

실무에서 바로 쓰는
SQL 튜닝 방법 20가지

이럴때는



방법1. SQL 의 실행계획을 확인하자 !

□ 학습 내용

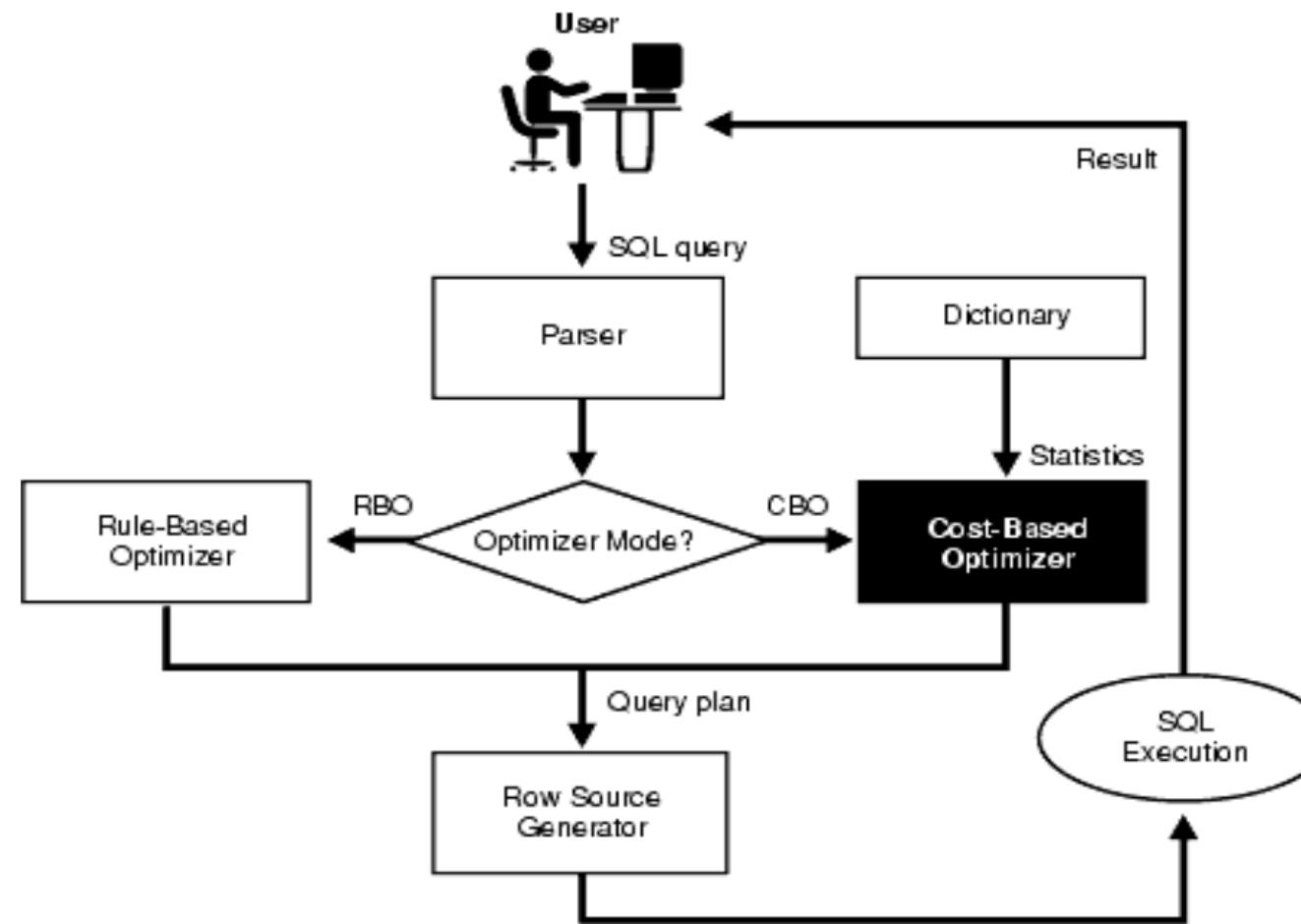
1. SQL의 실행계획이 무엇인지 학습합니다.
2. 예상 실행계획이 무엇인지 학습합니다.
3. 실제 실행계획이 무엇인지 학습합니다.
4. 실행계획을 제어하는 힌트가 무엇인지 학습합니다.

□ 학습 목표

SQL의 실행계획을 보면 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 옵티마이저로 하여금 힌트대로 실행계획을 생성해달라고 주문 "

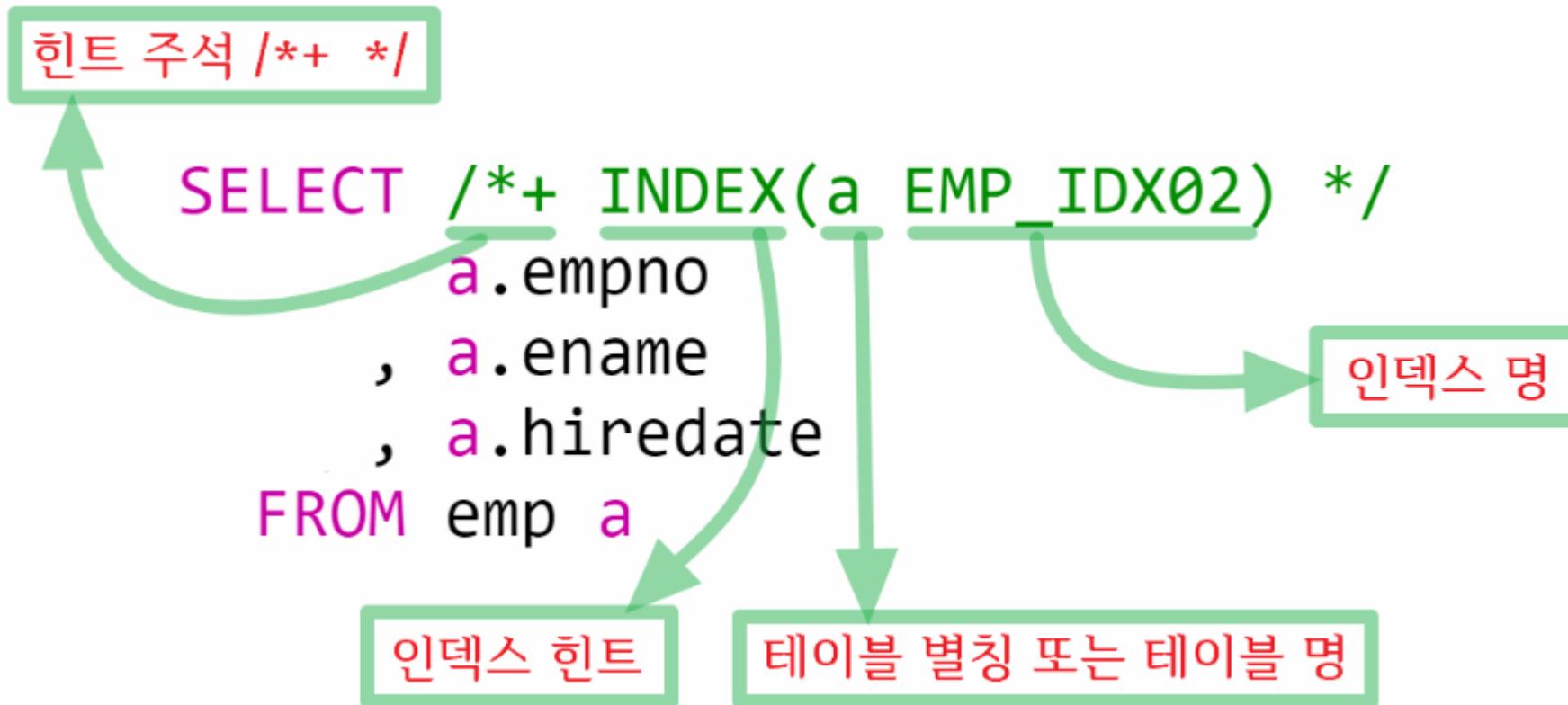
```
select /*+ gather_plan_statistics full(emp) */ ename, sal  
from emp  
where sal = 1300;
```

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	1	00:00:00.01	7
*	1	TABLE ACCESS FULL	EMP	1	1	00:00:00.01	7

5. 힌트(hint)의 종류



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



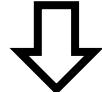
사원 테이블

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

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



사원 테이블

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

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

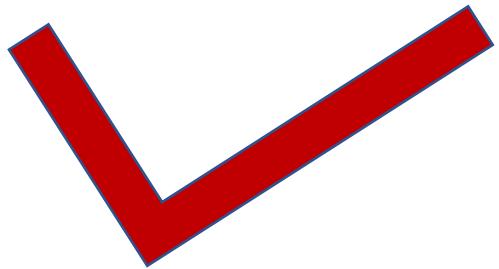


사원 테이블

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

ROWID	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
AAATc1AAHAAAHeAAA	7839	KING	PRESIDENT		1981-11-17	5000		10
AAATc1AAHAAAHeAAB	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
AAATc1AAHAAAHeAAC	7782	CLARK	MANAGER	7839	1981-05-09	2450		10
AAATc1AAHAAAHeAAD	7566	JONES	MANAGER	7839	1981-04-01	2975		20
AAATc1AAHAAAHeAAE	7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
AAATc1AAHAAAHeAAF	7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
AAATc1AAHAAAHeAAG	7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
AAATc1AAHAAAHeAAH	7900	JAMES	CLERK	7698	1981-12-11	950		30
AAATc1AAHAAAHeAAI	7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
AAATc1AAHAAAHeAAJ	7902	FORD	ANALYST	7566	1981-12-11	3000		20
AAATc1AAHAAAHeAAK	7369	SMITH	CLERK	7902	1980-12-09	800		20
AAATc1AAHAAAHeAAL	7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
AAATc1AAHAAAHeAAM	7876	ADAMS	CLERK	7788	1983-01-15	1100		20
AAATc1AAHAAAHeAAN	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	AAATc1AAHAAAAAHeAAM
ALLEN	AAATc1AAHAAAAAHeAAF
BLAKE	AAATc1AAHAAAAAHeAAB
CLARK	AAATc1AAHAAAAAHeAAC
FORD	AAATc1AAHAAAAAHeAAJ
JAMES	AAATc1AAHAAAAAHeAAH
JONES	AAATc1AAHAAAAAHeAAD
KING	AAATc1AAHAAAAAHeAAA
MARTIN	AAATc1AAHAAAAAHeAAE
MILLER	AAATc1AAHAAAAAHeAAN
SCOTT	AAATc1AAHAAAAAHeAAL
SMITH	AAATc1AAHAAAAAHeAAK
TURNER	AAATc1AAHAAAAAHeAAG
WARD	AAATc1AAHAAAAAHeAAl

emp 테이블

ROWID	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
AAATc1AAHAAAAAHeAAA	7839	KING	PRESIDENT		1981-11-17	5000		10
AAATc1AAHAAAAAHeAAB	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
AAATc1AAHAAAAAHeAAC	7782	CLARK	MANAGER	7839	1981-05-09	2450		10
AAATc1AAHAAAAAHeAAD	7566	JONES	MANAGER	7839	1981-04-01	2975		20
AAATc1AAHAAAAAHeAAE	7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30
AAATc1AAHAAAAAHeAAF	7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30
AAATc1AAHAAAAAHeAAG	7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30
AAATc1AAHAAAAAHeAAH	7900	JAMES	CLERK	7698	1981-12-11	950		30
AAATc1AAHAAAAAHeAAI	7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30
AAATc1AAHAAAAAHeAAJ	7902	FORD	ANALYST	7566	1981-12-11	3000		20
AAATc1AAHAAAAAHeAAK	7369	SMITH	CLERK	7902	1980-12-09	800		20
AAATc1AAHAAAAAHeAAL	7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
AAATc1AAHAAAAAHeAAM	7876	ADAMS	CLERK	7788	1983-01-15	1100		20
AAATc1AAHAAAAAHeAAN	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';
```

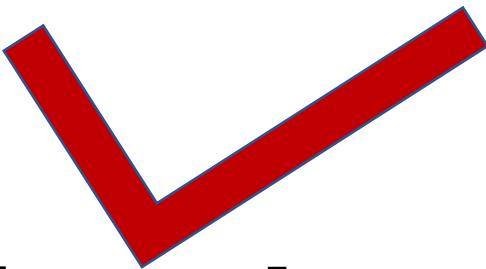
emp_job 인덱스

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
*	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	TABLE ACCESS BY INDEX ROWID BATCHED	EMP	1	4	4	00:00:00.01	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	10	00:00:00.01	7
*	1 TABLE ACCESS FULL	EMP	1	10	10	00:00:00.01	7

9. 튜닝하려면 ?

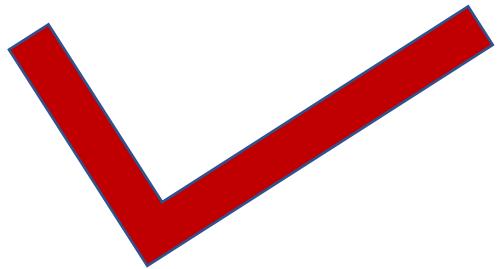
1981년도에 입사한 사원들의 이름과 입사일을 출력하시오 !

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
*	FILTER		1		10	00:00:00.01	2
2	TABLE ACCESS BY INDEX ROWID BATCHED	EMP	1	10	10	00:00:00.01	2
*	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
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);
```

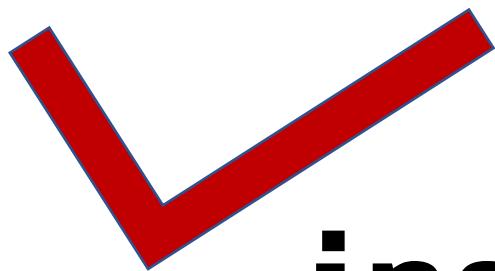
emp_empno_pk 인덱스

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



index full scan 으로 유도하자 !

인덱스 스캔 방법 7가지



	인덱스 엑세스 방법	관련 힌트
1	index range scan	index
2	index unique scan	index
3	index full scan	index
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 처음부터 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 있었을 때

인덱스를 통해서 테이블을 액세스 합니다.

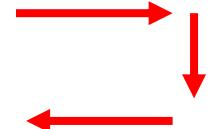
```
select ename,sal  
from emp  
where sal = 3000;
```

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.

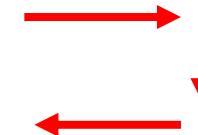
```
select ename,sal  
from emp  
where sal = 3000;
```

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 스캔하면서 원하는 데이터를 검색하는 스캔 방법입니다.

```
select ename,sal  
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.

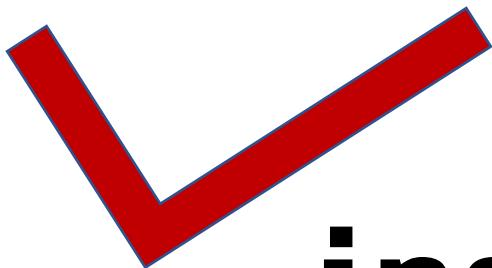
실행하기전에 다음 링크의 스크립트를 통해 emp와 dept를 다시 생성하세요

<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



index skip scan 으로 유도하자 !

index skip scan 이란 ?

인덱스 스kip 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
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

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.

**create index emp_deptno_job
on emp(deptno,job);**

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

```

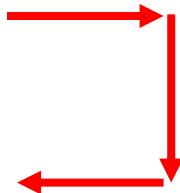
select /*+ index(emp emp_deptno_job) */ ename,deptno, job
from emp
where deptno = 10 and job ='MANAGER';


```

emp_deptno_job 인덱스 index range scan ! emp 테이블

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

ROWID	EMPNO	ENAME	JOB	MGR	...
AAATc1AHAAAAHeAAA	7839	KING	PRESIDENT		...
AAATc1AHAAAAHeAAB	7698	BLAKE	MANAGER	7839	...
AAATc1AHAAAAHeAAC	7782	CLARK	MANAGER	7839	...
AAATc1AHAAAAHeAAD	7566	JONES	MANAGER	7839	...
AAATc1AHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	...
AAATc1AHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	...
AAATc1AHAAAAHeAAG	7844	TURNER	SALESMAN	7698	...
AAATc1AHAAAAHeAAH	7900	JAMES	CLERK	7698	...
AAATc1AHAAAAHeAAI	7521	WARD	SALESMAN	7698	...
AAATc1AHAAAAHeAAJ	7902	FORD	ANALYST	7566	...
AAATc1AHAAAAHeAAK	7369	SMITH	CLERK	7902	...
AAATc1AHAAAAHeAAL	7788	SCOTT	ANALYST	7566	...
AAATc1AHAAAAHeAAM	7876	ADAMS	CLERK	7788	...
AAATc1AHAAAAHeAAN	7934	MILLER	CLERK	7782	...



```

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	...
AAATc1AHAAAAHeAAA	7839	KING	PRESIDENT		...
AAATc1AHAAAAHeAAB	7698	BLAKE	MANAGER	7839	...
AAATc1AHAAAAHeAAC	7782	CLARK	MANAGER	7839	...
AAATc1AHAAAAHeAAD	7566	JONES	MANAGER	7839	...
AAATc1AHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	...
AAATc1AHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	...
AAATc1AHAAAAHeAAG	7844	TURNER	SALESMAN	7698	...
AAATc1AHAAAAHeAAH	7900	JAMES	CLERK	7698	...
AAATc1AHAAAAHeAAI	7521	WARD	SALESMAN	7698	...
AAATc1AHAAAAHeAAJ	7902	FORD	ANALYST	7566	...
AAATc1AHAAAAHeAAK	7369	SMITH	CLERK	7902	...
AAATc1AHAAAAHeAAL	7788	SCOTT	ANALYST	7566	...
AAATc1AHAAAAHeAAM	7876	ADAMS	CLERK	7788	...
AAATc1AHAAAAHeAAN	7934	MILLER	CLERK	7782	...

index skip scan의 효과 높이는 결합 컬럼 인덱스는?

```
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);
```

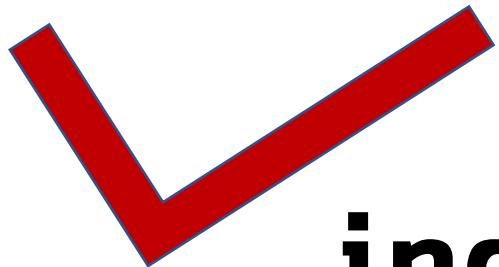
튜닝전:

```
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



index fast full scan 으로 유도하자 !

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
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;
```

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

직업에 인덱스는 있으나 not null 이 보장되지 않는다면 ?
full table scan !

emp 테이블

ROWID	EMPNO	ENAME	JOB	MGR	...
AAATc1AHAAAAHeAAA	7839	KING	PRESIDENT		...
AAATc1AHAAAAHeAAB	7698	BLAKE	MANAGER	7839	...
AAATc1AHAAAAHeAAC	7782	CLARK	MANAGER	7839	...
AAATc1AHAAAAHeAAD	7566	JONES	MANAGER	7839	...
AAATc1AHAAAAHeAAE	7654	MARTIN	SALESMAN	7698	...
AAATc1AHAAAAHeAAF	7499	ALLEN	SALESMAN	7698	...
AAATc1AHAAAAHeAAG	7844	TURNER	SALESMAN	7698	...
AAATc1AHAAAAHeAAH	7900	JAMES	CLERK	7698	...
AAATc1AHAAAAHeAAI	7521	WARD	SALESMAN	7698	...
AAATc1AHAAAAHeAAJ	7902	FORD	ANALYST	7566	...
AAATc1AHAAAAHeAAK	7369	SMITH	CLERK	7902	...
AAATc1AHAAAAHeAAL	7788	SCOTT	ANALYST	7566	...
AAATc1AHAAAAHeAAM	7876	ADAMS	CLERK	7788	...
AAATc1AHAAAAHeAAN	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;
```

- 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;
```



