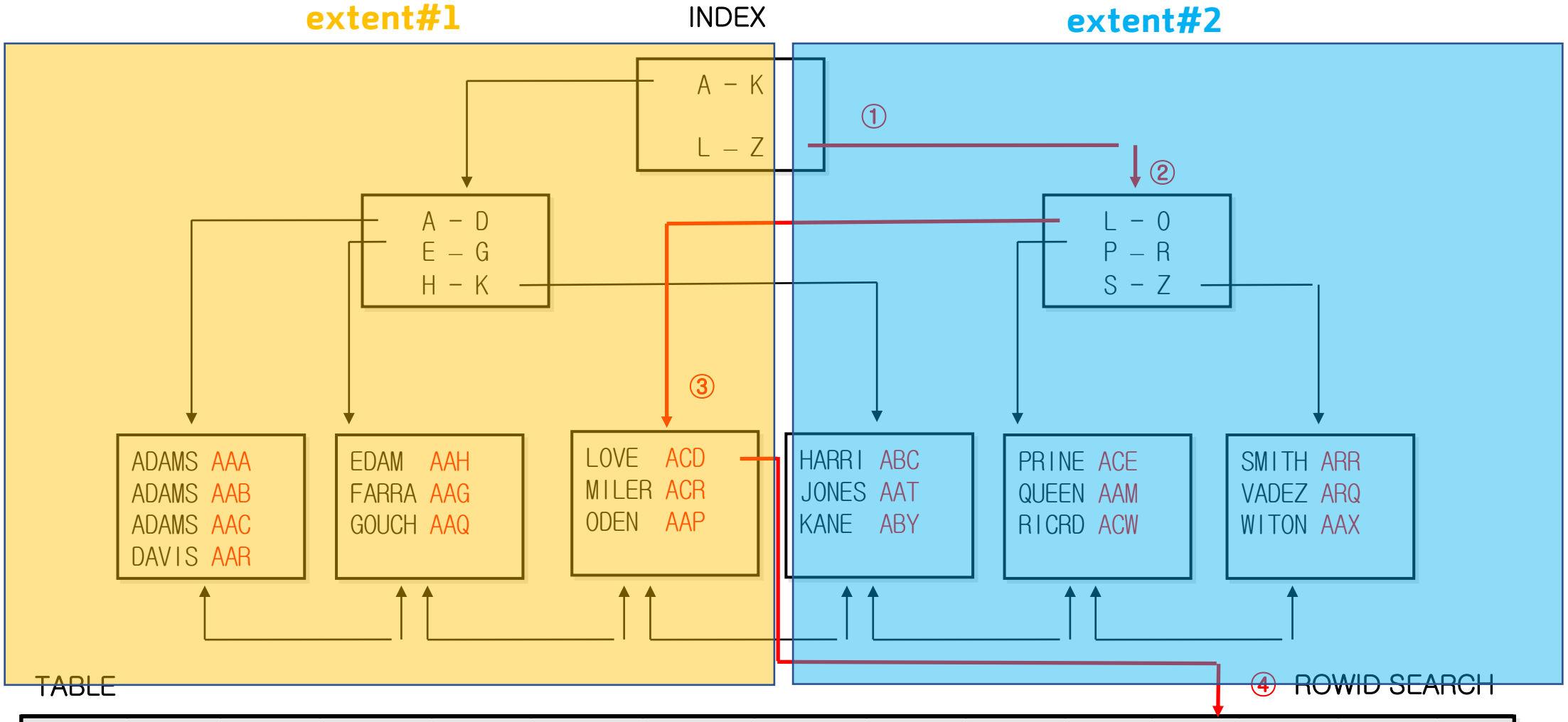


5. 논리적인 순서에 따라 배치된 index 의 구조



ROWID	AAH	AAG	AAC	AAR	AAA	AAB	AAQ	ABC	AAT	ABY	ACD	ACR	AAP	....
ENAME	EDAM	FARRA	ADAMS	DAVIS	ADAMS	ADAMS	GOUCH	HARRI	JONES	KANE	LOVE	MILER	ODEN	....
ETC	....	....	....	....	....	....	....	....	....	....	....	....	....	....

6. 물리적인 순서에 따라 배치된 index 의 구조



ROWID	AAH	AAG	AAC	AAR	AAA	AAB	AAQ	ABC	AAT	ABY	ACD	ACR	AAP	....
ENAME	EDAM	FARRA	ADAMS	DAVIS	ADAMS	ADAMS	GOUCH	HARRI	JONES	KANE	LOVE	MILER	ODEN	....
ETC	....	....	....	....	....	....	....	....	....	....	....	....	....	....

```
select /*+ */ deptno, count(*)
  from emp
  group by deptno;
```

index_fs(emp emp_deptno)

index full scan !

결과

DEPTNO	COUNT(*)
10	3
20	5
30	6

index_ffs(emp emp_deptno)

index fast full scan !

결과

DEPTNO	COUNT(*)
30	6
10	3
20	5

Quize

index fast full scan 으로 유도하도록 ? 에 알맞는 힌트를 넣으세요

telecom 에 단일 컬럼 인덱스를 생성합니다.

```
create index emp_telecom on emp(telecom);
```

튜닝전:

```
select /*+  ?  */  telecom, count(*)
        from emp
        group by telecom;
```

배운 내용 정리

1. index fast full scan 이 가능하려면 스캔하려는 인덱스 컬럼에 not null 이 보장되어야 합니다.

2. index full scan 보다 index fast full scan 이 빠른 이유는 물리적인 저장소에 있는 데이터를 그대로 읽어오기 때문입니다.

3. index full scan 은 정렬된 결과를 보장하지만 index fast full scan 은 정렬된 결과를 보장하지 않습니다.

4. index fast full scan 의 힌트는 index_ffs(테이블명 인덱스 이름) 입니다.

이럴때는

방법9. index merge scan 으로 유도하자 !

▣ 학습 내용

1. index merge scan 의 원리를 학습합니다.

2. 일반 트리구조의 인덱스와 비트맵 인덱스의 차이를 학습합니다.

3. index bitmap merge scan 의 원리를 학습합니다.

▣ 학습 목표

index merge scan 과 index bitmap merge scan 을 이용해서 SQL을 튜닝할 수 있습니다.

1. index merge scan 이란 ?

여러개의 인덱스를 같이 사용하여 하나의 인덱스만 사용했을때 보다 테이블 엑세스를 줄일 수 있는 인덱스 스캔방법

	인덱스 엑세스 방법	관련 힌트
1	index range scan	index
2	index unique scan	index
3	index full scan	index_fs
4	index skip scan	index_ss
5	index fast full scan	index_ffs
6	index merge scan	and_equal
7	index bitmap merge scan	index_combine

```
select /*+ gather_plan_statistics index(emp2 emp2_col1) */ count(*)
 from emp2
 where col1='A' and col2='D';
```

emp2_col1 인덱스

COL1	ROWID
A	AAAXDOAANAAAAHjAAJ
A	AAAXDOAANAAAAHjAAL
A	AAAXDOAANAAAAHjAAH
A	AAAXDOAANAAAAHjAAK
A	AAAXDOAANAAAAHjAAM
A	AAAXDOAANAAAAHjAAN
A	**AAAXDOAANAAAAHjAAB**
A	**AAAXDOAANAAAAHjAAC**
A	AAAXDOAANAAAAHjAAD
A	AAAXDOAANAAAAHjAAA
A	AAAXDOAANAAAAHjAAE
A	AAAXDOAANAAAAHjAAF
A	AAAXDOAANAAAAHjAAG
A	AAAXDOAANAAAAHjAAI

emp2 테이블

ROWID	COL1	COL2	...
AAATc1AAHAAAAHeAAA	A	C	···
AAATc1AAHAAAAHeAAB	A	D	···
AAATc1AAHAAAAHeAAC	A	D	···
AAATc1AAHAAAAHeAAD	A	C	···
AAATc1AAHAAAAHeAAE	A	C	···
AAATc1AAHAAAAHeAAF	A	C	···
AAATc1AAHAAAAHeAAG	A	C	···
AAATc1AAHAAAAHeAAH	A	C	···
AAATc1AAHAAAAHeAAI	A	C	···
AAATc1AAHAAAAHeAAJ	A	C	···
AAATc1AAHAAAAHeAAK	A	C	···
AAATc1AAHAAAAHeAAL	A	C	···
AAATc1AAHAAAAHeAAM	A	C	···
AAATc1AAHAAAAHeAAN	A	C	···

```
select /*+ gather_plan_statistics index(emp2 emp2_col2) */ count(*)
 from emp2
 where col1='A' and col2='D';
```

emp2_col2 인덱스

COL2	ROWID
D	AAAXDOAANAAAAHjAAZ
D	AAAXDOAANAAAAHjAAW
D	AAAXDOAANAAAAHjAAX
D	AAAXDOAANAAAAHjAAY
D	AAAXDOAANAAAAHjAQ
D	AAAXDOAANAAAAHjAAR
D	**AAAXDOAANAAAAHjAAB**
D	**AAAXDOAANAAAAHjAAC**
D	AAAXDOAANAAAAHjAAS
D	AAAXDOAANAAAAHjAAT
D	AAAXDOAANAAAAHjAAU
D	AAAXDOAANAAAAHjAAV
D	AAAXDOAANAAAAHjAAO
D	AAAXDOAANAAAAHjAAP

emp2 테이블

ROWID	COL1	COL2	...
AAATc1AAHAAAAHeAAP	B	D	···
AAATc1AAHAAAAHeAAO	B	D	···
AAATc1AAHAAAAHeAAV	B	D	···
AAATc1AAHAAAAHeAAU	B	D	···
AAATc1AAHAAAAHeAAT	B	D	···
AAATc1AAHAAAAHeAAS	B	D	···
AAATc1AAHAAAAHeAAR	B	D	···
AAATc1AAHAAAAHeAAQ	B	D	···
AAATc1AAHAAAAHeAAY	B	D	···
AAATc1AAHAAAAHeAAX	B	D	···
AAATc1AAHAAAAHeAAB	A	D	···
AAATc1AAHAAAAHeAAC	A	D	···
AAATc1AAHAAAAHeAAZ	B	D	···
AAATc1AAHAAAAHeAAW	B	D	···

```
select /*+ gather_plan_statistics and_equal(emp2 emp2_col1 emp2_col2) */ count(*)
 from emp2
 where col1='A' and col2='D';
```

emp2_col1 인덱스

COL1	ROWID
A	AAAXDOAANAAAAHjAAJ
A	AAAXDOAANAAAAHjAAL
A	AAAXDOAANAAAAHjAAH
A	AAAXDOAANAAAAHjAAK
A	AAAXDOAANAAAAHjAAM
A	AAAXDOAANAAAAHjAAN
A	**AAAXDOAANAAAAHjAAB**
A	**AAAXDOAANAAAAHjAAC**
A	AAAXDOAANAAAAHjAAD
A	AAAXDOAANAAAAHjAAA
A	AAAXDOAANAAAAHjAAE
A	AAAXDOAANAAAAHjAAF
A	AAAXDOAANAAAAHjAAG
A	AAAXDOAANAAAAHjAAI

emp2_col2 인덱스

COL2	ROWID
D	AAAXDOAANAAAAHjAAZ
D	AAAXDOAANAAAAHjAAW
D	AAAXDOAANAAAAHjAAX
D	AAAXDOAANAAAAHjAAY
D	AAAXDOAANAAAAHjAQ
D	AAAXDOAANAAAAHjAAR
D	**AAAXDOAANAAAAHjAAB**
D	**AAAXDOAANAAAAHjAAC**
D	AAAXDOAANAAAAHjAAS
D	AAAXDOAANAAAAHjAAT
D	AAAXDOAANAAAAHjAAU
D	AAAXDOAANAAAAHjAAV
D	AAAXDOAANAAAAHjAAO
D	AAAXDOAANAAAAHjAAP

