

SQL 튜닝 목차

2018년 4월 20일 금요일 오전 11:58

- 오라클 접속하는방법xe -> orcl

- 1. SQL 튜닝(인덱스 튜닝)

- SQL 튜닝 이란?
- 인덱스 튜닝
 - Index range scan
 - Index unique scan
 - Index skip scan
 - Index full scan
 - Index fast full scan
 - Index merge scan
 - Index bitmap merge scan
 - Index join

- 2. SQL 튜닝(조인문장 튜닝)

- 조인 SQL 문법의 종류
 - 오라클 조인 문법
 - 1999 ANSI 문법
- 2-1. 조인을 실행하는 방법 3가지
 - 2-1. Nested loop join
 - 2-2. HASH 조인
 - 해쉬 조인의 원리
 - 해쉬 조인 사용시 힌트
 - 해쉬 테이블과 prob테이블
 - 2-3. sort merge join
 - outer join 문장 튜닝
 - hash join 시에 병렬 작업 하는방법

- 3. SQL 튜닝(서브쿼리 튜닝)

- 3-1. 옵티마이저 란?
- 3-2. 쿼리변환 이란?
- 3-3. subquery unnesting

- [1.unest](#)
- [2.no_unnest](#)
 - [2-1. push_subq](#)
 - [2-2. no_push_subq](#)
- [3-4. view merging](#)
 - [3-4-1. merge](#)
 - [3-4-2. no_merge](#)
- [3-5. push_pred, no_push_pred](#)
- [3-6. 스칼라 서브쿼리 문장의 튜닝](#)
- [3-7. 수정 가능한 조인 뷰](#)
- [3-8. 데이터 분석함수를 이용한 SQL 튜닝](#)
- [3-9. WITH 절을 이용한 튜닝](#)

오라클 접속하는방법xe -> orcl

2018년 4월 23일 월요일 오후 1:45

1. Sqlplus / as sysdba 접속 후

1. Select instance_name from v\$instance; 하면 orcl이 나와야한다.

```
SQL> select instance_name from v$instance;

INSTANCE_NAME
-----
orcl
```

3. 잠긴 scott 계정의 lock을 해제한다. (alter user scott account unlock;)

```
SQL> alter user scott account unlock;
```

4. 사용자가 변경되었습니다.

5. Scott의 패스워드를 tiger로 변경한다.(alter user scott identified by tiger;)

```
SQL> alter user scott identified by tiger;
```

6. 사용자가 변경되었습니다.

7. Scott 유저에게 dba 권한을 부여한다. (grant dba to scott;)

```
SQL> grant dba to scott;
```

8. 권한이 부여되었습니다.

리스너 상태확인

```
C:\Users\Witwill>lsnrctl status;
```

```
LSNRCTL for 64-bit Windows: Version 11.2.0.1.0 - Production on 23-4월 -2018 13:52:01
```

```
Copyright (c) 1991, 2010, Oracle. All rights reserved.
```

```
NL-00853: 정의되지 않은 "status;" 명령어입니다. "help" 를 참조하십시오
```

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

```
LSNRCTL for 64-bit Windows: Version 11.2.0.1.0 - Production on 23-4월 -2018 13:52:04
```

```
Copyright (c) 1991, 2010, Oracle. All rights reserved.
```

```
<DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)<KEY=EXTPROC1522>>>)에 연결되었습니다
리스너의 상태
```

```
-----
별칭          LISTENER
버전          TNSLSNR for 64-bit Windows: Version 11.2.0.1.0 - Produc
tion
시작 날짜      23-4월 -2018 11:27:47
업타임        0 일 2 시간. 24 분. 18 초
```

```

시작 날짜                23-4월 -2018 11:27:47
업타임                  0 일 2 시간. 24 분. 18 초
트레이스 수준           off
보안                    ON: Local OS Authentication
SNMP                    OFF리스너 매개변수 파일   D:\tt\product\11.2.0\ddhome_1
\tnetwork\admin\listener.ora
리스너 로그 파일        d:\tt\diag\tnlsnr\itwill-PC\listener\alert\log.xml
끝점 요약 청취 중...
  <DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)<PIPENAME=\\.\pipe\EXTPROC1522ipc>>>
  <DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)<HOST=127.0.0.1><PORT=1522>>>
서비스 요약...
"CLRExtProc" 서비스는 1개의 인스턴스를 가집니다.
  "CLRExtProc" 인스턴스<UNKNOWN 상태>는 이 서비스에 대해 1 처리기를 가집니다.
"orcl" 서비스는 1개의 인스턴스를 가집니다.
  "orcl" 인스턴스<READY 상태>는 이 서비스에 대해 1 처리기를 가집니다.
"orclXDB" 서비스는 1개의 인스턴스를 가집니다.
  "orcl" 인스턴스<READY 상태>는 이 서비스에 대해 1 처리기를 가집니다.
명령이 성공적으로 수행되었습니다

C:\Users\itwill>

```

오스트(H): localhost

사용자(U): scott

암호(P): tiger

☐ 암호 저장(W)

☒ 유니코드 사용(S)

연결 모드(M): Normal 포트(R): 1522

서비스 이름(A): ord

문자 집합(E):

xe에서 원래 데이터 가져오기

```

SQL> create public database link db_link9
2 connect to scott
3 identified by tiger
4 using '192.168.19.51:1521/xe';

```

데이터베이스 링크가 생성되었습니다.

1. SQL 튜닝(인덱스 튜닝)

2018년 4월 18일 수요일 오후 4:15

SQL 튜닝 이란?

- 쿼리의 검색속도를 높이기 위해서 반드시 알아야하는 기술

SQL 튜닝 목차

1. 인덱스 튜닝
2. 조인 문장 튜닝
3. 서브쿼리 문장 튜닝

1. 인덱스 튜닝

%% 인덱스 액세스 방법 8가지

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

1. Index range scan

: 인덱스를 부분 검색하는 스캔 방법

```
SELECT ename, sal  
FROM EMP  
WHERE job = 'SALESMAN';
```

인덱스

	JOB	ROWID
1	ANALYST	AAAFChAABAAALDBAAJ
2	ANALYST	AAAFChAABAAALDBAAL
3	CLERK	AAAFChAABAAALDBAAH
4	CLERK	AAAFChAABAAALDBAAK
5	CLERK	AAAFChAABAAALDBAAM
6	CLERK	AAAFChAABAAALDBAAN
7	MANAGER	AAAFChAABAAALDBAAB
8	MANAGER	AAAFChAABAAALDBAAC
9	MANAGER	AAAFChAABAAALDBAAD
10	PRESIDENT	AAAFChAABAAALDBAAA
11	SALESMAN	AAAFChAABAAALDBAAE
12	SALESMAN	AAAFChAABAAALDBAAF
13	SALESMAN	AAAFChAABAAALDBAAG
14	SALESMAN	AAAFChAABAAALDBAAI

테이블

ROWID	ENAME	SAL	JOB
AAAFChAABAAALDBAAA	KING	5000	PRESIDENT
AAAFChAABAAALDBAAB	BLAKE	2850	MANAGER
AAAFChAABAAALDBAAC	CLARK	2450	MANAGER
AAAFChAABAAALDBAAD	JONES	2975	MANAGER
AAAFChAABAAALDBAAE	MARTIN	1250	SALESMAN
AAAFChAABAAALDBAAF	ALLEN	1600	SALESMAN
AAAFChAABAAALDBAAG	TURNER	1500	SALESMAN
AAAFChAABAAALDBAAH	JAMES	950	CLERK
AAAFChAABAAALDBAAI	WARD	1250	SALESMAN
AAAFChAABAAALDBAAJ	FORD	3000	ANALYST
AAAFChAABAAALDBAAK	SMITH	800	CLERK
AAAFChAABAAALDBAAL	SCOTT	3000	ANALYST
AAAFChAABAAALDBAAM	ADAMS	1100	CLERK
AAAFChAABAAALDBAAN	MILLER	1300	CLERK

13	SALESMAN	AAAFChAABAAALDBAAG
14	SALESMAN	AAAFChAABAAALDBAAI

AAAFChAABAAALDBAAL	SCOTT	3000	ANALYST
AAAFChAABAAALDBAAM	ADAMS	1100	CLERK
AAAFChAABAAALDBAAN	MILLER	1300	CLERK

	ENAME	SAL
1	MARTIN	1250
2	ALLEN	1600
3	TURNER	1500
4	WARD	1250

문제1) 직업에 인덱스를 생성하고 직업의 인덱스의 구조가 어떻게 생겼는지 조회하시오.

`CREATE INDEX emp_job ON EMP(job);` <--인덱스 생성

`SELECT job, ROWID
FROM EMP
WHERE job > ' ';`

	JOB	ROWID
1	ANALYST	AAAFChAABAAALDBAAJ
2	ANALYST	AAAFChAABAAALDBAAL
3	CLERK	AAAFChAABAAALDBAAH
4	CLERK	AAAFChAABAAALDBAAK
5	CLERK	AAAFChAABAAALDBAAM
6	CLERK	AAAFChAABAAALDBAAN
7	MANAGER	AAAFChAABAAALDBAAB
8	MANAGER	AAAFChAABAAALDBAAC
9	MANAGER	AAAFChAABAAALDBAAD
10	PRESIDENT	AAAFChAABAAALDBAAA

문제2) 사원이름에 인덱스를 걸고 아래의 SQL이 어느 컬럼에 인덱스를 사용하는지 확인하시오.

`SELECT ename, sal, job
FROM EMP
WHERE job = 'SALESMAN' AND ename = 'ALLEN';`

@@ sqlplus에서 인덱스 보는방법

`Set autot traceonly explain` <-- 실행계획만 보겠다

`Set autot on` <-- 하면 계속 볼 수 있고, 블록 개수도 볼 수 있다.

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		1	13
1	TABLE ACCESS BY INDEX ROWID	EMP	1	13
2	INDEX RANGE SCAN	EMP_ENAME	1	

왜 job 컬럼에 인덱스를 타지 않고 ename 컬럼에 인덱스를 액세스 했을까?