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

```
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	AAAXDOAANAAAHHjAAJ
ANALYST	AAAXDOAANAAAHHjAAL
CLERK	AAAXDOAANAAAHHjAAH
CLERK	AAAXDOAANAAAHHjAAK
CLERK	AAAXDOAANAAAHHjAAM
CLERK	AAAXDOAANAAAHHjAAN
MANAGER	AAAXDOAANAAAHHjAAB
MANAGER	AAAXDOAANAAAHHjAAC
MANAGER	AAAXDOAANAAAHHjAAD
PRESIDENT	AAAXDOAANAAAHHjAAA
SALESMAN	AAAXDOAANAAAHHjAAE
SALESMAN	AAAXDOAANAAAHHjAAF
SALESMAN	AAAXDOAANAAAHHjAAG
SALESMAN	AAAXDOAANAAAHHjAAI

결과

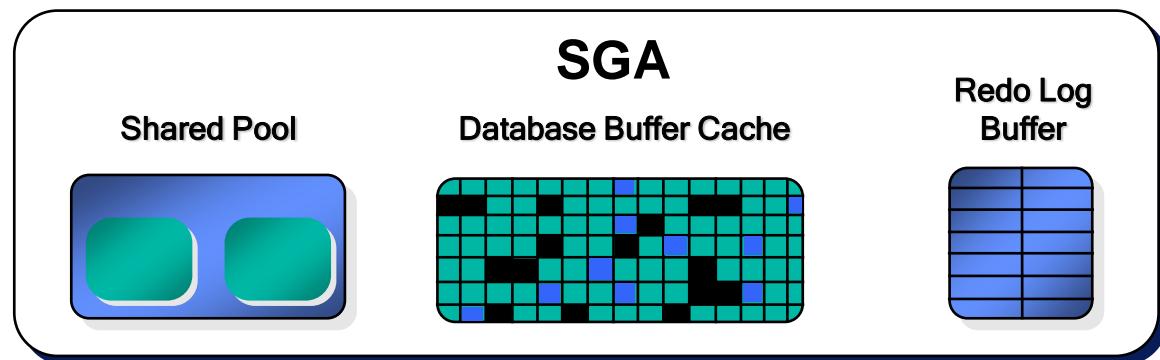
JOB	COUNT(*)
ANALYST	2
CLERK	4
SALESMAN	4
MANAGER	3
PRESIDENT	1

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

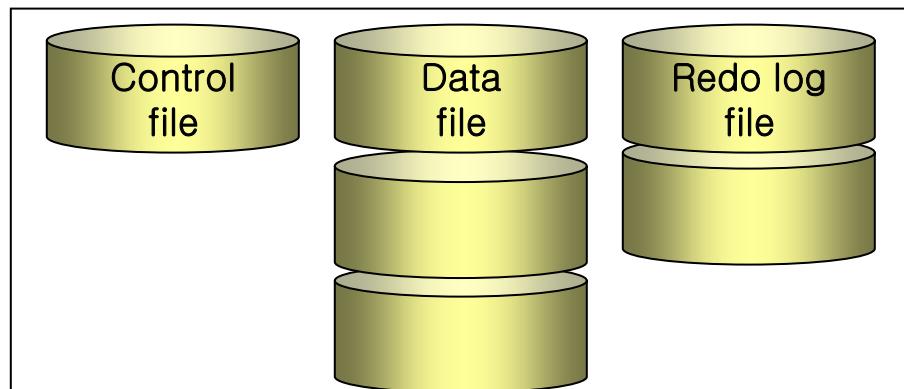
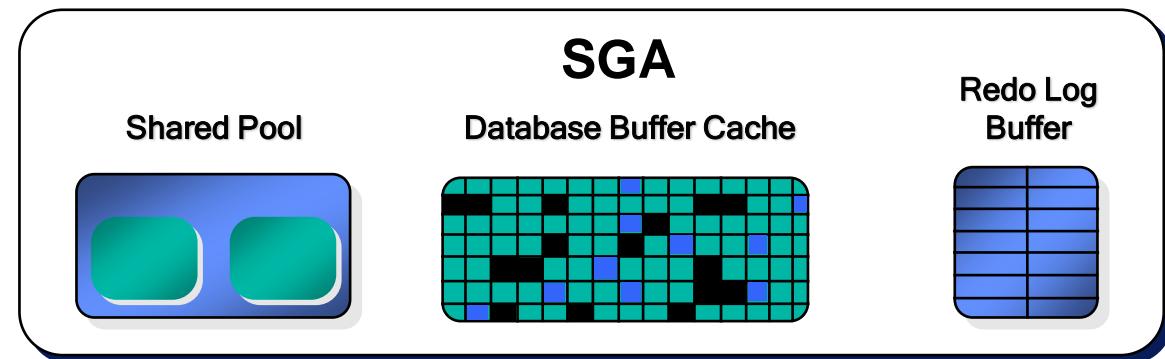
	index full scan	index fast full scan
I/O 방식	single block i/o	multi block i/o
정렬	정렬 보장	정렬 안됨
속도	느림	빠름
병렬읽기	지원 안됨	지원됨

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

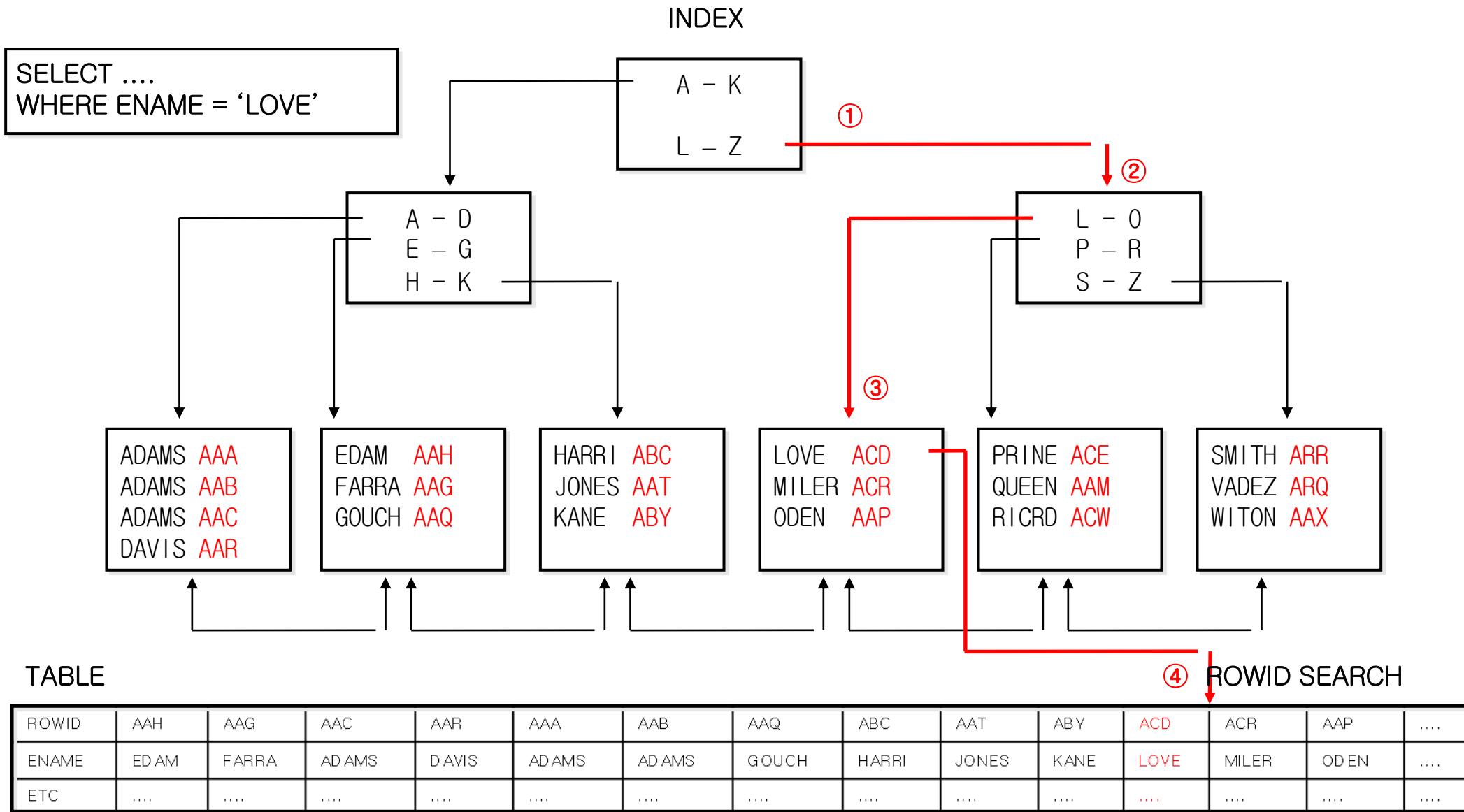
single block i/o



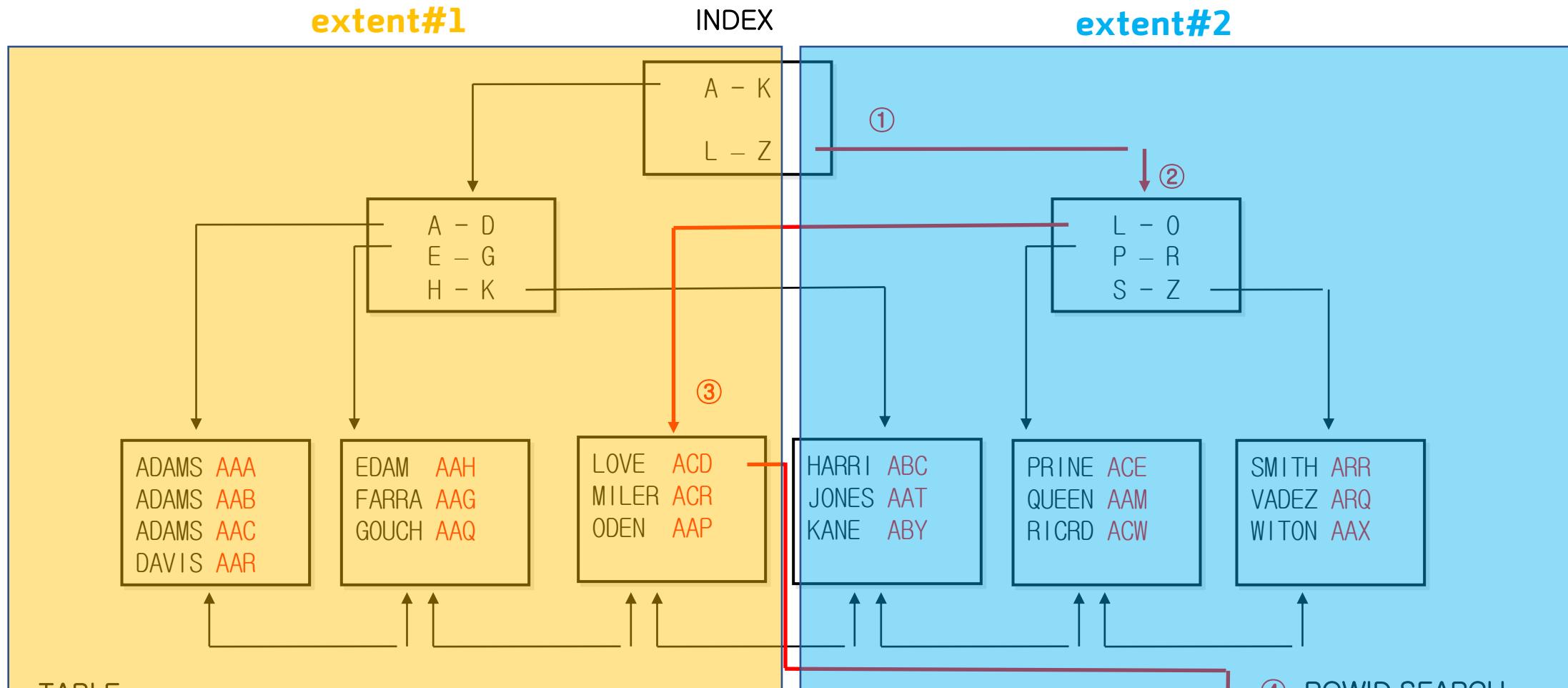
multi block i/o



5. 논리적인 순서에 따라 배치된 index 의 구조



6. 물리적인 순서에 따라 배치된 index 의 구조



```
select /*+
      from emp
      group by deptno;
```

index_fs(emp emp_deptno) **index_ffs(emp emp_deptno)**

index full scan !

결과

DEPTNO	COUNT(*)
10	3
20	5
30	6

```
/*+ deptno, count(*)
```

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);`

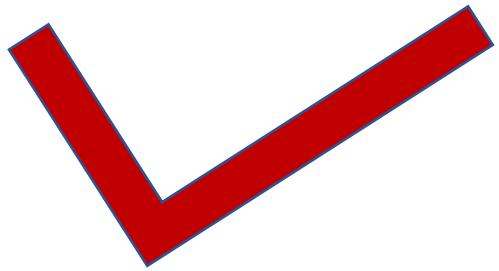
튜닝전:

```
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 투닝할 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
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';

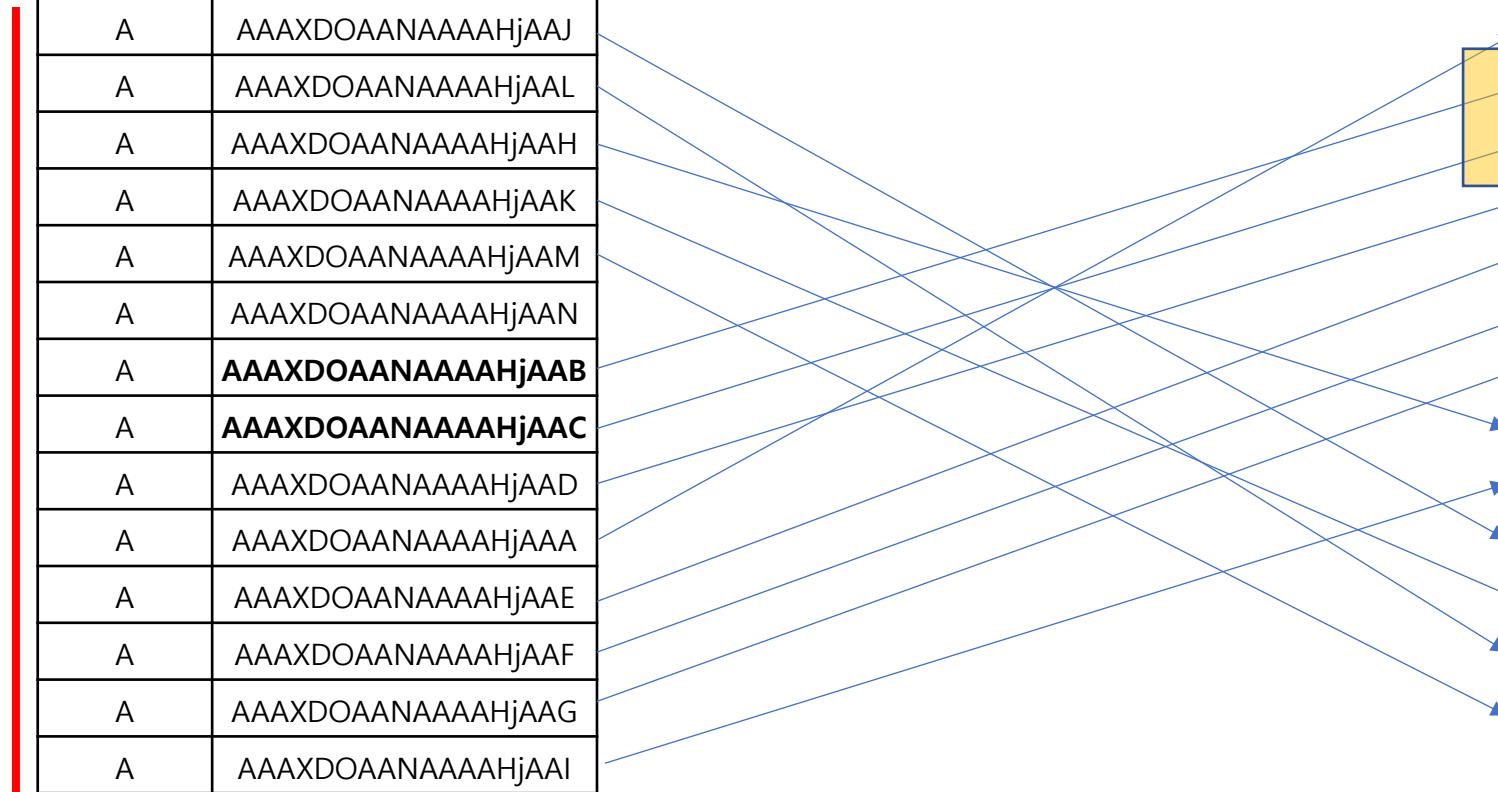
```

emp2_col1 인덱스

COL1	ROWID
A	AAAXDOAANAAAHeAAJ
A	AAAXDOAANAAAHeAAL
A	AAAXDOAANAAAHeAAH
A	AAAXDOAANAAAHeAAK
A	AAAXDOAANAAAHeAAM
A	AAAXDOAANAAAHeAAN
A	AAAXDOAANAAAHeAAB
A	AAAXDOAANAAAHeAAC
A	AAAXDOAANAAAHeAAD
A	AAAXDOAANAAAHeAAJ
A	AAAXDOAANAAAHeAAE
A	AAAXDOAANAAAHeAAF
A	AAAXDOAANAAAHeAAG
A	AAATc1AAHAAAAHeAAH
A	AAATc1AAHAAAAHeAAI
A	AAATc1AAHAAAAHeAAJ
A	AAATc1AAHAAAAHeAAK
A	AAATc1AAHAAAAHeAAL
A	AAATc1AAHAAAAHeAAM
A	AAATc1AAHAAAAHeAAN

emp2 테이블

ROWID	COL1	COL2	...
AAATc1AAHAAAAHeAAH	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	AAAXDOAANAAAHeAAZ
D	AAAXDOAANAAAHeAAW
D	AAAXDOAANAAAHeAAX
D	AAAXDOAANAAAHeAAY
D	AAAXDOAANAAAHeAQ
D	AAAXDOAANAAAHeAAR
D	AAAXDOAANAAAHeAAB
D	AAAXDOAANAAAHeAAC
D	AAAXDOAANAAAHeAAS
D	AAAXDOAANAAAHeAAT
D	AAAXDOAANAAAHeAAU
D	AAAXDOAANAAAHeAAV
D	AAAXDOAANAAAHeAAO
D	AAAXDOAANAAAHeAAP

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 ?

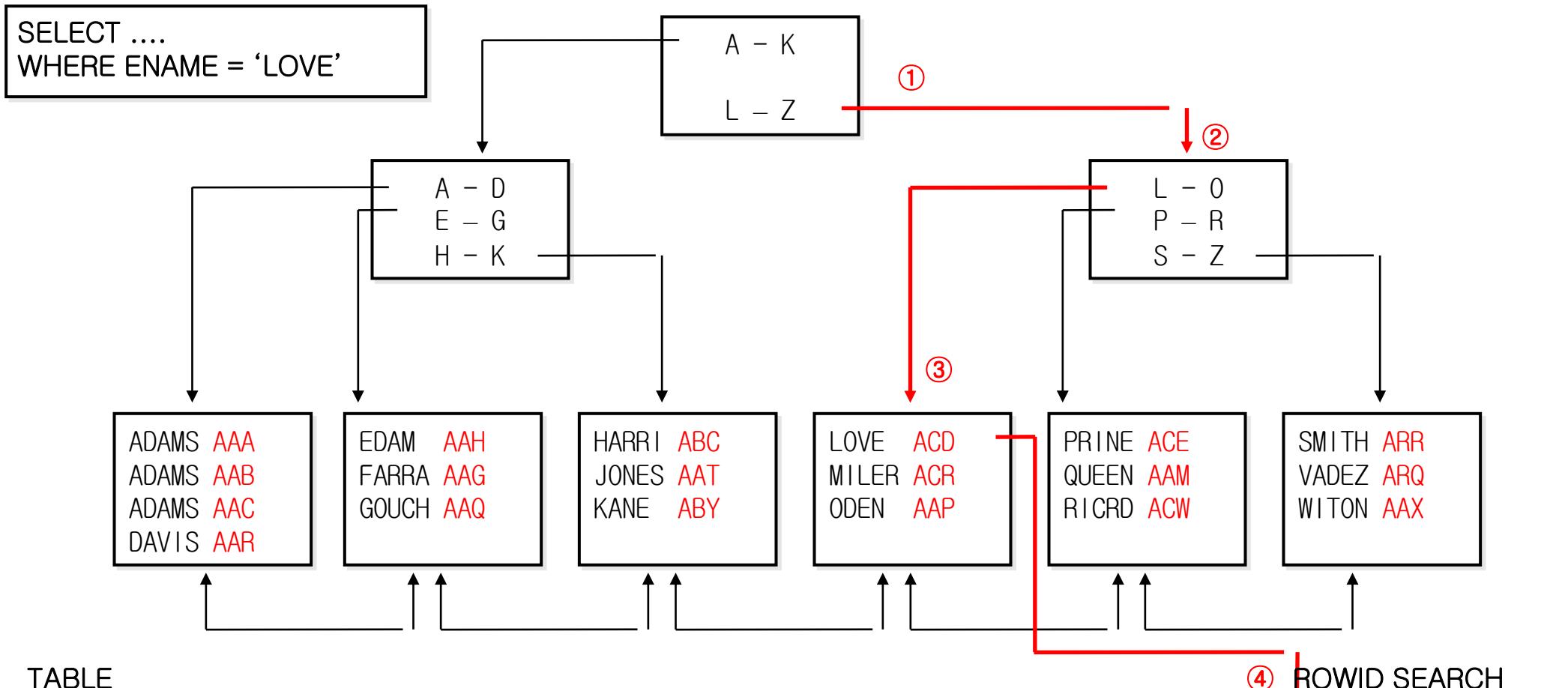
일반 인덱스를 크기가 아주 작은 비트맵 인덱스로 변환하고 비트맵 인덱스들을 하나로 합쳐서 스캔하는 스캔방법

	인덱스 엑세스 방법	관련 힌트
1	index range scan	index
2	index unique scan	index
3	index full scan	index
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

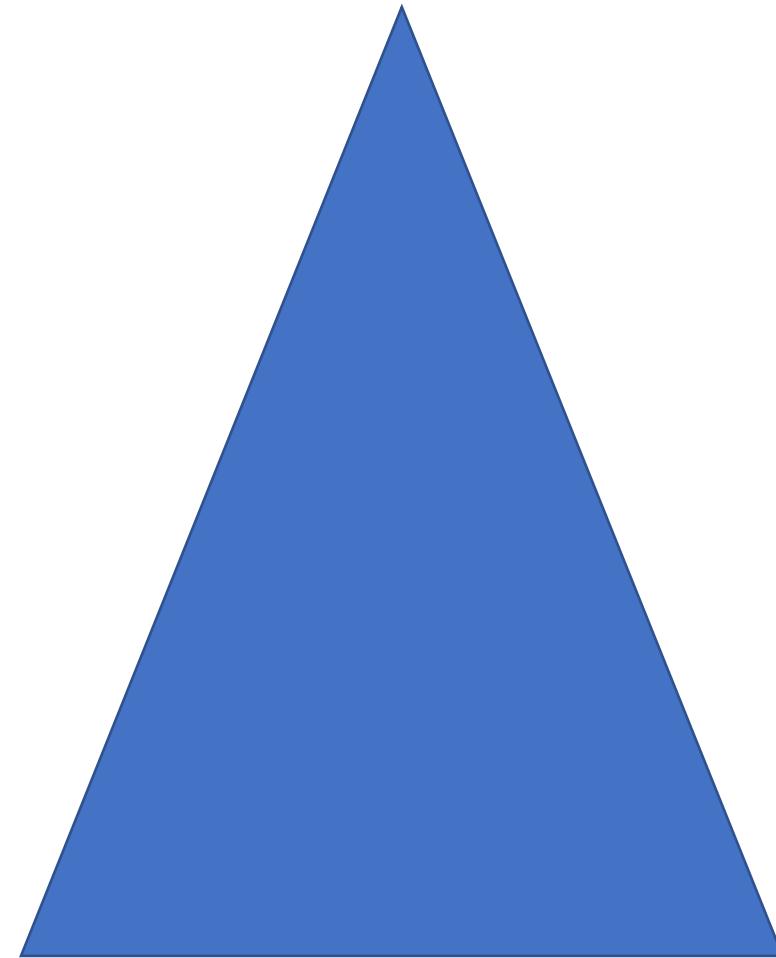


일반적인 tree 구조의 인덱스

INDEX

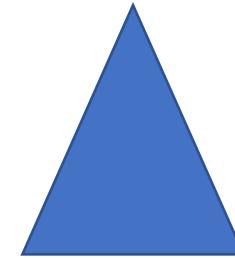


일반 인덱스 → 비트맵 인덱스로 변환하게 되면?



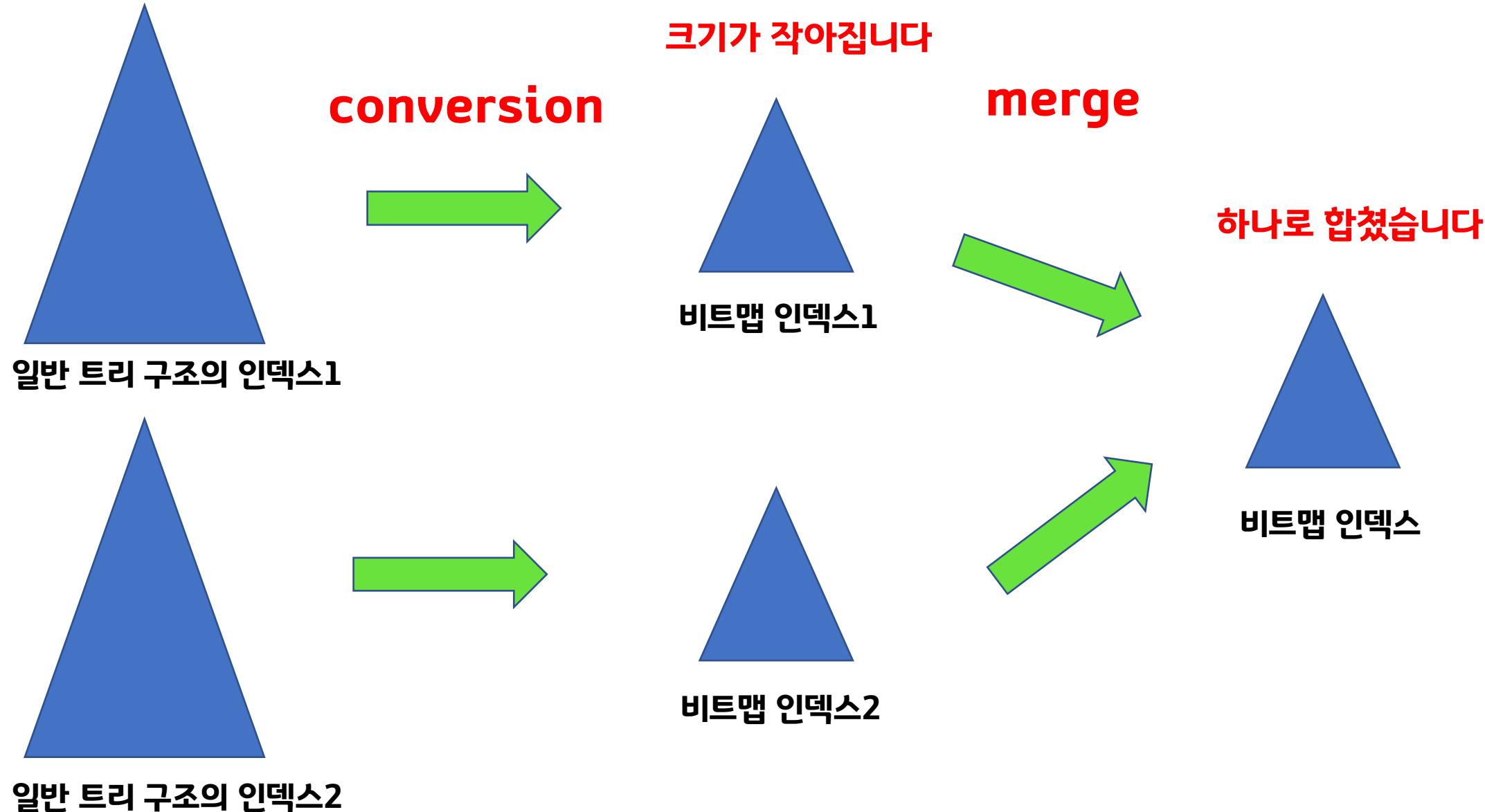
일반 트리 구조의 인덱스

크기가 작아집니다



비트맵 인덱스

비트맵으로 변환하고 합칩니다



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

emp_col1 인덱스

bitmap conversion

emp_col2 인덱스

bitmap conversion

emp2 테이블

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

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

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);
```