emp2 테이블

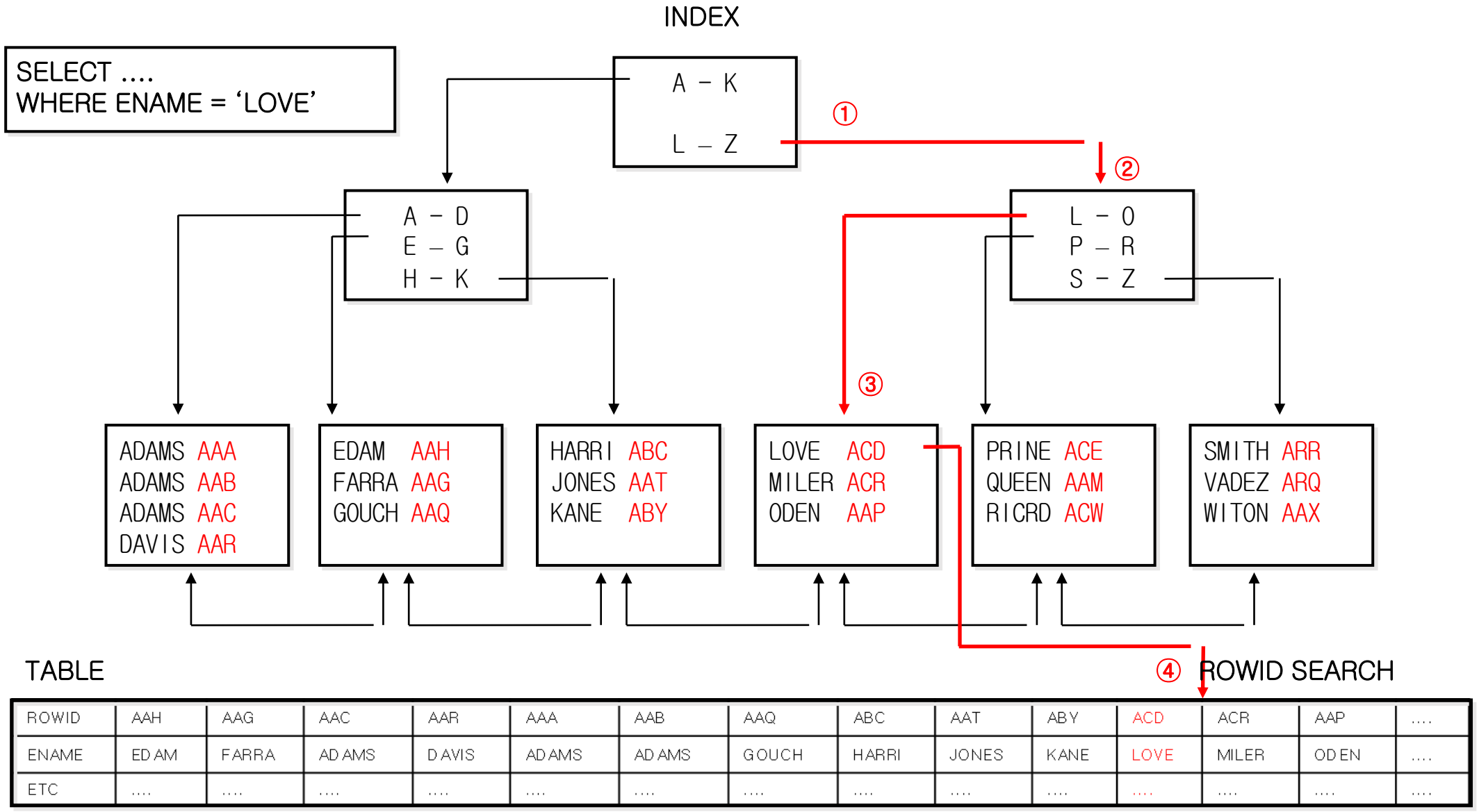
ROWID	COL1	COL2	...
AAATc1AAHAAAAHeAAA	A	C	···
AAATc1AAHAAAAHeAAB	A	D	···
AAATc1AAHAAAAHeAAC	A	D	···
AAATc1AAHAAAAHeAAD	A	C	···
AAATc1AAHAAAAHeAAE	A	C	···
AAATc1AAHAAAAHeAAF	A	C	···
AAATc1AAHAAAAHeAAG	A	C	···
AAATc1AAHAAAAHeAAH	A	C	···
AAATc1AAHAAAAHeAAI	A	C	···
AAATc1AAHAAAAHeAAJ	A	C	···
AAATc1AAHAAAAHeAAK	B	C	···
AAATc1AAHAAAAHeAAL	B	C	···
AAATc1AAHAAAAHeAAM	B	C	···
AAATc1AAHAAAAHeAAN	B	C	···

2. index bitmap merge scan 이란 ?

일반 인덱스를 크기가 아주 작은 비트맵 인덱스로 변환하고 비트맵 인덱스들을 하나로 합쳐서 스캔하는 스캔방법

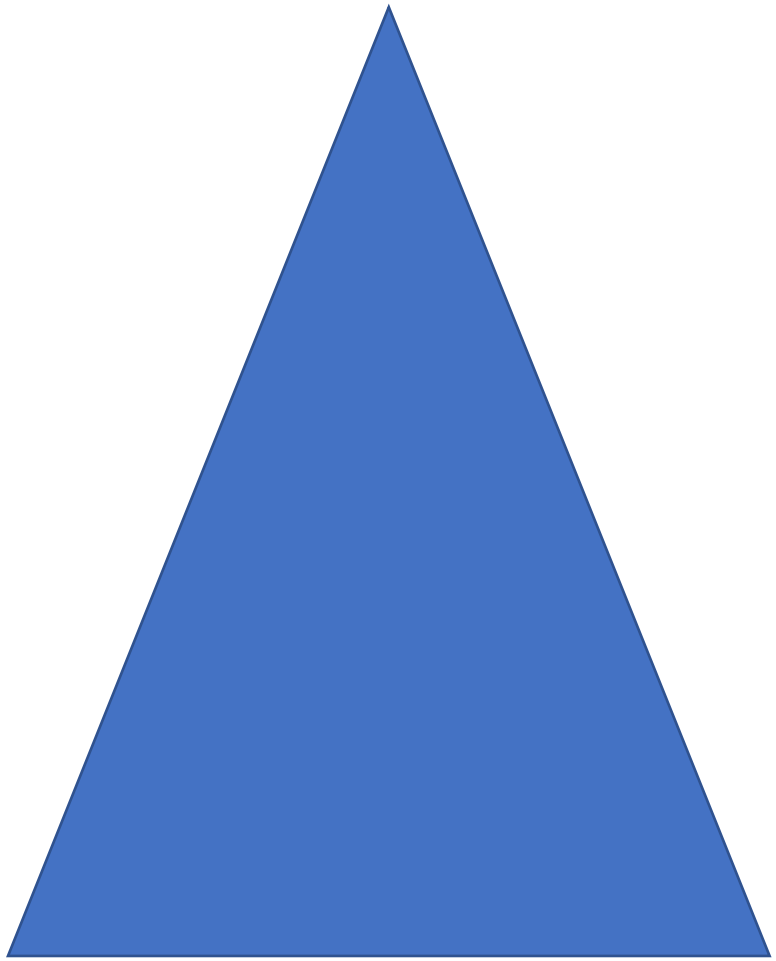
	인덱스 엑세스 방법	관련 힌트
1	index range scan	index
2	index unique scan	index
3	index full scan	index_fs
4	index skip scan	index_ss
5	index fast full scan	index_ffs
6	index merge scan	and_equal
7	index bitmap merge scan	index_combine

일반적인 tree 구조의 인덱스



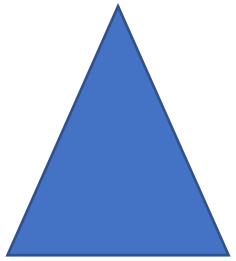
ROWID	AAH	AAG	AAC	AAR	AAA	AAB	AAQ	ABC	AAT	ABY	ACD	ACR	AAP	....
ENAME	EDAM	FARRA	ADAMS	DAVIS	ADAMS	ADAMS	GOUCH	HARRI	JONES	KANE	LOVE	MILER	ODEN	....
ETC	....	....	....	....	....	....	....	....	....	....	....	....	....	....

일반 인덱스 → 비트맵 인덱스로 변환하게 되면?



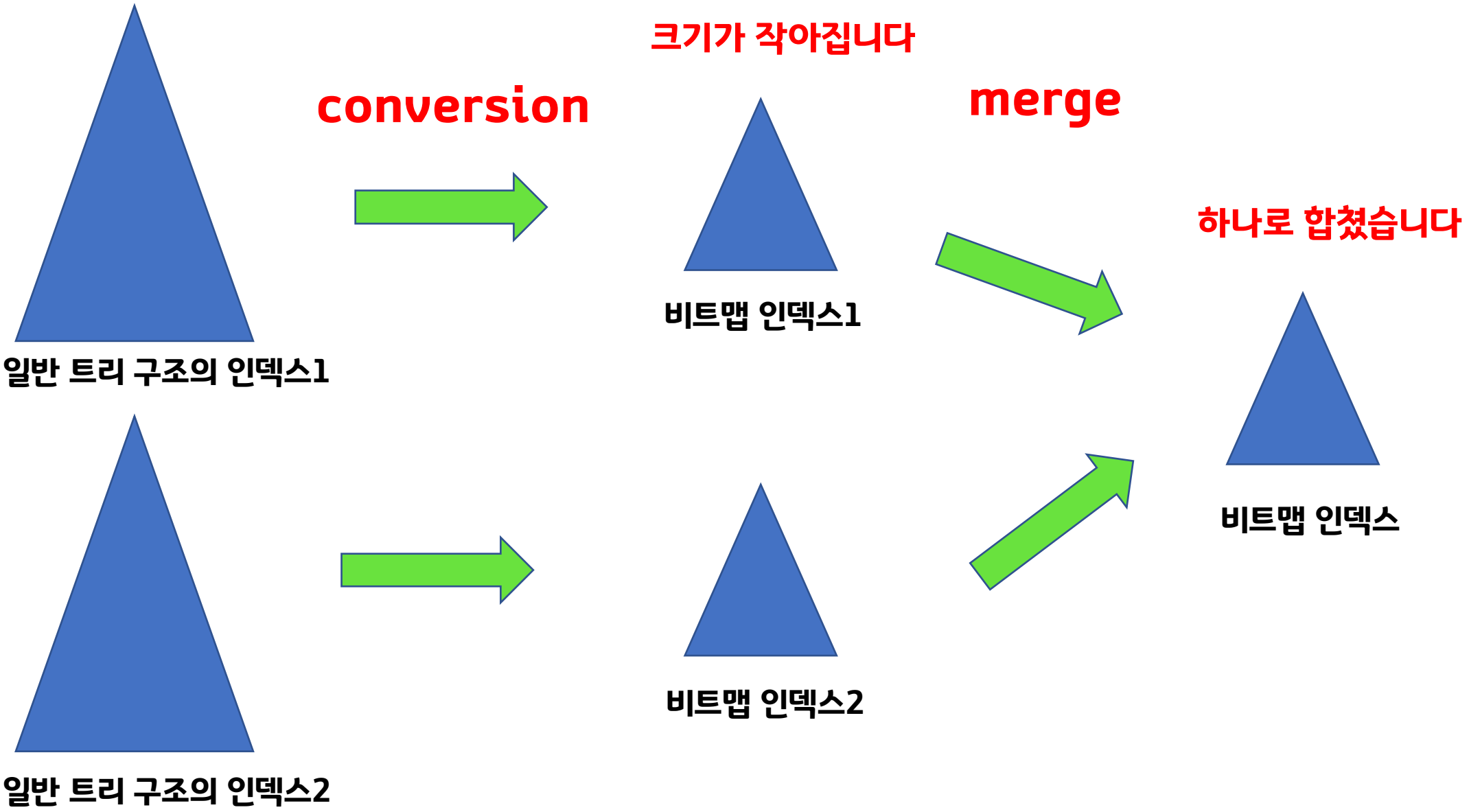
일반 트리 구조의 인덱스

크기가 작아집니다



비트맵 인덱스

비트맵으로 변환하고 합칩니다



```
select /*+ gather_plan_statistics index_combine(emp2) */ count(*)
 from emp2
 where col1='A' and col2='D';
```

bitmap conversion

bitmap conversion

emp_col1 인덱스

COL1	ROWID
A	AAAXDOAANAAAAHjAAJ
A	AAAXDOAANAAAAHjAAL
A	AAAXDOAANAAAAHjAAH
A	AAAXDOAANAAAAHjAAK
A	AAAXDOAANAAAAHjAAM
A	AAAXDOAANAAAAHjAAN
A	**AAAXDOAANAAAAHjAAB**
A	**AAAXDOAANAAAAHjAAC**
A	AAAXDOAANAAAAHjAAD
A	AAAXDOAANAAAAHjAAA
A	AAAXDOAANAAAAHjAAE
A	AAAXDOAANAAAAHjAAF
A	AAAXDOAANAAAAHjAAG
A	AAAXDOAANAAAAHjAAI

emp_col2 인덱스

COL2	ROWID
D	AAAXDOAANAAAAHjAAZ
D	AAAXDOAANAAAAHjAAW
D	AAAXDOAANAAAAHjAAX
D	AAAXDOAANAAAAHjAAY
D	AAAXDOAANAAAAHjAQ
D	AAAXDOAANAAAAHjAAR
D	**AAAXDOAANAAAAHjAAB**
D	**AAAXDOAANAAAAHjAAC**
D	AAAXDOAANAAAAHjAAS
D	AAAXDOAANAAAAHjAAT
D	AAAXDOAANAAAAHjAAU
D	AAAXDOAANAAAAHjAAV
D	AAAXDOAANAAAAHjAAO
D	AAAXDOAANAAAAHjAAP

emp2 테이블

ROWID	COL1	COL2	...
AAATc1AAHAAAAHeAAA	A	D	···
AAATc1AAHAAAAHeAAB	A	D	···
AAATc1AAHAAAAHeAAC	A	D	···
AAATc1AAHAAAAHeAAD	A	D	···
AAATc1AAHAAAAHeAAE	A	D	···
AAATc1AAHAAAAHeAAF	A	D	···
AAATc1AAHAAAAHeAAG	A	D	···
AAATc1AAHAAAAHeAAH	A	D	···
AAATc1AAHAAAAHeAAI	A	D	···
AAATc1AAHAAAAHeAAJ	A	D	···
AAATc1AAHAAAAHeAAK	A	D	···
AAATc1AAHAAAAHeAAL	A	D	···
AAATc1AAHAAAAHeAAM	A	D	···
AAATc1AAHAAAAHeAAN	A	D	···

Quize

index bitmap merge scan 으로 유도하도록 ? 에 알맞는 힌트를 넣으세요

```
@demo
create index emp_job on  emp(job);
create index emp_deptno on emp(deptno);
```

튜닝전:

```
select /*+   ?   */   empno, ename, job, deptno
        from emp
        where deptno = 30 and job='SALESMAN';
```

배운 내용 정리

1. 하나의 인덱스를 사용했을때 보다 여러개의 인덱스를 동시에 사용했을때 테이블 엑세스를 줄일 수 있다면 더 좋은 검색 성능을 보입니다.