	JOB	ROWID
1	ANALYST	AAAFChAABAAALDBAAJ
2	ANALYST	AAAFChAABAAALDBAAL
3	CLERK	AAAFChAABAAALDBAAH

	ENAME	ROWID
1	ADAMS	AAAFChAABAAALDBAAM
2	ALLEN	AAAFChAABAAALDBAAF
3	BLAKE	AAAFChAABAAALDBAAB

1	ANALYST	AAAFChAABAAALDBAAJ
2	ANALYST	AAAFChAABAAALDBAAL
3	CLERK	AAAFChAABAAALDBAAH
4	CLERK	AAAFChAABAAALDBAAK
5	CLERK	AAAFChAABAAALDBAAM
6	CLERK	AAAFChAABAAALDBAAN
7	MANAGER	AAAFChAABAAALDBAAB
8	MANAGER	AAAFChAABAAALDBAAC
9	MANAGER	AAAFChAABAAALDBAAD
10	PRESIDENT	AAAFChAABAAALDBAAA
11	SALESMAN	AAAFChAABAAALDBAAE
12	SALESMAN	AAAFChAABAAALDBAAF
13	SALESMAN	AAAFChAABAAALDBAAG
14	SALESMAN	AAAFChAABAAALDBAAI

인덱스_emp_job

```
SELECT job, rowid
FROM EMP
WHERE job > ' ';
```

4개에서 찾는거보다 1개에서 찾는게 더빨라서

1	ADAMS	AAAFChAABAAALDBAAM
2	ALLEN	AAAFChAABAAALDBAAF
3	BLAKE	AAAFChAABAAALDBAAB
4	CLARK	AAAFChAABAAALDBAAC
5	FORD	AAAFChAABAAALDBAAJ
6	JAMES	AAAFChAABAAALDBAAH
7	JONES	AAAFChAABAAALDBAAD
8	KING	AAAFChAABAAALDBAAA
9	MARTIN	AAAFChAABAAALDBAAE
10	MILLER	AAAFChAABAAALDBAAN
11	SCOTT	AAAFChAABAAALDBAAL
12	SMITH	AAAFChAABAAALDBAAK
13	TURNER	AAAFChAABAAALDBAAG
14	WARD	AAAFChAABAAALDBAAI

인덱스_emp_ename

```
SELECT ename, ROWID
FROM EMP
WHERE ename > ' ';
```

문제3) 만약 문제2번에서 emp_ename 인덱스를 액세스 하지 않고 emp_job 인덱스를 액세스 했을때 인덱스 검색 범위가 더 작은 emp_ename 인덱스를 타게 하려면?

```
SELECT /*+ index(emp emp_ename) */ ename, sal, job
FROM EMP
WHERE job = 'SALESMAN' AND ename = 'ALLEN';
```

--Emp_ename 인덱스를 액세스 하라고 힌트를 준다.

문제4) 81년도에 입사했고 월급이 3000인 사원의 이름과 입사일과 월급을 출력하는데 월급에도 인덱스를 걸고 입사일에도 인덱스를 걸어서 쿼리하시오.

```
CREATE INDEX emp_sal ON EMP(sal);
CREATE INDEX emp_hiredate ON EMP(hiredate);
```

```
SELECT ename, hiredate, sal
FROM EMP
WHERE hiredate BETWEEN TO_DATE('1981/01/01', 'RRRR/MM/DD')
AND To_date('1981/12/31', 'RRRR/MM/DD')
AND sal = '3000';
```

	ENAME	HIREDATE	SAL
1	FORD	1981-12-11 오전 12:00:00	3000

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
FILTER		
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	1
INDEX RANGE SCAN	SCOTT.EMP_HIREDATE	1

원래는 날짜보다 월급이 더 적어서 날짜로 인덱스를 타게되는데 컴퓨터가 덜 길들여서 그렇다.

문제5) 문제 4번에서 월급의 인덱스를 액세스 하지 못하고 인사일의 인덱스를 액세스
했다면 월급의 인덱스를 액세스 할 수 있도록 힌트를 주시오.

```
SELECT /*+ index(emp emp_sal) */ename, hiredate, sal
FROM EMP
WHERE hiredate BETWEEN TO_DATE('1981/01/01', 'RRRR/MM/DD')
AND to_date('1981/12/31', 'RRRR/MM/DD')
AND sal = '3000';
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
FILTER		
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	1
INDEX RANGE SCAN	SCOTT.EMP_SAL	1

문제6) 부서번호와 커미션에 각각 인덱스를 생성하고 부서번호가 30번이고 커미션이 300인
사원의 이름, 월급, 커미션, 부서번호를 출력하시오.

```
SELECT ename, sal, COMM
FROM EMP
WHERE deptno = '30' AND COMM = '300';
```

	ENAME	SAL	COMM
1	ALLEN	1600	300

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	1
INDEX RANGE SCAN	SCOTT.EMP_COMM	1

2. Index unique scan

: primary key 또는 unique 제약을 컬럼에
걸면자동으로 unique 인덱스가 생성되는데 이 인덱스를
엑세스하는 스캔방법

Ex) 사원 테이블에 primary key를 거시오.

```
ALTER TABLE EMP
ADD CONSTRAINT emp_empno_pk PRIMARY KEY(empno);
```

문제7) 사원 테이블에 걸린 인덱스 리스트를 조회하시오.

```
SELECT index_name, uniqueness
FROM user_indexes
WHERE table_name = 'EMP';
```

	INDEX_NAME	UNIQUENESS
1	EMP_JOB	NONUNIQUE
2	EMP_ENAME	NONUNIQUE
3	EMP_SAL	NONUNIQUE
4	EMP_HIREDATE	NONUNIQUE
5	EMP_DEPTNO	NONUNIQUE
6	EMP_COMM	NONUNIQUE
7	EMP_EMPNO_PK	UNIQUE

Nonunique 인덱스와 unique 인덱스 차이

문제8) 사원번호에 empno 에 인덱스의 구조를 확인하시오.

```
SELECT empno, ROWID
FROM EMP
WHERE empno >= 0;
```

EMPNO	ROWID
1	7369
2	7499
3	7521
4	7566
5	7654
6	7698
7	7782
8	7788
9	7839
10	7844
11	7876
12	7900
13	7902
14	7934

문제9) 사원번호가 7654인 사원의 사원번호와 이름을 조회하는데, 인덱스를 통해서 조회될 수 있도록 힌트를 주시오.

```
SELECT /*+ index (emp emp_empno_pk) */ empno, ename
FROM EMP
WHERE empno = '7654';
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	1
INDEX UNIQUE SCAN	SCOTT.EMP_EMPNO_PK	1

EMPNO	ROWID	인덱스	ROWID	EMPNO	ENAME	JOB	SAL
7369	AAAFChAABAAALDBAAK		1	7839	KING	PRESIDENT	5000
7499	AAAFChAABAAALDBAAF		2	7698	BLAKE	MANAGER	2850
7521	AAAFChAABAAALDBAAI		3	7782	CLARK	MANAGER	2450
7566	AAAFChAABAAALDBAAD		4	7566	JONES	MANAGER	2975
7654	AAAFChAABAAALDBAAE		5	7654	MARTIN	SALESMAN	1250
7698	AAAFChAABAAALDBAAB		6	7499	ALLEN	SALESMAN	1600
7782	AAAFChAABAAALDBAAC		7	7844	TURNER	SALESMAN	1500
7788	AAAFChAABAAALDBAAL		8	7900	JAMES	CLERK	950
7839	AAAFChAABAAALDBAAA		9	7521	WARD	SALESMAN	1250
7844	AAAFChAABAAALDBAAG		10	7902	FORD	ANALYST	3000
7876	AAAFChAABAAALDBAAM		11	7369	SMITH	CLERK	800
7900	AAAFChAABAAALDBAAH		12	7788	SCOTT	ANALYST	3000
7902	AAAFChAABAAALDBAAJ		13	7876	ADAMS	CLERK	1100
7934	AAAFChAABAAALDBAAN		14	7934	MILLER	CLERK	1300

Range스캔은 스캔해서 찾는데 unique스캔은 스캔하지않고 바로 가서 딱!! 찍음

문제11) 사원번호가 7654이고, 이름이 MARTIN인 사원의 사원번호와 이름과 월급을 조회하고, 사원번호의 인덱스를 액세스 했는지? 사원이름의 인덱스를 액세스 했는지? 확인하시오.

```
SELECT ename, sal
```

FROM EMP

WHERE empno = '7654' AND ename = 'MARTIN'; <- empno와 ename의 순서를 바꿔도 결과는 같다.

	ENAME	SAL
1	MARTIN	1250

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	1
INDEX UNIQUE SCAN	SCOTT.EMP_EMPNO_PK	1

==> Primary key 제약이 걸렸다는 것은 이미 데이터가 유니크 하다는게 보장이 된 상태이다.
따라서 유니크 인덱스를 액세스하고, 사원번호의 인덱스를 액세스한다.

문제12) 문제11번의 인덱스를 ename에 걸린 인덱스를 액세스를 하도록 힌트를 주고 실행하시오.

```
SELECT /*+ index(emp emp_ename) */ ename, sal
FROM EMP
WHERE empno = '7654' AND ename = 'MARTIN';
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	1

ENAME	ROWID
1	ADAMS
2	ALLEN
3	BLAKE
4	CLARK
5	FORD
6	JAMES
7	JONES
8	KING
9	MARTIN
10	MILLER
11	SCOTT
12	SMITH
13	TURNER
14	WARD

ROWID	EMPNO	ENAME	JOB
1	7839	KING	PRESIDENT
2	7698	BLAKE	MANAGER
3	7782	CLARK	MANAGER
4	7566	JONES	MANAGER
5	7654	MARTIN	SALESMAN
6	7499	ALLEN	SALESMAN
7	7844	TURNER	SALESMAN
8	7900	JAMES	CLERK
9	7521	WARD	SALESMAN
10	7902	FORD	ANALYST
11	7369	SMITH	CLERK
12	7788	SCOTT	ANALYST
13	7876	ADAMS	CLERK
14	7934	MILLER	CLERK

Range 스캔은 MARTIN을 찾고 아래 하나 더 읽고 찾으러 간다.

문제13) (점심시간문제) 직업이 SALESMAN인 사원들의 이름과 월급과 직업을 출력하는 쿼리의 실행 계획 그림을 인덱스와 테이블로 나눠서 그리시오. (주의사항 : index range scan 그림이어야함)