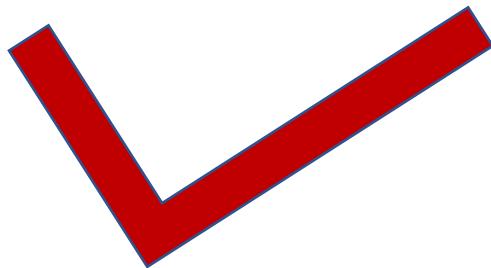
튜닝전:

```
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;
```

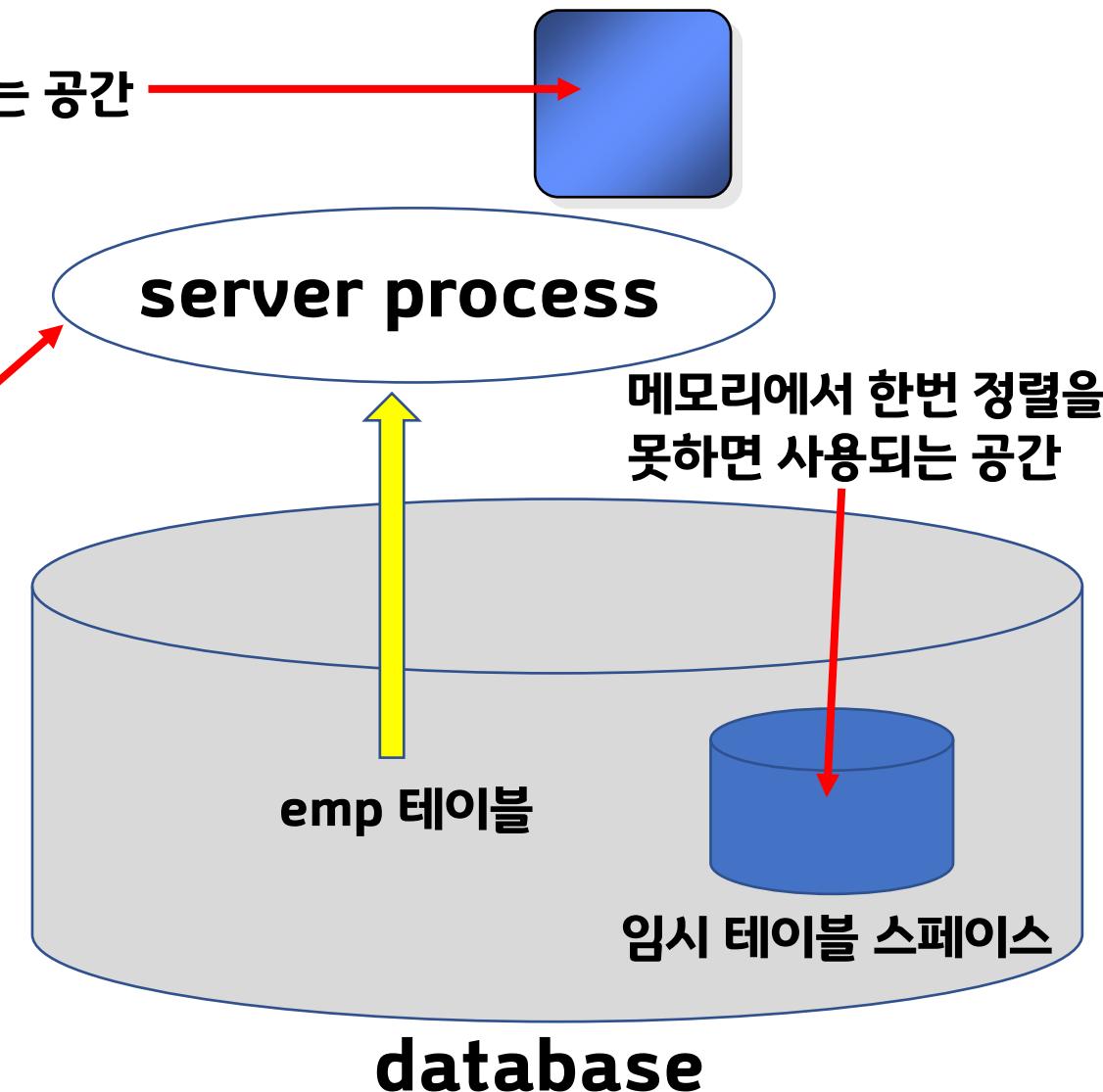
1. 정렬작업이 database 에 왜 부하를 주는지 ?

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

위의 SQL 을 수행하는 프로세서

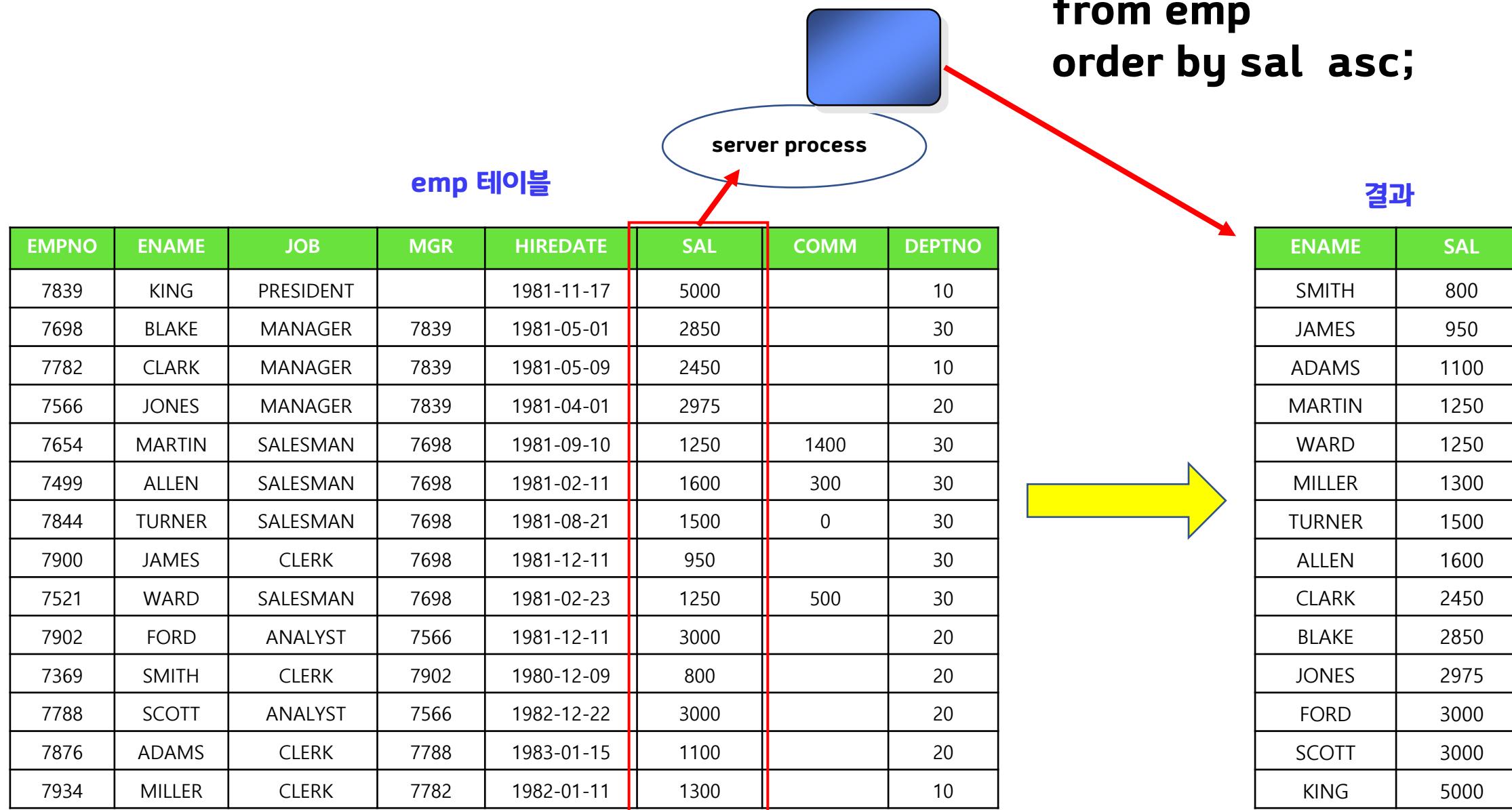
정렬작업을 위해서 사용되는 공간

개별 메모리 공간



인덱스를 활용하지 못한다면?

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



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.

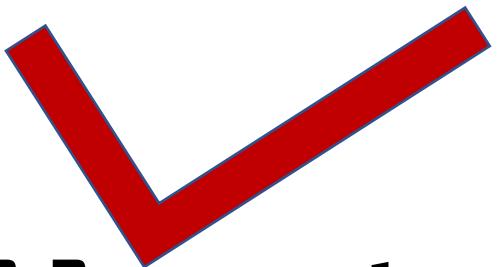
튜닝전:

```
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



방법 11. 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

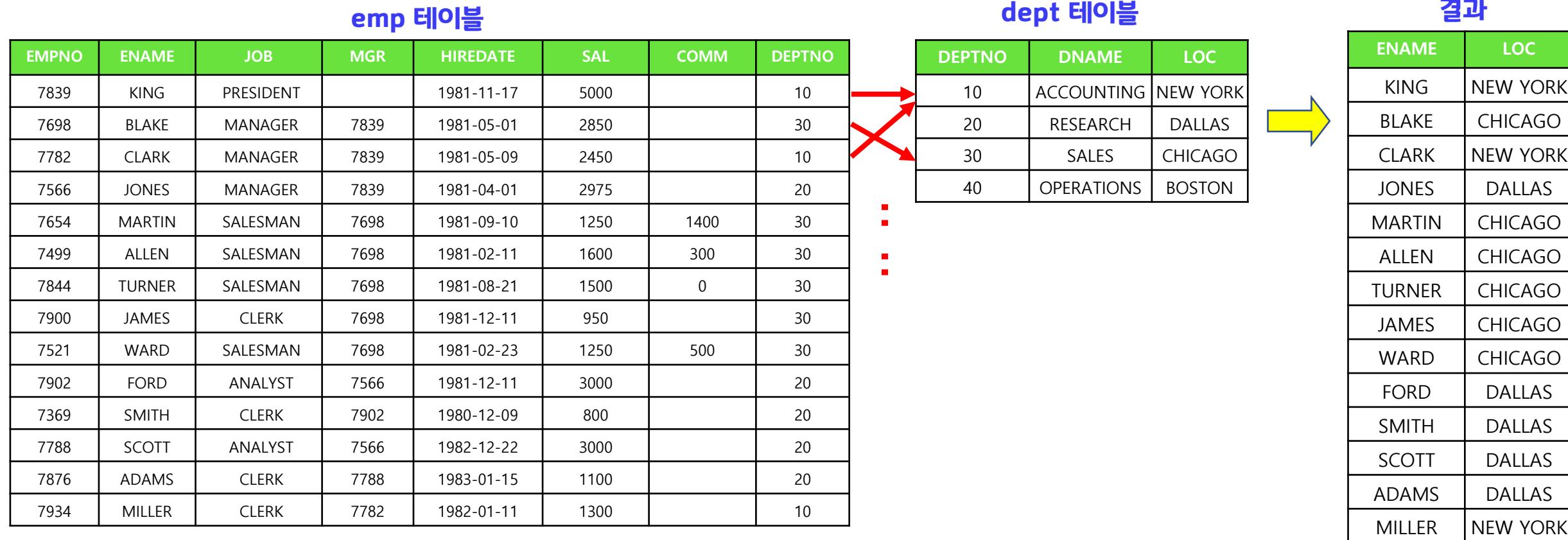
조인 방법

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 엑세스하면서
그 추출된 테이블로 연결할 테이블을 조인하는 방식



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;
```



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

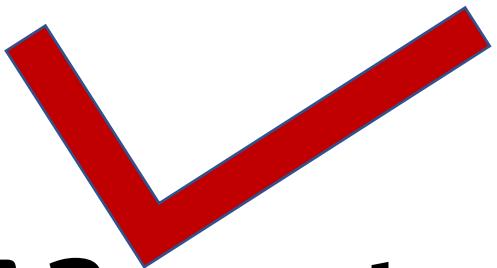
튜닝전:

```
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 수행 테이블의 처리 범위를 하나씩 엑세스하면서 그 추출된 테이블로 연결할 테이블을 조인하는 방식입니다.
2. 테이블 조인 순서는 조인 되는 데이터를 적게 할 수 있는 테이블을 수행 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	LOSL	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	LOSQL	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	LOSL	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	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	LOSQL	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	LOSL	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	LOSL	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

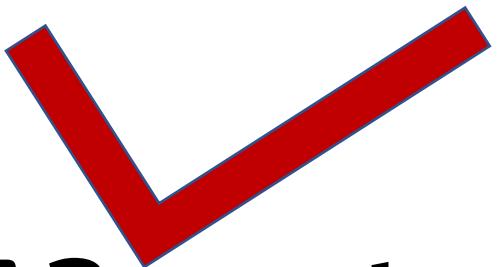
튜닝전:

```
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 수행 테이블의 처리 범위를 하나씩 엑세스하면서 그 추출된 테이블로 연결할 테이블을 조인하는 방식입니다.
2. 여러개의 테이블 조인 순서도 조인 되는 데이터를 적게 할 수 있는 테이블을 수행 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 없었을 때

```
select /*+ leading(d e) use_nl(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
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

full table scan !

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 있었을 때

```
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 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

index range scan !

emp_deptno 인덱스

ROWID	DEPTNO
xzDAANAAAC2eAAA	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 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_deptno 인덱스

DEPTNO	ROWID
10	AAAXzCAANAA ACz+AAA
20	AAAXzCAANAA ACz+AAB
30	AAAXzCAANAA ACz+AAC
40	AAAXzCAANAA ACz+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';
```

emp_ename인덱스

ENAME	ROWID
ADAMS	AAATifAAHAAA DL1AAM
ALLEN	AAATifAAHAAA DL1AAF
BLAKE	AAATifAAHAAA DL1AAB
CLARK	AAATifAAHAAA DL1AAC
FORD	AAATifAAHAAA DL1AJ
JAMES	AAATifAAHAAA DL1AH
JONES	AAATifAAHAAA DL1AAD
KING	AAATifAAHAAA DL1AAA
MARTIN	AAATifAAHAAA DL1AAE
MILLER	AAATifAAHAAA DL1AAN
SCOTT	AAATifAAHAAA DL1AAL
SMITH	AAATifAAHAAA DL1AAK
TURNER	AAATifAAHAAA DL1AAG
WARD	AAATifAAHAAA DL1AAI

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

index 생성

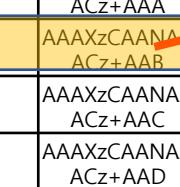
dept_deptno인덱스

DEPTNO	ROWID
10	AAAXzCAANAA ACz+AAA
20	AAAXzCAANAA ACz+AAB
30	AAAXzCAANAA ACz+AAC
40	AAAXzCAANAA ACz+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	AAAXzDAANAAAC2eAAI
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	AAAXzDAANAAAC2eAAK
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_deptno 인덱스

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
emp 테이블

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

emp_deptno 인덱스

ROWID	DEPTNO
AAAXzDAANAAAeAAA	10
AAAXzDAANAAAeAAC	10
AAAXzDAANAAAeAAN	10
AAAXzDAANAAAeAAD	20
AAAXzDAANAAAeAAJ	20
AAAXzDAANAAAeAAK	20
AAAXzDAANAAAeAAL	20
AAAXzDAANAAAeAAM	20
AAAXzDAANAAAeAAB	30
AAAXzDAANAAAeAAE	30
AAAXzDAANAAAeAAF	30
AAAXzDAANAAAeAAG	30
AAAXzDAANAAAeAAH	30
AAAXzDAANAAAeAAI	30

index 생성
dept 테이블

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

dept_loc 인덱스

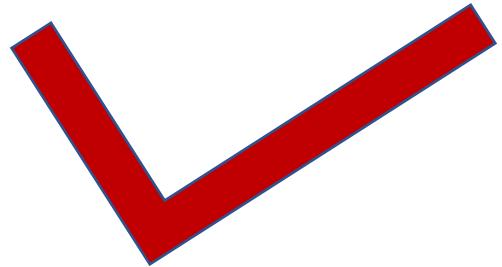
ROWID	LOC
AAAXzDAANAAAeAAB	BOSTON
AAAXzDAANAAAeAAE	CHICAGO
AAAXzDAANAAAeAAC	DALLAS
AAAXzDAANAAAeAAK	NEW YORK

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 ;
```

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

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



dept 테이블

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

2. 어느 테이블을 해쉬 테이블로 구성해야 할까?

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

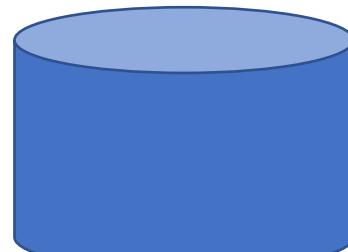
오라클 메모리 영역 PGA

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



PROB 테이블
dept 테이블



temp disk

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 ;
```

오라클 메모리 영역 PGA

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

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';
```

오라클 메모리 영역 PGA

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

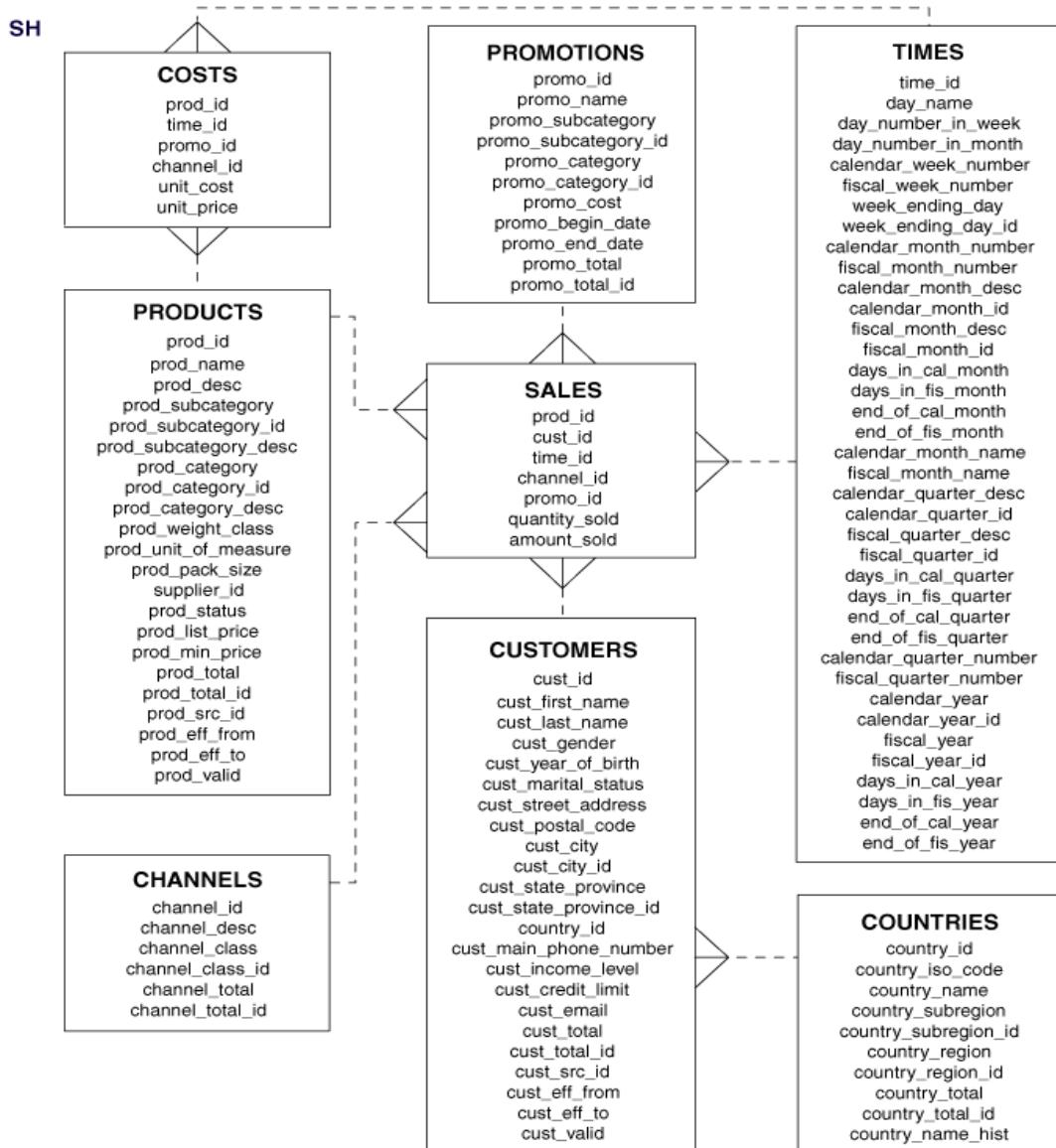
해쉬 테이블			
해쉬값	DEPTNO	DNAME	LOC
dkfle21jfn..	30	SALES	CHICAGO



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 ?