2. where 절에 사용된 여러개의 인덱스를 동시에 이용하려면 and_equal 힌트를 사용하면 됩니다.

3. index bitmap merge scan은 b-tree 인덱스를 bitmap 인덱스로 변환하여 수행합니다.

4. index bitmap merge scan 의 힌트는 index_combine(테이블명) 입니다.

이럴때는

방법10. index descending scan 으로 유도하자 !

▣ 학습 내용

1. 정렬작업이 왜 데이터베이스에 부하를 주는지 학습합니다.

2. index range scan ascending 을 학습합니다.

3. index range scan descending 을 학습합니다.

▣ 학습 목표

정렬작업을 피하는 SQL 을 작성할 수 있습니다.

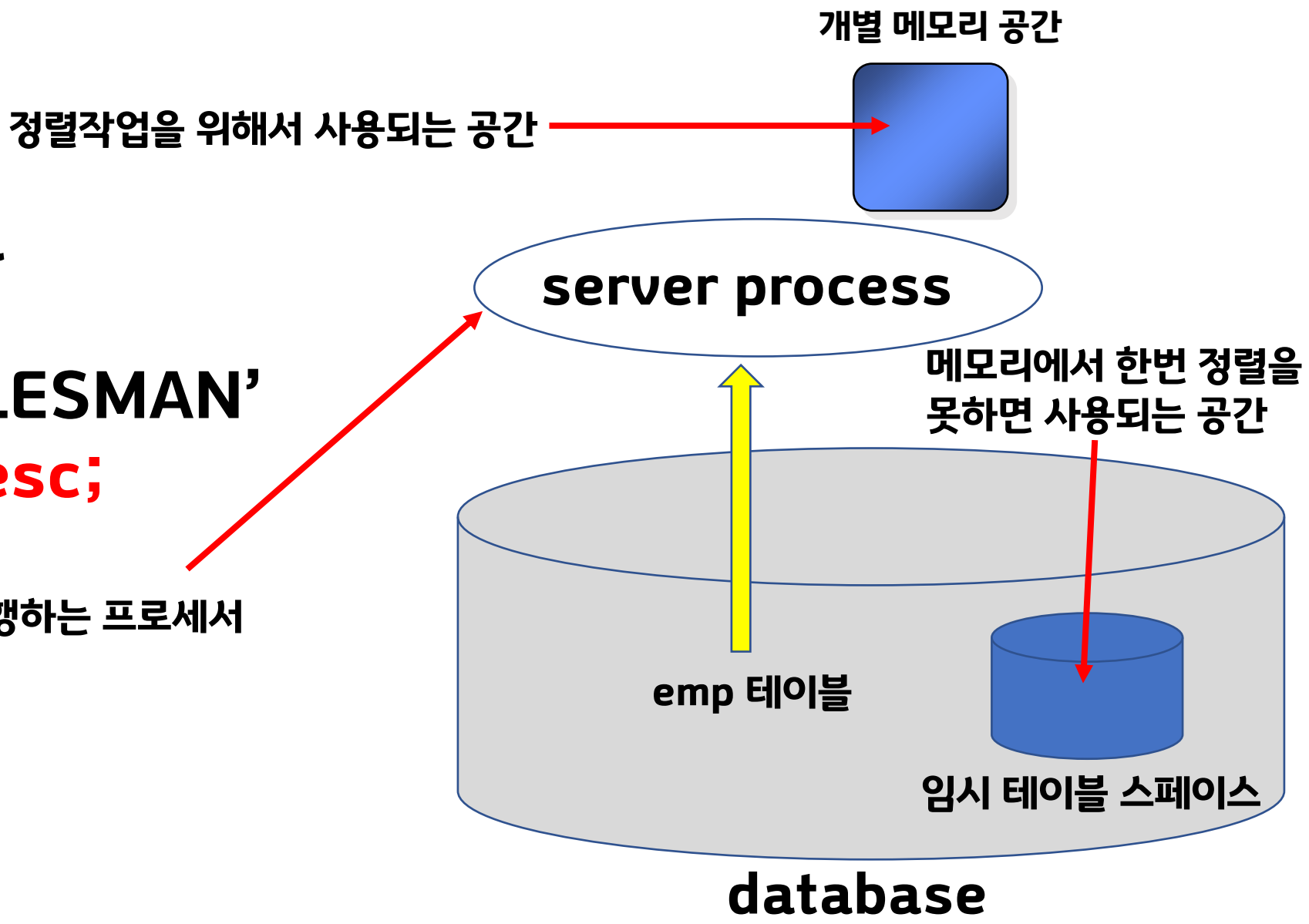
1. 정렬작업을 일으키는 SQL 은?

- order by 절
- sort merge join
- create index 문

```
select ename, sal
  from emp
  where job='SALESMAN'
  order by sal desc;
```

1. 정렬작업이 database 에 왜 부하를 주는지 ?

```
select ename, sal
  from  emp
  where  job='SALESMAN'
  order  by sal  desc;
```



인덱스를 활용하지 못한다면?

```
select  ename,sal
 from emp
 order by sal  asc;
```

server process

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

결과

ENAME	SAL
SMITH	800
JAMES	950
ADAMS	1100
MARTIN	1250
WARD	1250
MILLER	1300
TURNER	1500
ALLEN	1600
CLARK	2450
BLAKE	2850
JONES	2975
FORD	3000
SCOTT	3000
KING	5000

인덱스를 활용한다면?

```
select /*+ index_asc(emp emp_sal) */  ename,sal
 from emp
 where sal > 0 ;
```

emp_sal 인덱스

SAL	ROWID
800	AAATdNAAHAAAAFeAAK
950	AAATdNAAHAAAAFeAAH
1100	AAATdNAAHAAAAFeAAM
1250	AAATdNAAHAAAAFeAAE
1250	AAATdNAAHAAAAFeAAI
1300	AAATdNAAHAAAAFeAAN
1500	AAATdNAAHAAAAFeAAG
1600	AAATdNAAHAAAAFeAAF
2450	AAATdNAAHAAAAFeAAC
2850	AAATdNAAHAAAAFeAAB
2975	AAATdNAAHAAAAFeAAD
3000	AAATdNAAHAAAAFeAAJ
3000	AAATdNAAHAAAAFeAAL
5000	AAATdNAAHAAAAFeAAA

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

결과

ENAME	SAL
SMITH	800
JAMES	950
ADAMS	1100
MARTIN	1250
WARD	1250
MILLER	1300
TURNER	1500
ALLEN	1600
CLARK	2450
BLAKE	2850
JONES	2975
FORD	3000
SCOTT	3000
KING	5000

2. index range scan ascending 이란?

```
select /*+ index_asc(emp emp_sal) */  ename,sal
 from emp
 where sal > 0 ;
```

index 를 위에서 아래로 스캔

emp_sal 인덱스

SAL	ROWID
800	AAATdNAAHAAAAFeAAK
950	AAATdNAAHAAAAFeAAH
1100	AAATdNAAHAAAAFeAAM
1250	AAATdNAAHAAAAFeAAE
1250	AAATdNAAHAAAAFeAAI
1300	AAATdNAAHAAAAFeAAN
1500	AAATdNAAHAAAAFeAAG
1600	AAATdNAAHAAAAFeAAF
2450	AAATdNAAHAAAAFeAAC
2850	AAATdNAAHAAAAFeAAB
2975	AAATdNAAHAAAAFeAAD
3000	AAATdNAAHAAAAFeAAJ
3000	AAATdNAAHAAAAFeAAL
5000	AAATdNAAHAAAAFeAAA

결과

ENAME	SAL
SMITH	800
JAMES	950
ADAMS	1100
MARTIN	1250
WARD	1250
MILLER	1300
TURNER	1500
ALLEN	1600
CLARK	2450
BLAKE	2850
JONES	2975
FORD	3000
SCOTT	3000
KING	5000

3. index range scan descending 이란?

```
select /*+ index_desc(emp emp_sal) */  ename,sal
 from emp
 where sal > 0 ;
```

index 를 아래에서 위로 스캔

emp_sal 인덱스

SAL	ROWID
800	AAATdNAAHAAAAFeAAK
950	AAATdNAAHAAAAFeAAH
1100	AAATdNAAHAAAAFeAAM
1250	AAATdNAAHAAAAFeAAE
1250	AAATdNAAHAAAAFeAAI
1300	AAATdNAAHAAAAFeAAN
1500	AAATdNAAHAAAAFeAAG
1600	AAATdNAAHAAAAFeAAF
2450	AAATdNAAHAAAAFeAAC
2850	AAATdNAAHAAAAFeAAB
2975	AAATdNAAHAAAAFeAAD
3000	AAATdNAAHAAAAFeAAJ
3000	AAATdNAAHAAAAFeAAL
5000	AAATdNAAHAAAAFeAAA

결과

ENAME	SAL
KING	5000
SCOTT	3000
FORD	3000
JONES	2975
BLAKE	2850
CLARK	2450
ALLEN	1600
TURNER	1500
MILLER	1300
WARD	1250
MARTIN	1250
ADAMS	1100
JAMES	950
SMITH	800

Quize

도스창에서 @demo를 돌리고 아래의 SQL을 튜닝하시오.

결합 컬럼 인덱스를 직접 생성하고 튜닝하시오.

튜닝전:

```
select   /*+ gather_plan_statistics */  ename, job, sal
         from  emp
         where  substr(job, 1, 5 )='SALES'
         order by  sal  desc;
```

배운 내용 정리

1. 과도한 데이터 정렬 작업을 데이터 베이스에 부하를 줍니다.

2. 정렬작업을 수행하는 데이터 베이스의 메모리 공간은 한정되어 있습니다.

3. 정렬작업을 피하기 위해서는 이미 정렬이 되어져 있는 인덱스를 활용하는 방법이 있습니다.

4. 정렬된 결과를 볼 필요가 없다면 굳이 정렬을 일으키는 SQL을 작성하지 않는것이 바람직 합니다.

이럴때는

방법11. nested loop join 으로 유도하라 ! 첫번째

▣ 학습 내용

1.조인 문법과 조인 방법에 대해서 학습합니다.

2. nested loop 조인이란 무엇인지 설명합니다.

3. 조인 순서의 중요성과 조인순서를 정하는 힌트를 배웁니다.

▣ 학습 목표

nested loop 조인으로 유도할 수 있고 조인 순서를 조정할 수 있습니다.