```
SELECT ename, sal, job
FROM EMP
WHERE job = 'SALESMAN';
```

ENAME	SAL	JOB
1	MARTIN	1250 SALESMAN
2	ALLEN	1600 SALESMAN
3	TURNER	1500 SALESMAN
4	WARD	1250 SALESMAN

3	TURNER	1500	SALESMAN
4	WARD	1250	SALESMAN

Operation	Object Name
SELECT STATEMENT Optimizer Mode=ALL_ROWS	
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP
INDEX RANGE SCAN	SCOTT.EMP_JOB

JOB	ROWID
1 ANALYST	AAAFChAABAAALDBAAJ
2 ANALYST	AAAFChAABAAALDBAAL
3 CLERK	AAAFChAABAAALDBAAH
4 CLERK	AAAFChAABAAALDBAAK
5 CLERK	AAAFChAABAAALDBAAM
6 CLERK	AAAFChAABAAALDBAAN
7 MANAGER	AAAFChAABAAALDBAAB
8 MANAGER	AAAFChAABAAALDBAAC
9 MANAGER	AAAFChAABAAALDBAAD
10 PRESIDENT	AAAFChAABAAALDBAAA
11 SALESMAN	AAAFChAABAAALDBAAE
12 SALESMAN	AAAFChAABAAALDBAAF
13 SALESMAN	AAAFChAABAAALDBAAG
14 SALESMAN	AAAFChAABAAALDBAAI

ROWID	EMPNO	ENAME	JOB
1 AAAFChAABAAALDBAAA	7839	KING	PRESIDENT
2 AAAFChAABAAALDBAAB	7698	BLAKE	MANAGER
3 AAAFChAABAAALDBAAC	7782	CLARK	MANAGER
4 AAAFChAABAAALDBAAD	7566	JONES	MANAGER
5 AAAFChAABAAALDBAAE	7654	MARTIN	SALESMAN
6 AAAFChAABAAALDBAAF	7499	ALLEN	SALESMAN
7 AAAFChAABAAALDBAAG	7844	TURNER	SALESMAN
8 AAAFChAABAAALDBAAH	7900	JAMES	CLERK
9 AAAFChAABAAALDBAAI	7521	WARD	SALESMAN
10 AAAFChAABAAALDBAAJ	7902	FORD	ANALYST
11 AAAFChAABAAALDBAAK	7369	SMITH	CLERK
12 AAAFChAABAAALDBAAL	7788	SCOTT	ANALYST
13 AAAFChAABAAALDBAAM	7876	ADAMS	CLERK
14 AAAFChAABAAALDBAAN	7934	MILLER	CLERK

3. Index skip scan :

인덱스를 전부 스캔하지 않고 스킵해서 스캔하는 액세스 방법

(결합컬럼을 이해하면 이해가 간다.)

문제14) demobld.sql 돌리고 아래와 같이 결합 컬럼 인덱스를 생성하시오.

```
CREATE INDEX emp_deptno_sal
ON EMP(deptno, sal);
```

문제15) 방금 만든 emp_deptno_sal의 인덱스의 구조가 어떻게 되었는지 확인하시오.

```
SELECT deptno, sal, ROWID
FROM EMP
```

```
WHERE deptno >= 0 AND sal >= 0
```

<-- and sal >= 0은 결합컬럼때문에 안써줘도됨

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
INDEX RANGE SCAN	SCOTT.EMP_DEPTNO_SAL	14

문제16) 부서번호가 20번인 직원들의 이름과 월급과 부서번호를 출력하는데 emp_deptno_sal 인덱스를 액세스 할 수 있도록 힌트를 주시오.

```
SELECT /*+ index(emp emp_deptno_sal) */ ename, sal, deptno, ROWID
FROM EMP
WHERE deptno >= 0;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	14
INDEX RANGE SCAN	SCOTT.EMP_DEPTNO_SAL	1

문제17) 월급이 3000인 사원의 이름과 월급을 출력하는데 실행계획에 emp_deptno_sal 인덱스를 액세스 하는지 확인하시오.(★★★★★)

```
SELECT ename, sal
FROM EMP
WHERE sal >= 3000;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		3
TABLE ACCESS FULL	SCOTT.EMP	3

왜 인덱스 emp_deptno_sal 인덱스를 액세스 하지 못하고 full table scan을 하는가?

- 결합 컬럼 인덱스의 첫번째 컬럼이 where 절에 검색 조건으로 존재해야 그 인덱스를 액세스 할 수 있다.

==> 이런경우에 emp_deptno_sal 인덱스를 액세스하려면 index skip scan을 사용해야한다.

%% Index skip scan 그림

```
Select /*+index_ss(emp emp_deptno_sal) */ ename, sal
from emp
where sal = 1100;
```

	DEPTNO	SAL	ROWID
1	10	1300	AAAFcQAAABAAALDBAAN
2	10	2450	AAAFcQAAABAAALDBAAC
3	10	5000	AAAFcQAAABAAALDBAAA
4	20	800	AAAFcQAAABAAALDBAAK
5	20	1100	AAAFcQAAABAAALDBAAM
6	20	2975	AAAFcQAAABAAALDBAAD
7	20	3000	AAAFcQAAABAAALDBAAJ
8	20	3000	AAAFcQAAABAAALDBAAL
9	30	950	AAAFcQAAABAAALDBAAH
10	30	1250	AAAFcQAAABAAALDBAAE
11	30	1250	AAAFcQAAABAAALDBAAI
12	30	1500	AAAFcQAAABAAALDBAAG
13	30	1600	AAAFcQAAABAAALDBAAF
14	30	2850	AAAFcQAAABAAALDBAAB

	ROWID	DEPTNO	EMPNO	ENAME	JOB
1	AAAFcQAAABAAALDBAAA	10	7839	KING	PRESIDENT
2	AAAFcQAAABAAALDBAAB	30	7698	BLAKE	MANAGER
3	AAAFcQAAABAAALDBAAC	10	7782	CLARK	MANAGER
4	AAAFcQAAABAAALDBAAD	20	7566	JONES	MANAGER
5	AAAFcQAAABAAALDBAAE	30	7654	MARTIN	SALESMAN
6	AAAFcQAAABAAALDBAAF	30	7499	ALLEN	SALESMAN
7	AAAFcQAAABAAALDBAAG	30	7844	TURNER	SALESMAN
8	AAAFcQAAABAAALDBAAH	30	7900	JAMES	CLERK
9	AAAFcQAAABAAALDBAAI	30	7521	WARD	SALESMAN
10	AAAFcQAAABAAALDBAAJ	20	7902	FORD	ANALYST
11	AAAFcQAAABAAALDBAAK	20	7369	SMITH	CLERK
12	AAAFcQAAABAAALDBAAL	20	7788	SCOTT	ANALYST
13	AAAFcQAAABAAALDBAAM	20	7876	ADAMS	CLERK
14	AAAFcQAAABAAALDBAAN	10	7934	MILLER	CLERK

sal 이 1100이 보이면 한칸 더내려가 스캔하고 스킵하여 나가서 찾으러감

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	1
INDEX SKIP SCAN	SCOTT.EMP_DEPTNO_SAL	1

문제18) 직업 + 월급으로 결합 컬럼 인덱스를 생성하고 아래의 SQL이 index skip scan 할 수 있도록 힌트를 주시오.

```
Select ename, sal, job, hiredate
from emp
where sal = 1250;
```

```
CREATE INDEX emp_job_sal
ON EMP(job, sal);
```

```
SELECT /*+ index_ss(emp emp_job_sal) */ ename, sal, job, hiredate
FROM EMP
WHERE sal = 1250;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		2
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	2
INDEX SKIP SCAN	SCOTT.EMP_JOB_SAL	1

%% 인덱스 스킵 스캔의 효과를 보기위한 조건

- 결합 컬럼 인덱스의 첫번째 컬럼의 데이터의 종류가 몇 가지 안되어야 효과를 볼 수 있다.

```
Select /*+ index_ss(emp emp_col1_col2) */ *
from emp
where col2 = 30;
```

Col 1	Col 2
A	10
A	20
A	30
A	40
A	50
A	60
B	10
B	20
B	30
B	40
B	50
B	60
C	10
C	20
C	30
C	40
C	50
C	60

Col 1의 종류가 몇 가지 안되어야 효과가 좋다.

(만약, A~C와 A~Z 랑 비교를 해보면 A~C가 더빠르다)

문제19) 사원 테이블의 직업의 종류가 몇 개인가?

```
SELECT COUNT(DISTINCT job)
FROM EMP;
```

COUNT(DISTINCT JOB)	
1	5

4. Index full scan

: 인덱스 전체를 처음부터 끝까지 스캔하는 스캔방법

문제20) 사원 테이블의 인원수가 몇 명인지 카운트 하시오.

```
SELECT COUNT(empno) FROM EMP;
```

```
SELECT COUNT(*) FROM EMP;
```

```
SELECT COUNT(1) FROM EMP;
```

3가지 방법이있다. 인덱스를 생성 안하고 하면 테이블 full scan으로 나온다.

문제21) 사원 테이블의 사원번호에 인덱스를 생성하고, 사원의 인원수가 몇 명 인지 카운트 하시오.

```
ALTER TABLE EMP
ADD constraint emp_empno_pk
PRIMARY KEY(empno);
```

```
SELECT COUNT(empno) FROM EMP;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
SORT AGGREGATE		1
INDEX FULL SCAN	SCOTT.EMP_EMPNO_PK	14

문제22) 문제21번을 다시 실행하는데 테이블 full scan으로 수행되게 하시오.

```
SELECT /*+ full(emp) */ COUNT(empno)
FROM EMP;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
SORT AGGREGATE		1
TABLE ACCESS FULL	SCOTT.EMP	14

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	3

문제23) 다시 index full scan 힌트를 주고 실행한 후에 몇 개의 블록을 읽었는지 확인하시오.

```
SELECT /*+ index_fs(emp emp_empno_pk) */ COUNT(empno)
FROM EMP;
```

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	1

5. Index fast full scan

: 인덱스 full 스캔보다 더 성능이 좋은 스캔 방법

Why? 병렬 처리가 가능하다.

문제24) 문제23번의 index full scan 실행계획을 index fast full scan이 되도록 힌트를 주시오.

```
SELECT /*+ index_ffs(emp emp_empno_pk) */ COUNT(empno)
FROM EMP;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
SORT AGGREGATE		1
INDEX FAST FULL SCAN	SCOTT.EMP_EMPNO_PK	14

만약 fast full scan이 안나오는 경우는 not null 제약을 걸어주거나 where is not null 을써준다.

문제25) 직업, 직업별 인원수를 출력하는데 아래의 인덱스를 이용해서 수행하시오.

```
CREATE INDEX emp_job_empno
ON EMP(job, empno);
```

```
SELECT /*+ index_ffs(emp emp_job_empno) *//job, COUNT(job)
FROM EMP
GROUP BY job;
```

Operation	Object Name	Rows	Bytes	Cost
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14	84	3
HASH GROUP BY		14	84	3
INDEX FAST FULL SCAN	SCOTT.EMP_JOB_EMPNO	14	84	2

문제26) 부서번호와 부서번호별 인원수를 출력하는데 가장 빠르게 출력될 수 있도록 적절한 인덱스를 생성하고, 힌트를 주고 수행하시오.