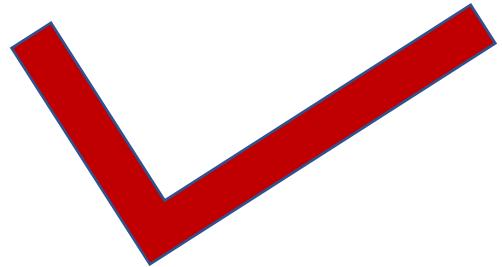
조금 더 큰 테이블이 필요합니다.

<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

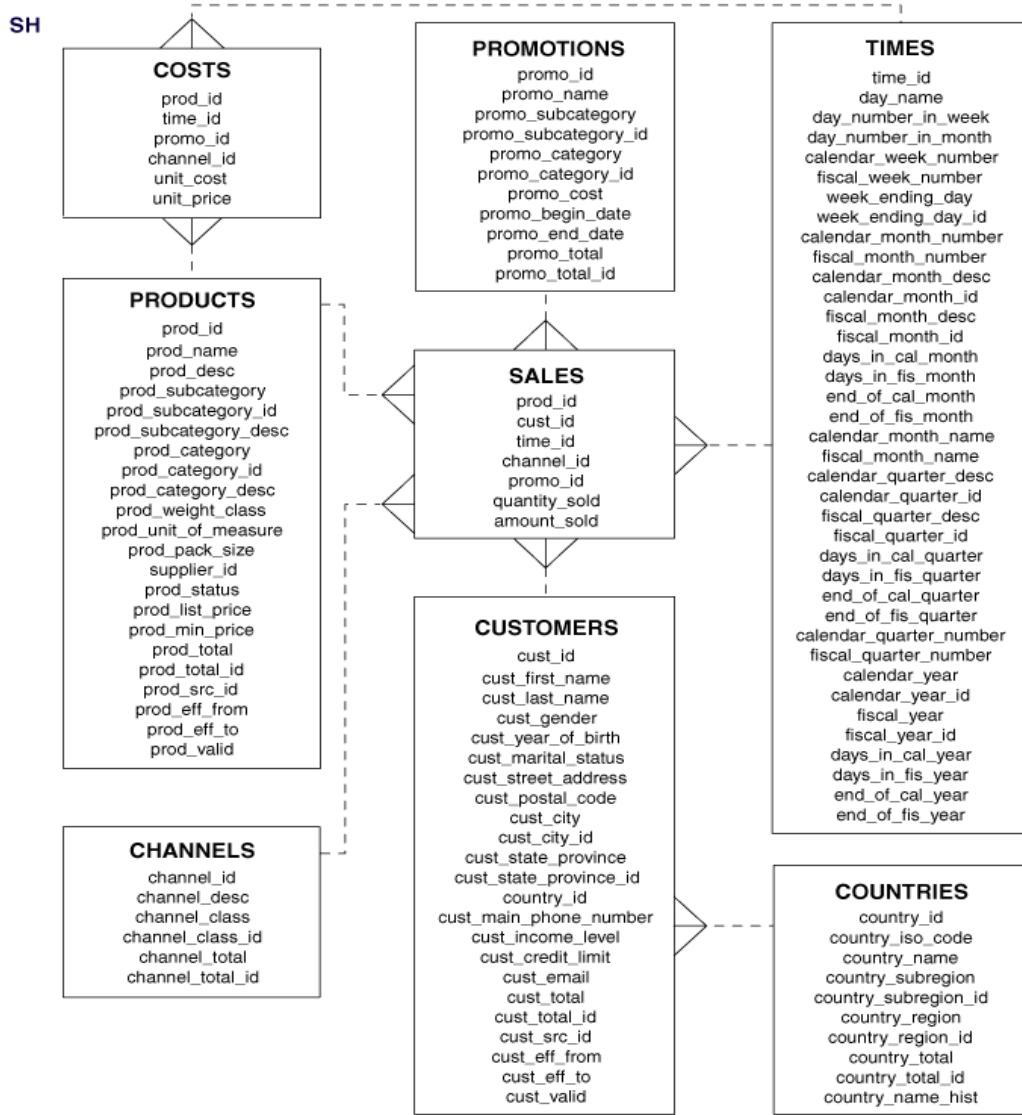
▣ 학습 내용

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.

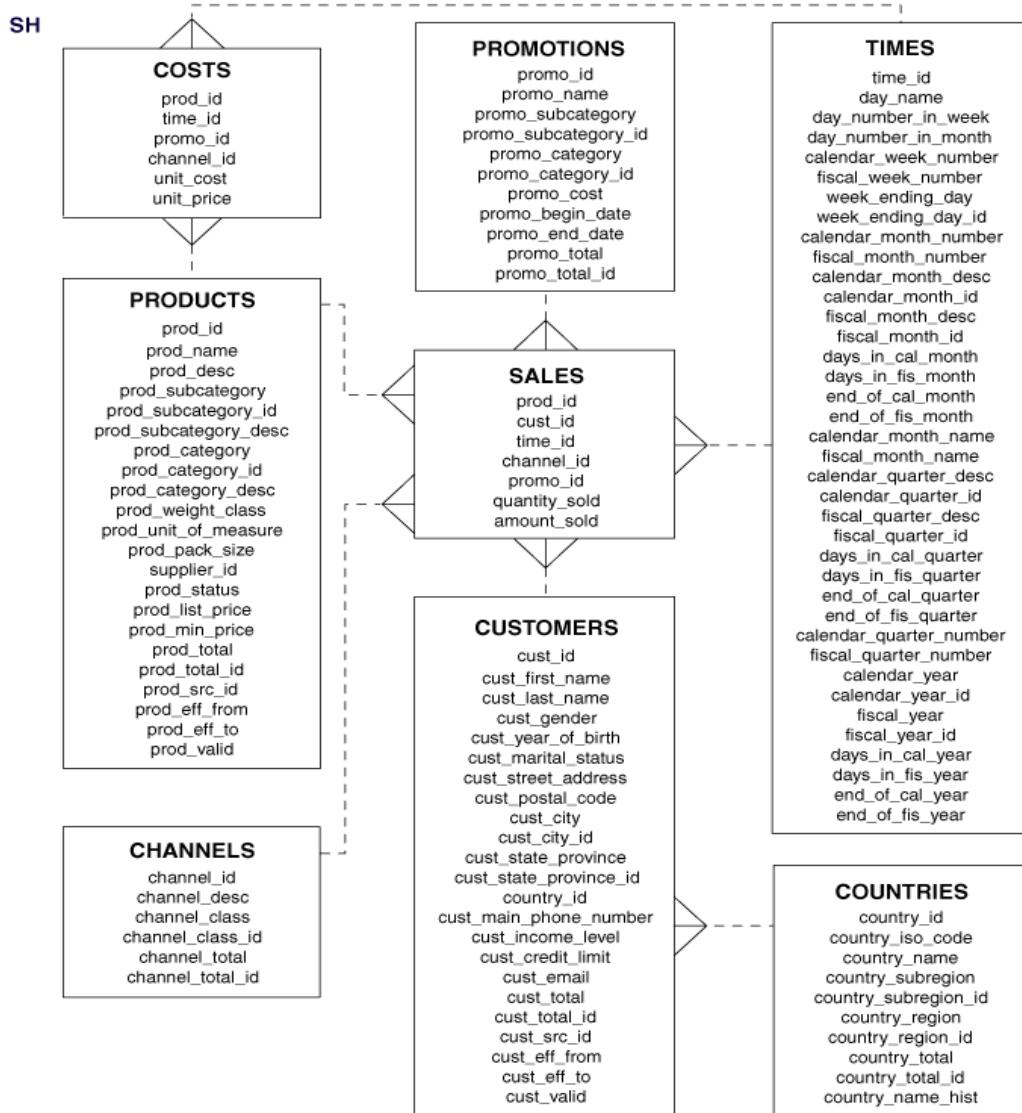
1. 3개 이상의 테이블을 해쉬조인 할때 조인순서는?



```
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       | 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?

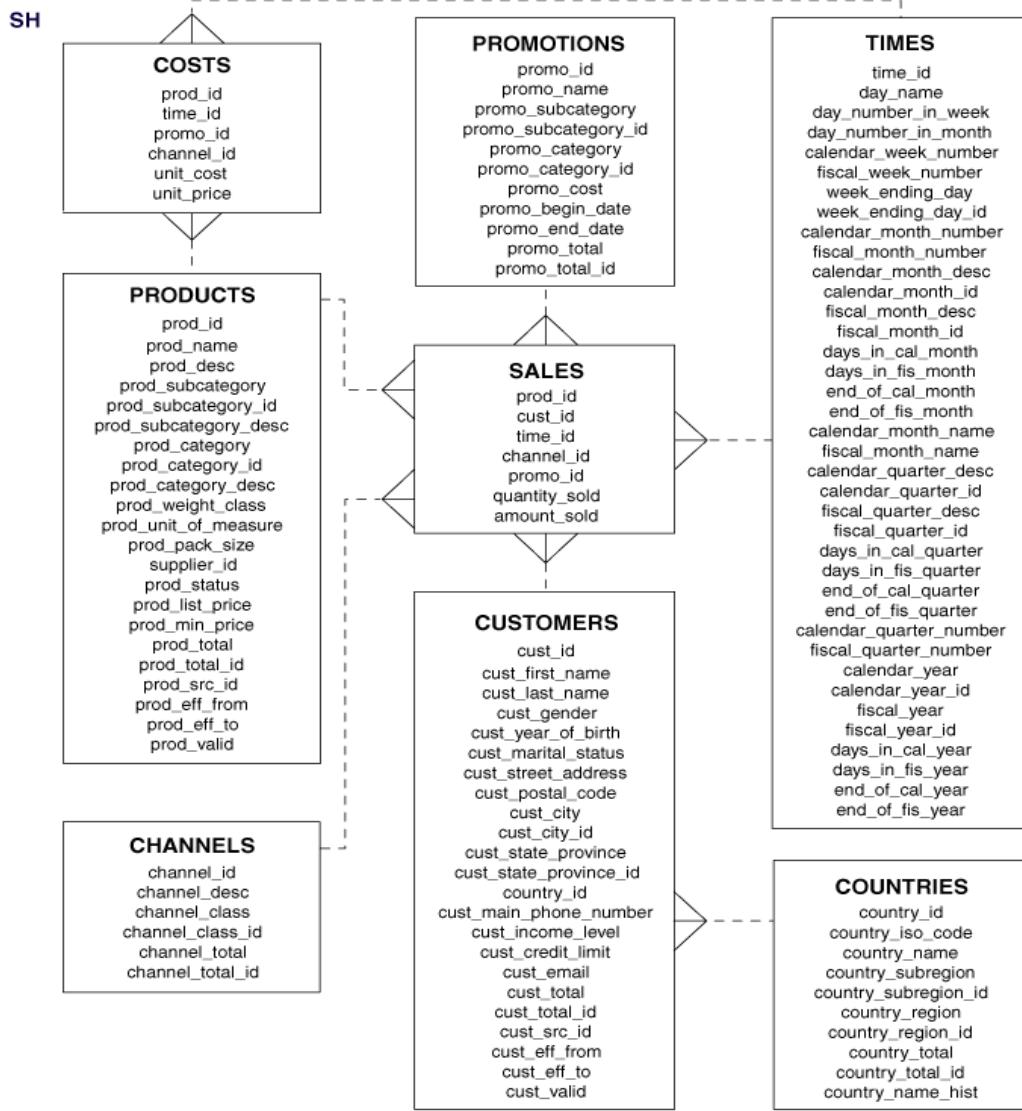
```
select /*+ leading(t s p) use_hash(s) use_hash(p) swap_join_inputs(p) */  
    p.prod_name, t.CALENDAR_YEAR, sum(s.amount_sold)  
  from sales100 s, times100 t, products100 p  
 where s.time_id = t.time_id  
   and s.prod_id = p.prod_id  
 group by p.prod_name, t.calendar_year;
```

11	12	13	Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	1Mem	Used-Mem
15	16	17	0	SELECT STATEMENT		1		2	00:00:00.07	4499			
16	17	18	1	HASH GROUP BY		1	2	2	00:00:00.07	4499	1223K	1223K	639K (0)
17	18	19	2	HASH JOIN		1	3195	6669	00:00:00.13	4499	1538K	1538K	606K (0)
18	19	20	3	TABLE ACCESS FULL	PRODUCTS100	1	1	1	00:00:00.01	3			
19	20	21	4	HASH JOIN		1	460K	492K	00:00:00.06	4495	1744K	1744K	1542K (0)
20	21	22	5	TABLE ACCESS FULL	TIMES100	1	731	731	00:00:00.01	54			
21	22	23	6	TABLE ACCESS FULL	SALES100	1	918K	918K	00:00:00.03	4440			

24 Predicate Information (identified by operation id):

```
25  
26  
27 2 - access("S"."PROD ID"="P"."PROD ID")  
28 3 - filter("P"."PROD NAME" LIKE 'Deluxe%')  
29 4 - access("S"."TIME ID"="T"."TIME ID")  
30 5 - filter(("T"."CALENDAR YEAR"=2000 OR "T"."CALENDAR YEAR"=2001))
```

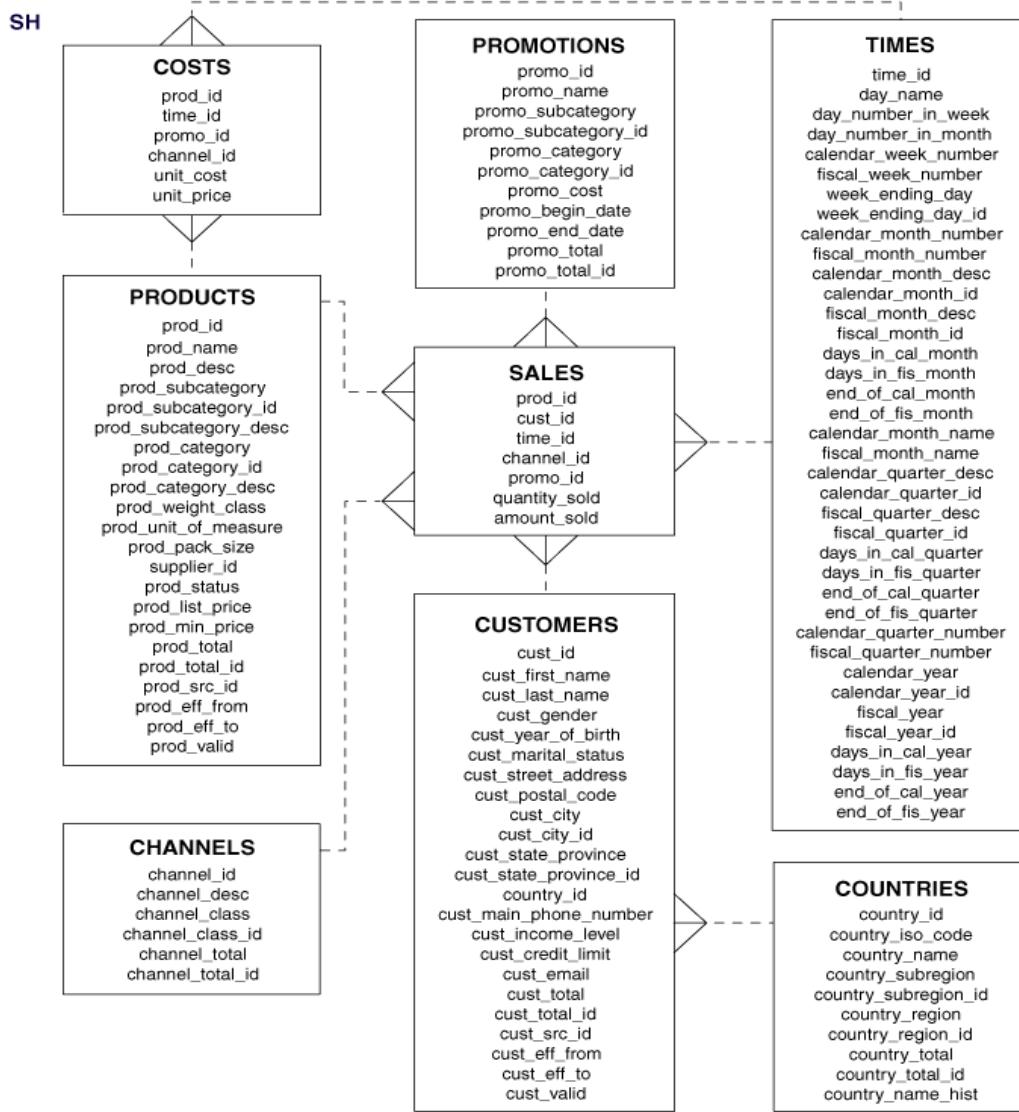
5. 검색조건이 있었을때의 해쉬조인 순서는?



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

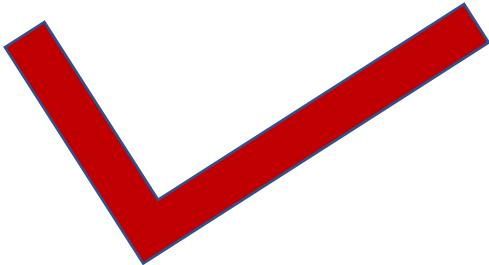
② select count(*) from sales100; 918843건

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_inputs` 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	LOSQL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

2. sort merge 조인의 원리

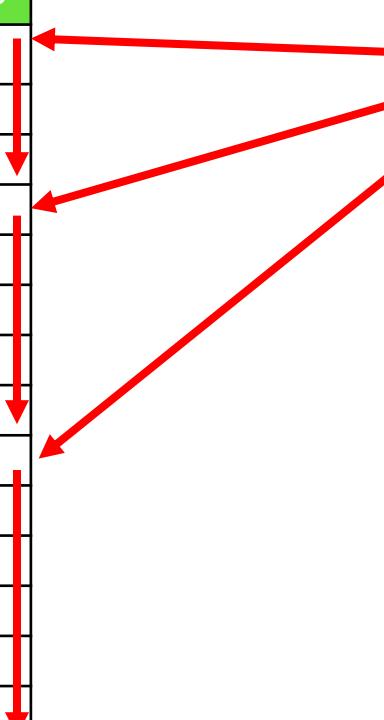
```
select /*+ leading(d e) use_merge(e) */ e.ename, d.loc  
from emp e, dept d  
where e.deptno = d.deptno;
```

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?

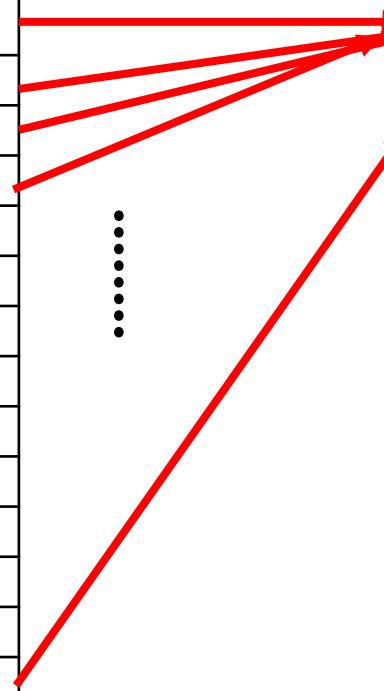
```
select /*+ leading(e d) use_merge(d) */ e.ename, d.loc  
from emp e, dept d  
where e.deptno = d.deptno;
```

emp 테이블

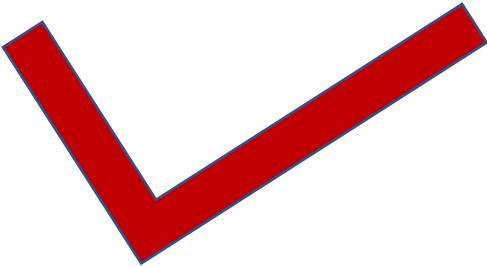
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		1981-11-17	5000		10
7782	CLARK	MANAGER	7839	1981-05-09	2450		10
7934	MILLER	CLERK	7782	1982-01-11	1300		10
7876	ADAMS	CLERK	7788	1983-01-15	1100		20
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20
7369	SMITH	CLERK	7902	1980-12-09	800		20
7902	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 ←

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

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

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 ←

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

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

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 ;
```

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

해쉬 함수

오라클 메모리 영역 PGA

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

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 (+);
```

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

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



dept 테이블

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

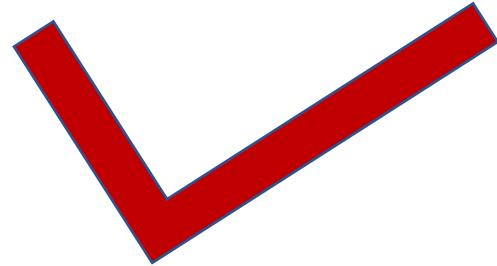
4. 아우터 조인순서와 관계없이 해쉬 테이블을 정할 수 있는가?

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

```
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 (+) ;
```

10	Id	Operation		Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	1Mem	Used-Mem
12	0	SELECT STATEMENT			1		14	00:00:00.01	12			
13	*	HASH JOIN RIGHT OUTER			1	14	14	00:00:00.01	12	1797K	1797K	975K (0)
14	2	TABLE ACCESS FULL	DEPT		1	4	4	00:00:00.01	6			
15	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 ← main query  
      from emp 14건  
      where deptno in ( select deptno ← sub query  
                           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

2. 서브쿼리문의 실행순서 두번째

메인쿼리 부터 실행되게 되면?

```
select ename, sal ← main query  
      from emp 14건  
      where deptno in ( select deptno ← sub query  
                           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

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 !**

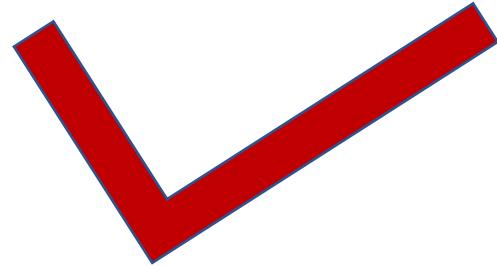
② **unnest** 뜻? **감싸지 말아라 !**

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 학습합니다.
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 많을 때

서브쿼리 부터 실행되게 되면?

```
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 );
```

main query

sub query

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	00:00:16.03	2013K
* 2 TABLE ACCESS FULL	SALES100	497	2	00:00:18.00	2013K

2. 서브쿼리의 데이터가 많을 때

메인쿼리 부터 실행되게 되면?

```
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 );
```

main query

sub query

	Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
3	0	SELECT STATEMENT		1		50	00:00:05.98	2013K
4	1	FILTER		1		50	00:00:05.98	2013K
5	2	TABLE ACCESS FULL	CUSTOMERS100	1	55500	497	00:00:00.01	17
6	3	TABLE ACCESS FULL	SALES100	497	2	50	00:00:18.09	2013K

3. 서브쿼리의 데이터가 많을 때

해쉬 세미조인으로 실행되게 되면?

```
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 );
```

main query

sub query

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

4. 서브쿼리의 데이터가 많을 때

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	0Mem	1Mem	Used-Mem
0	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 앤티 조인이 되게하라 !