1. 조인문법과 조인방법

```
select  e.ename, d.loc
 from emp e, dept  d
 where  e.deptno = d.deptno;
```

결과

ENAME	LOC
KING	NEW YORK
BLAKE	CHICAGO
CLARK	NEW YORK
JONES	DALLAS
MARTIN	CHICAGO
ALLEN	CHICAGO
TURNER	CHICAGO
JAMES	CHICAGO
WARD	CHICAGO
FORD	DALLAS
SMITH	DALLAS
SCOTT	DALLAS
ADAMS	DALLAS
MILLER	NEW YORK

조인 문법

오라클 조인 문법

1. equi join
2. non equi join
3. outer join
4. self join

1999 ansi 문법

1. on 절을 사용한 조인
2. using 절을 사용한 조인
3. left/right/full 아우터 조인
4. cross 조인

조인 방법

1. nested loop 조인
2. hash 조인
3. sort merge 조인

2. nested loop 조인이란?

```
select  e.ename, d.loc
 from emp e, dept  d
 where  e.deptno = d.deptno;
```

먼저 선행 테이블의 처리 범위를 하나씩 엑세스 하면서
그 추출된 테이블로 연결할 테이블을 조인하는 방식

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

결과

ENAME	LOC
KING	NEW YORK
BLAKE	CHICAGO
CLARK	NEW YORK
JONES	DALLAS
MARTIN	CHICAGO
ALLEN	CHICAGO
TURNER	CHICAGO
JAMES	CHICAGO
WARD	CHICAGO
FORD	DALLAS
SMITH	DALLAS
SCOTT	DALLAS
ADAMS	DALLAS
MILLER	NEW YORK

3. 조인하는 테이블 순서는?

```
select  e.ename, d.loc
 from emp e, dept  d
 where  e.deptno = d.deptno;
```

① emp 테이블 → dept 테이블

② dept 테이블 → emp 테이블

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

결과

ENAME	LOC
KING	NEW YORK
BLAKE	CHICAGO
CLARK	NEW YORK
JONES	DALLAS
MARTIN	CHICAGO
ALLEN	CHICAGO
TURNER	CHICAGO
JAMES	CHICAGO
WARD	CHICAGO
FORD	DALLAS
SMITH	DALLAS
SCOTT	DALLAS
ADAMS	DALLAS
MILLER	NEW YORK

4. 조인하는 순서를 조정하는 힌트는?

```
select  /*+ gather_plan_statistics leading(e d) use_nl(d) */ e.ename, d.loc
 from emp e, dept  d
 where  e.deptno = d.deptno;
```

emp 테이블 → dept 테이블

driving table　　driven table

선행 테이블　　연결할 테이블

5. 조건이 있었을 때의 테이블 조인 순서는?

```
select  e.ename, d.loc
 from emp e, dept  d
 where  e.deptno = d.deptno and e.ename='SCOTT';
```

① emp 테이블 → dept 테이블

② dept 테이블 → emp 테이블

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

결과

ENAME	LOC
KING	NEW YORK
BLAKE	CHICAGO
CLARK	NEW YORK
JONES	DALLAS
MARTIN	CHICAGO
ALLEN	CHICAGO
TURNER	CHICAGO
JAMES	CHICAGO
WARD	CHICAGO
FORD	DALLAS
SMITH	DALLAS
SCOTT	DALLAS
ADAMS	DALLAS
MILLER	NEW YORK

Quize

도스창에서 @demo를 돌리고 아래의 SQL을 튜닝하시오.

다음 SQL이 조인 방법을 nested loop 조인이 되게하고 가장 좋은 조인 순서로 실행되게 힌트를 주시오

튜닝전:

```
select e.ename, e.sal, d.loc
  from  emp  e,  dept  d
  where  e.deptno = d.deptno  and e.job='SALESMAN'
  and d.loc='CHICAGO';
```

배운 내용 정리

1. nested loop 조인이란 먼저 선행 테이블의 처리 범위를 하나씩 엑세스 하면서 그 추출된 테이블로 연결할 테이블을 조인하는 방식 입니다.

2. 테이블 조인 순서는 조인 되는 데이터를 적게 할 수 있는 테이블을 선행 테이블로 선택하면서 조인해야 합니다.

3. 조인 순서를 결정하는 힌트는 leading 입니다.

4. nested loop 조인으로 유도하는 힌트는 use_nl 입니다.

이럴때는

방법12. nested loop join 으로 유도하라 ! 두번째

▣ 학습 내용

1. 3개 이상의 테이블 조인시 조인 순서 정하기

2. 3개 이상의 테이블 조인시 검색 조건이 있을때 조인 순서 정하기

3. 3개 이상의 테이블 조인시 검색 조건이 여러개 있을때 조인 순서 정하기

▣ 학습 목표

여러개의 테이블을 조인할 때 nested loop 조인으로 유도할 수 있고 조인 순서를 조정할 수 있습니다.

1. 3개 이상의 테이블 조인시 조인순서는?

```
select /*+ leading(s e d ) use_nl(e) use_nl(d) */  e.ename, d.loc, s.grade
 from emp e, dept  d, salgrade  s
 where  e.deptno = d.deptno  and e.sal between s.losal and s.hisal;
```

salgrade 테이블

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

buffer 139개

2. 반대로 조인하면 성능이 좋아질까?

```
select /*+ leading(d e s ) use_nl(e) use_nl(s) */  e.ename, d.loc, s.grade
 from emp e, dept  d, salgrade  s
 where  e.deptno = d.deptno  and e.sal between s.losal and s.hisal;
```

salgrade 테이블

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

buffer 119개

3. 검색 조건이 있었을때의 조인순서는?

```
select /*+ leading(e d s) use_nl(d) use_nl(s) */ e.ename, d.loc, s.grade
 from emp e, dept  d, salgrade  s
 where  e.deptno = d.deptno  and e.sal between s.losal and s.hisal
 and  e.ename='SCOTT';
```

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

salgrade 테이블

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

buffer 21개

4. 반대로 조인하면 성능이 좋아질까?

```
select /*+ leading(s d e) use_nl(d) use_nl(e) */ e.ename, d.loc, s.grade
 from emp e, dept  d, salgrade  s
 where  e.deptno = d.deptno  and e.sal between s.losal and s.hisal
 and  e.ename='SCOTT';
```

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

salgrade 테이블

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

buffer 182개

5. 검색 조건이 여러개 있었을때의 조인순서는?

```
select /*+ leading(s e d ) use_nl(e) use_nl(d) */ e.ename, d.loc, s.grade
 from emp e, dept  d, salgrade  s
 where  e.deptno = d.deptno  and e.sal between s.losal and s.hisal
 and  s.grade  in (3,4) and d.loc='DALLAS';
```

salgrade 테이블

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

buffer 70개

6. 반대로 조인하면 성능이 좋아질까?

```
select /*+ leading(d e s ) use_nl(e) use_nl(s) */ e.ename, d.loc, s.grade
 from emp e, dept  d, salgrade  s
 where  e.deptno = d.deptno  and e.sal between s.losal and s.hisal
 and  s.grade in (3,4) and d.loc='DALLAS';
```

salgrade 테이블

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

buffer 49개

Quize

도스창에서 @demo를 돌리고 아래의 SQL을 튜닝하시오.

다음 SQL이 조인 방법을 nested loop 조인이 되게하고 가장 좋은 조인 순서로 실행되게 힌트를 주시오

튜닝전:

```
select e.ename, e.sal, d.loc
  from  emp  e,  dept  d, salgrade s
  where  e.deptno = d.deptno
  and e.sal between s.losal and s.hisal  and d.loc='DALLAS';
```

배운 내용 정리

1. nested loop 조인이란 먼저 선행 테이블의 처리 범위를 하나씩 엑세스 하면서 그 추출된 테이블로 연결할 테이블을 조인하는 방식 입니다.

2. 여러개의 테이블 조인 순서도 조인 되는 데이터를 적게 할 수 있는 테이블을 선행 테이블로 선택하면서 조인하면 됩니다.

3. 조인 순서를 결정하는 힌트는 leading 입니다.

4. nested loop 조인으로 유도하는 힌트는 use_nl 입니다.

이럴때는

방법13. nested loop join 으로 유도하라 ! 세번째

▣ 학습 내용

1. 연결고리가 되는 컬럼에 인덱스가 있었을때 두개의 테이블 조인시 성능을 높이는 방법 배우기

2. 연결고리가 되는 컬럼에 인덱스가 있고 검색조건이 별도로 있었을때 조인의 성능을 높이는 방법 배우기

▣ 학습 목표

여러개의 테이블을 조인할 때 인덱스를 사용할 수 있으며 조인되는 순서를 올바르게 제시할 수 있습니다.

1. 조인하려는 테이블 사이의 연결고리에 인덱스가 없었을때

```
select  /*+ leading(d e)  use_nl(e) */ e.ename, d.loc
 from emp e, dept  d
 where  e.deptno = d.deptno;
```

full table scan !

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

buffer 35개

2. 조인하려는 테이블 사이의 연결고리에 인덱스가 있었을때

```
select  /*+ leading(d e)  use_nl(e) index(emp emp_deptno) */ e.ename, d.loc
 from emp e, dept  d
 where  e.deptno = d.deptno;
```

index range scan !

index 생성

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	ROWID
7839	KING	PRESIDENT		1981-11-17	5000		10	AAAXzDAANAAAC2eAAA
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30	AAAXzDAANAAAC2eAAB
7782	CLARK	MANAGER	7839	1981-05-09	2450		10	AAAXzDAANAAAC2eAAC
7566	JONES	MANAGER	7839	1981-04-01	2975		20	AAAXzDAANAAAC2eAAD
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30	AAAXzDAANAAAC2eAAE
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30	AAAXzDAANAAAC2eAAF
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30	AAAXzDAANAAAC2eAAG
7900	JAMES	CLERK	7698	1981-12-11	950		30	AAAXzDAANAAAC2eAAH
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30	AAAXzDAANAAAC2eAAI
7902	FORD	ANALYST	7566	1981-12-11	3000		20	AAAXzDAANAAAC2eAAJ
7369	SMITH	CLERK	7902	1980-12-09	800		20	AAAXzDAANAAAC2eAAK
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20	AAAXzDAANAAAC2eAAL
7876	ADAMS	CLERK	7788	1983-01-15	1100		20	AAAXzDAANAAAC2eAAM
7934	MILLER	CLERK	7782	1982-01-11	1300		10	AAAXzDAANAAAC2eAAN

emp_deptno인덱스

ROWID	DEPTNO
XzDAANAAAC2eAAA	10
AAAXzDAANAAAC2eAAC	10
AAAXzDAANAAAC2eAAN	10
AAAXzDAANAAAC2eAAD	20
AAAXzDAANAAAC2eAAJ	20
AAAXzDAANAAAC2eAAK	20
AAAXzDAANAAAC2eAAL	20
AAXzDAANAAAC2eAAM	20
AAAXzDAANAAAC2eAAB	30
AAAXzDAANAAAC2eAAE	30
AAAXzDAANAAAC2eAAF	30
AAAXzDAANAAAC2eAAG	30
AAAXzDAANAAAC2eAAH	30
AAAXzDAANAAAC2eAAI	30

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

buffer 10개

3. 검색 조건이 있었을때의 조인문장 튜닝

```
select  /*+ leading(e d)  use_nl(d)  index(dept dept_deptno) */ e.ename, d.loc
 from emp e, dept  d
 where  e.deptno = d.deptno  and e.ename='SCOTT';
```

index 생성

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept_deptno인덱스

DEPTNO	ROWID
10	AAAXzCAANAAACz+AAA
20	AAAXzCAANAAACz+AAB
30	AAAXzCAANAAACz+AAC
40	AAAXzCAANAAACz+AAD

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

buffer 9개

3. 검색조건에 인덱스까지 걸어준다면 ?

```
select  /*+ leading(e d)  use_nl(d) index(emp emp_ename)
            index(dept dept_deptno) */ e.ename, d.loc
 from emp e, dept  d
 where  e.deptno = d.deptno  and e.ename='SCOTT';
```

index 생성 index 생성

emp_ename인덱스

ENAME	ROWID
ADAMS	AAATifAAHAAADL1AAM
ALLEN	AAATifAAHAAADL1AAF
BLAKE	AAATifAAHAAADL1AAB
CLARK	AAATifAAHAAADL1AAC
FORD	AAATifAAHAAADL1AAJ
JAMES	AAATifAAHAAADL1AAH
JONES	AAATifAAHAAADL1AAD
KING	AAATifAAHAAADL1AAA
MARTIN	AAATifAAHAAADL1AAE
MILLER	AAATifAAHAAADL1AAN
SCOTT	AAATifAAHAAADL1AAL
SMITH	AAATifAAHAAADL1AAK
TURNER	AAATifAAHAAADL1AAG
WARD	AAATifAAHAAADL1AAI

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept_deptno인덱스

DEPTNO	ROWID
10	AAAXzCAANAAACz+AAA
20	AAAXzCAANAAACz+AAB
30	AAAXzCAANAAACz+AAC
40	AAAXzCAANAAACz+AAD

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

buffer 4개

4. 조건이 여러개일 때의 조인문장 튜닝

```
select /*+ gather_plan_statistics leading(d e) use_nl(e) */ e.ename, e.sal, d.loc
  from  emp  e,  dept  d
  where  e.deptno = d.deptno  and e.job='SALESMAN' and d.loc='CHICAGO';
```

index 생성

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	ROWID
7839	KING	PRESIDENT		1981-11-17	5000		10	AAAXzDAANAAAC2eAAA
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30	AAAXzDAANAAAC2eAAB
7782	CLARK	MANAGER	7839	1981-05-09	2450		10	AAAXzDAANAAAC2eAAC
7566	JONES	MANAGER	7839	1981-04-01	2975		20	AAAXzDAANAAAC2eAAD
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30	AAAXzDAANAAAC2eAAE
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30	AAAXzDAANAAAC2eAAF
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30	AAAXzDAANAAAC2eAAG
7900	JAMES	CLERK	7698	1981-12-11	950		30	AAAXzDAANAAAC2eAAH
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30	AAAXzDAANAAAC2eAAI
7902	FORD	ANALYST	7566	1981-12-11	3000		20	AAAXzDAANAAAC2eAAJ
7369	SMITH	CLERK	7902	1980-12-09	800		20	AAAXzDAANAAAC2eAAK
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20	AAAXzDAANAAAC2eAAL
7876	ADAMS	CLERK	7788	1983-01-15	1100		20	AAAXzDAANAAAC2eAAM
7934	MILLER	CLERK	7782	1982-01-11	1300		10	AAAXzDAANAAAC2eAAN

emp_deptno인덱스

ROWID	DEPTNO
AAAXzDAANAAAC2eAAA	10
AAAXzDAANAAAC2eAAC	10
AAAXzDAANAAAC2eAAN	10
AAAXzDAANAAAC2eAAD	20
AAAXzDAANAAAC2eAAJ	20
AAAXzDAANAAAC2eAAK	20
AAAXzDAANAAAC2eAAL	20
AAAXzDAANAAAC2eAAM	20
AAAXzDAANAAAC2eAAB	30
AAAXzDAANAAAC2eAAE	30
AAAXzDAANAAAC2eAAF	30
AAAXzDAANAAAC2eAAG	30
AAAXzDAANAAAC2eAAH	30
AAAXzDAANAAAC2eAAI	30