```
SELECT /*+ index_ffs(emp emp_deptno_empno) */ deptno, COUNT(deptno)
FROM EMP
GROUP BY deptno;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
HASH GROUP BY		14
INDEX FAST FULL SCAN	SCOTT.EMP_DEPTNO_EMPNO	14

문제27) 문제26번의 SQL을 병렬 처리 할 수 있도록 힌트를 주고 실행 하시오.

```
SELECT /*+ index_ffs(emp emp_deptno_empno)
parallel_index(emp, emp_deptno_deptno, 4) */ deptno, COUNT(deptno)
FROM EMP
GROUP BY deptno;
```

table index 병렬도

Optimizer 모드 : Default		<input checked="" type="checkbox"/> 실행 계획 개체 보기(O)	
트리 뷰 텍스트 순서도			
Operation	Object Name	Rows	Bytes Cost Object Node In/Out PStart PStop
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14	182 6
PX COORDINATOR			
PX SEND QC (RANDOM)	SYS.:TQ10001	14	182 6 :Q1001 P->S QC (RANDOM)
HASH GROUP BY		14	182 6 :Q1001 PCWP
PX RECEIVE		14	182 6 :Q1001 PCWP
PX SEND HASH	SYS.:TQ10000	14	182 6 :Q1000 P->P HASH
HASH GROUP BY		14	182 6 :Q1000 PCWP
PX BLOCK ITERATOR		14	182 2 :Q1000 PCWC
INDEX FAST FULL SCAN	SCOTT.EMP_DEPTNO_EMPNO	14	182 2 :Q1000 PCWP

문제28) 이름에 EN 또는 IN 을 포함하고 있는 사원들의 이름, 월급, 직업을 출력하시오.

CREATE INDEX emp_ename ON EMP(ename);

SELECT ename, sal, job

FROM EMP

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

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		3
TABLE ACCESS FULL	SCOTT.EMP	3

<-- like에 %가 앞에있으면
인덱스가있어도 테이블 full이다.

	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	41

	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	41

문제29) 아래의 SQL을 튜닝하시오.

SELECT ename, sal, job

FROM EMP

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

풀이)

이름에 EN 또는 IN 을 포함하고 있는 사원들의 데이터의 rowid를 출력하시오.

SELECT /*+ index_ffs(emp emp_ename) */ rowid

FROM EMP

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

	ROWID
1	AAAFcQAABAAALDBAAF
2	AAAFcQAABAAALDBAAA
3	AAAFcQAABAAALDBAAE

위에서 구한 rowid3건을 가지고 해당 rowid의 테이블의 이름과 월급과 직업을 출력하시오.

SELECT ename, sal, job

FROM emp

WHERE ROWID in(


```
SELECT /*+ index_ffs(emp emp_ename) */ rowid
FROM EMP
WHERE ename LIKE '%EN%' OR ename LIKE '%IN%'
);
```

문제30) 이름에 EN 또는 IN을 포함하고 있는 직원들의 이름, 월급, 직업을 출력하는데 정규식을 사용해서 출력하면 성능이 더 좋아지는지 확인해보시오.

```
SELECT /*+ index_ffs(emp emp_ename) */ ename, sal, job
FROM EMP
WHERE regexp_like(ename, '(EN|IN)');
```

6. Index merge scan

: where 절에 조건이 여러 개 가 있고 그 조건에 사용되는 컬럼이 다 단일 컬럼 인덱스로 구성되어 있을때 하나의 인덱스를 사용하는게 아니라 여러 개의 인덱스를 동시에 사용해서 시너지 효과를 보는 스캔 방법

문제31) 직업이 SALESMAN이고 부서번호가 30번인 직원들의 이름, 월급, 직업, 부서번호를 출력.

```
SELECT ename, sal, job, deptno
FROM EMP
WHERE job = 'SALESMAN' AND deptno = 30;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		4
TABLE ACCESS FULL	SCOTT.EMP	4

인덱스 스캔이 될 수 있도록 인덱스를 걸어본다.

문제32) 직업과 부서번호에 각각 단일 컬럼 인덱스를 생성하고 문제31번 쿼리의 실행계획을 보면 직업과 부서번호2개중에 어느 컬럼에 인덱스를 액세스 하겠는가?

```
CREATE INDEX emp_job ON emp(job);
CREATE INDEX emp_deptno ON EMP(deptno);
```

Operation	Object Name
SELECT STATEMENT Optimizer Mode=ALL_ROWS	
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP
INDEX RANGE SCAN	SCOTT.EMP_JOB

문제33) 그렇다면 왜 deptno의 인덱스를 타지 않고 job의 인덱스를 탔을까?

Emp_deptno와 emp_job 두개의 인덱스 중에서 index range scan을 더 짧게하는 인덱스를 선택하기 때문

```
SELECT COUNT(*) FROM EMP WHERE deptno = 30;
```

COUNT(*)	6
1	

컴퓨터가 이상하게 인덱스할때는
count를 해봐서 확인해서 힌트를 주면된다.

```
SELECT COUNT(*) FROM EMP WHERE job = 'SALESMAN';
```

COUNT(*)	4
1	

```
SELECT ename, sal, job, deptno
```

FROM EMP
WHERE deptno = 30 AND job = 'SALESMAN';

DEPTNO	ROWID	JOB	ROWID
1	10 AAAFC4AABAAALDBAAA	1 ANALYST	AAAF4AABAAALDBAAJ
2	10 AAAFC4AABAAALDBAAC	2 ANALYST	AAAF4AABAAALDBAAL
3	10 AAAFC4AABAAALDBAAN	3 CLERK	AAAF4AABAAALDBAAH
4	20 AAAFC4AABAAALDBAAD	4 CLERK	AAAF4AABAAALDBAAK
5	20 AAAFC4AABAAALDBAAJ	5 CLERK	AAAF4AABAAALDBAAM
6	20 AAAFC4AABAAALDBAAK	6 CLERK	AAAF4AABAAALDBAAN
7	20 AAAFC4AABAAALDBAAL	7 MANAGER	AAAF4AABAAALDBAAB
8	20 AAAFC4AABAAALDBAAM	8 MANAGER	AAAF4AABAAALDBAAC
9	30 AAAFC4AABAAALDBAAB	9 MANAGER	AAAF4AABAAALDBAAD
10	30 AAAFC4AABAAALDBAAE	10 PRESIDENT	AAAF4AABAAALDBAAA
11	30 AAAFC4AABAAALDBAAF	11 SALESMAN	AAAF4AABAAALDBAAE
12	30 AAAFC4AABAAALDBAAG	12 SALESMAN	AAAF4AABAAALDBAAF
13	30 AAAFC4AABAAALDBAAH	13 SALESMAN	AAAF4AABAAALDBAAG
14	30 AAAFC4AABAAALDBAAI	14 SALESMAN	AAAF4AABAAALDBAAI

문제34) 그러면 emp_deptno 인덱스를 액세스 하게끔 힌트를 주고 실행해보시오.

```
SELECT /*+ index (emp emp_deptno) */ ename, sal, job, deptno
FROM EMP
WHERE job = 'SALESMAN' AND deptno = 30;
```

Operation	Object Name	Rows	자동 추적
SELECT STATEMENT Optimizer Mode=ALL_ROWS		4	속성
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	4	값
INDEX RANGE SCAN	SCOTT.EMP_DEPTNO	1	1 recursive calls 0
			2 db block gets 0
			3 consistent gets 40

(merge scan)

문제35) 위의 SQL이 index merge scan이 될 수 있도록 힌트를 주시오.

```
SELECT /*+ and_equal (emp emp_job emp_deptno) */ ename, sal, job, deptno
FROM EMP
WHERE job = 'SALESMAN' AND deptno = 30;
```

Operation	Object Name	Rows	자동 추적
SELECT STATEMENT Optimizer Mode=ALL_ROWS		4	속성
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	4	값
AND-EQUAL			1 recursive calls 0
INDEX RANGE SCAN	SCOTT.EMP_DEPTNO	1	2 db block gets 0
INDEX RANGE SCAN	SCOTT.EMP_JOB	1	3 consistent gets 43

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		4
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	4
AND-EQUAL		
INDEX RANGE SCAN	SCOTT.EMP_DEPTNO	1
INDEX RANGE SCAN	SCOTT.EMP_JOB	1

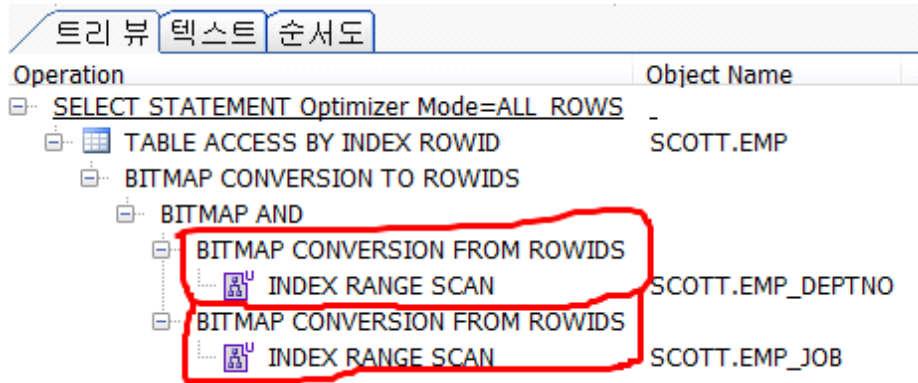
==> 두개의 인덱스를 활용하는거라 블록의 개수가 늘어나 좋은 튜닝은 아니다
그래서 요즘은 index bitmap merge scan 을 쓴다

7. Index bitmap merge scan

: index merge scan의 진화된 기술로 여러 개의 인덱스를 각각 bitmap으로 변환해서 하나로 합쳐서 구한 rowid로 테이블을 액세스 하는 스캔방법.

Ex)

```
SELECT /*+ index_combine(emp) */ ename, sal, job, deptno
FROM EMP
WHERE job = 'SALESMAN' AND deptno = 30;
```



문제36) 우리반 테이블에 전공과 통신사에 각각 단일 컬럼 인덱스를 걸고, 전공이 경제학이고, 통신사가 sk인 학생들의 이름, 전공, 통신사를 출력하는 쿼리의 실행계획이 index bitmap merge scan이 되게하시오.

```
SELECT /*index_combine(emp) */ ename, major, telecom
FROM EMP2
WHERE major = '경제학' AND telecom = 'sk';
```

8. Index join

: 테이블을 액세스 하지 않으면서 인덱스만 가지고 조인을 해서 결과를 보여주는 스캔 방법

Ex) demobld 파일을 돌리고 사원번호와 이름에 각각 단일 컬럼 인덱스를 생성하고 아래의 실행계획을 확인해본다.

```
SELECT empno, ename
FROM EMP
WHERE empno = 7788;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	1
INDEX RANGE SCAN	SCOTT.EMP_EMPNO	1

문제37) 아래의 SQL을 index join 액세스 방법으로 힌트를 줘서 테이블 액세스를 하지 말게 하시오.

```
SELECT /*+ index_join(emp) */ empno, ename
FROM EMP
WHERE empno = 7788;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	1
INDEX RANGE SCAN	SCOTT.EMP_EMPNO	1

Not null 제약이 없어서 안변한다.

문제38) 문제38번 SQL 쿼리를 not null 제약을 걸고 확인해보시오.