□ 학습 내용

1. 순수하게 서브쿼리로 실행되었을때와 세미조인으로 수행되었을때의 차이를 학습합니다.
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 많을 때

서브쿼리 부터 실행되게 되면?

```
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 );
```

main query

sub query

U	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	13

2. 서브쿼리의 데이터가 많을 때

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
                         where amount_sold between 0 and 10000 );
```

918843건

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	55500	57	00:00:00.01	5
*	3	TABLE ACCESS FULL	SALES100	57	2	00:00:02.08	226K

3. 서브쿼리의 데이터가 많을 때

해쉬 안티조인으로 실행되게 되면?

main query
sub query

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

9	0											
1	Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	Reads	0Mem	1Mem	Used-Mem
3	0	SELECT STATEMENT		1	50	00:00:00.41	5960	1511				
4	*	1	HASH JOIN ANTI	1	48441	50	00:00:00.41	5960	1511	14M	2068K	15M (0)
5	*	2	TABLE ACCESS FULL	CUSTOMERS100	55500	55500	00:00:00.01	1520	1511			
6	*	3	TABLE ACCESS FULL	SALES100	918K	918K	00:00:00.03	4440	0			

4. 서브쿼리의 데이터가 많을 때

해쉬 right 안티조인으로 실행되게 되면?

main query

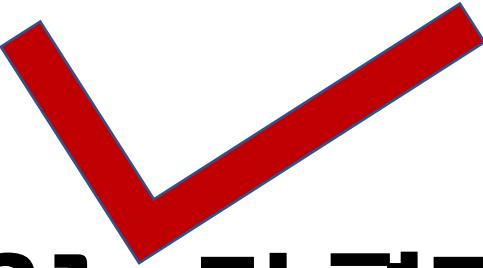
sub query

```
select /*+ gather_plan_statistics */ *
from customers100 c  55500건
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 );
```

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

끝까지 잘 공부 하셨습니다. 감사합니다.

이럴때는



방법21. 더 편리한 도구를 이용해서 튜닝하세요

□ 학습 내용

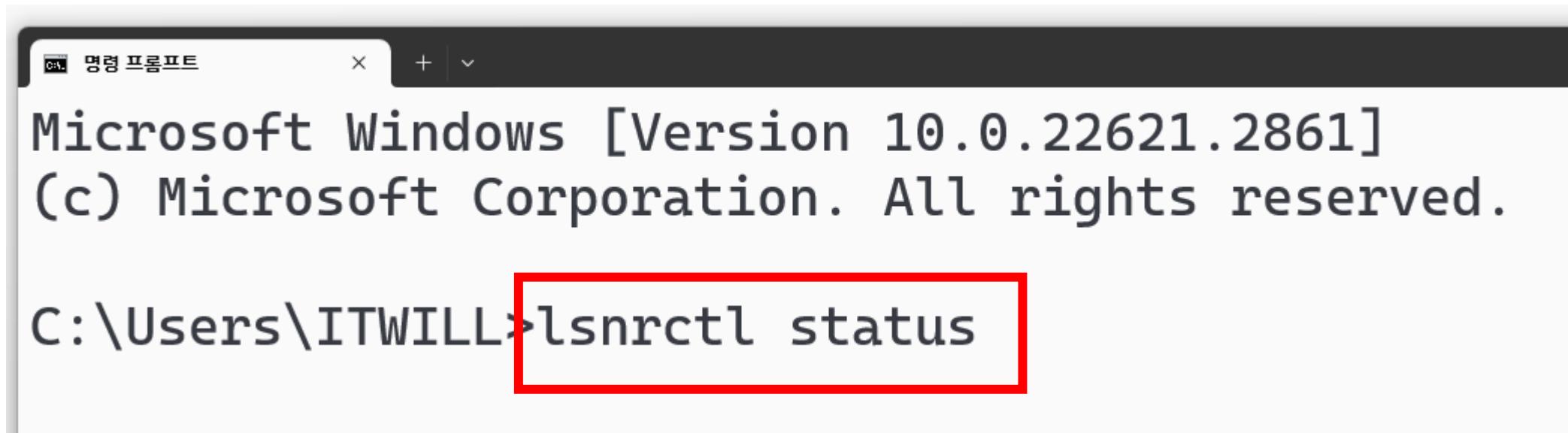
- 1. dbever 소프트 웨어 설치전에 확인해야할 사항을 확인합니다.**
- 2. dbever 를 설치 합니다.**
- 3. dbever 에서 예상 실행계획 보는 방법과 실제 실행계획 보는 방법을 배웁니다.**

□ 학습 목표

dbever 를 이용하여 SQL 튜닝을 빠르고 편하게 할 수 있습니다.

dbever 설치전에 반드시 확인해야할 사항

리스너의 상태가 정상인지 확인합니다 !



```
명령 프롬프트
x + v
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ITWILL>lsnrctl status
```

host 와 port 와 서비스 이름을 확인합니다

호스트 이름과 포트와 서비스 이름으로 오라클에 접속

```
명령 프롬프트 - sqlplus c##scott/tiger@localhost:1521/xe

C:\Users\ssuser>
C:\Users\ssuser>sqlplus c##scott/tiger@localhost:1521/xe
SQL*Plus: Release 21.0.0.0.0 - Production on 금 1월 12 12:58:18 2024
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

마지막 성공한 로그인 시간: 목 1월 11 2024 19:13:30 +09:00

다음에 접속됨:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL>
SQL>
SQL>
```

dbever 설치

A screenshot of a Google search results page. The search bar at the top contains the query "db 에버설치". A red box highlights this search term. Below the search bar, the first result is for "DBeaver Community" from the URL <https://dbeaver.io>. The result includes a small icon of a beaver, the text "Advanced security: master password and strong credentials encryption to secure and easy database connection. ... Install: Windows installer – run installer ...", and links for "Archive Files · News · Of /update/latest · Early Access". A red arrow points to the "Download" link.

Google db 에버설치

DBeaver Community
<https://dbeaver.io> › download

→ Download

Advanced security: master password and strong credentials encryption to secure and easy database connection. ... Install: Windows installer – run installer ...

Archive Files · News · Of /update/latest · Early Access

dbever 설치

Download

DBeaver Community 23.3.2

Released on january 8th 2024 ([Milestones](#)).

It is free and open source ([license](#)).

Also you can get it from the [GitHub mirror](#).



- [Windows \(installer\)](#)
- [Windows \(zip\)](#)

- [Chocolatey](#) (choco install dbeaver)
- [Install from Microsoft Store](#)

DBeaver PRO 23.3

Released on December 11th, 2023

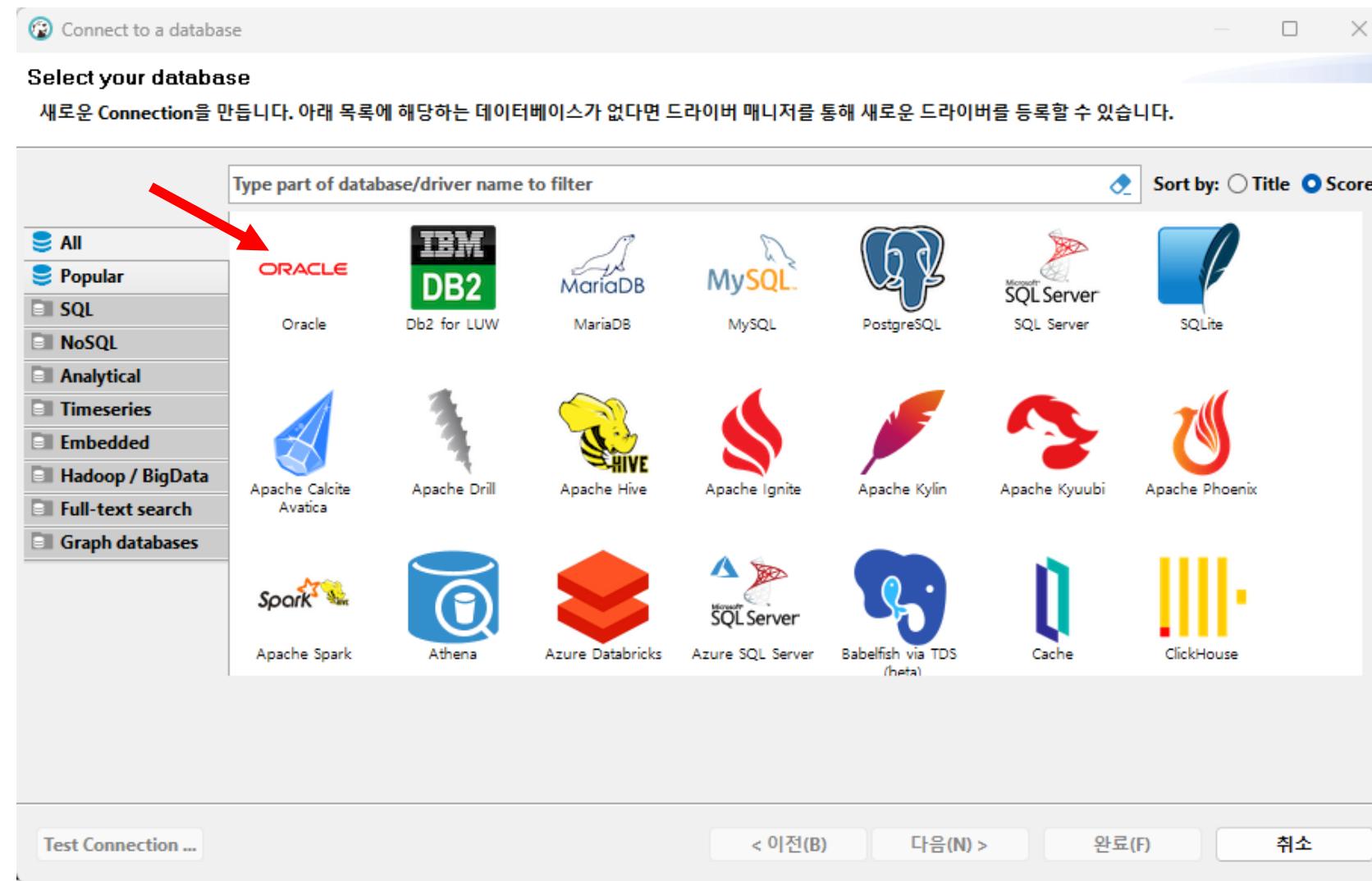
PRO version website: [dbeaver.com](#)

[Trial](#) version is available.

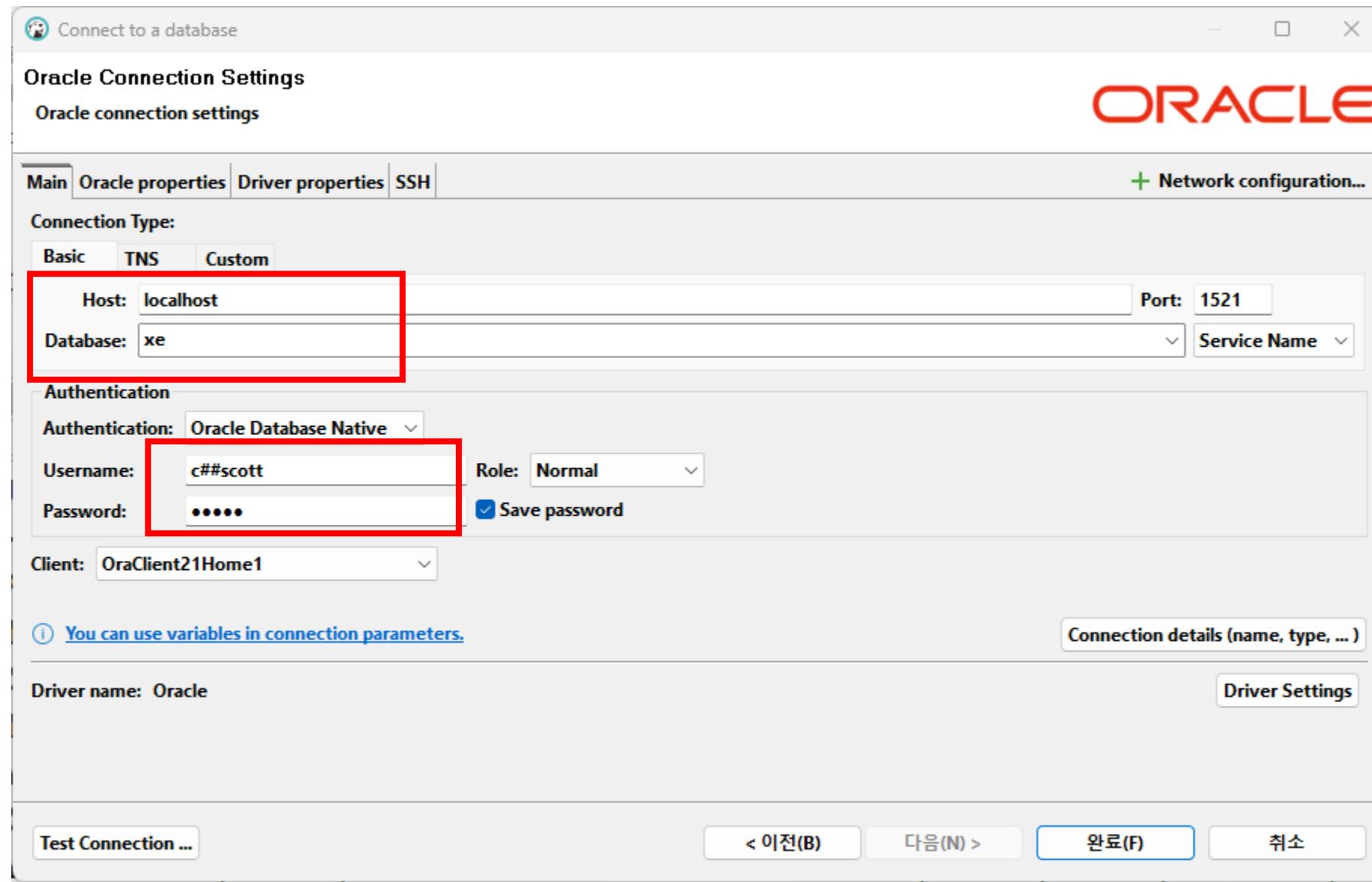
DBeaver PRO features:

- **Advanced security:** master password and strong credentials encryption to secure and easy database connection.
- **Enterprise-level authentication methods:** SAML, SSO, OKTA, and Kerberos. SSO authentication cloud services such as GCP, AWS, and Azure.

dbever 설치후 접속하기



dbever 설치후 접속하기



폰트 변경

DBeaver 23.3.0 - <ORCL 2> Script

파일(F) 편집(E) 탐색(N) 검색(A) SQL 편집기 데이터베이스(D) 원도우(W) 도움말(H)

Connections <ORCL 2>

Enter a part of object name

ORCL 2 - localhost:1522

- Schemas
- Global metadata
- Storage
- Security
- Administer

-- 예제

-- 예제1.

explain
select

f
w 설정 300;

select * from table(dbms_xplan.display);

-- 예제2. 실제 실행계획 확인하는 방법

select /*+ gather_plan_statistics */ ename, sal

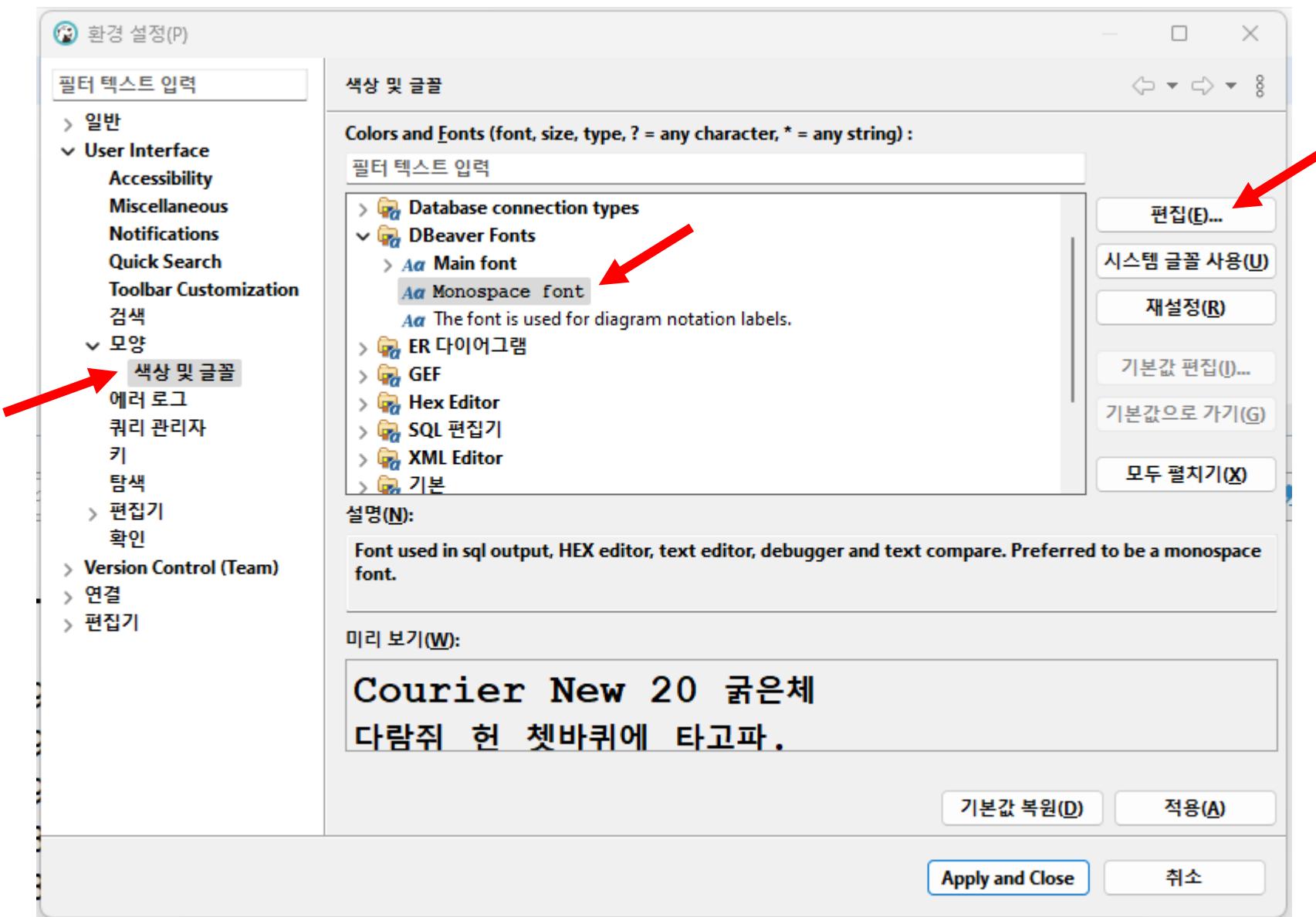
새 창에서 열기(N)
편집기
모양
Database Navigator
프로젝트
프로젝트 탐색기
Database Tasks
Show View
퍼스펙티브 화면구성
퍼스펙티브 초기화

SCOTT

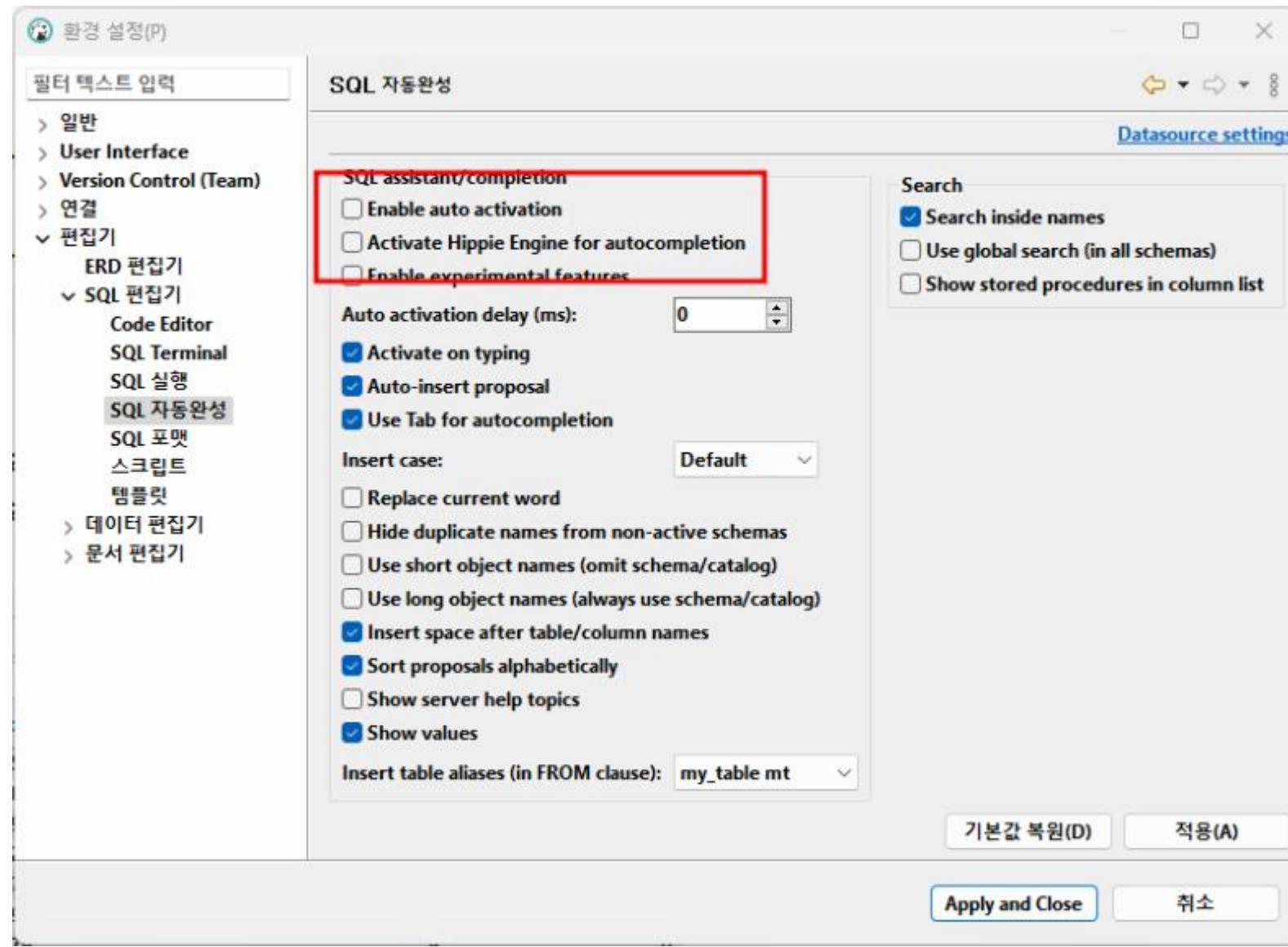
확인하자!
하는 방법

The screenshot shows the DBeaver interface with a context menu open over a SQL query. The 'Settings' option in the menu is highlighted with a red rectangle. The 'Settings' dialog is displayed, showing the 'Font' tab with a dropdown menu set to '설정' (Custom) and a value of '300'. The background shows parts of the SQL editor and a database browser.

폰트 변경

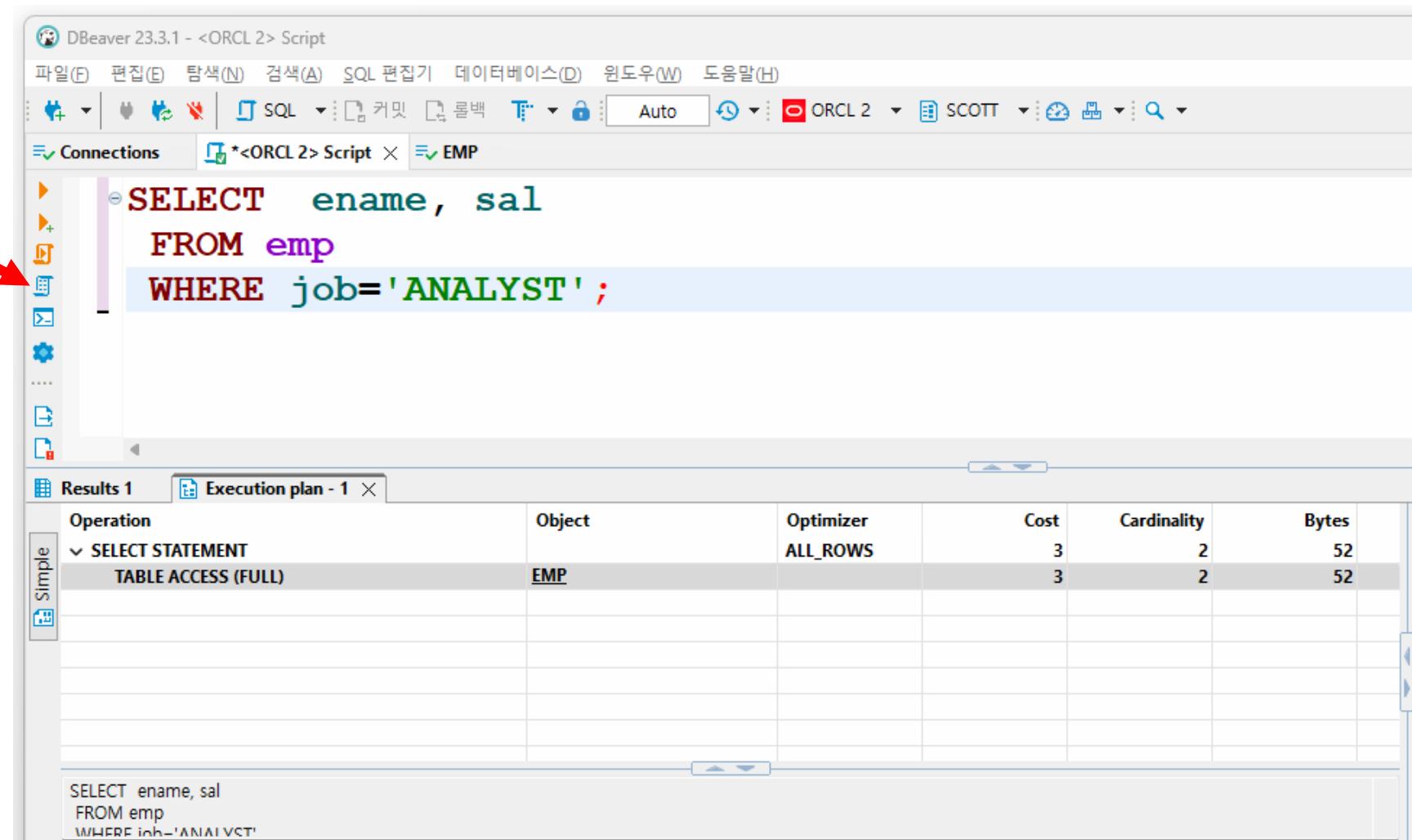


SQL 자동 완성 기능 끄기



dbever에서 예상 실행계획 보기

실행계획 보기 버튼



dbever에서 실제 실행계획 보기

The screenshot shows the DBeaver interface with the following details:

- Toolbar:** 파일(F), 편집(E), 탐색(N), 검색(A), SQL 편집기, 데이터베이스(D), 원도우(W), 도움말(H).
- Connections:** ORCL 2, SCOTT.
- Script Tab:** SELECT ename, sal FROM emp WHERE job='ANALYST';
- SQL Tab:** select sql_id, child number, sql_text from v\$sql where sql_text like '%emp%' AND sql_text NOT LIKE '%v\$sql%' ORDER BY last load time DESC;
- Results 1 Tab:** Results table with columns: SQL_ID, CHILD_NUMBER, SQL_TEXT. The table shows several rows of SQL statements, with the first row being the original query and subsequent rows being parts of the execution plan.
- Bottom Status Bar:** 7 row(s) fetched - 12ms, on 2024-01-11 at 09:14:13. KST | ko | 쓰기 가능 | 스마트 삽입 | 2 : 1 : 2 | Sel: 0 | 0.

v\$sql로
방금 수행한 SQL
을 확인합니다

dbever에서 실제 실행계획 보기

DBeaver 23.1 - <ORCL 2> Script

파일(F) 편집(E) 탐색(N) 검색(A) SQL 편집기 데이터베이스(D) 윈도우(W) 도움말(H)

Connections *<ORCL 2> Script SCOTT