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

buffer 9개

5. 조건이 여러개일 때 검색조건에 인덱스가 있다면?

```
select /*+ gather_plan_statistics leading(d e) use_nl(e)
           index(dept dept_loc)  index(emp emp_deptno) */ e.ename, e.sal, d.loc
  from  emp  e,  dept  d
  where  e.deptno = d.deptno  and e.job='SALESMAN' and d.loc='CHICAGO';
```

index 생성

index 생성

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	ROWID
7839	KING	PRESIDENT		1981-11-17	5000		10	AAAXzDAANAAAC2eAAA
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30	AAAXzDAANAAAC2eAAB
7782	CLARK	MANAGER	7839	1981-05-09	2450		10	AAAXzDAANAAAC2eAAC
7566	JONES	MANAGER	7839	1981-04-01	2975		20	AAAXzDAANAAAC2eAAD
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30	AAAXzDAANAAAC2eAAE
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30	AAAXzDAANAAAC2eAAF
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30	AAAXzDAANAAAC2eAAG
7900	JAMES	CLERK	7698	1981-12-11	950		30	AAAXzDAANAAAC2eAAH
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30	AAAXzDAANAAAC2eAAI
7902	FORD	ANALYST	7566	1981-12-11	3000		20	AAAXzDAANAAAC2eAAJ
7369	SMITH	CLERK	7902	1980-12-09	800		20	AAAXzDAANAAAC2eAAK
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20	AAAXzDAANAAAC2eAAL
7876	ADAMS	CLERK	7788	1983-01-15	1100		20	AAAXzDAANAAAC2eAAM
7934	MILLER	CLERK	7782	1982-01-11	1300		10	AAAXzDAANAAAC2eAAN

emp_deptno인덱스

ROWID	DEPTNO
AAAXzDAANAAAC2eAAA	10
AAAXzDAANAAAC2eAAC	10
AAAXzDAANAAAC2eAAN	10
AAAXzDAANAAAC2eAAD	20
AAAXzDAANAAAC2eAAJ	20
AAAXzDAANAAAC2eAAK	20
AAAXzDAANAAAC2eAAL	20
AAAXzDAANAAAC2eAAM	20
AAAXzDAANAAAC2eAAB	30
AAAXzDAANAAAC2eAAE	30
AAAXzDAANAAAC2eAAF	30
AAAXzDAANAAAC2eAAG	30
AAAXzDAANAAAC2eAAH	30
AAAXzDAANAAAC2eAAI	30

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

dept_loc 인덱스

ROWID	LOC
AAAXzDAANAAAC2eAAA	BOSTON
AAAXzDAANAAAC2eAAC	CHCAGO
AAAXzDAANAAAC2eAAN	DALLAS
AAAXzDAANAAAC2eAAD	NEW YORK

buffer 4개

배운 내용 정리

1. 두개의 테이블을 조인할 때 조인의 연결고리가 되는 컬럼에 인덱스가 있다면 조인시 검색성능을 높일 수 있습니다.

2. 두개의 테이블 조인시 검색조건이 따로 있다면 검색조건이 되는 컬럼에 인덱스가 있다면 검색성능을 높일 수 있습니다.

이럴때는

방법14. 해쉬조인으로 유도하자 ! 첫번째

▣ 학습 내용

1. 해쉬조인의 원리를 이해합니다.

2. 검색조건이 있었을때의 해쉬조인 순서를 지정하는 방법을 이해합니다.

▣ 학습 목표

해쉬조인으로 대용량 테이블의 조인 성능을 높일 수 있습니다.

1. 해쉬조인의 원리

```
select /*+ gather_plan_statistics  leading(d e) use_hash(e) */  e.ename, d.loc
  from  emp  e, dept  d
  where  e.deptno = d.deptno  ;
```

해쉬 함수

오라클 메모리 영역 PGA

해쉬 테이블

해쉬값	DEPTNO	DNAME	LOC
kdl21fnae..	10	ACCOUNTING	NEW YORK
fekfh32en..	20	RESEARCH	DALLAS
dkfle21jfn..	30	SALES	CHICAGO
eekfn09dn...	40	OPERATIONS	BOSTON

PROB 테이블

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

2. 어느 테이블을 해쉬 테이블로 구성해야할까?

buffer 10887 개

```
select /*+ gather_plan_statistics  leading(e d) use_hash(d) */  e.ename, d.loc
  from  emp  e, dept  d
  where  e.deptno = d.deptno  ;
```

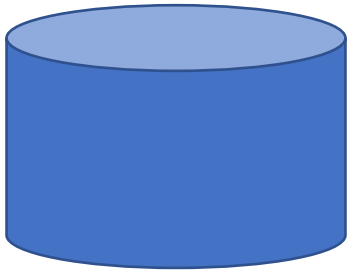
emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

오라클 메모리 영역 PGA

해쉬 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30



temp disk

PROB 테이블

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

3. dept 테이블을 해쉬 테이블로 구성한다면?

buffer 13 개

```
select /*+ gather_plan_statistics  leading(d e) use_hash(e) */  e.ename, d.loc
  from  emp  e, dept  d
  where  e.deptno = d.deptno  ;
```

PROB 테이블

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

오라클 메모리 영역 PGA

해쉬 테이블

해쉬값	DEPTNO	DNAME	LOC
kdl21fnae..	10	ACCOUNTING	NEW YORK
fekfh32en..	20	RESEARCH	DALLAS
dkfle21jfn..	30	SALES	CHICAGO
eekfn09dn...	40	OPERATIONS	BOSTON

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

4. 검색조건이 있었을때의 해쉬조인 순서

```
select /*+ gather_plan_statistics  leading(d e) use_hash(e) */  e.ename, d.loc
  from  emp  e, dept  d
  where  e.deptno = d.deptno  and  e.job='SALESMAN'
   and  d.loc='CHICAGO';
```

PROB 테이블

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

오라클 메모리 영역 PGA

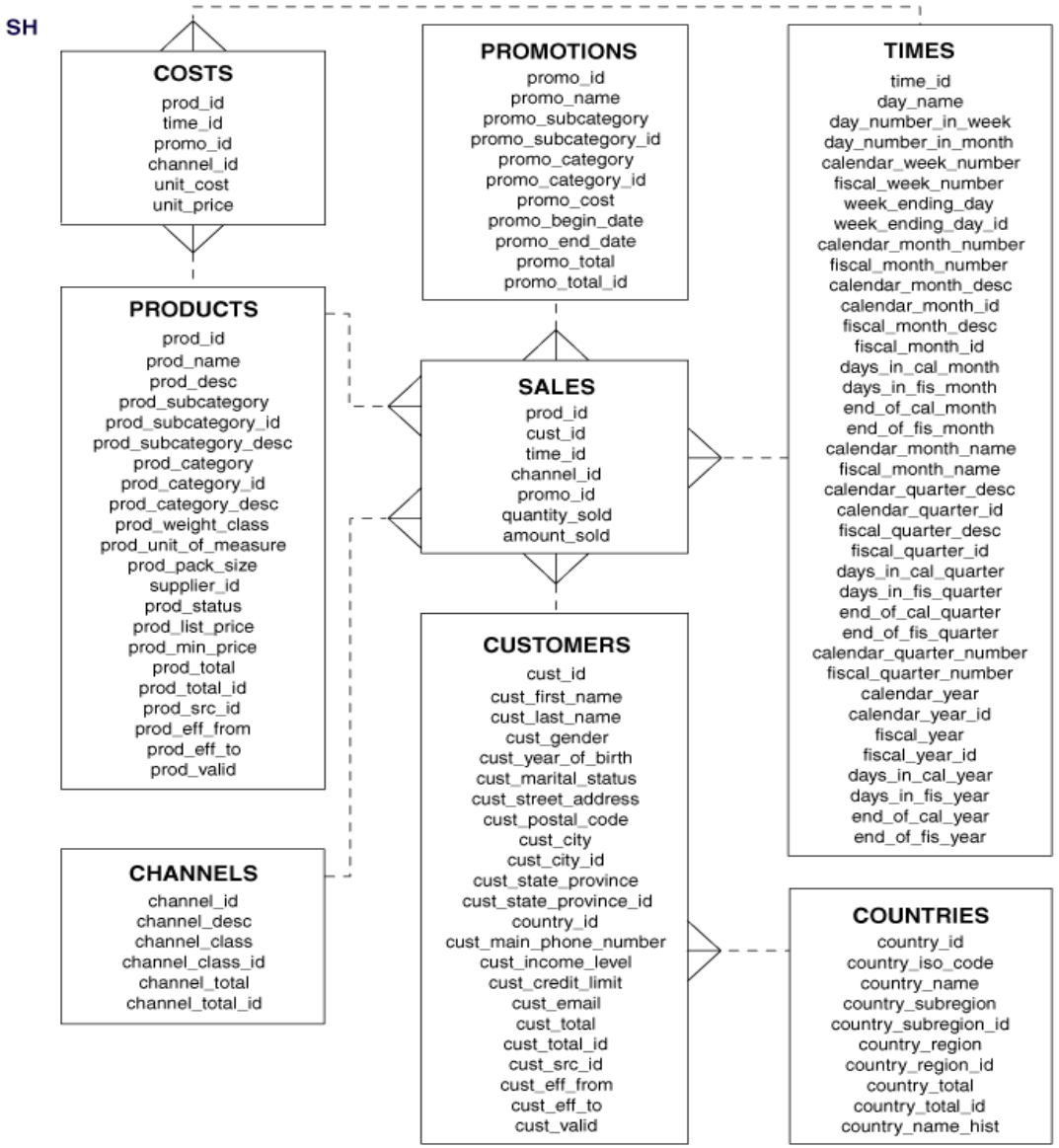
해쉬 테이블

해쉬값	DEPTNO	DNAME	LOC
dkfle21jfn..	30	SALES	CHICAGO

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

5. 오라클 교육용 테이블 ER 다이어 그램



해쉬조인 튜닝 연습을 하려면 ?

조금 더 큰 테이블이 필요합니다.

https://cafe.daum.net/oracleoracle/Sdyr/1048

배운 내용 정리

1. 검색조건이 없는 두개의 테이블을 해쉬조인할 때는 크기가 작은 테이블을 해쉬 테이블로 구성합니다.

2. 검색조건이 있는 두개의 테이블을 해쉬조인할 때는 검색조건으로 엑세스 되는 건수가 작은 테이블을 해쉬 테이블로 구성합니다.