```
ALTER TABLE EMP
  MODIFY empno CONSTRAINT emp_empno_nn NOT NULL;
```

```
ALTER TABLE EMP
  MODIFY ename CONSTRAINT emp_ename_nn NOT NULL;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
VIEW	SCOTT.index\$_join\$_001	1
HASH JOIN		
INDEX RANGE SCAN	SCOTT.EMP_EMPNO	1
INDEX FAST FULL SCAN	SCOTT.EMP_ENAME	1

문제39) 전공이 통계학이고, 나이가 20대인 학생의 학생번호, 이름, 전공, 나이를 출력하는 쿼리를 작성하는데 쿼리의 성능이 높아지도록 적절하게 인덱스를 생성하고, 인덱스를 잘 탈 수 있도록 힌트를 사용해서 SQL을 작성하시오.

```
SELECT /*+ index(emp2 emp2_major) */ empno, ename, major, age
FROM EMP2
```

```
WHERE major = '통계학' AND age BETWEEN 20 AND 29;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		2
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP2	2
INDEX RANGE SCAN	SCOTT.EMP2_MAJOR	1

문제40) 아래의 SQL을 튜닝하시오.

```
Create index emp2_ename on emp2(ename);
```

```
Select ename, age, major
From emp2
Where substr(ename,1,1) = 김;
```

```
-----
SELECT ename, age, major
FROM EMP2
WHERE ename like '김%';
```

문제41) 아래의 SQL을 튜닝하시오.

```
Create index emp2_birth on emp2(birth);
```

```
Select ename, age, major
From emp2
Where to_char(birth, 'RR/MM/DD') = '92/05/25';
```

```
-----
SELECT ename, age, major
FROM EMP2
WHERE birth = TO_DATE('92/05/25', 'RR/MM/DD');
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP2	1
INDEX RANGE SCAN	SCOTT.EMP2_BIRTH	1

2. SQL 튜닝(조인문장 튜닝)

2018년 4월 20일 금요일 오후 1:52

조인 SQL 문법의 종류

1. 오라클 조인 문법

- equi join
- non equi join
- outer join
- self join

2. 1999 ANSI 문법

- on 절을 사용한 조인
- using 절을 사용한 조인
- left/right/full outer 조인
- natural 조인
- cross 조인

2-1. 조인을 실행하는 방법 3가지

1. Nested loop join
2. Hash join
3. Sort merge join

문제43) 이름, 월급, 부서위치를 출력하시오.

```
SELECT e.ename, e.sal, d.loc  
FROM EMP e, DEPT d  
WHERE e.DEPTNO = d.DEPTNO;
```

<input type="checkbox"/>	ENAME	SAL	LOC
1	KING	5000	NEW YORK
2	BLAKE	2850	CHICAGO
3	CLARK	2450	NEW YORK
4	JONES	2975	DALLAS
5	MARTIN	1250	CHICAGO
6	ALLEN	1600	CHICAGO
7	TURNER	1500	CHICAGO
8	JAMES	950	CHICAGO
9	WARD	1250	CHICAGO
10	FORD	3000	DALLAS
11	SMITH	800	DALLAS
12	SCOTT	3000	DALLAS
13	ADAMS	1100	DALLAS
14	MILLER	1300	NEW YORK

문제44) 문제43번의 SQL 조인 순서가 어떻게 되는가?

1. Emp --> dept
2. Dept ----> emp

Operation	Object Name	Rows
4 SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
3 HASH JOIN		14
1 TABLE ACCESS FULL	SCOTT.DEPT	4
2 TABLE ACCESS FULL	SCOTT.EMP	14

Why ?

```
SELECT e.ename, e.sal, d.loc
FROM EMP e, DEPT d
WHERE e.DEPTNO = d.DEPTNO;
```

ENAME	SAL	JOB	DEPTNO	DEPTNO	DNAME	LOC
1 KING	5000	PRESIDENT	10	1	10 ACCOUNTING	NEW YORK
2 BLAKE	2850	MANAGER	30	2	20 RESEARCH	DALLAS
3 CLARK	2450	MANAGER	10	3	30 SALES	CHICAGO
4 JONES	2975	MANAGER	20	4	40 OPERATIONS	BOSTON
5 MARTIN	1250	SALESMAN	30			
6 ALLEN	1600	SALESMAN	30			
7 TURNER	1500	SALESMAN	30			
8 JAMES	950	CLERK	30			
9 WARD	1250	SALESMAN	30			
10 FORD	3000	ANALYST	20			
11 SMITH	800	CLERK	20			
12 SCOTT	3000	ANALYST	20			
13 ADAMS	1100	CLERK	20			
14 MILLER	1300	CLERK	10			

조인 시도를 14번했다.

반면에

LOC	DNAME	DEPTNO	DEPTNO	ENAME	SAL	JOB
1 NEW YORK	ACCOUNTING	10	1	10 KING	5000	PRESIDENT
2 DALLAS	RESEARCH	20	2	30 BLAKE	2850	MANAGER
3 CHICAGO	SALES	30	3	10 CLARK	2450	MANAGER
4 BOSTON	OPERATIONS	40	4	20 JONES	2975	MANAGER
			5	30 MARTIN	1250	SALESMAN
			6	30 ALLEN	1600	SALESMAN
			7	30 TURNER	1500	SALESMAN
			8	30 JAMES	950	CLERK
			9	30 WARD	1250	SALESMAN
			10	20 FORD	3000	ANALYST
			11	20 SMITH	800	CLERK
			12	20 SCOTT	3000	ANALYST
			13	20 ADAMS	1100	CLERK
			14	10 MILLER	1300	CLERK

Dept는 조인시도를 4번만한다.

★★★★★★★★★★

%조인 튜닝시 가장 중요한 2가지

1. 조인 순서

1-1) ordered : from 절에서 기술한 테이블 순서대로 조인 해라.

1-2) leading : leading 힌트 안에쓴 테이블 순서대로 조인 해라.

2. 조인 방법

- 2-1) use_nl : nested loop 조인으로 유도해라.
- 2-2) use_hash : hash 조인으로 유도해라.
- 2-3) use_merge : sort merge 조인으로 조인 해라

1) ordered

문제45) ordered 힌트를 이용해서 아래 SQL의 조인 순서를 emp -> dept 순으로 조인되게 하시오.

```
SELECT e.ename, e.sal, d.loc
FROM EMP e, DEPT d
WHERE e.DEPTNO = d.DEPTNO;
```

```
SELECT /*+ ordered */ e.ename, e.sal, d.loc
FROM EMP e, DEPT d
WHERE e.DEPTNO = d.DEPTNO;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
HASH JOIN		14
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.DEPT	4

문제46) 아래의 SQL을 아래의 방법으로 조인하시오.

```
SELECT e.ename, e.sal, d.loc
FROM EMP e, DEPT d
WHERE e.DEPTNO = d.DEPTNO;
```

조인순서 : emp --> dept

조인방법 : nested loop join

```
SELECT /*+ ordered use_nl(d) */ e.ename, e.sal, d.loc
FROM EMP e, DEPT d
WHERE e.DEPTNO = d.DEPTNO;
```

Operation	Object Name
SELECT STATEMENT Optimizer Mode=ALL_ROWS	
NESTED LOOPS	
TABLE ACCESS FULL	SCOTT.EMP
TABLE ACCESS FULL	SCOTT.DEPT

자동 추적	
속성	값
1 recursive calls	0
2 db block gets	0
3 consistent gets	45

문제47) 아래의 조인 SQL을 튜닝하시오.

```
SELECT e.ename, e.sal, d.loc
FROM EMP e, DEPT d
WHERE e.DEPTNO = d.DEPTNO;
```

```
SELECT /*+ ordered use_nl(e) */ e.ename, e.sal, d.loc
FROM dept d, emp e
WHERE e.DEPTNO = d.DEPTNO;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
NESTED LOOPS		14
TABLE ACCESS FULL	SCOTT.DEPT	4
TABLE ACCESS FULL	SCOTT.EMP	4

자동 추적	
속성	값
1 recursive calls	0
2 db block gets	0
3 consistent gets	15

문제48) 아래의 조인 SQL에 적절한 힌트를 주시오.

```
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';
```

```
-----
SELECT /*+ ordered use_nl(e)*/ e.ename, e.sal, d.loc
FROM DEPT d, EMP e
WHERE e.DEPTNO = d.DEPTNO
      and e.job = 'SALESMAN'
      and d.loc = 'CHICAGO';
```

=> SALESMAN 카운트가 4, CHICAGO 카운트가 1이라 CHICAGO먼저 쓰게한다.

답2)

1-2)leading

```
SELECT /*+ leading(d e) use_nl(e)*/ e.ename, e.sal, d.loc
FROM DEPT d, EMP e
WHERE e.DEPTNO = d.DEPTNO
      and e.job = 'SALESMAN'
      and d.loc = 'CHICAGO';
```

< leading 을 쓰면 from 절 순서가 상관 x

2-1. Nested loop join

: 두개의 테이블을 조인할 때 한건 한건씩 순차적으로 반복해서 조인하는 조인 방법.

문제49) sk텔레콤 통신사인 학생들의 학생 이름, 주소, 나이, 통신사, 통신사 월정액을 출력하시오.

(emp2와 telecom_price 조인)

```
SELECT /*+ leading(t e) use_nl(e)*/ e2.ename, e2.address, e2.age, e2.telecom, tp.t_price
FROM EMP2 e2, TELECOM_PRICE tp
WHERE E2.TELECOM = tp.TELECOM
      AND tp.telecom = 'sk';
```

<--e2에서 가져오면 8개를 조인하는데 tp에서 가져오면 1개 만하면된다.