```
SELECT ename, sal
  FROM emp
 WHERE job='ANALYST';

select sql_id, child_number, sql_text
from v$sql
where sql_text like '%emp%' AND sql_text NOT LIKE '%v$sql%'
ORDER BY last_load_time DESC;

SELECT * FROM TABLE(dbms_xplan.display_cursor('6x1sj7bt9c2gv', 0 , 'ALLSTATS LAST'));
```

Results 1 Execution plan - 1

```
SELECT * FROM TABLE(dbms_xplan.display_cursor('6x1sj7bt9c2gv', 0 , 'ALLSTATS LAST')) Enter a SQL expression to filter results (use Ctrl+Space)
```

Plan hash value: 3956160932

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	00:00:00.01	7

buffers 가 출력
되는지 확인합니다



ChatGPT와 함께



방법22. SQL을 재작성해서 튜닝하세요. 첫번째!

□ 학습 내용

1. SQL 재작성하는 튜닝의 필요성에 대해서 학습합니다.
2. 악성 SQL을 분석하여 왜 튜닝이 필요한지 이해합니다.
3. ChatGPT 를 활용하여 악성 SQL 튜닝하는 방법을 학습합니다.

□ 학습 목표

데이터 분석함수로 SQL을 재작성하여 튜닝할 수 있게 됩니다.

□ SQL 재작성 튜닝이 필요한 이유

1. 힌트 만으로는 성능이 더이상 좋아지지 않는 SQL 일때
2. 같은 데이터가 있는 다른 테이블을 선택하는게 더 바람직할 때
3. 같은 결과가 나오는 더 좋은 SQL이 있을 때

튜닝전 : select decode(no,1, deptno, 2, null) as deptno, sum(sal)
from emp e, (select rownum no
from dual
connect by level <= 2) d
group by decode(no, 1, deptno, 2, null)
order by 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

출력결과

DEPTNO	SUM(SAL)
10	8750
20	10875
30	9400
	29025



튜닝전 :

```
select decode(no,1, deptno, 2, null) as deptno, sum(sal)
  from emp e, ( select rownum no
                  from dual
                 connect by level <= 2 ) v
 group by decode(no, 1, deptno, 2, null )
order by deptno;
```

**악성 SQL분석을 통해 SQL 튜닝 스킬을
향상 시키세요 !**

출력결과

DEPTNO	SUM(SAL)
10	8750
20	10875
30	9400
	29025

튜닝전:

```
select deptno, sal, no  
from emp e, ( select rownum no  
from dual  
connect by level <= 2 ) v
```

emp 테이블

DEPTNO	SAL
10	5000
30	2850
10	2450
20	2975
30	1250
30	1600
30	1500
30	950
30	1250
20	3000
20	800
20	3000
20	1100
10	1300

In line view

NO
1
2

출력결과

DEPTNO	SAL	NO
10	5000	1
30	2850	1
10	2450	1
20	2975	1
30	1250	1
30	1600	1
30	1500	1
30	950	1
30	1250	1
20	3000	1
20	800	1
20	3000	1
20	1100	1
10	1300	1

DEPTNO	SAL	NO
10	5000	2
30	2850	2
10	2450	2
20	2975	2
30	1250	2
30	1600	2
30	1500	2
30	950	2
30	1250	2
20	3000	2
20	800	2
20	3000	2
20	1100	2
10	1300	2

튜닝전:

```
select decode(no,1,deptno,2, null) as deptno, sal, no  
from emp e, ( select rownum no  
from dual  
connect by level <= 2 ) v
```

emp 테이블

DEPTNO	SAL
10	5000
30	2850
10	2450
20	2975
30	1250
30	1600
30	1500
30	950
30	1250
20	3000
20	800
20	3000
20	1100
10	1300

In line view

NO
1
2

출력결과

DEPTNO	SAL	NO
10	5000	1
30	2850	1
10	2450	1
20	2975	1
30	1250	1
30	1600	1
30	1500	1
30	950	1
30	1250	1
20	3000	1
20	800	1
20	3000	1
20	1100	1
10	1300	1

null	5000	2
null	2850	2
null	2450	2
null	2975	2
null	1250	2
null	1600	2
null	1500	2
null	950	2
null	1250	2
null	3000	2
null	800	2
null	3000	2
null	1100	2
null	1300	2

튜닝전:

```
select decode(no,1,deptno,2, null) as deptno, sum(sal)
  from emp e, ( select rownum no
                  from dual
               connect by level <= 2 ) v
 group by decode(no,1,deptno,2,null);
```

emp 테이블

SAL	DEPTNO
5000	10
2850	30
2450	10
2975	20
1250	30
1600	30
1500	30
950	30
1250	30
3000	20
800	20
3000	20
1100	20
1300	10

In line view

NO
1
2

DEPTNO	SAL	NO
10	5000	1
30	2850	1
10	2450	1
20	2975	1
30	1250	1
30	1600	1
30	1500	1
30	950	1
30	1250	1
20	3000	1
20	800	1
20	3000	1
20	1100	1
10	1300	1

DEPTNO	SAL	NO
null	5000	2
null	2850	2
null	2450	2
null	2975	2
null	1250	2
null	1600	2
null	1500	2
null	950	2
null	1250	2
null	3000	2
null	800	2
null	3000	2
null	1100	2
null	1300	2

출력결과

DEPTNO	SUM(SAL)
10	8750
20	10875
30	9400
	29025

튜닝전 : select decode(no,1, deptno, 2, null) as deptno, sum(sal)
from emp e, (select rownum no
from dual
connect by level <= 2) v
group by decode(no, 1, deptno, 2, null)
order by deptno;



튜닝후 : select deptno, sum(sal)
from emp
group by rollup(deptno)
order by deptno asc;

출력결과

DEPTNO	SUM(SAL)
10	8750
20	10875
30	9400
	29025

rollup을 안썼을 때 :

```
select deptno, sum(sal)
      from emp
     group by deptno
    order by deptno asc;
```

출력결과

DEPTNO	SUM(SAL)
10	8750
20	10875
30	9400

rollup을 썼을 때 :

```
select deptno, sum(sal)
      from emp
     group by rollup(deptno)
    order by deptno asc;
```

출력결과

DEPTNO	SUM(SAL)
10	8750
20	10875
30	9400
	29025

튜닝전 :

```
select decode(no,1, deptno, 2, null) as deptno, sum(sal)
from emp e, ( select rownum no
               from dual
             connect by level <= 2 ) v
group by decode(no, 1, deptno, 2, null )
order by deptno;
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	1Mem	Used-Mem
0	SELECT STATEMENT		1	1	4	00:00:00.01	7			
1	SORT GROUP BY		1	3	4	00:00:00.01	7	2048	2048	2048 (0)
2	MERGE JOIN CARTESIAN		1	14	28	00:00:00.01	7			
3	VIEW		1	1	2	00:00:00.01	0			
4	COUNT		1		2	00:00:00.01	0			
5	CONNECT BY WITHOUT FILTERING		1		2	00:00:00.01	0	2048	2048	2048 (0)
6	FAST DUAL		1	1	1	00:00:00.01	0			
7	BUFFER SORT		2	14	28	00:00:00.01	7	2048	2048	2048 (0)
8	TABLE ACCESS FULL	EMP	1	14	14	00:00:00.01	7			

```
select deptno, sum(sal)
  from emp
 group by rollup(deptno)
 order by deptno asc;
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	1Mem	Used-Mem
0 SELECT STATEMENT 1 4 00:00:00.01 6										
1 SORT GROUP BY ROLLUP 1 3 4 00:00:00.01 6 2048 2048 2048 (0)										
2 TABLE ACCESS FULL EMP 1 14 14 00:00:00.01 6										



ChatGPT 를 통해서 쉽게 SQL 튜닝하기



ChatGPT 에게 요청할 튜닝전 SQL :

```
select decode(no,1, job, 2, null) as job, sum(sal)
from emp e, ( select rownum no
               from dual
             connect by level <= 2 ) v
group by decode(no, 1, job, 2, null )
order by job;
```

출력결과

JOB	SUM(SAL)
ANALYST	6000
CLERK	4150
MANAGER	8275
PRESIDENT	5000
SALESMAN	5600
	29025

ChatGPT 를 통해서 얻어낼 최종 튜닝후 결과:



ChatGPT

주어진 SQL 쿼리를 참고하여 주어진 SQL 쿼리를 튜닝하겠습니다.

기존의 쿼리는 특정 'JOB'에 대한 월급 합계를 계산하고 있습니다. 이를 'ROLLUP'을 사용하여 간결하게 변경해 보겠습니다.

```
sql
-- 튜닝 후:
SELECT
    job,
    SUM(sal)
FROM
    emp
GROUP BY
    ROLLUP(job)
ORDER BY
    job NULLS FIRST;
```



ChatGPT와 함께



방법23. SQL을 재작성해서 튜닝하세요. 두번째!

▣ 학습 내용

1. 악성 SQL을 분석하여 왜 튜닝이 필요한지 이해합니다.
2. 튜닝후 SQL이 왜 더 성능이 우수한지 학습합니다.
3. ChatGPT를 나에게 더 최적화 해서 SQL튜닝하는 tip을 배웁니다.

▣ 학습 목표

데이터 분석함수로 SQL을 재작성하여 튜닝할 수 있게 됩니다.

직업별 부서번호별 토탈월급과 부서번호별 관리자 번호별 토탈월급을 출력하시오

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

출력결과

JOB	DEPTNO	MGR	SUM(SAL)
ANALYST	20		6000
CLERK	10		1300
CLERK	20		1900
CLERK	30		950
MANAGER	10		2450
MANAGER	20		2975
MANAGER	30		2850
PRESIDENT	10		5000
SALESMAN	30		5600
	10	7782	1300
	10	7839	2450
	10		5000
	20	7566	6000
	20	7788	1100
	20	7839	2975
	20	7902	800
	30	7698	6550
	30	7839	2850



튜닝전 :

```
select job, deptno, null as mgr, sum(sal)
  from emp
 group by job, deptno
union all
select null as job, deptno, mgr, sum(sal)
  from emp
 group by deptno, mgr
order by job, deptno, mgr;
```

출력결과

JOB	DEPTNO	MGR	SUM(SAL)
ANALYST	20		6000
CLERK	10		1300
CLERK	20		1900
CLERK	30		950
MANAGER	10		2450
MANAGER	20		2975
MANAGER	30		2850
PRESIDENT	10		5000
SALESMAN	30		5600
	10	7782	1300
	10	7839	2450
	10		5000
	20	7566	6000
	20	7788	1100
	20	7839	2975
	20	7902	800
	30	7698	6550
	30	7839	2850

튜닝전 :

```
select job, deptno, null as mgr, sum(sal)  
from emp  
group by job, deptno
```

union all

```
select null as job, deptno, mgr, sum(sal)  
from emp  
group by deptno, mgr  
order by job, deptno, mgr;
```

출력결과

JOB	DEPTNO	MGR	SUM(SAL)
ANALYST	20		6000
CLERK	10		1300
CLERK	20		1900
CLERK	30		950
MANAGER	10		2450
MANAGER	20		2975
MANAGER	30		2850
PRESIDENT	10		5000
SALESMAN	30		5600
	10	7782	1300
	10	7839	2450
	10		5000
	20	7566	6000
	20	7788	1100
	20	7839	2975
	20	7902	800
	30	7698	6550
	30	7839	2850

튜닝전 :

```
select job, deptno, null as mgr, sum(sal)
from emp
group by job, deptno
```

union all

```
select null as job, deptno, mgr, sum(sal)
from emp
group by deptno, mgr
order by job, deptno, mgr;
```

출력결과

JOB	DEPTNO	MGR	SUM(SAL)
ANALYST	20		6000
CLERK	10		1300
CLERK	20		1900
CLERK	30		950
MANAGER	10		2450
MANAGER	20		2975
MANAGER	30		2850
PRESIDENT	10		5000
SALESMAN	30		5600
	10	7782	1300
	10	7839	2450
	10		5000
	20	7566	6000
	20	7788	1100
	20	7839	2975
	20	7902	800
	30	7698	6550
	30	7839	2850

emp 테이블을 2번 select 했습니다 !

튜닝전 :

```
select job, deptno, null as mgr, sum(sal)
  from emp
 group by job, deptno
union all
select null as job, deptno, mgr, sum(sal)
  from emp
 group by deptno, mgr
 order by job, deptno, mgr;
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0 SELECT STATEMENT			1		18	00:00:00.01	12
1 UNION-ALL			1		18	00:00:00.01	12
2 HASH GROUP BY			1	11	9	00:00:00.01	6
3 TABLE ACCESS FULL	EMP		1	14	14	00:00:00.01	6
4 HASH GROUP BY			1	13	9	00:00:00.01	6
5 TABLE ACCESS FULL	EMP		1	14	14	00:00:00.01	6

튜닝후 :

```
select job, deptno, mgr, sum(sal)
from emp
group by grouping sets( (job,deptno) , (deptno,mgr) )
order by job, deptno, mgr ;
```

출력결과

JOB	DEPTNO	MGR	SUM(SAL)
ANALYST	20		6000
CLERK	10		1300
CLERK	20		1900
CLERK	30		950
MANAGER	10		2450
MANAGER	20		2975
MANAGER	30		2850
PRESIDENT	10		5000
SALESMAN	30		5600
	10	7782	1300
	10	7839	2450
	10		5000
	20	7566	6000
	20	7788	1100
	20	7839	2975
	20	7902	800
	30	7698	6550
	30	7839	2850

emp 테이블을 1번만 select 했습니다 !

튜닝후 :

```
select job, deptno, mgr, sum(sal)
from emp
group by grouping sets( (job,deptno) , (deptno,mgr) )
order by job, deptno, mgr ;
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		18	00:00:00.01	8
1	TEMP TABLE TRANSFORMATION		1		18	00:00:00.01	8
2	LOAD AS SELECT (CURSOR DURATION MEMORY)	SYS_TEMP_0FD9D6638_10E2D0A	1		0	00:00:00.01	7
3	TABLE ACCESS FULL	EMP	1	14	14	00:00:00.01	6
4	LOAD AS SELECT (CURSOR DURATION MEMORY)	SYS_TEMP_0FD9D6639_10E2D0A	1		0	00:00:00.01	0
5	HASH GROUP BY		1	13	9	00:00:00.01	0
6	TABLE ACCESS FULL	SYS_TEMP_0FD9D6638_10E2D0A	1	14	14	00:00:00.01	0
7	LOAD AS SELECT (CURSOR DURATION MEMORY)	SYS_TEMP_0FD9D6639_10E2D0A	1		0	00:00:00.01	0
8	HASH GROUP BY		1	11	9	00:00:00.01	0
9	TABLE ACCESS FULL	SYS_TEMP_0FD9D6638_10E2D0A	1	14	14	00:00:00.01	0
10	SORT ORDER BY		1	14	18	00:00:00.01	0
11	VIEW		1	13	18	00:00:00.01	0
12	TABLE ACCESS FULL	SYS_TEMP_0FD9D6639_10E2D0A	1	13	18	00:00:00.01	0



ChatGPT 를 통해서 쉽게 SQL 튜닝하기



ChatGPT 에게 요청할 때 좋은 결과를 얻기 위한 tip 3가지!

1. 튜닝하기 전에 이제부터 너는 SQL 튜닝 선생님이야라고 정체성을 부여합니다.
2. 내가 보고자 하는 다른 튜닝 케이스의 튜닝전 SQL과 튜닝후 SQL을 예시로 주면서 지금 튜닝하고자 하는 SQL을 튜닝해달라고 요청합니다.
3. chatgpt가 출력해준 결과를 반드시 확인하여 chatgpt의 착시현상으로 잘못된 결과를 도출한건 아닌지 반드시 점검합니다.



ChatGPT 에게 정체성 부여하기 !



You

이제부터 너는 sql 튜닝 선생님이야



ChatGPT

알겠습니다! SQL 튜닝에 관련된 질문이나 문제가 있으면 언제든지 물어보세요. SQL 쿼리의 최적화, 인덱싱, 실행 계획 분석 등에 대한 도움을 드리겠습니다. 무엇을 도와드릴까요?



ChatGPT에게 요청할 튜닝전 SQL :

```
select job, deptno, null as mgr, sum(sal)
from emp
group by job, deptno
union all
select null as job, deptno, mgr, sum(sal)
from emp
group by deptno, mgr
union all
select to_char(null) as job,to_number(null) as deptno,to_number(null) as mgr,sum(sal)
from emp;
```



ChatGPT 를 통해서 얻어낼 최종 튜닝후 결과:



ChatGPT

주어진 SQL 쿼리를 튜닝하기 위해 "GROUPING SETS"를 사용하여 하나의 쿼리로 원하는 결과를 얻을 수 있습니다. "GROUPING SETS"를 사용하면 여러 개의 UNION ALL을 통합하여 더 간결하고 효율적인 쿼리를 작성할 수 있습니다.

다음은 튜닝된 SQL 쿼리입니다:

sql

Copy code

```
select job, deptno, mgr, sum(sal)
from emp
group by grouping sets((job, deptno), (deptno, mgr), ());
```

이렇게 하면 원래의 세 개의 SELECT 문을 하나의 쿼리로 통합하여 효율적으로 결과를 얻을 수 있습니다.





ChatGPT 강화학습 시키기 !



ChatGPT

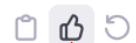
주어진 SQL 쿼리를 튜닝하기 위해 "GROUPING SETS"를 사용하여 하나의 쿼리로 원하는 결과를 얻을 수 있습니다. "GROUPING SETS"를 사용하면 여러 개의 UNION ALL을 통합하여 더 간결하고 효율적인 쿼리를 작성할 수 있습니다.

다음은 튜닝된 SQL 쿼리입니다:

```
sql
Copy code

select job, deptno, mgr, sum(sal)
from emp
group by grouping sets((job, deptno), (deptno, mgr), ());
```

이렇게 하면 원래의 세 개의 SELECT 문을 하나의 쿼리로 통합하여 효율적으로 결과를 얻을 수 있습니다.



Provide additional feedback

너무 잘했어 ! 최고 !

Submit feedback

나에게 더 맞춤형 인공지능이
되려면 보상을 해주세요 !



ChatGPT와 함께



방법24. SQL을 재작성해서 튜닝하세요. 세번째!

▣ 학습 내용

1. 악성 SQL을 분석하여 왜 튜닝이 필요한지 이해합니다.
2. 튜닝후 SQL이 왜 더 성능이 우수한지 학습합니다.
3. ChatGPT를 나에게 더 최적화 해서 SQL튜닝하는 tip을 배웁니다.

▣ 학습 목표

데이터 분석함수로 SQL을 재작성하여 튜닝할 수 있게 됩니다.

사원번호, 이름, 월급, 월급의 누적치를 출력하세요

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

출력결과

EMPNO	ENAME	SAL	누적치
7369	SMITH	800	800
7499	ALLEN	1600	2400
7521	WARD	1250	3650
7566	JONES	2975	6625
7654	MARTIN	1250	7875
7698	BLAKE	2850	10725
7782	CLARK	2450	13175
7788	SCOTT	3000	16175
7839	KING	5000	21175
7844	TURNER	1500	22675
7876	ADAMS	1100	23775
7900	JAMES	950	24725
7902	FORD	3000	27725
7934	MILLER	1300	29025



튜닝전 :

```
select empno, ename,sal,  
       (select sum(sal) from emp e where e.empno <= b.empno) sumsal  
from emp b  
order by empno;      emp 테이블
```

7369 7369

출력결과

EMPNO	ENAME	SAL	SUMSAL
7369	SMITH	800	
7499	ALLEN	1600	
7521	WARD	1250	
7566	JONES	2975	
7654	MARTIN	1250	
7698	BLAKE	2850	
7782	CLARK	2450	
7788	SCOTT	3000	
7839	KING	5000	
7844	TURNER	1500	
7876	ADAMS	1100	
7900	JAMES	950	
7902	FORD	3000	
7934	MILLER	1300	



EMPNO	ENAME	SAL	SUMSAL
7369	SMITH	800	800

튜닝전 :

```
select empno, ename, sal,  
       (select sum(sal) from emp e where e.empno <= b.empno) sumsal  
from emp b  
order by empno;      emp 테이블
```

7369

7499

7499

출력결과

EMPNO	ENAME	SAL	SUMSAL
7369	SMITH	800	
7499	ALLEN	1600	
7521	WARD	1250	
7566	JONES	2975	
7654	MARTIN	1250	
7698	BLAKE	2850	
7782	CLARK	2450	
7788	SCOTT	3000	
7839	KING	5000	
7844	TURNER	1500	
7876	ADAMS	1100	
7900	JAMES	950	
7902	FORD	3000	
7934	MILLER	1300	



EMPNO	ENAME	SAL	SUMSAL
7369	SMITH	800	800
7499	ALLEN	1600	2400

튜닝전 :

```
select empno, ename, sal,  
       (select sum(sal) from emp e where e.empno <= b.empno) sumsal  
from emp b  
order by empno;      emp 테이블
```

EMPNO	ENAME	SAL	SUMSAL
7369	SMITH	800	
7499	ALLEN	1600	
7521	WARD	1250	
7566	JONES	2975	
7654	MARTIN	1250	
7698	BLAKE	2850	
7782	CLARK	2450	
7788	SCOTT	3000	
7839	KING	5000	
7844	TURNER	1500	
7876	ADAMS	1100	
7900	JAMES	950	
7902	FORD	3000	
7934	MILLER	1300	



EMPNO	ENAME	SAL	SUMSAL
7369	SMITH	800	800
7499	ALLEN	1600	2400
7521	WARD	1250	3650

출력결과

튜닝전 :

```
select empno, ename,sal,  
       (select sum(sal) from emp e where e.empno <= b.empno) sumsal  
from emp b  
order by empno;    emp 테이블
```

EMPNO	ENAME	SAL	SUMSAL
7369	SMITH	800	
7499	ALLEN	1600	
7521	WARD	1250	
7566	JONES	2975	
7654	MARTIN	1250	
7698	BLAKE	2850	
7782	CLARK	2450	
7788	SCOTT	3000	
7839	KING	5000	
7844	TURNER	1500	
7876	ADAMS	1100	
7900	JAMES	950	
7902	FORD	3000	
7934	MILLER	1300	



출력결과

EMPNO	ENAME	SAL	SUMSAL
7369	SMITH	800	800
7499	ALLEN	1600	2400
7521	WARD	1250	3650
7566	JONES	2975	6625
7654	MARTIN	1250	7875
7698	BLAKE	2850	10725
7782	CLARK	2450	13175
7788	SCOTT	3000	16175
7839	KING	5000	21175
7844	TURNER	1500	22675
7876	ADAMS	1100	23775
7900	JAMES	950	24725
7902	FORD	3000	27725
7934	MILLER	1300	29025

튜닝전 :

```
select empno, ename, sal,  
      (select sum(sal) from emp e where e.empno <= b.empno) sumsal  
from emp b  
order by empno;
```

select 절의 서브쿼리를 14번
실행했습니다 !

출력결과

EMPNO	ENAME	SAL	SUMSAL
7369	SMITH	800	800
7499	ALLEN	1600	2400
7521	WARD	1250	3650
7566	JONES	2975	6625
7654	MARTIN	1250	7875
7698	BLAKE	2850	10725
7782	CLARK	2450	13175
7788	SCOTT	3000	16175
7839	KING	5000	21175
7844	TURNER	1500	22675
7876	ADAMS	1100	23775
7900	JAMES	950	24725
7902	FORD	3000	27725
7934	MILLER	1300	29025

튜닝후 :

```
select empno, ename, sal,  
       sum(sal) over ( order by empno asc ) sumsal  
from emp;
```

출력결과

EMPNO	ENAME	SAL	SUMSAL
7369	SMITH	800	800
7499	ALLEN	1600	2400
7521	WARD	1250	3650
7566	JONES	2975	6625
7654	MARTIN	1250	7875
7698	BLAKE	2850	10725
7782	CLARK	2450	13175
7788	SCOTT	3000	16175
7839	KING	5000	21175
7844	TURNER	1500	22675
7876	ADAMS	1100	23775
7900	JAMES	950	24725
7902	FORD	3000	27725
7934	MILLER	1300	29025

튜닝전 :

```
select empno, ename,sal,  
       (select sum(sal) from emp e where e.empno <= b.empno) sumsal  
from emp b order by empno;
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	1Mem	Used-Mem	
0 SELECT STATEMENT 1 14 00:00:00.01 90	1 SORT AGGREGATE 14 1 14 00:00:00.01 84	* 2 TABLE ACCESS FULL EMP 14 1 105 00:00:00.01 84	3 SORT ORDER BY 1 14 14 00:00:00.01 90 2048 2048 2048 (0)	4 TABLE ACCESS FULL EMP 1 14 14 00:00:00.01 6							

튜닝후 :

```
select empno, ename, sal,  
       sum(sal) over ( order by empno asc ) sumsal  
from emp;
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	1Mem	Used-Mem
0	SELECT STATEMENT		1	14	14	00:00:00.01	6			
1	WINDOW SORT		1	14	14	00:00:00.01	6	2048	2048	2048 (0)
2	TABLE ACCESS FULL	EMP	1	14	14	00:00:00.01	6			



ChatGPT 를 통해서 쉽게 SQL 튜닝하기



ChatGPT 에게 요청할 때 좋은 결과를 얻기 위한 tip 4가지!

1. 튜닝하기 전에 이제부터 너는 SQL 튜닝 선생님이야라고 정체성을 부여합니다.
2. 내가 보고자 하는 다른 튜닝 케이스의 튜닝전 SQL과 튜닝후 SQL을 예시로 주면서 지금 튜닝하고자 하는 SQL을 튜닝해달라고 요청합니다.
3. chatgpt가 출력해준 결과를 반드시 확인하여 chatgpt의 착시현상으로 잘못된 결과를 도출한건 아닌지 반드시 점검합니다.
4. 결과가 잘 출력되었으면 좋아요 버튼으로 반드시 보상을 해줍니다.



ChatGPT에게 요청할 튜닝전 SQL :

```
select deptno, empno, ename,sal, ( select sum(sal)
    from emp e
    where e.empno <= b.empno and e.deptno = b.deptno
    and job in ('CLERK', 'ANALYST')
) sumsal
from emp b
where job in ('CLERK', 'ANALYST')
order by deptno, empno;
```



ChatGPT 를 통해서 얻어낼 최종 튜닝후 결과:



ChatGPT

주어진 SQL 쿼리를 튜닝하기 위해 "Window Functions"를 사용하여 더 효율적인 쿼리를 작성할 수 있습니다. 특히, `SUM(sal) OVER (PARTITION BY deptno ORDER BY empno ASC)`와 같은 윈도우 함수를 사용하면, 서브쿼리 없이도 원하는 결과를 얻을 수 있습니다.

다음은 튜닝된 SQL 쿼리입니다:

```
sql Copy code  
  
select deptno, empno, ename, sal, sum(sal) over (partition by deptno or  
from emp  
where job in ('CLERK', 'ANALYST')  
order by deptno, empno;
```

이렇게 하면 원래의 서브쿼리를 제거하고 `SUM(sal) OVER`를 사용하여 동일한 결과를 더 효율적으로 얻을 수 있습니다.





ChatGPT 강화학습 시키기 !



ChatGPT

주어진 SQL 쿼리를 튜닝하기 위해 "Window Functions"를 사용하여 더 효율적인 쿼리를 작성할 수 있습니다. 특히, `SUM(sal) OVER (PARTITION BY deptno ORDER BY empno ASC)`와 같은 윈도우 함수를 사용하면, 서브쿼리 없이도 원하는 결과를 얻을 수 있습니다.

다음은 튜닝된 SQL 쿼리입니다:

```
sql Copy code
select deptno, empno, ename, sal, sum(sal) over (partition by deptno or
from emp
where job in ('CLERK', 'ANALYST')
order by deptno, empno;
```

이렇게 하면 원래의 서브쿼리를 제거하고 `SUM(sal) OVER`를 사용하여 동일한 결과를 더 효율적으로 얻을 수 있습니다.



Provide additional feedback

너무 잘했어! 최고!



Submit feedback

나에게 더 맞춤형 인공지능이
되려면 보상을 해주세요 !



ChatGPT와 함께



방법25. SQL을 재작성해서 튜닝하세요. 네번째!

▣ 학습 내용

1. 악성 SQL을 분석하여 왜 튜닝이 필요한지 이해합니다.
2. 튜닝후 SQL이 왜 더 성능이 우수한지 학습합니다.
3. ChatGPT를 나에게 더 최적화 해서 SQL튜닝하는 tip을 배웁니다.

▣ 학습 목표

데이터 분석함수로 SQL을 재작성하여 튜닝할 수 있게 됩니다.

사원번호, 이름, 월급, 이전행의 월급이 출력되게하시오

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

출력결과

EMPNO	ENAME	SAL	BEFORE_SAL
7369	SMITH	800	
7499	ALLEN	1600	800
7521	WARD	1250	1600
7566	JONES	2975	1250
7654	MARTIN	1250	2975
7698	BLAKE	2850	1250
7782	CLARK	2450	2850
7788	SCOTT	3000	2450
7839	KING	5000	3000
7844	TURNER	1500	5000
7876	ADAMS	1100	1500
7900	JAMES	950	1100
7902	FORD	3000	950
7934	MILLER	1300	3000



튜닝전 :

```
select a.empno ,a.ename, a.sal, b.sal as before_sal
from ( select rownum no1, empno,ename, sal
      from ( select empno, ename, sal
              from emp
             order by empno ) ) a,
( select rownum + 1 no2, empno, ename, sal
      from ( select empno, ename, sal
              from emp
             order by empno ) ) b
where a.no1 = b.no2 (+)
order by no1;
```

출력결과

EMPNO	ENAME	SAL	BEFORE_SAL
7369	SMITH	800	
7499	ALLEN	1600	800
7521	WARD	1250	1600
7566	JONES	2975	1250
7654	MARTIN	1250	2975
7698	BLAKE	2850	1250
7782	CLARK	2450	2850
7788	SCOTT	3000	2450
7839	KING	5000	3000
7844	TURNER	1500	5000
7876	ADAMS	1100	1500
7900	JAMES	950	1100
7902	FORD	3000	950
7934	MILLER	1300	3000

튜닝전 :

```
select a.empno ,a.ename, a.sal, b.sal as before_sal
from ( select rownum no1, empno,ename, sal
      from ( select empno, ename, sal
              from emp
              order by empno ) ) a,
( select rownum + 1 no2, empno, ename, sal
      from ( select empno, ename, sal
              from emp
              order by empno ) ) b
where a.no1 = b.no2 (+)
order by no1;
```

출력결과

EMPNO	ENAME	SAL	BEFORE_SAL
7369	SMITH	800	
7499	ALLEN	1600	800
7521	WARD	1250	1600
7566	JONES	2975	1250
7654	MARTIN	1250	2975
7698	BLAKE	2850	1250
7782	CLARK	2450	2850
7788	SCOTT	3000	2450
7839	KING	5000	3000
7844	TURNER	1500	5000
7876	ADAMS	1100	1500
7900	JAMES	950	1100
7902	FORD	3000	950
7934	MILLER	1300	3000

첫번째 in line view

```
select rownum no1, empno,ename, sal  
from ( select empno, ename, sal  
       from emp  
      order by empno );
```

첫번째 in line view

```
select rownum no1, empno,ename, sal  
from ( select empno, ename, sal  
       from emp  
      order by empno ) ;
```

NO1	EMPNO	ENAME	SAL
1	7369	SMITH	800
2	7499	ALLEN	1600
3	7521	WARD	1250
4	7566	JONES	2975
5	7654	MARTIN	1250
6	7698	BLAKE	2850
7	7782	CLARK	2450
8	7788	SCOTT	3000
9	7839	KING	5000
10	7844	TURNER	1500
11	7876	ADAMS	1100
12	7900	JAMES	950
13	7902	FORD	3000
14	7934	MILLER	1300

첫번째 in line view

```
select rownum no1, empno,ename, sal  
from ( select empno, ename, sal  
       from emp  
      order by empno ) ;
```

NO1	EMPNO	ENAME	SAL
1	7369	SMITH	800
2	7499	ALLEN	1600
3	7521	WARD	1250
4	7566	JONES	2975
5	7654	MARTIN	1250
6	7698	BLAKE	2850
7	7782	CLARK	2450
8	7788	SCOTT	3000
9	7839	KING	5000
10	7844	TURNER	1500
11	7876	ADAMS	1100
12	7900	JAMES	950
13	7902	FORD	3000
14	7934	MILLER	1300

두번째 in line view

```
select rownum + 1 no2, empno, ename, sal  
from ( select empno, ename, sal  
       from emp  
      order by empno ) ;
```

첫번째 in line view

```
select rownum no1, empno,ename, sal  
from ( select empno, ename, sal  
       from emp  
      order by empno ) ;
```

NO1	EMPNO	ENAME	SAL
1	7369	SMITH	800
2	7499	ALLEN	1600
3	7521	WARD	1250
4	7566	JONES	2975
5	7654	MARTIN	1250
6	7698	BLAKE	2850
7	7782	CLARK	2450
8	7788	SCOTT	3000
9	7839	KING	5000
10	7844	TURNER	1500
11	7876	ADAMS	1100
12	7900	JAMES	950
13	7902	FORD	3000
14	7934	MILLER	1300

두번째 in line view

```
select rownum + 1 no2, empno, ename, sal  
from ( select empno, ename, sal  
       from emp  
      order by empno ) ;
```

NO2	EMPNO	ENAME	SAL
2	7369	SMITH	800
3	7499	ALLEN	1600
4	7521	WARD	1250
5	7566	JONES	2975
6	7654	MARTIN	1250
7	7698	BLAKE	2850
8	7782	CLARK	2450
9	7788	SCOTT	3000
10	7839	KING	5000
11	7844	TURNER	1500
12	7876	ADAMS	1100
13	7900	JAMES	950
14	7902	FORD	3000
15	7934	MILLER	1300

두 집합을 조인한다면 ?

```
select rownum no1, empno,ename, sal  
from ( select empno, ename, sal  
       from emp  
      order by empno ) ;
```

NO1	EMPNO	ENAME	SAL
1	7369	SMITH	800
2	7499	ALLEN	1600
3	7521	WARD	1250
4	7566	JONES	2975
5	7654	MARTIN	1250
6	7698	BLAKE	2850
7	7782	CLARK	2450
8	7788	SCOTT	3000
9	7839	KING	5000
10	7844	TURNER	1500
11	7876	ADAMS	1100
12	7900	JAMES	950
13	7902	FORD	3000
14	7934	MILLER	1300

```
select rownum + 1 no2, empno, ename, sal  
from ( select empno, ename, sal  
       from emp  
      order by empno ) ;
```

NO2	EMPNO	ENAME	SAL
2	7369	SMITH	800
3	7499	ALLEN	1600
4	7521	WARD	1250
5	7566	JONES	2975
6	7654	MARTIN	1250
7	7698	BLAKE	2850
8	7782	CLARK	2450
9	7788	SCOTT	3000
10	7839	KING	5000
11	7844	TURNER	1500
12	7876	ADAMS	1100
13	7900	JAMES	950
14	7902	FORD	3000
15	7934	MILLER	1300

튜닝전 :

```
select a.empno ,a.ename, a.sal, b.sal as before_sal
from ( select rownum no1, empno,ename, sal
      from ( select empno, ename, sal
              from emp
              order by empno ) ) a,
( select rownum + 1 no2, empno, ename, sal
      from ( select empno, ename, sal
              from emp
              order by empno ) ) b
where a.no1 = b.no2 (+) ←
order by no1;
```

출력결과

EMPNO	ENAME	SAL	BEFORE_SAL
7369	SMITH	800	
7499	ALLEN	1600	800
7521	WARD	1250	1600
7566	JONES	2975	1250
7654	MARTIN	1250	2975
7698	BLAKE	2850	1250
7782	CLARK	2450	2850
7788	SCOTT	3000	2450
7839	KING	5000	3000
7844	TURNER	1500	5000
7876	ADAMS	1100	1500
7900	JAMES	950	1100
7902	FORD	3000	950
7934	MILLER	1300	3000

튜닝후 :

```
select empno, ename, sal,  
       lag(sal,1) over ( order by empno ) as sal_lag  
from emp;
```

출력결과

EMPNO	ENAME	SAL	BEFORE_SAL
7369	SMITH	800	
7499	ALLEN	1600	800
7521	WARD	1250	1600
7566	JONES	2975	1250
7654	MARTIN	1250	2975
7698	BLAKE	2850	1250
7782	CLARK	2450	2850
7788	SCOTT	3000	2450
7839	KING	5000	3000
7844	TURNER	1500	5000
7876	ADAMS	1100	1500
7900	JAMES	950	1100
7902	FORD	3000	950
7934	MILLER	1300	3000

튜닝전 :

```
select a.empno ,a.ename, a.sal, b.sal as before_sal
  from ( select rownum no1, empno,ename, sal
            from ( select empno, ename, sal
                      from emp
                  order by empno ) ) a,
        ( select rownum + 1 no2, empno, ename, sal
            from ( select empno, ename, sal
                      from emp
                  order by empno ) ) b
   where a.no1 = b.no2 (+)
  order by no1;
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	1Mem	Used-Mem
0	SELECT STATEMENT		1		14	00:00:00.01	14			
1	SORT ORDER BY		1	14	14	00:00:00.01	14	2048	2048	2048 (0)
* 2	HASH JOIN OUTER		1	14	14	00:00:00.01	14	1476K	1476K	1055K (0)
3	VIEW		1	14	14	00:00:00.01	7			
4	COUNT		1		14	00:00:00.01	7			
5	VIEW		1	14	14	00:00:00.01	7			
6	SORT ORDER BY		1	14	14	00:00:00.01	7	2048	2048	2048 (0)
7	TABLE ACCESS FULL	EMP	1	14	14	00:00:00.01	7			
8	VIEW		1	14	14	00:00:00.01	7			
9	COUNT		1		14	00:00:00.01	7			
10	VIEW		1	14	14	00:00:00.01	7			
11	SORT ORDER BY		1	14	14	00:00:00.01	7	2048	2048	2048 (0)
12	TABLE ACCESS FULL	EMP	1	14	14	00:00:00.01	7			

튜닝후 :

```
select empno, ename, sal,  
       lag(sal,1) over ( order by empno ) as sal_lag  
from emp;
```

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	1Mem	Used-Mem	
0 SELECT STATEMENT 1 14 00:00:00.01 6											
1 WINDOW SORT 1 14 14 00:00:00.01 6 2048 2048 2048 (0)											
2 TABLE ACCESS FULL EMP 1 14 14 00:00:00.01 6											



ChatGPT 를 통해서 쉽게 SQL 튜닝하기



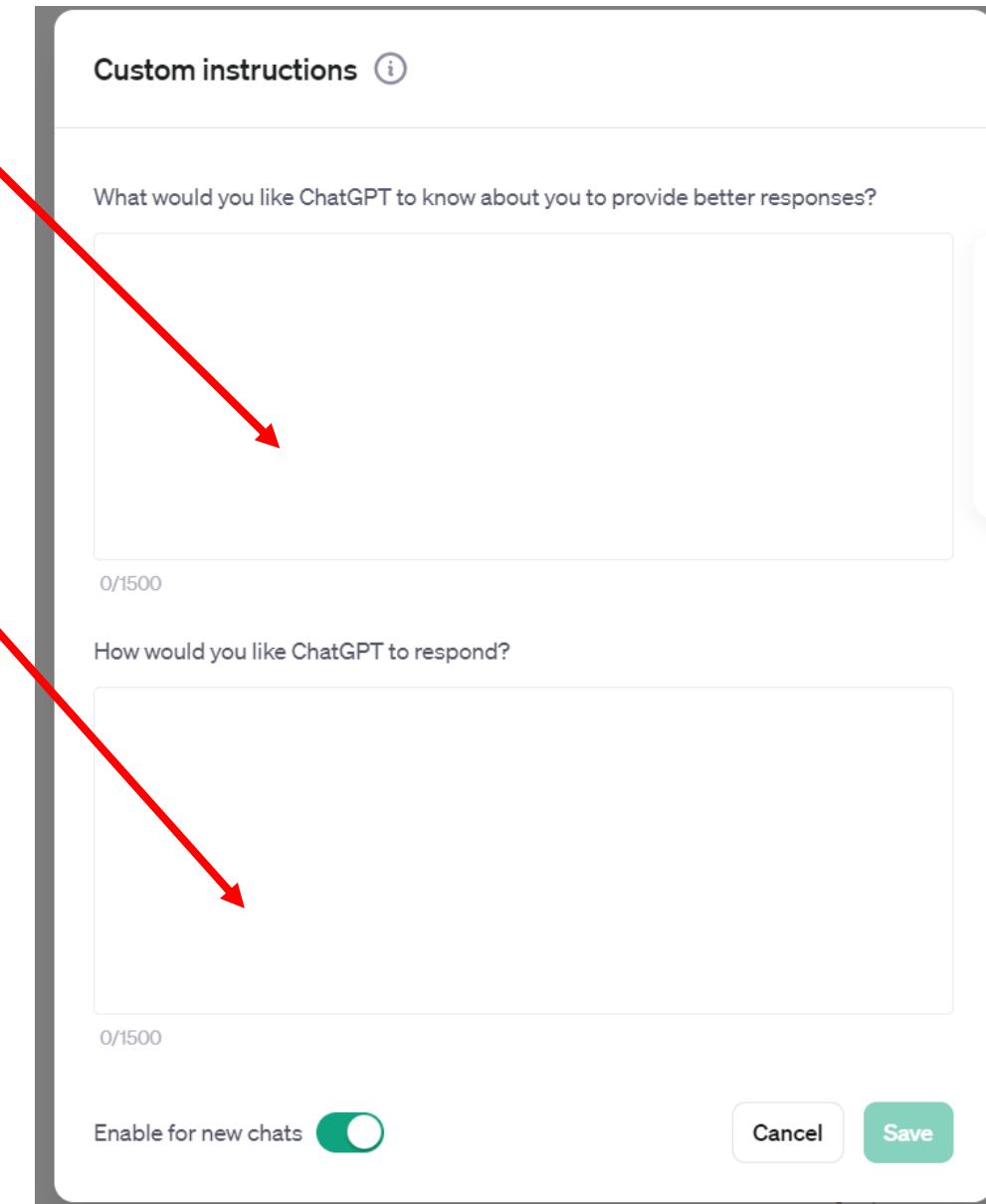
앞에서 얘기 했던 ChatGPT 활용 tip 4가지!