이럴때는

방법15. 해쉬조인으로 유도하자 ! 두번째

▣ 학습 내용

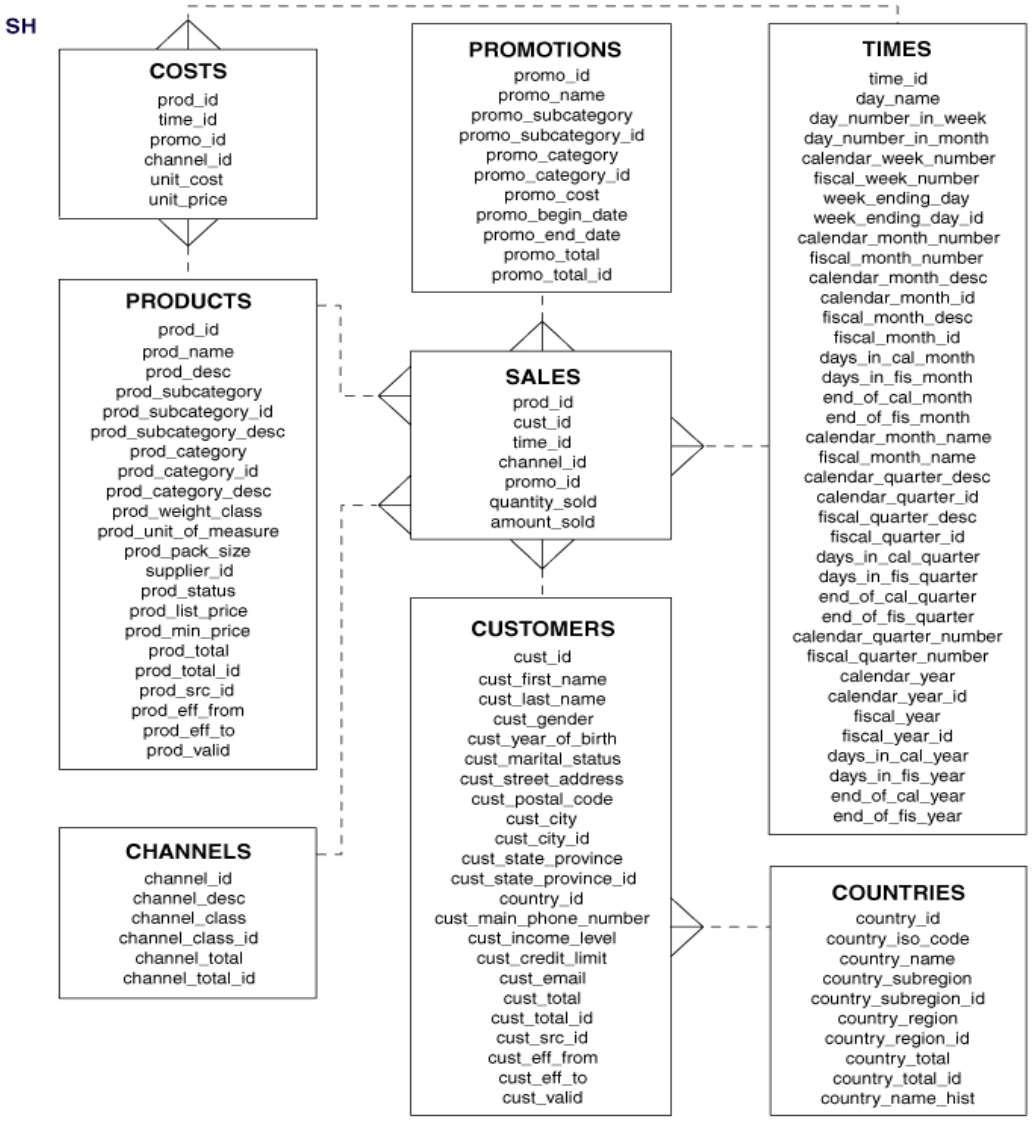
1. 3개 이상의 테이블을 해쉬조인 할때 해쉬 테이블을 선정하는법을 이해합니다.

2. 검색조건이 있었을때의 해쉬조인시 해쉬 테이블을 선정하는 방법을 이해합니다.

▣ 학습 목표

3개이상의 해쉬조인시 힌트를 이용하여 해쉬 테이블을 지정할 수 있습니다.

1. 3개 이상의 테이블을 해쉬조인 할때 조인순서는?



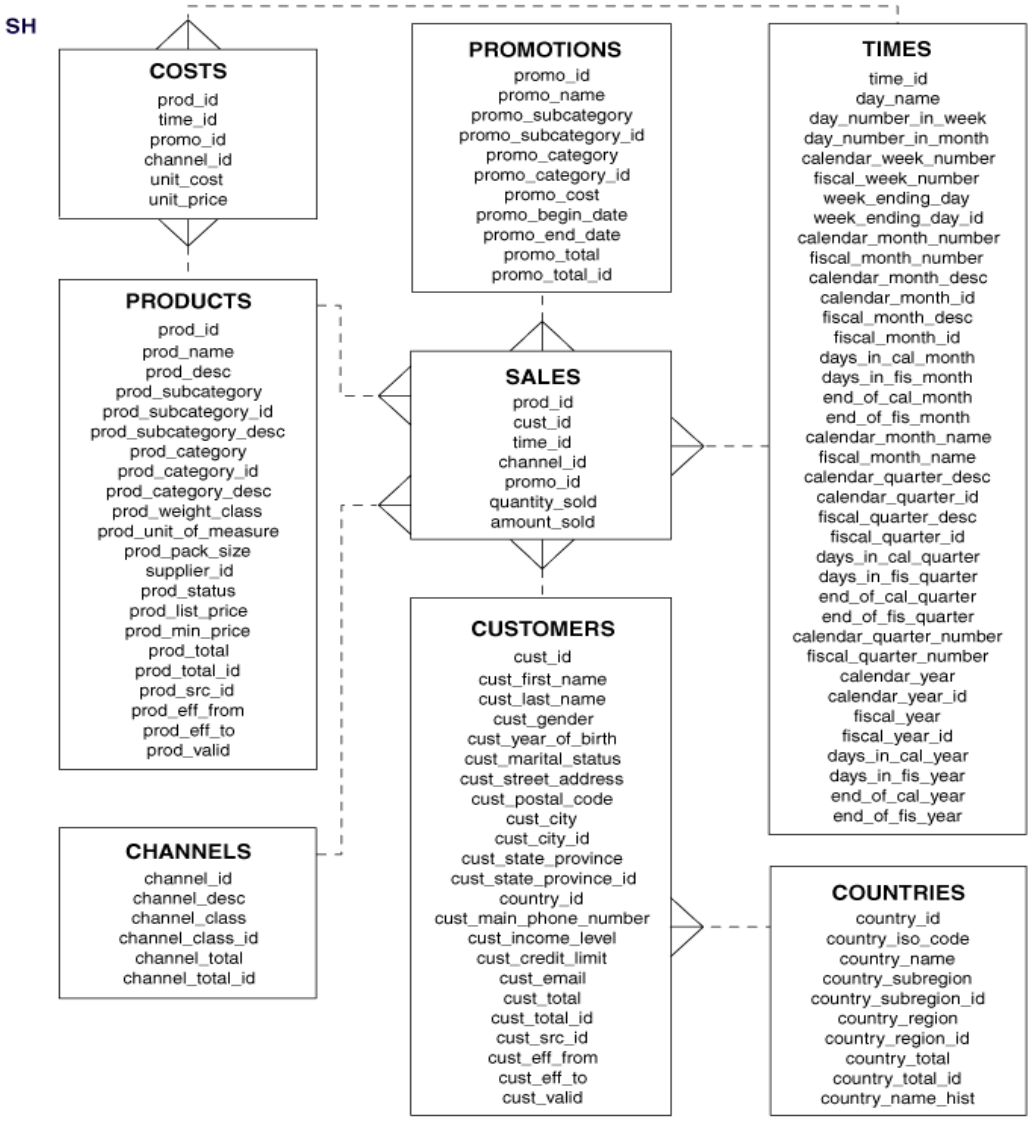

```
select /*+                    ?                    */
 p.prod_name, t.CALENDAR_YEAR,  sum(s.amount_sold)
 from   sales100  s, times100  t, products100  p
 where   s.time_id = t.time_id
 and  s.prod_id = p.prod_id
 group  by  p.prod_name, t.calendar_year;
```

① select count(*) from products100; 72건

② select count(*) from sales100; 918843건

③ select count(*) from times100; 1826건

2. 3개 이상의 테이블을 해쉬조인 할때 조인순서의 답




```
select /*+ leading(p s t) use_hash(s) use_hash(t) */
 p.prod_name, t.CALENDAR_YEAR, sum(s.amount_sold)
 from sales100 s, times100 t, products100 p
 where s.time_id = t.time_id
 and s.prod_id = p.prod_id
 group by p.prod_name, t.calendar_year;
```

```
select /*+ leading(t s p) use_hash(s) use_hash(p) */
 p.prod_name, t.CALENDAR_YEAR, sum(s.amount_sold)
 from sales100 s, times100 t, products100 p
 where s.time_id = t.time_id
 and s.prod_id = p.prod_id
 group by p.prod_name, t.calendar_year;
```

3. 3개 이상의 테이블을 해쉬조인 할때 해쉬 테이블 구성방법

```
select /*+ leading(t s p) use_hash(s) use_hash(p) */
  p.prod_name, t.CALENDAR_YEAR, sum(s.amount_sold)
  from sales100 s, times100 t, products100 p
  where s.time_id = t.time_id
  and s.prod_id = p.prod_id
  group by p.prod_name, t.calendar_year;
```

```
11
12 ----------------------------------------------------------------------------------------------------------------------
13 | Id  | Operation              | Name         | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
14 ----------------------------------------------------------------------------------------------------------------------
15 |   0 | SELECT STATEMENT       |              |      1 |        |      2 |00:00:00.12 |    4499 |       |       |          |
16 |   1 |  HASH GROUP BY         |              |      1 |      2 |      2 |00:00:00.12 |    4499 |  1223K|  1223K|  628K (0)|
17 |*  2 |   HASH JOIN            |              |      1 |   3195 |   6669 |00:00:00.12 |    4499 |   28M|  4517K|   35M (0)|
18 |*  3 |    HASH JOIN           |              |      1 |   460K|   492K|00:00:00.06 |    4495 |  1744K|  1744K| 1671K (0)|
19 |*  4 |     TABLE ACCESS FULL| TIMES100     |      1 |    731 |    731 |00:00:00.01 |      54 |       |       |          |
20 |   5 |     TABLE ACCESS FULL| SALES100     |      1 |   918K|   918K|00:00:00.03 |    4440 |       |       |          |
21 |*  6 |    TABLE ACCESS FULL | PRODUCTS100  |      1 |      1 |      1 |00:00:00.01 |       3 |       |       |          |
22 ----------------------------------------------------------------------------------------------------------------------
23
24 Predicate Information (identified by operation id):
25 ---------------------------------------------------
26
27    2 - access("S"."PROD_ID"="P"."PROD_ID")
28    3 - access("S"."TIME_ID"="T"."TIME_ID")
29    4 - filter(("T"."CALENDAR_YEAR"=2000 OR "T"."CALENDAR_YEAR"=2001))
30    6 - filter("P"."PROD_NAME" LIKE 'Deluxe%')
```

4. products100 테이블을 해쉬 테이블로 구성하려면?

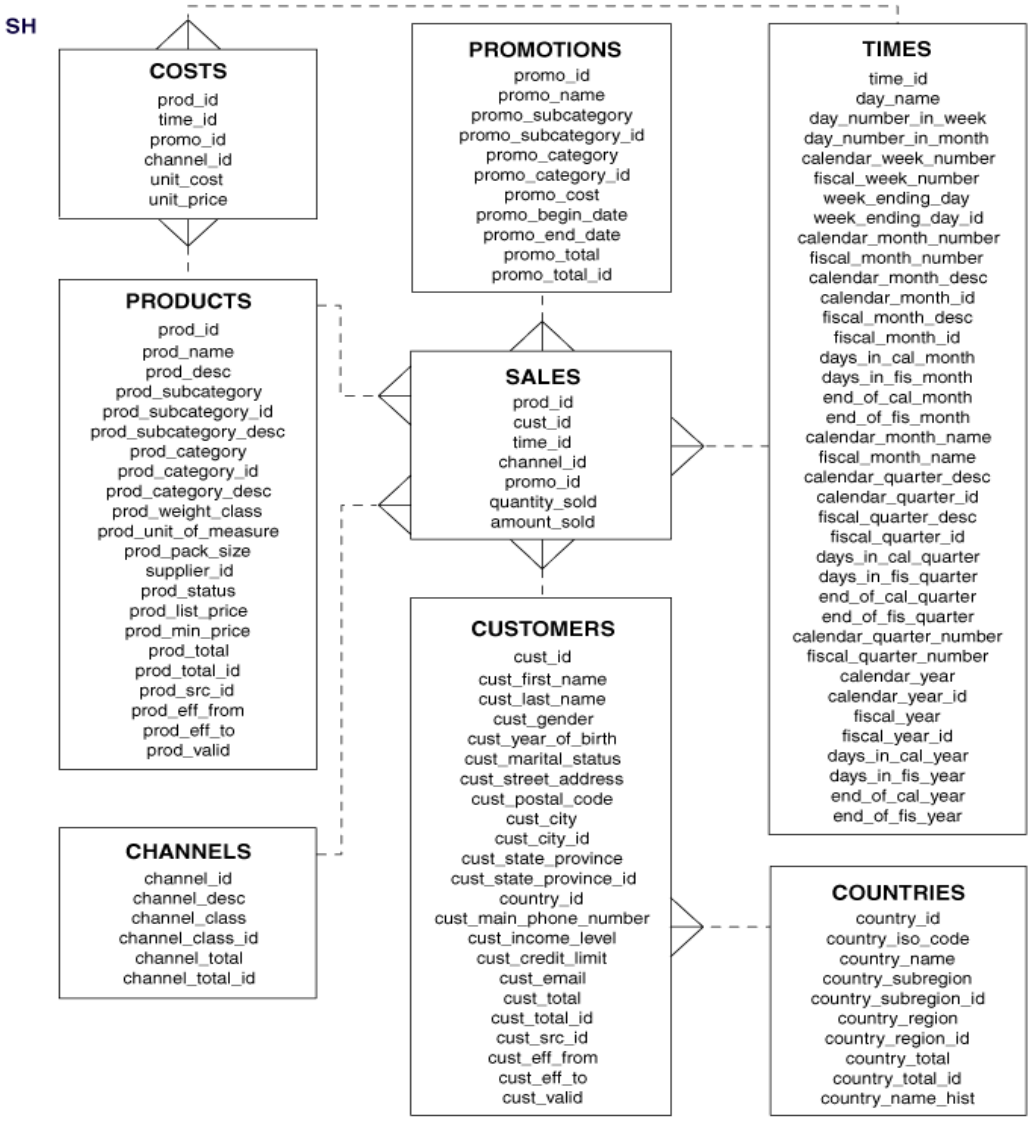
```
select /*+ leading(t s p) use_hash(s) use_hash(p) swap_join_inputs(p) */
  p.prod_name, t.CALENDAR_YEAR, sum(s.amount_sold)
  from sales100 s, times100 t, products100 p
  where s.time_id = t.time_id
  and s.prod_id = p.prod_id
  group by p.prod_name, t.calendar_year;
```

```
---------------------------------------------------------------------------------------------------------------
| Id  | Operation            | Name        | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
---------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |             |      1 |        |      2 |00:00:00.07 |    4499 |       |       |          |
|   1 |  HASH GROUP BY       |             |      1 |      2 |      2 |00:00:00.07 |    4499 |  1223K|  1223K|  639K (0)|
|*  2 |   HASH JOIN          |             |      1 |   3195 |   6669 |00:00:00.13 |    4499 |  1538K|  1538K|  606K (0)|
|*  3 |    TABLE ACCESS FULL | PRODUCTS100 |      1 |      1 |      1 |00:00:00.01 |       3 |       |       |          |
|*  4 |    HASH JOIN         |             |      1 |   460K|   492K|00:00:00.06 |    4495 |  1744K|  1744K| 1542K (0)|
|*  5 |     TABLE ACCESS FULL| TIMES100    |      1 |   731 |   731 |00:00:00.01 |      54 |       |       |          |
|   6 |     TABLE ACCESS FULL| SALES100    |      1 |   918K|   918K|00:00:00.03 |    4440 |       |       |          |
---------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("S"."PROD_ID"="P"."PROD_ID")
   3 - filter("P"."PROD_NAME" LIKE 'Deluxe%')
   4 - access("S"."TIME_ID"="T"."TIME_ID")
   5 - filter(("T"."CALENDAR_YEAR"=2000 OR "T"."CALENDAR_YEAR"=2001))
```