%%Price 테이블 예제 만들기(대용량 데이터)

```
SELECT COUNT(*) FROM PRICE;
```

```
CREATE TABLE market_code
AS
SELECT DISTINCT m_type_code, m_type_name
FROM PRICE;
```

```
SELECT COUNT(*) FROM market_code;
```

```
select * FROM market_code;
```

```
CREATE TABLE gu_code
AS
SELECT DISTINCT m_gu_code, m_gu_name
FROM PRICE;
```

```
SELECT COUNT(*) FROM gu_code;
```

```
CREATE TABLE price2
```

```

AS
SELECT * FROM PRICE;

ALTER TABLE PRICE
DROP COLUMN m_type_name;

ALTER TABLE PRICE
DROP COLUMN m_gu_name;
-----

```

문제50) price 테이블과 market_code와 조인을 해서 a_name, a_price, m_type_name을 출력하시오.

```

SELECT /*+ leading(p mc) use_nl(mc) */ p.a_name, p.a_price, mc.m_type_name
FROM PRICE p, market_code mc
WHERE p.m_type_code = mc.M_TYPE_CODE;

```

P를 먼저 읽으면

Operation	Object Name	자동 추적
SELECT STATEMENT Optimizer Mode=ALL_ROWS		
HASH JOIN		
TABLE ACCESS FULL	SCOTT.MARKET_CODE	
TABLE ACCESS FULL	SCOTT.PRICE	

	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	234

그러나 mc를 먼저 읽게하면

```

SELECT /*+ leading(mc p) use_nl(p) */ p.a_name, p.a_price, mc.m_type_name
FROM PRICE p, market_code mc
WHERE p.m_type_code = mc.M_TYPE_CODE;

```

Operation	Object Name	자동 추적
SELECT STATEMENT Optimizer Mode=ALL_ROWS		
NESTED LOOPS		
TABLE ACCESS FULL	SCOTT.MARKET_CODE	
TABLE ACCESS FULL	SCOTT.PRICE	

	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	10
4	physical reads	0

문제51) 이름이 SCOTT인 사원의 이름과 월급과 부서위치를 출력하는데 적절한 조인 순서와 조인방법을 힌트를 주어서 작성하시오.

```

SELECT /*+ leading(e d) use_nl(d) */ e.ename, e.sal, d.loc
FROM EMP e, DEPT d
WHERE e.DEPTNO = d.DEPTNO
AND ename = 'SCOTT';

```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
NESTED LOOPS		1
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	1
INDEX RANGE SCAN	SCOTT.EMP_ENAME	1
TABLE ACCESS FULL	SCOTT.DEPT	1

문제52. price 테이블과 gu_code와 조인을 해서 a_name, a_price, m_gu_name을 출력하는데 적절한 조인 힌트를 줘서 작성하시오.

```

SELECT /*+ leading(g p) use_nl(p) */ a_name, a_price, m_gu_name
FROM price p, GU_CODE g
WHERE p.m_gu_code = g.M_GU_CODE;

```

문제53. price 테이블과 gu_code와 조인을 해서 a_name, a_price, m_gu_name을 출력하는데 이마트 역삼점만 출력하고 적절한 조인 순서와 조인방법을 힌트로 기술하시오.

```

SELECT COUNT(*) FROM PRICE WHERE m_name='이마트 역삼점';

```

COUNT(*)	
1	96

SELECT COUNT(*) FROM GU_CODE;

COUNT(*)	
1	25

==> gu_code가 더 적어서 먼저 써야 한다.

```
SELECT /*+ leading(g p) use_nl(p) */ a_name, a_price, m_gu_name
FROM price p, GU_CODE g
WHERE p.m_gu_code = g.M_GU_CODE
AND p.m_name = '이마트 역삼점';
```

문제54. 이름, 월급, 부서위치, 급여등급(grade)을 출력하시오.

```
SELECT /*+ leading(d s e) use_nl(s) use_nl(e) */ e.ename, e.sal, 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;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
NESTED LOOPS		1
NESTED LOOPS		20
TABLE ACCESS FULL	SCOTT.DEPT	4
TABLE ACCESS FULL	SCOTT.SALGRADE	5
TABLE ACCESS FULL	SCOTT.EMP	1

문제55. price, gu_code, market_code 3개의 테이블을 조인해서 a_name, a_price, m_gu_name, m_type_name을 출력하는데 적절한 조인순서와 힌트를 주고 실행하시오.

```
SELECT /*+ leading(m p g) use_nl(p) use_nl(g) */ a_name, a_price, m_gu_name, m_type_name
FROM PRICE p, GU_CODE g, MARKET_CODE m
WHERE p.m_gu_code = g.M_GU_CODE
AND p.m_type_CODE = m.m_type_code;
```

Operation	Object Name
SELECT STATEMENT Optimizer Mode=ALL_ROWS	
NESTED LOOPS	
NESTED LOOPS	
TABLE ACCESS FULL	SCOTT.MARKET_CODE
TABLE ACCESS FULL	SCOTT.PRICE
TABLE ACCESS FULL	SCOTT.GU_CODE

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	288

문제56. 급여등급이 2등급인 직원들의 이름, 월급, 부서위치, 급여등급을 출력하는데 적절한 조인 순서와 조인 방법을 써서 구현하시오.(가장 적은 블록수)

```
SELECT /*+ leading(s e d) use_nl(e) use_nl(d) */ e.ename, e.sal, 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 = 2;
```

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	15

2-2. HASH 조인

: (상대방의 SQL을 느리게해서 가급적 사용하면 안된다.)

Hash 조인하기 전에 테이블 셋팅 방법

```
CREATE TABLE sales200
AS
SELECT * FROM sh.SALES;
```

```
CREATE TABLE times200
AS
SELECT * FROM sh.TIMES;
```

문제57. 아래의 SQL을 튜닝하시오.

튜닝전

```
Select /*+ leading(s t) use_nl(t) */
      t.calendar_year, sum(s.amount_sold)
from sales200 s, times200 t
WHERE s.time_id = t.times_id
      AND t.week_ending_day_id = 1581
GROUP BY t.calendar_year;
```

```
SELECT COUNT(*) FROM sales200;
```

COUNT(*)	
1	918843

```
SELECT COUNT(*) FROM times200 WHERE week_ending_day_id = 1581;
```

COUNT(*)	
1	918843

```
Select /*+ leading(t s) use_nl(s) */
      t.calendar_year, sum(s.amount_sold)
from sales200 s, times200 t
WHERE s.time_id = t.time_id
      AND t.week_ending_day_id = 1581
GROUP BY t.calendar_year;
```

```
CREATE TABLE customers200
AS
SELECT * FROM sh.CUSTOMERS;
```

```
SELECT COUNT(*) FROM customers200;
```

COUNT(*)	
1	55500

문제59. 아래의 SQL을 튜닝하시오.

튜닝전

```
SELECT /*+ leading(s c) use_nl(c) */ COUNT(*)
  FROM sales200 s, customers200 c
 WHERE s.cust_id = c.cust_id
    AND c.COUNTRY_ID = 52790
    AND s.TIME_ID BETWEEN TO_DATE('1999/01/01','RRRR/MM/DD')
                        AND TO_DATE('1999/12/31','RRRR/MM/DD');
```

```
SELECT /*+ leading(c s) use_nl(s) */ COUNT(*)
  FROM sales200 s, customers200 c
 WHERE s.cust_id = c.cust_id
    AND c.COUNTRY_ID = 52790
    AND s.TIME_ID BETWEEN TO_DATE('1999/01/01','RRRR/MM/DD')
                        AND TO_DATE('1999/12/31','RRRR/MM/DD');
```

==튜닝을 해도 오래걸린다. 그래서 hash 조인을 써주면된다.

```
SELECT /*+ leading(c s) use_hash(s) */ COUNT(*)
  FROM sales200 s, customers200 c
 WHERE s.cust_id = c.cust_id
    AND c.COUNTRY_ID = 52790
    AND s.TIME_ID BETWEEN TO_DATE('1999/01/01','RRRR/MM/DD')
                        AND TO_DATE('1999/12/31','RRRR/MM/DD');
```

문제60. 아래의 SQL을 nested loop 조인으로 수행하면서 좋은 성능을 보이도록 인덱스를 생성하시오.

튜닝전

```
SELECT /*+ leading(c s) use_nl(s) */ COUNT(*)
  FROM sales200 s, customers200 c
 WHERE s.cust_id = c.cust_id
    AND c.COUNTRY_ID = 52790
    AND s.TIME_ID BETWEEN TO_DATE('1999/01/01','RRRR/MM/DD')
                        AND TO_DATE('1999/12/31','RRRR/MM/DD');
```

```
CREATE INDEX sales200_cust_id ON sales200(cust_id);
CREATE INDEX customers200_cust_id ON customers200(cust_id);
```

```
SELECT /*+ leading(c s) use_nl(s) */ COUNT(*)
  FROM sales200 s, customers200 c
 WHERE s.cust_id = c.cust_id
    AND c.COUNTRY_ID = 52790
    AND s.TIME_ID BETWEEN TO_DATE('1999/01/01','RRRR/MM/DD')
                        AND TO_DATE('1999/12/31','RRRR/MM/DD');
```

Operation	Object Name	Rows
8 SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
7 SORT AGGREGATE		1
6 FILTER		
5 NESTED LOOPS		
3 NESTED LOOPS		253 K
1 TABLE ACCESS FULL	SCOTT.CUSTOMERS200	21 K
2 INDEX RANGE SCAN	SCOTT.SALES200_CUST_ID	202
4 TABLE ACCESS BY INDEX ROWID	SCOTT.SALES200	12

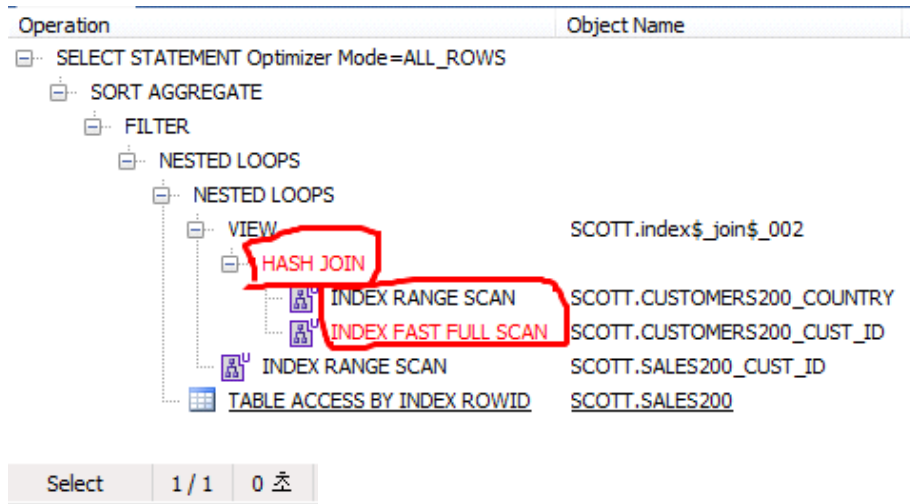
Select	1 / 1	1 초
--------	-------	-----

<---- 1초 걸림

더 빨라지는 방법

Country_id 에 인덱스를 걸고 하면 더 빨라 질 수 있다.

```
CREATE INDEX customers200_country ON customers200(country_id);
```



Index끼리 조인한다. 여기서 해시조인이 있는데 이 정도는 양이 작아서 괜찮다.

문제61. 아래의 SQL을 튜닝하시오.