1. 튜닝하기 전에 이제부터 너는 SQL 튜닝 선생님이야라고 정체성을 부여합니다.
2. 내가 보고자 하는 다른 튜닝 케이스의 튜닝전 SQL과 튜닝후 SQL을 예시로 주면서 지금 튜닝하고자 하는 SQL을 튜닝해달라고 요청합니다.
3. chatgpt가 출력해준 결과를 반드시 확인하여 chatgtp의 착시현상으로 잘못된 결과를 도출한건 아닌지 반드시 점검합니다.
4. 결과가 잘 출력되었으면 좋아요 버튼으로 반드시 보상을 해줍니다.



더 좋은 결과를 얻기 위한 또 하나의 chatgpt 사용 tip !

SQL 튜너로써의 정체성을 더 분명히 합니다.





더 좋은 결과를 얻기 위한 또 하나의 chatgpt 사용 tip !

나의 목표

The screenshot shows the 'Custom instructions' section of the ChatGPT settings. The top part asks, 'What would you like ChatGPT to know about you to provide better responses?' Below it is a text area containing the Korean text: '나는 SQL 재작성을 잘하는 SQL 튜너가 되고 싶어.' A red arrow points from the text '나는 SQL 재작성을 잘하는 SQL 튜너가 되고 싶어.' to the text '나의 목표' on the left. The bottom part asks, 'How would you like ChatGPT to respond?' It contains a text area with the following Korean text:
나는 SQL 티닝 선생님을 원해 너는 이제부터 SQL 튜너야
앞으로 나에게 친절하게 대답해줬으면해.
모른다거나 죄송하다는 말은 하지 말아줘.
잘 티닝해서 꼭 알려줘야해 ~
A red arrow points from this text area to the text 'chatGPT에게 내가 원하는것' on the right.

Custom instructions ⓘ

What would you like ChatGPT to know about you to provide better responses?

나는 SQL 재작성을 잘하는 SQL 튜너가 되고 싶어.

30/1500

How would you like ChatGPT to respond?

나는 SQL 티닝 선생님을 원해 너는 이제부터 SQL 튜너야
앞으로 나에게 친절하게 대답해줬으면해.
모른다거나 죄송하다는 말은 하지 말아줘.
잘 티닝해서 꼭 알려줘야해 ~

97/1500

Enable for new chats

Cancel Save

chatGPT에게
내가 원하는것



ChatGPT에게 요청할 튜닝전 SQL :

```
select a.deptno , a.empno, a.ename, a.sal, b.sal as before_sal
  from ( select rownum no1, deptno, empno,ename, sal
          from ( select deptno, empno, ename, sal
                  from emp
                  order by deptno, sal ) ) a,
       ( select rownum +1 no2, deptno, empno, ename,
              from ( select deptno, empno, ename, sal
                      from emp
                      order by deptno,sal ) ) b
 where a.no1 = b.no2 (+)
order by no1;
```

출력결과

DEPTO	EMPNO	ENAME	SAL	BEFORE_SAL
10	7934	MILLER	1300	
10	7782	CLARK	2450	1300
10	7839	KING	5000	2450
20	7369	SMITH	800	5000
20	7876	ADAMS	1100	800
20	7566	JONES	2975	1100
20	7788	SCOTT	3000	2975
20	7902	FORD	3000	3000
30	7900	JAMES	950	3000
30	7521	WARD	1250	950
30	7654	MARTIN	1250	1250
30	7844	TURNER	1500	1250
30	7499	ALLEN	1600	1500
30	7698	BLAKE	2850	1600



ChatGPT 를 통해서 얻어낼 최종 튜닝후 결과:



ChatGPT

튜닝 전 SQL의 패턴을 유지하면서 튜닝 후 SQL을 작성하면 다음과 같습니다:

sql

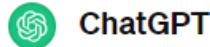
Copy code

```
SELECT empno, ename, sal, LAG(sal, 1) OVER (ORDER BY deptno, sal) AS sa
FROM emp
ORDER BY empno;
```

이 SQL은 튜닝 전 SQL의 구조를 유지하면서 'LAG' 함수를 사용하여 이전 행의 'sal' 값을 가져오도록 했습니다. 튜닝 후 SQL은 조인 없이도 'emp' 테이블을 1번만 액세스하여 필요한 정보를 얻을 수 있도록 작성되었습니다.

< 2 / 2 >

ChatGPT 강화학습 시키기 !



ChatGPT

튜닝 전 SQL의 패턴을 유지하면서 튜닝 후 SQL을 작성하면 다음과 같습니다:

```
sql Copy code
SELECT empno, ename, sal, LAG(sal, 1) OVER (ORDER BY deptno, sal) AS sa
FROM emp
ORDER BY empno;
```

이 SQL은 튜닝 전 SQL의 구조를 유지하면서 'LAG' 함수를 사용하여 이전 행의 'sal' 값을 가져오도록 했습니다. 튜닝 후 SQL은 조인 없이도 'emp' 테이블을 1번만 액세스하여 필요한 정보를 얻을 수 있도록 작성되었습니다.

< 2 / 2 > Copy Up Down Left Right

너무 잘했어! 최고!

나에게 더 맞춤형 인공지능이
되려면 보상을 해주세요 !

Submit feedback



ChatGPT 를 이용하여



방법26. 조인 SQL문장을 전문가 처럼 튜닝하기1

▣ 학습 내용

1. ChatGPT에게 튜닝시키기 전에 조인 순서의 중요성을 학습합니다.
2. ChatGPT에게 제공할 테이블 통계정보와 컬럼 통계정보를 학습합니다.
3. ChatGPT에게 조인 문장 튜닝시 좋은 결과를 얻기 위한 방법을 학습합니다.

▣ 학습 목표

ChatGPT를 이용하여 조인 문장의 조인순서와 조인 방법을 결정할 수 있다.

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
4. cross 조인

조인 방법

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

Nested loop 조인이란?

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

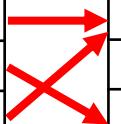
먼저 선행 테이블의 처리 범위를 하나씩 순차적으로 엑세스하면서 그 추출된 연결 고리 컬럼으로 연결할 테이블을 조인하는 방식

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

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



조인하는 순서를 조정하는 힌트는?

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





ChatGPT에게 요청할 튜닝전 SQL :

```
select 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

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



튜닝전 :

```
select /*+ leading(e d) use_nl(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
0	SELECT STATEMENT		1	1	14	00:00:00.01	92
1	NESTED LOOPS		1	14	14	00:00:00.01	92
2	TABLE ACCESS FULL	EMP	1	14	14	00:00:00.01	7
* 3	TABLE ACCESS FULL	DEPT	14	1	14	00:00:00.01	85

튜닝후 :

```
select /*+ leading(d e) use_nl(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	
	0	SELECT STATEMENT				1				14		00:00:00.01		32	
	1	NESTED LOOPS				1		14		14		00:00:00.01		32	
	2	TABLE ACCESS FULL		DEPT		1		4		4		00:00:00.01		7	
*	3	TABLE ACCESS FULL		EMP		4		4		14		00:00:00.01		25	



테이블 통계정보와 컬럼 통계정보 수집하기

begin

```
dbms_stats.gather_table_stats( user, 'emp') ;
dbms_stats.gather_table_stats( user, 'dept') ;
END ;
```



Chatgpt 에게 전해줄 테이블 통계정보

```
SELECT table_name, num_rows, avg_row_len, last_analyzed
FROM dba_tables
WHERE table_name IN ('EMP', 'DEPT');
```

TABLE_NAME	NUM_ROWS	AVG_ROW_LEN	LAST_ANALYZED
DEPT	4	23	2024-01-19 10:08:10.000
EMP	14	40	2024-01-19 10:08:10.000



Chatgpt 에게 전해줄 컬럼 통계정보

```
select table_name, column_name, num_distinct,  
       num_nulls, low_value, high_value, density,  
       num_buckets, last_analyzed  
from dba_tab_col_statistics  
where table_name in ( 'EMP' , 'DEPT' ) ;
```

TABLE_NAME	COLUMN_NAME	NUM_DISTINCT	NUM_NULLS	LOW_VALUE HIGH_VALUE DENSITY			NUM_BUCKETS	LAST_ANALYZED
				+	+	+		
DEPT	DEPTNO	4	0	Á	Á)	0.25	1	2024-01-19 11:35:57.000
DEPT	DNAME	4	0	ACCOUNTING	SALES	0.25	1	2024-01-19 11:35:57.000
DEPT	LOC	4	0	BOSTON	NEW YORK	0.25	1	2024-01-19 11:35:57.000
EMP	EMPNO	14	0	ÀJF	ÀP#	0.0714285714285714	1	2024-01-19 11:40:46.000
EMP	ENAME	14	0	ADAMS	WARD	0.0714285714285714	1	2024-01-19 11:40:46.000
EMP	JOB	5	0	ANALYST	SALESMAN	0.2	1	2024-01-19 11:40:46.000
EMP	MGR	6	1	ÀLC	ÀP	0.1666666666666667	1	2024-01-19 11:40:46.000
EMP	HIREDATE	13	0	Àw'	Àw·	0.0769230769230769	1	2024-01-19 11:40:46.000
EMP	SAL	12	0	À	À3	0.0833333333333333	1	2024-01-19 11:40:46.000
EMP	COMM	4	10	À	À	0.25	1	2024-01-19 11:40:46.000
EMP	DEPTNO	3	0	Á	Á	0.0357142857142857	3	2024-01-19 11:40:46.000



ChatGPT 를 통해서 얻어낼 최종 튜닝후 결과:



ChatGPT

기존의 SQL 쿼리에 성능 최적화를 위한 힌트를 추가하여, 데이터베이스의 쿼리 옵티마이저가 더 효율적인 실행 계획을 선택할 수 있도록 도울 수 있습니다. 다음은 수정된 SQL 쿼리입니다. 이 쿼리는 'DEPT' 테이블을 먼저 읽고, 그 다음에 'EMP' 테이블과 조인하는 것을 목표로 합니다.

sql

Copy code

```
SELECT /*+ LEADING(d e) USE_NL(e) */ e.ename, d.loc
FROM dept d, emp e
WHERE e.deptno = d.deptno;
```

여기서 사용된 힌트는 다음과 같습니다:

ChatGPT 강화학습 시키기 !



ChatGPT

기존의 SQL 쿼리에 성능 최적화를 위한 힌트를 추가하여, 데이터베이스의 쿼리 옵티마이저가 더 효율적인 실행 계획을 선택할 수 있도록 도울 수 있습니다. 다음은 수정된 SQL 쿼리입니다. 이 쿼리는 'DEPT' 테이블을 먼저 읽고, 그 다음에 'EMP' 테이블과 조인하는 것을 목표로 합니다.

sql

Copy code

```
SELECT /*+ LEADING(d e) USE_NL(e) */ e.ename, d.loc
FROM dept d, emp e
WHERE e.deptno = d.deptno;
```

여기서 사용된 힌트는 다음과 같습니다:



Provide additional feedback

너무 잘했어 ! 최고 !

X

Submit feedback



ChatGPT 를 이용하여



방법27. 조인 SQL문장을 전문가 처럼 튜닝하기2

▣ 학습 내용

1. 인덱스가 있는 테이블들의 nested loop 조인시 가장 좋은 조인 순서를 이해합니다.
2. 인덱스가 있는 테이블들의 조인튜닝을 하기 위해 ChatGPT 에게 더 좋은 질문을 하는 방법을 학습합니다.

▣ 학습 목표

ChatGPT 를 이용하여 인덱스가 존재하는 테이블의 조인 문장의 조인순서와 조인 방법을 결정할 수 있다.

emp 테이블이 driving 테이블이 된다면?

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

↑
index 생성 ↑
index 생성 ↑
index 생성 ↑
index 생성

emp_ename 인덱스

ENAME	ROWID
ADAMS	AAATifAAHAAA DL1AAM
ALLEN	AAATifAAHAAA DL1AAF
BLAKE	AAATifAAHAAA DL1AAB
CLARK	AAATifAAHAAA DL1AAC
FORD	AAATifAAHAAA DL1AAJ
JACK	AAATifAAHAAA DL1AAH
JAMES	AAATifAAHAAA DL1AAD
JONES	AAATifAAHAAA DL1AAA
MARTIN	AAATifAAHAAA DL1AAE
MILLER	AAATifAAHAAA DL1AAN
SCOTT	AAATifAAHAAA DL1AAL
SMITH	AAATifAAHAAA DL1AAK
TURNER	AAATifAAHAAA DL1AAG
WARD	AAATifAAHAAA DL1AAI

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	JACK	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_deptno 인덱스

DEPTNO	ROWID
10	AAAXzCAANAA ACz+AAA
20	AAAXzCAANAA ACz+AAB
30	AAAXzCAANAA ACz+AAC
40	AAAXzCAANAA ACz+AAD

dept 테이블

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

dept 테이블이 driving 테이블이 된다면?

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

↑
index 생성 ↑
index 생성 ↑
index 생성 ↑
index 생성

emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	ROWID
7839	JACK	PRESIDENT		1981-11-17	5000		10	AAAXzDAANAAAAC2eAAA
7698	BLAKE	MANAGER	7839	1981-05-01	2850		30	AAAXzDAANAAAAC2eAAB
7788	CLARK	MANAGER	7839	1981-05-09	2450		10	AAAXzDAANAAAAC2eAAC
7566	JONES	MANAGER	7839	1981-04-01	2975		20	AAAXzDAANAAAAC2eAAD
7654	MARTIN	SALESMAN	7698	1981-09-10	1250	1400	30	AAAXzDAANAAAAC2eAAE
7499	ALLEN	SALESMAN	7698	1981-02-11	1600	300	30	AAAXzDAANAAAAC2eAAF
7844	TURNER	SALESMAN	7698	1981-08-21	1500	0	30	AAAXzDAANAAAAC2eAAG
7900	JAMES	CLERK	7698	1981-12-11	950		30	AAAXzDAANAAAAC2eAAH
7521	WARD	SALESMAN	7698	1981-02-23	1250	500	30	AAAXzDAANAAAAC2eAAI
7902	FORD	ANALYST	7566	1981-12-11	3000		20	AAAXzDAANAAAAC2eAAJ
7369	SMITH	CLERK	7902	1980-12-09	800		20	AAAXzDAANAAAAC2eAAK
7788	SCOTT	ANALYST	7566	1982-12-22	3000		20	AAAXzDAANAAAAC2eAA
7876	ADAMS	CLERK	7788	1983-01-15	1100		20	AAAXzDAANAAAAC2eAM
7904	MILLER	CLERK	7782	1982-01-11	1300		10	AAAXzDAANAAAAC2eAA

emp_deptno 인덱스

ROWID	DEPTNO
AAAXzDAANAAAAC2eAAA	10
AAAXzDAANAAAAC2eAAC	10
AAAXzDAANAAAAC2eAAN	10
AAAXzDAANAAAAC2eAAD	20
AAAXzDAANAAAAC2eAAJ	20
AAAXzDAANAAAAC2eAAK	20
AAAXzDAANAAAAC2eAAL	20
AAAXzDAANAAAAC2eAAM	20
AAAXzDAANAAAAC2eAAB	30
AAAXzDAANAAAAC2eAAE	30
AAAXzDAANAAAAC2eAAF	30
AAAXzDAANAAAAC2eAAG	30
AAAXzDAANAAAAC2eAAH	30
AAAXzDAANAAAAC2eAAI	30

dept 테이블

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

dept_loc 인덱스

ROWID	LOC
AAAXzDAANAAAAC2eAAA	BOSTON
AAAXzDAANAAAAC2eAAC	CHICAGO
AAAXzDAANAAAAC2eAAN	DALLAS
AAAXzDAANAAAAC2eAAD	NEW YORK



ChatGPT에게 요청할 튜닝전 SQL :

```
select e.ename, e.sal, d.loc  
from emp e, dept d  
where e.deptno = d.deptno and e.ename='JACK' and d.loc='NEW YORK';
```

index 있음 index 있음 index 있음
emp 테이블

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	JACK	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	SAL	LOC
MARTIN	1250	CHICAGO
ALLEN	1600	CHICAGO
TURNER	1500	CHICAGO
WARD	1250	CHICAGO

튜닝전 :

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

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1		1	00:00:00.46	11543
1	NESTED LOOPS		1	43691	1	00:00:00.46	11543
2	NESTED LOOPS		1	611K	393K	00:00:00.14	771
3	TABLE ACCESS BY INDEX ROWID BATCHED	DEPT	1	1	1	00:00:00.01	2
* 4	INDEX RANGE SCAN	DEPT_LOC	1	1	1	00:00:00.01	1
* 5	INDEX RANGE SCAN	EMP_DEPTNO	1	611K	393K	00:00:00.08	769
* 6	TABLE ACCESS BY INDEX ROWID	EMP	393K	43691	1	00:00:00.23	10772

튜닝후 :

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

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1	1	1	00:00:00.01	6
1	NESTED LOOPS		1	43691	1	00:00:00.01	6
2	NESTED LOOPS		1	131K	1	00:00:00.01	5
3	TABLE ACCESS BY INDEX ROWID BATCHED	EMP	1	131K	1	00:00:00.01	4
* 4	INDEX RANGE SCAN	EMP_ENAME	1	131K	1	00:00:00.01	3
* 5	INDEX RANGE SCAN	DEPT_DEPTNO	1	1	1	00:00:00.01	1
* 6	TABLE ACCESS BY INDEX ROWID	DEPT	1	1	1	00:00:00.01	1



테이블 통계정보와 컬럼 통계정보 수집하기

begin

```
dbms_stats.gather_table_stats( user, 'emp') ;
dbms_stats.gather_table_stats( user, 'dept') ;
END ;
```



Chatgpt 에게 전해줄 테이블 통계정보

```
SELECT table_name, num_rows, avg_row_len  
FROM dba_tables  
WHERE table_name IN ('EMP', 'DEPT');
```



Chatgpt 에게 전해줄 컬럼 통계정보

```
select table_name, column_name, num_distinct,  
       num_nulls, low_value, high_value, density,  
       num_buckets  
  from dba_tab_col_statistics  
 where table_name in ( 'EMP' , 'DEPT' ) ;
```



Chatgpt 에게 전해줄 인덱스 정보

```
SELECT i.table_name, i.index_name, i.uniqueness,
       t.num_rows, i.leaf_blocks,
       i.clustering_factor, c.column_name, c.column_position
  FROM
    dba_indexes i,
    dba_ind_columns c,
    dba_tab_statistics t
 WHERE
    i.index_name = c.index_name
    AND i.table_owner = c.index_owner
    AND i.table_name = t.table_name
    AND i.table_owner = t.owner
    AND i.table_name in ('EMP', 'DEPT');
```



ChatGPT에게 제공해줘야하는 3가지 정보

1. 테이블 통계정보를 주어야 합니다.
2. 컬럼 통계정보를 주어야 합니다.
3. 인덱스 리스트를 주어야 합니다.



ChatGPT 를 이용하여



방법28. 조인 SQL문장을 전문가 처럼 튜닝하기 3

□ 학습 내용

1. 해쉬조인 SQL 문장 튜닝시 조인 순서의 중요성을 복습합니다.
2. ChatGPT를 활용하여 튜닝전 SQL의 실행계획을 분석합니다.
3. 해쉬조인 튜닝을 잘하기 위해서 ChatGPT 를 활용하는 팁을 배웁니다.

□ 학습 목표

ChatGPT 를 이용하여 해쉬조인을 튜닝할 수 있다.

아래의 SQL을 해석조인한다면 조인순서는 ?

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

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

dept 테이블이 해쉬 테이블이 된다면?

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	

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

buffer 19 개

dept 테이블

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

emp 테이블이 해쉬 테이블이 된다면?

buffer 10883 개

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

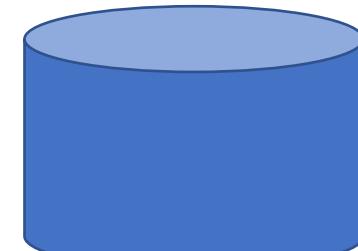
오라클 메모리 영역 PGA

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

PROB 테이블

dept 테이블



temp disk

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

튜닝전 :

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

<hr/>							
Id Operation	Name Starts E-Rows A-Rows A-Time Buffers Reads						
0 SELECT STATEMENT		1	200 00:00:00.20 10883 2865				
* 1 HASH JOIN		1 179K 200 00:00:00.20 10883 2865					
* 2 TABLE ACCESS FULL EMP		1 538K 524K 00:00:00.05 10878 0					
* 3 TABLE ACCESS FULL DEPT		1 1 1 00:00:00.01 4 0					

튜닝후 :

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

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	
0	SELECT STATEMENT		1		200	00:00:00.01	19		
* 1	HASH JOIN		1	179K	200	00:00:00.01	19	1476K	
* 2	TABLE ACCESS FULL	DEPT	1	1	1	00:00:00.01	7		
* 3	TABLE ACCESS FULL	EMP	1	538K	200	00:00:00.01	11		