5. 검색조건이 있었을때의 해쉬조인 순서는?



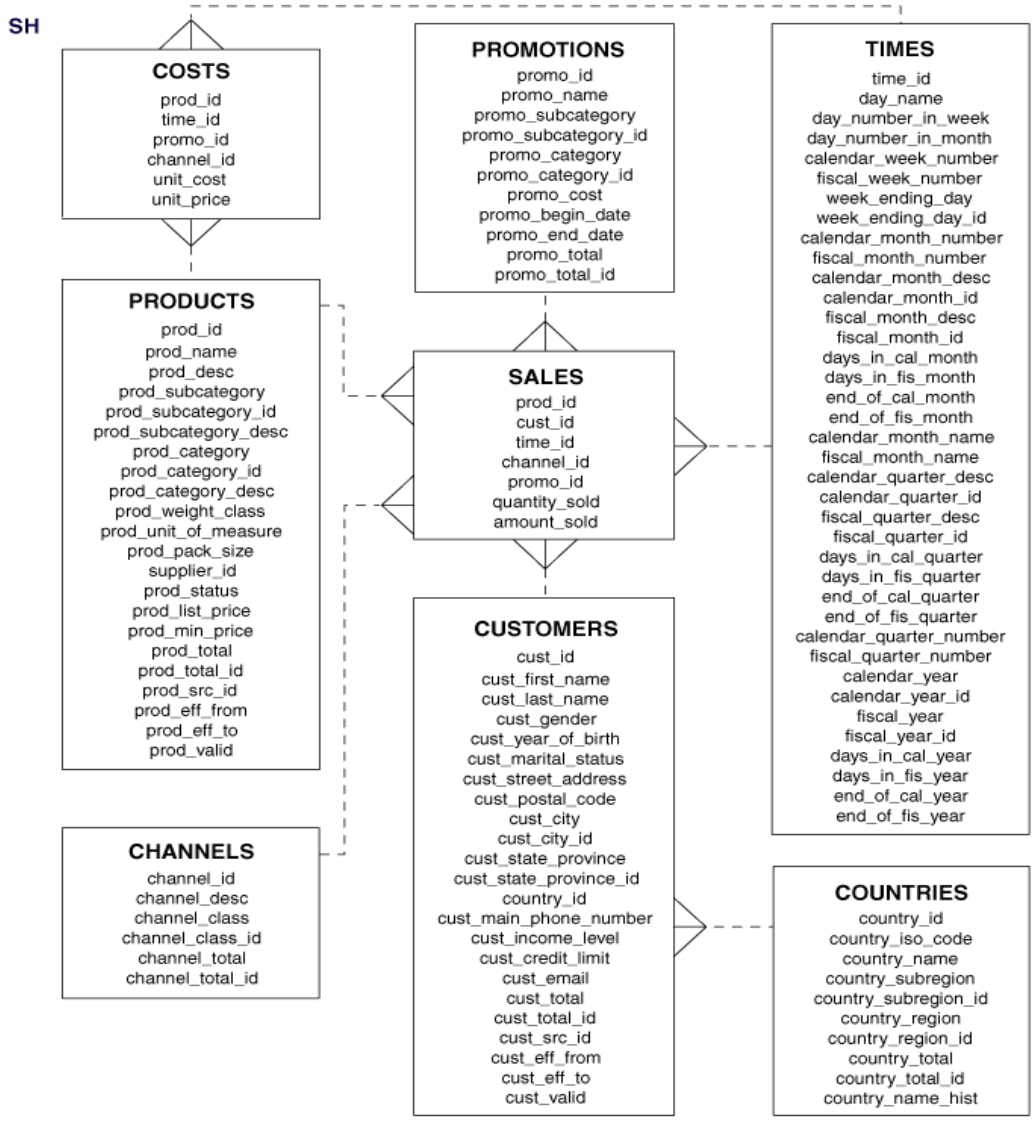

```
select /*+                    ?                    */
p.prod_name, t.CALENDAR_YEAR, sum(s.amount_sold)
 from  sales100  s, times100  t, products100  p
 where  s.time_id = t.time_id
 and  s.prod_id = p.prod_id
  and  t.CALENDAR_YEAR in (2000,2001)
  and p.prod_name like 'Deluxe%'
  group  by  p.prod_name, t.calendar_year;
```

① select count(*) from products100
where prod_name like 'Deluxe%'; 1건

② select count(*) from sales100; 918843건

③ select count(*)
from times100
where CALENDAR_YEAR in (2000,2001); 731건

6. 검색조건이 있었을때의 해쉬조인 순서는 답




```
select /*+  leading(p s t) use_hash(s) use_hash(t)
            swap_join_inputs(p)      */
p.prod_name, t.CALENDAR_YEAR, sum(s.amount_sold)
 from  sales100  s, times100  t, products100  p
 where  s.time_id = t.time_id
 and  s.prod_id = p.prod_id
  and  t.CALENDAR_YEAR in (2000,2001)
  and p.prod_name like 'Deluxe%'
  group  by  p.prod_name, t.calendar_year;
```

① select count(*) from products100
where prod_name like 'Deluxe%'; 1건

② select count(*) from sales100; 918843건

③ select count(*)
from times100
where CALENDAR_YEAR in (2000,2001); 731건

배운 내용 정리

1. 3개 이상의 테이블 해쉬 조인시 해쉬 테이블을 지정하는 오라클 힌트는 swap_join_ionputs 입니다.

2. 3개 이상의 테이블 해쉬 조인시 탐색 테이블을 지정하는 오라클 힌트는 no_swap_join_inputs 입니다.

이럴때는

방법16. sort merge 조인으로 유도하자 !

▣ 학습 내용

1. sort merge join 으로 유도해야하는 조인문이 어떤 문장인지 이해합니다.

2. sort merge join 문의 원리를 이해합니다.

3. sort merge join 문의 조인 순서를 이해합니다.

▣ 학습 목표

sort merge join 문으로 SQL 조인문장을 튜닝할 수 있습니다.

1. non equi join 을 해쉬조인으로 수행할 수 있을까?

```
select  /*+ leading(s e)  use_merge(e) */ e.ename, e.sal, s.grade
 from emp e, salgrade  s
 where  e.sal  between  s.losal and s.hisal;
```

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

salgrade 테이블

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

2. sort merge 조인의 원리

```
select  /*+ leading(d e)  use_merge(e) */ e.ename, d.loc
 from emp e, dept  d
 where  e.deptno = d.deptno;
```

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7934	MILLER	CLERK	7782	1982-01-11	1300		10
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

3. sort merge 조인도 조인 순서가 중요할까?

```
select  /*+ leading(e d)  use_merge(d) */ e.ename, d.loc
 from emp e, dept  d
 where  e.deptno = d.deptno;
```

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7934	MILLER	CLERK	7782	1982-01-11	1300		10
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

이럴때

방법17. outer join 은 이렇게 튜닝하라

▣ 학습 내용

1. outer join 이 실행될 때 조인되는 순서를 이해합니다.

2. outer join 이 실행될때 해쉬 테이블을 구성하는 방법을 학습합니다.

▣ 학습 목표

outer join 조인문장을 튜닝할 수 있습니다.

1. 아우터 조인의 조인순서는?

```
select  e.ename, d.loc
 from  emp  e, dept  d
 where  e.deptno (+) = d.deptno  ;
```

①

```
select  e.ename, d.loc
 from  emp  e, dept  d
 where  e.deptno  = d.deptno (+)  ;
```

②

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10
2921	JACK				4500		70

①

②

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

1. 아우터 조인의 조인순서는?

```
select  e.ename, d.loc
 from  emp  e, dept  d
 where  e.deptno (+) = d.deptno  ;
```

①

```
select  e.ename, d.loc
 from  emp  e, dept  d
 where  e.deptno  = d.deptno (+)  ;
```

②

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10
2921	JACK				4500		70

①

②

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

2. 아우터 조인의 해쉬 조인시에 해쉬 테이블은?

```
select /*+ gather_plan_statistics  leading(d e) use_hash(e) */  e.ename, d.loc
  from  emp  e, dept  d
  where  e.deptno (+) = d.deptno  ;
```

해쉬 함수

오라클 메모리 영역 PGA

해쉬 테이블

해쉬값	DEPTNO	DNAME	LOC
kdl21fnae..	10	ACCOUNTING	NEW YORK
fekfh32en..	20	RESEARCH	DALLAS
dkfle21jfn..	30	SALES	CHICAGO
eekfn09dn...	40	OPERATIONS	BOSTON

PROB 테이블

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

3. 아우터 조인순서와 관계없이 해쉬 테이블을 정할 수 있는가?

```
select /*+ gather_plan_statistics  leading(d e) use_hash(e) */  e.ename, d.loc
  from  emp  e, dept  d
  where  e.deptno  = d.deptno (+)  ;
```

해쉬 함수

오라클 메모리 영역 PGA

해쉬 테이블

해쉬값	DEPTNO	DNAME	LOC
kdl21fnae..	10	ACCOUNTING	NEW YORK
fekfh32en..	20	RESEARCH	DALLAS
dkfle21jfn..	30	SALES	CHICAGO
eekfn09dn...	40	OPERATIONS	BOSTON

PROB 테이블

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7566	JONES	MANAGER	7839	1981-04-01	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
7900	JAMES	CLERK	7698	1981-12-11	950		30
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
7902	FORD	ANALYST	7566	1981-12-11	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7934	MILLER	CLERK	7782	1982-01-11	1300		10

dept 테이블

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

4. 아우터 조인순서와 관계없이 해쉬 테이블을 정할 수 있는가?