```
Select /*+ leading(t s) use_nl(s) */
      t.calendar_year, sum(s.amount_sold)
from sales200 s, time200 t
WHERE s.time_id = t.times_id
      AND t.week_ending_day_id = 1581
GROUP BY t.calendar_year;
```

	자동 추적	
	속성	값
1	recursive calls	792
2	db block gets	0
3	consistent gets	31730

```
CREATE INDEX sales200_time_id ON sales200(time_id);
CREATE INDEX times200_time_id ON times200(time_id);
CREATE INDEX times200_week_ending_day_id ON times200(week_ending_day_id);
```

인덱스를 3개 생성해주면

	자동 추적	
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	286

많이 줄어든다.

문제62. 테이블 생성후 아래 SQL을 튜닝하시오. (nl 조인으로)

```
CREATE TABLE products200
AS
SELECT * FROM sh.PRODUCTS;
```

튜닝전

```
SELECT /*+ leading(s t p) use_nl(t) use_nl(p) */ p.prod_name, t.calendar_year, SUM(s.amount_sold)
FROM sales200 s, times200 t, products200 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;
```

자동 추적		
	속성	값
1	recursive calls	10
2	db block gets	0
3	consistent gets	1996995

```

CREATE INDEX sales200_time_id ON sales200(time_id);
CREATE INDEX times200_time_id ON times200(time_id);
CREATE INDEX products200_prod_id ON products200(prod_id);
CREATE INDEX sales200_prod_id ON sales200(prod_id);
CREATE INDEX products200_prod_name ON products200(prod_name);
CREATE INDEX times200_calendar_year ON times200(calendar_year);

```

```

SELECT /*+ leading(p s t) use_nl(s) use_nl(t) */ p.prod_name, t.calendar_year, SUM(s.amount_sold)
  FROM sales200 s, times200 t, products200 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;

```

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	439

문제63. 위의 SQL을 HASH조인으로 수행하시오.

```

SELECT /*+ leading(p s t) use_hash(s) use_hash(t) */ p.prod_name, t.calendar_year, SUM(s.amount_sold)
  FROM sales200 s, times200 t, products200 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;

```

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	4471

해쉬 조인의 원리

```

Select /*+ leading(d e) use_hash(e) */ e.ename, d.loc
  From emp e, dept d
 Where e.deptno = d.deptno

```

==> 해쉬 조인을 하게 되면 dept 테이블을 메모리에 올려놓고 메모리에 있는 dept 테이블과 emp와 조인을 시도한다.(작은 테이블이거나 where 절에서 작은 건수를 메모리에 올려 놓는다.)

ENAME	DEPTNO
1 SMITH	20
2 ALLEN	30
3 WARD	30
4 JONES	20
5 MARTIN	30
6 BLAKE	30
7 CLARK	10
8 SCOTT	20
9 KING	10
10 TURNER	30
11 ADAMS	20
12 JAMES	30
13 FORD	20
14 MILLER	10

해시함수

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

데이터베이스

문제64. 아래의 SQL을 hash 조인으로 수행하는데, itmes 테이블을 해쉬 테이블로 구성하는 것과 sales 테이블을 해쉬 테이블로 구성하는 것과의 성능차이를 확인 하시오.

```
Select /*+ leading(t s) use_nl(s) */
      t.calendar_year, sum(s.amount_sold)
  from sales200 s, times200 t
 WHERE s.time_id = t.time_id
    AND t.week_ending_day_id = 1581
 GROUP BY t.calendar_year;
```

```
Select /*+ leading(t s) use_hash(s) */
      t.calendar_year, sum(s.amount_sold)
  from sales200 s, times200 t
 WHERE s.time_id = t.time_id
    AND t.week_ending_day_id = 1581
 GROUP BY t.calendar_year;
```

문제65. 아래의 SQL을 조인 순서와 조인 방법을 아래와 같이 설정하시오.

```
SELECT /*+ leading(t s p) use_hash(s) use_hash(p) */
      p.prod_name, t.calendar_year, SUM(s.amount_sold)
  FROM sales200 s, times200 t, products200 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;
```

조인 순서 : times - > sales - > products

조인 방법 : 해쉬조인 해쉬조인

실행계획)

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14656	1388K	1284 (4)	00:00:16
1	HASH GROUP BY		14656	1388K	1284 (4)	00:00:16
* 2	HASH JOIN		14656	1388K	1282 (3)	00:00:16
* 3	TABLE ACCESS FULL	PRODUCTS200	1	40	3 (0)	00:00:01
* 4	HASH JOIN		1055K	57M	1271 (3)	00:00:16
* 5	TABLE ACCESS FULL	TIMES200	731	16082	17 (0)	00:00:01
* 6	TABLE ACCESS FULL	SALES200	1055K	35M	1246 (2)	00:00:15

문제66. 위의 결과를 아래와 같이 실행계획이 나오게 하시오.

```
SELECT /*+ leading(t s p) use_hash(s) use_hash(p) no_swap_join_inputs(p)*/
      p.prod_name, t.calendar_year, SUM(s.amount_sold)
FROM sales200 s, times200 t, products200 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;
```

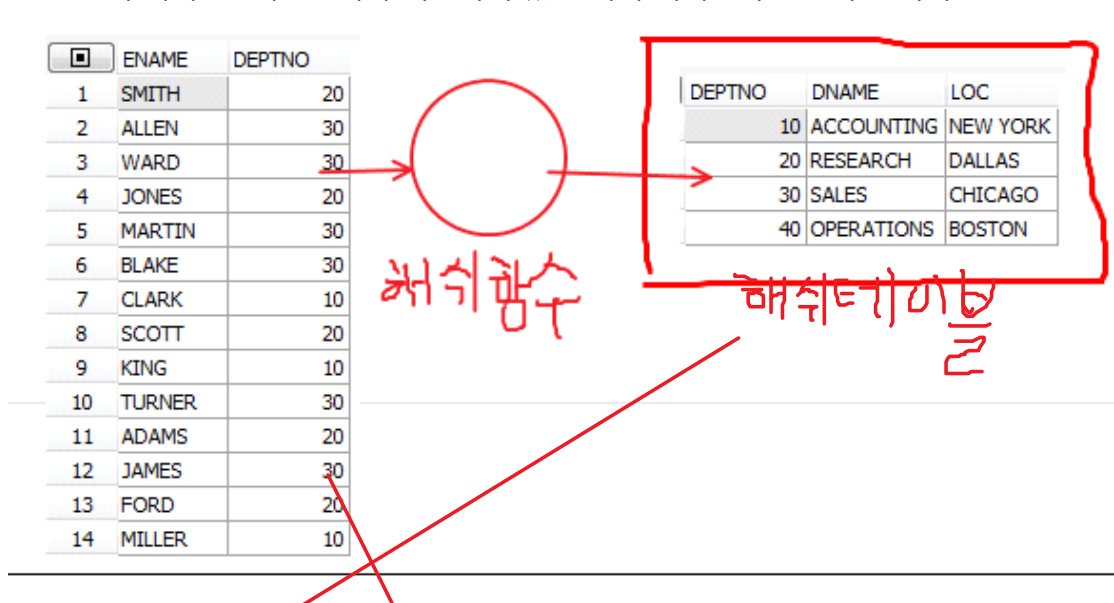
Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		14656	1388K		4732 (1)	00:00:57
1	HASH GROUP BY		14656	1388K		4732 (1)	00:00:57
* 2	HASH JOIN		14656	1388K	69M	4731 (1)	00:00:57
* 3	HASH JOIN		1055K	57M		1271 (3)	00:00:16
* 4	TABLE ACCESS FULL	TIMES200	731	16082		17 (0)	00:00:01
5	TABLE ACCESS FULL	SALES200	1055K	35M		1246 (2)	00:00:15
* 6	TABLE ACCESS FULL	PRODUCTS200	1	40		3 (0)	00:00:01

해쉬 조인 사용시 힌트

1. `Use_hash`(테이블명): 해쉬 조인 해라~
2. `Swap_join_inputs`: 해쉬 테이블을 선정할 때 사용하는 힌트
3. `No_swap_join_inputs`: prob 테이블을 선정할 때 사용하는 힌트

해쉬 테이블과 prob테이블

1. 해쉬 테이블 : 메모리로 올라가는 테이블
2. 탐색 테이블 : 디스크에서 메모리에 있는 해쉬 테이블과 조인하는 테이블



14	MILLER	10
----	--------	----

Select /*+ leading(d e) use_hash(e)*/ e.ename, d.loc
 From emp e, dept d
 Where e.deptno = d.deptno;

테이블이 2개면 leading 힌트만으로도 해쉬 테이블과 탐색 테이블을 선정할 수 있다.

테이블이 3개면 leading 힌트만으로도 해쉬 테이블과 탐색 테이블을 선정하기가 어려워진다.
 => 그래서 필요한 힌트가 swap_join_inputs와 no_swap_join_inputs 이다.

Select /*+ leading(d e b) use_hash(e) use_hash(b)*/ e.ename, d.loc, b.bonus
 From emp e, dept d, bonus b
 Where e.deptno = d.deptno
 and e.empno = b.empno;

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
HASH JOIN		14
HASH JOIN		14
TABLE ACCESS FULL	SCOTT.DEPT	4
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.BONUS	14

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
HASH JOIN		14
HASH JOIN		14
TABLE ACCESS FULL	SCOTT.DEPT	4
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.BONUS	14

문제67. 아래와 같이 실행계획이 나오게 하시오.

Operation	Object Name
SELECT STATEMENT Optimizer Mode=ALL_ROWS	
HASH JOIN	
TABLE ACCESS FULL	SCOTT.BONUS
HASH JOIN	
TABLE ACCESS FULL	SCOTT.DEPT
TABLE ACCESS FULL	SCOTT.EMP

Select /*+ leading(d e b) use_hash(e) use_hash(b) swap_join_inputs(b)*/ e.ename, d.loc, b.bonus
 From emp e, dept d, bonus b
 Where e.deptno = d.deptno
 and e.empno = b.empno;

문제68. 아래와 같이 실행계획이 나오게 하시오.

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
HASH JOIN		14
HASH JOIN		14
TABLE ACCESS FULL	SCOTT.BONUS	14
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.DEPT	4

```

Select /*+ leading(b e d) use_hash(e) use_hash(d)*/ e.ename, d.loc, b.bonus
From emp e, dept d, bonus b
Where e.deptno = d.deptno
and e.empno = b.empno;

```

%% 테이블 컬럼에 개수를 자동으로 출력하는 방법 (컴퓨터를 켜놓으면 밤10시에 자동으로함)
Set autot off

```

SELECT table_name, num_rows, last_analyzed
FROM user_tables;

```

TABLE_NAME	NUM_ROWS	LAST_ANALYZED
BONUS	(null)	(null)
PRODUCTS200	(null)	(null)
CUSTOMERS200	(null)	(null)
TIMES200	(null)	(null)
SALES200	(null)	(null)
EMP2_SK	(null)	(null)
EMP2_KT	(null)	(null)

%% 테이블 emp에 대해 수동으로 분석작업을 수행하는 방법

```
Analyze table emp COMPUTE statistics;
```

51	SALGRADE	(null)	(null)
52	EMP	14	2018-04-24 오전 10:
53	DEPT	(null)	(null)

문제69. 아래와 같이 실행계획이 나오게하시오.

Operation	Object Name
SELECT STATEMENT Optimizer Mode=ALL_ROWS	
HASH JOIN	
NESTED LOOPS	
TABLE ACCESS FULL	SCOTT.BONUS
TABLE ACCESS FULL	SCOTT.EMP
TABLE ACCESS FULL	SCOTT.DEPT

```

Select /*+ leading(b e d) use_nl(e) use_hash(d)*/ e.ename, d.loc, b.bonus
From emp e, dept d, bonus b
Where e.deptno = d.deptno
and e.empno = b.empno;

```

=> 만약 아래와 같이 Nested loops 를안타고 인덱스를 타면 제약을 없애준다.

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
HASH JOIN		14
NESTED LOOPS		14
NESTED LOOPS		14
TABLE ACCESS FULL	SCOTT.BONUS	14
INDEX UNIQUE SCAN	SCOTT.PK_EMP	1
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	1
TABLE ACCESS FULL	SCOTT.DEPT	4

문제 70. emp와 salgrade를 조인해서 이름, 월급, 급여등급을 출력하는데, 조인방법이 해쉬 조인이 되게 하시오.