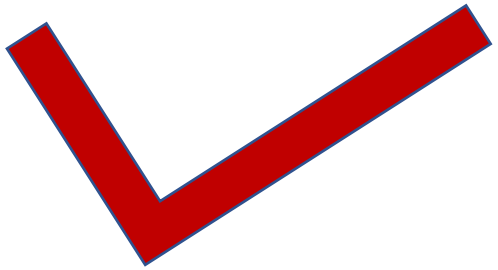
```
select /*+ gather_plan_statistics  leading(d e) use_hash(e) */  e.ename, d.loc
  from  emp  e, dept  d
  where  e.deptno  = d.deptno (+)  ;
```

```
---------------------------------------------------------------------------------------------------------
| Id  | Operation           | Name | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
---------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |      |      1 |        |     14 |00:00:00.01 |      12 |       |       |          |
|*  1 |  HASH JOIN OUTER    |      |      1 |     14 |     14 |00:00:00.01 |      12 |  1856K|  1856K|  906K (0)|
|   2 |   TABLE ACCESS FULL| EMP  |      1 |     14 |     14 |00:00:00.01 |       6 |       |       |          |
|   3 |   TABLE ACCESS FULL| DEPT |      1 |      4 |      4 |00:00:00.01 |       6 |       |       |          |
---------------------------------------------------------------------------------------------------------
```

```
select /*+ gather_plan_statistics  leading(d e) use_hash(e)
           swap_join_inputs(d) */  e.ename, d.loc
  from  emp  e, dept  d
  where  e.deptno  = d.deptno (+)  ;
```

```
---------------------------------------------------------------------------------------------------------
| Id  | Operation              | Name | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
---------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |      |      1 |        |     14 |00:00:00.01 |      12 |       |       |          |
|*  1 |  HASH JOIN RIGHT OUTER|      |      1 |     14 |     14 |00:00:00.01 |      12 |  1797K|  1797K|  975K (0)|
|   2 |   TABLE ACCESS FULL    | DEPT |      1 |      4 |      4 |00:00:00.01 |       6 |       |       |          |
|   3 |   TABLE ACCESS FULL    | EMP  |      1 |     14 |     14 |00:00:00.01 |       6 |       |       |          |
---------------------------------------------------------------------------------------------------------
```

이럴때는



방법18. 서브쿼리부터 실행되게하라 !

▣ 학습 내용

1. 순수하게 서브쿼리문으로 수행되는 방법을 학습합니다.

2. 서브쿼리문에서 서브쿼리문부터 실행되게하는 방법을 학습합니다.

3. 서브쿼리문에서 메인쿼리문부터 실행되게하는 방법을 학습합니다.

▣ 학습 목표

서브쿼리문의 실행순서를 힌트로 조정할 수 있습니다.

1.서브쿼리문의 실행순서 첫번째

서브쿼리 부터 실행되게 되면?

```
select  ename, sal
  from  emp 14건
  where  deptno  in  ( select   deptno
                       from  dept 4건
                       where  deptno = 10 );
```

main query

sub query

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		3	00:00:00.01	14
* 1	TABLE ACCESS FULL	EMP	1	1	3	00:00:00.01	14
* 2	FILTER		3		1	00:00:00.01	7
* 3	TABLE ACCESS FULL	DEPT	1	1	1	00:00:00.01	7

2. 서브쿼리문의 실행순서 두번째

메인쿼리 부터 실행되게 되면?

```
select  ename, sal
  from  emp 14건
  where  deptno  in  ( select   deptno
                       from  dept 4건
                       where  deptno = 10 );
```

main query

sub query

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		3	00:00:00.01	14
* 1	FILTER		1		3	00:00:00.01	14
2	TABLE ACCESS FULL	EMP	1	14	14	00:00:00.01	7
* 3	FILTER		3		1	00:00:00.01	7
* 4	TABLE ACCESS FULL	DEPT	1	1	1	00:00:00.01	7

3. 서브쿼리문의 실행순서 세번째

서브쿼리 부터 실행되게 되면?

```
select  ename, sal
  from  emp 14건
  where  deptno  in  ( select   /*+ no_unnest push_subq */ deptno
                          from  dept 4건
                          where  deptno = 10 );
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		3	00:00:00.01	14
* 1	TABLE ACCESS FULL	EMP	1	1	3	00:00:00.01	14
* 2	FILTER		3		1	00:00:00.01	7
* 3	TABLE ACCESS FULL	DEPT	1	1	1	00:00:00.01	7

4. 서브쿼리문의 실행순서 네번째

메인쿼리 부터 실행되게 되면?

```
select ename, sal
  from emp 14건
  where deptno in ( select /*+ no_unnest no_push_subq */
                           deptno
                    from dept  4건
                    where deptno = 10 );
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		3	00:00:00.01	14
* 1	FILTER		1		3	00:00:00.01	14
2	TABLE ACCESS FULL	EMP	1	14	14	00:00:00.01	7
* 3	FILTER		3		1	00:00:00.01	7
* 4	TABLE ACCESS FULL	DEPT	1	1	1	00:00:00.01	7

5. 순수하게 서브쿼리문으로 실행되었을 때의 힌트

① nest 뜻? 감싸라 !

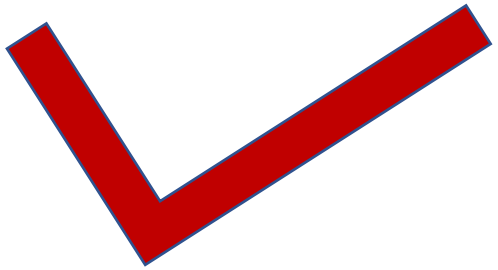
② unnest 뜻? 감싸지 말아라 !

③ no_unnest 뜻? 강하게 감싸라 ~~~~~~

④ push_subq 뜻? 서브쿼리부터 실행해라 !

⑤ no_push_subq 뜻? 메인쿼리부터 실행해라 !

이럴때는



방법19. 서브쿼리를 세미조인이 되게하라 !

▣ 학습 내용

1. 순수하게 서브쿼리로 실행되었을때와 안티조인으로 수행되었을때의 차이를 학습

2. 해쉬 안티조인으로 수행되게 하는 방법을 학습합니다.

3. 해쉬 right 안티조인으로 수행되게 하는 방법을 학습합니다.

▣ 학습 목표

세미조인으로 수행되게 서브쿼리문을 튜닝할 수 있습니다.

1.서브쿼리의 데이터가 많을때

서브쿼리 부터 실행되게 되면?

main query

sub query

```
select  /*+ gather_plan_statistics */ *
 from customers100  c   55500건
 where c.cust_id in ( select /*+ no_unnest  push_subq */ cust_id
                      from  sales100 s   918843건
                      where  amount_sold between 0 and 10000 );
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		50	00:00:16.03	2013K
* 1	TABLE ACCESS FULL	CUSTOMERS100	1	50	50	00:00:16.03	2013K
* 2	TABLE ACCESS FULL	SALES100	497	2	50	00:00:18.00	2013K

2.서브쿼리의 데이터가 많을때

메인쿼리 부터 실행되게 되면?

main query

sub query

```
select  /*+ gather_plan_statistics */ *
 from customers100  c    55500건
 where c.cust_id in ( select /*+ no_unnest no_push_subq */
                                cust_id
                        from  sales100 s      918843건
                        where  amount_sold between 0 and 10000 );
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		50	00:00:05.98	2013K
* 1	FILTER		1		50	00:00:05.98	2013K
2	TABLE ACCESS FULL	CUSTOMERS100	1	55500	497	00:00:00.01	17
* 3	TABLE ACCESS FULL	SALES100	497	2	50	00:00:18.09	2013K

3. 서브쿼리의 데이터가 많을때

해쉬 세미조인으로 실행되게 되면?

main query

sub query

```
select  /*+ gather_plan_statistics */ *
 from customers100  c   55500건
 where c.cust_id in ( select /*+ unnest  hash_sj */ cust_id
                      from  sales100 s  918843건
                      where  amount_sold between 0 and 10000 );
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	1Mem	Used-Mem
0	SELECT STATEMENT		1		50	00:00:00.05	1523			
* 1	HASH JOIN SEMI		1	7059	50	00:00:00.05	1523	14M	2068K	15M (0)
2	TABLE ACCESS FULL	CUSTOMERS100	1	55500	55500	00:00:00.02	1520			
* 3	TABLE ACCESS FULL	SALES100	1	918K	50	00:00:00.01	3			

4. 서브쿼리의 데이터가 많을때

해쉬 right 세미조인으로 실행되게 되면?

main query

sub query

```
select  /*+ gather_plan_statistics */ *
 from customers100  c   55500건
 where c.cust_id in ( select /*+ unnest  hash_sj
                                  swap_join_inputs(s) */ cust_id
                 from  sales100 s   918843건
                 where  amount_sold between 0 and 10000 );
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	1Mem	Used-Mem
0	SELECT STATEMENT		1		50	00:00:00.15	4456			
* 1	HASH JOIN RIGHT SEMI		1	7059	50	00:00:00.15	4456	43M	6071K	47M (0)
* 2	TABLE ACCESS FULL	SALES100	1	918K	918K	00:00:00.03	4440			
3	TABLE ACCESS FULL	CUSTOMERS100	1	55500	497	00:00:00.01	16			

이럴때는

방법20. 서브쿼리를 해쉬 안티 조인이 되게하라 !

▣ 학습 내용

1. 순수하게 서브쿼리로 실행되었을때와 세미조인으로 수행되었을때의 차이를 학습

2. 해쉬 세미조인으로 수행되게 하는 방법을 학습합니다.

3. 해쉬 right 세미조인으로 수행되게 하는 방법을 학습합니다.

▣ 학습 목표

세미조인으로 수행되게 서브쿼리문을 튜닝할 수 있습니다.

1.서브쿼리의 데이터가 많을때

서브쿼리 부터 실행되게 되면?

main query

sub query

```
select  /*+ gather_plan_statistics */ *
 from customers100  c    55500건
 where c.cust_id not in ( select /*+ no_unnest  push_subq */
                                cust_id
                            from  sales100 s        918843건
                          where  amount_sold between 0 and 10000 );
```

```
0
1 ----------------------------------------------------------------------------------------------------
2 | Id  | Operation           | Name          | Starts | E-Rows | A-Rows |   A-Time    | Buffers | Reads  |
3 ----------------------------------------------------------------------------------------------------
4 |   0 | SELECT STATEMENT    |               |      1 |        |     50 |00:00:02.75 |    226K|   4440 |
5 |*  1 |  TABLE ACCESS FULL  | CUSTOMERS100  |      1 |   2775 |     50 |00:00:02.75 |    226K|   4440 |
6 |*  2 |   TABLE ACCESS FULL| SALES100      |     57 |      2 |      7 |00:00:02.74 |    226K|   4434 |
7 ----------------------------------------------------------------------------------------------------
```

2. 서브쿼리의 데이터가 많을때

메인쿼리 부터 실행되게 되면?

main query

sub query

```
select  /*+ gather_plan_statistics */ *
 from customers100  c    55500건
 where c.cust_id not in ( select /*+ no_unnest no_push_subq */
                                cust_id
                                from  sales100 s                918843건
                              where  amount_sold between 0 and 10000 );
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		50	00:00:02.16	226K
* 1	FILTER		1		50	00:00:02.16	226K
2	TABLE ACCESS FULL	CUSTOMERS100	1	55500	57	00:00:00.01	5
* 3	TABLE ACCESS FULL	SALES100	57	2	7	00:00:02.08	226K

3. 서브쿼리의 데이터가 많을때

해쉬 안티조인으로 실행되게 되면?

main query

sub query

```
select  /*+ gather_plan_statistics */ *
 from customers100  c
 where c.cust_id not in ( select /*+ unnest  hash_aj */ cust_id
                          from  sales100 s
                          where  amount_sold between 0 and 10000 );
```

55500건

918843건

```
---------------------------------------------------------------------------------------------------------------------
| Id  | Operation           | Name          | Starts | E-Rows | A-Rows |   A-Time    | Buffers | Reads  |  OMem |  1Mem | Used-Mem |
---------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |               |      1 |        |     50 |00:00:00.41 |    5960 |   1511 |       |       |          |
|*  1 |  HASH JOIN ANTI     |               |      1 |  48441 |     50 |00:00:00.41 |    5960 |   1511 |   14M|  2068K|   15M (0)|
|   2 |   TABLE ACCESS FULL| CUSTOMERS100  |      1 |  55500 |  55500 |00:00:00.01 |    1520 |   1511 |       |       |          |
|*  3 |   TABLE ACCESS FULL| SALES100      |      1 |   918K|   918K|00:00:00.03 |    4440 |      0 |       |       |          |
---------------------------------------------------------------------------------------------------------------------
```

4. 서브쿼리의 데이터가 많을때

해쉬 right 안티조인으로 실행되게 되면?

main query

sub query

```
select  /*+ gather_plan_statistics */ *
 from customers100  c
 where c.cust_id not in ( select /*+ unnest  hash_sj
                                swap_join_inputs(s) */ cust_id
                          from  sales100 s
                         where  amount_sold between 0 and 10000 );
```

55500건

918843건

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	1Mem	Used-Mem
0	SELECT STATEMENT		1		50	00:00:00.15	4444			
* 1	HASH JOIN RIGHT ANTI		1	48441	50	00:00:00.15	4444	43M	6071K	47M (0)
* 2	TABLE ACCESS FULL	SALES100	1	918K	918K	00:00:00.03	4440			
3	TABLE ACCESS FULL	CUSTOMERS100	1	55500	57	00:00:00.01	4			

끝까지 잘 공부 하셨습니다. 감사합니다.