```
SELECT /*+ leading(s e) use_hash(e)*/ e.ename, e.sal, s.grade
FROM EMP e, SALGRADE s
WHERE e.sal between s.LOSAL AND s.HISAL;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
NESTED LOOPS		1
TABLE ACCESS FULL	SCOTT.SALGRADE	5
TABLE ACCESS FULL	SCOTT.EMP	1

Non equi join은 해쉬 조인을 쓸 수 없다.

해쉬 조인이 가능하려면 조인의 연결고리가 이퀄 조건이어야 한다.

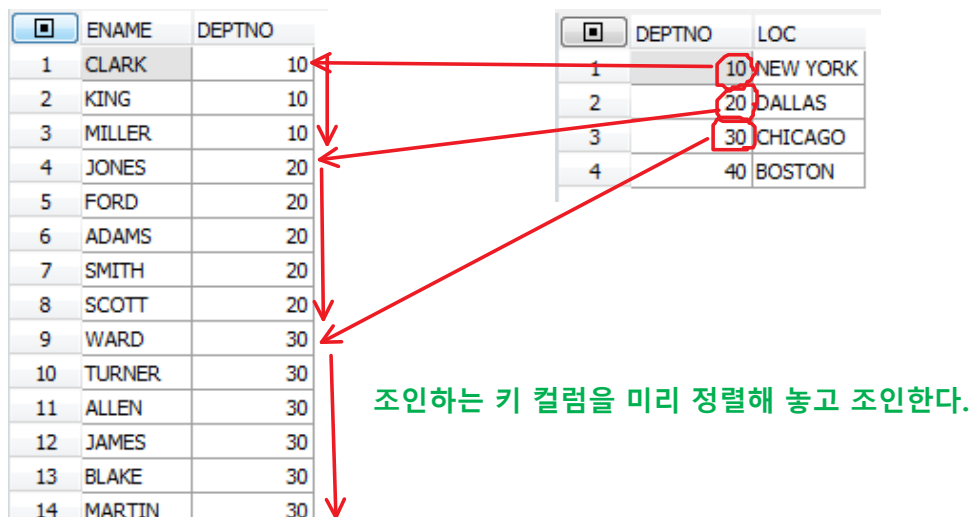
2-3. sort merge join

: emp와 salgrade의 조인처럼 조인의 연결고리가 이퀄 조건이 아니면서
대용량 테이블의 조인 이어서 해쉬 조인을 사용 할수 없을 때 유리한 조인방법

Ex) select /*+ leading(d e) use_merge(e) */ e.ename, d.loc, e.deptno
From emp e, dept d
Where e.deptno = d.deptno;

	ENAME	LOC	DEPTNO
1	CLARK	NEW YORK	10
2	KING	NEW YORK	10
3	MILLER	NEW YORK	10
4	JONES	DALLAS	20
5	FORD	DALLAS	20
6	ADAMS	DALLAS	20
7	SMITH	DALLAS	20
8	SCOTT	DALLAS	20
9	WARD	CHICAGO	30
10	TURNER	CHICAGO	30
11	ALLEN	CHICAGO	30
12	JAMES	CHICAGO	30
13	BLAKE	CHICAGO	30
14	MARTIN	CHICAGO	30

=> 부서번호가 자동적으로 정렬된다.



문제71. emp 와 salgrade 를 조인해서 이름, 월급, 급여등급을 출력하는데 조인 방법이 sort merge join이 되게 하시오.

```
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;
```

	ENAME	SAL	GRADE
1	SMITH	800	1
2	JAMES	950	1
3	ADAMS	1100	1
4	WARD	1250	2
5	MARTIN	1250	2
6	MILLER	1300	2
7	TURNER	1500	3
8	ALLEN	1600	3
9	CLARK	2450	4
10	BLAKE	2850	4
11	JONES	2975	4
12	SCOTT	3000	4
13	FORD	3000	4
14	KING	5000	5

	Operation	Object Name	Rows
1			
2			
3			
4	SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
5	MERGE JOIN		1
6	SORT JOIN		5
7	TABLE ACCESS FULL	SCOTT.SALGRADE	5
8	FILTER		
9	SORT JOIN		14
10	TABLE ACCESS FULL	SCOTT.EMP	14

과 를 merge 조인을 한다.

문제72. 부서위치, 부서위치별 토탈월급을 출력하는데, 적절한 조인 순서와 조인방법을 명시해서 출력하시오.

```
SELECT /*+ leading(d e) use_nl(e)*/ d.loc, SUM(sal)
FROM EMP e, DEPT d
WHERE e.DEPTNO = d.DEPTNO
GROUP BY d.loc;
```

현재 Emp 테이블과 dept 테이블은 데이터가 적어서 nl조인을 써도된다.

문제73. (점심시간문제) 이름, 월급, 부서위치, 급여등급을 출력하는데, 아래와 같이 실행계획이 출력되게 하시오.

	Operation	Object Name	Rows
1			
2			
3			
4	SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
5	HASH JOIN		1
6	NESTED LOOPS		1
7	TABLE ACCESS FULL	SCOTT.SALGRADE	5
8	TABLE ACCESS FULL	SCOTT.EMP	1
9	TABLE ACCESS FULL	SCOTT.DEPT	4

```
SELECT /*+ leading(s e d) use_nl(e) use_hash(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;

outer join 문장 튜닝

Ex) 이름, 부서위치를 출력하는데, outer join 사인을 사용해서 BOSTON도 출력되게 하시오.

```
SELECT e.ename, d.loc
FROM EMP e, DEPT d
WHERE e.deptno(+) = d.deptno;
```

2	ALLEN	CHICAGO
3	WARD	CHICAGO
4	JONES	DALLAS
5	MARTIN	CHICAGO
6	BLAKE	CHICAGO
7	CLARK	NEW YORK
8	SCOTT	DALLAS
9	KING	NEW YORK
10	TURNER	CHICAGO
11	ADAMS	DALLAS
12	JAMES	CHICAGO
13	FORD	DALLAS
14	MILLER	NEW YORK
15	(null)	BOSTON

Operation	Object Name
SELECT STATEMENT Optimizer Mode=ALL_ROWS	
HASH JOIN OUTER	
TABLE ACCESS FULL	SCOTT.DEPT
TABLE ACCESS FULL	SCOTT.EMP

위의 결과를 nested loop 조인이 되게끔 해보시오.

```
SELECT /*+ leading(d e) use_nl(e) */ e.ename, d.loc
FROM EMP e, DEPT d
WHERE e.deptno(+) = d.deptno;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
NESTED LOOPS OUTER		14
TABLE ACCESS FULL	SCOTT.DEPT	4
TABLE ACCESS FULL	SCOTT.EMP	4

문제74. 이름, 부서위치를 출력하는 outer join 문장을 아래와 같이 작성하는데, 지금 입력한 JACK도 출력되게 하시오.

```
Insert into emp(empno, ename, sal, deptno)
Values(1092, 'JACK', 3400, 70);
```

```
SELECT e.ename, d.loc
FROM EMP e, DEPT d
WHERE e.deptno = d.deptno(+);
```

6	SCOTT	DALLAS
7	JONES	DALLAS
8	SMITH	DALLAS
9	JAMES	CHICAGO
10	TURNER	CHICAGO
11	BLAKE	CHICAGO
12	MARTIN	CHICAGO
13	WARD	CHICAGO
14	ALLEN	CHICAGO
15	JACK	(null)

문제75. 문제74번의 조인방법을 nested loop 조인이 되게 하고 조인순서를 dept->emp순으로 조인되게하시오.

```
SELECT /*+ leading(d e) use_nl(e) */e.ename, d.loc
FROM EMP e, DEPT d
WHERE e.deptno = d.deptno(+);
```

조인순서 →

이렇게 하면 될 거 같지만, 조인할때 outer join 사인을 사용하게 되면 조인순서가 무조건 outer join 사인이 없는쪽에서 outer join 사인이 있는 쪽으로 조인을 한다.

문제76. 문제74번의 조인방법을 hash 조인이 되게 하고 조인순서를 dept->emp순으로 조인되게하시오.

Hash 조인도 안된다.

그러나, swap_join_inputs 로하면 된다.

```
SELECT /*+ leading(d e) use_hash(e) swap_join_inputs(d) */e.ename, d.loc
FROM EMP e, DEPT d
WHERE e.deptno = d.deptno(+);
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
HASH JOIN RIGHT OUTER		14
TABLE ACCESS FULL	SCOTT.DEPT	4
TABLE ACCESS FULL	SCOTT.EMP	14

문제77. 아래의 SQL을 튜닝하시오.

```
SELECT e.ename, d.loc, e.job
FROM EMP e, DEPT d
WHERE e.deptno = d.deptno(+)
and d.loc(+) = 'DALLAS';
```

=> dept 테이블 먼저 조인해야되서 swap 조인 인풋을 써준다.

```
SELECT /*+ swap_join_inputs(d) use_hash(e) */ e.ename, d.loc, e.job
FROM EMP e, DEPT d
WHERE e.deptno = d.deptno(+)
and d.loc(+) = 'DALLAS';
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
HASH JOIN RIGHT OUTER		14
TABLE ACCESS FULL	SCOTT.DEPT	1
TABLE ACCESS FULL	SCOTT.EMP	14

문제78. 아래의 full outer join 을 튜닝 하시오.

튜닝전

```
SELECT /*+ opt_param('_optimizer_native_full_outer_join', 'off')*/ e.ename, d.loc
FROM EMP e FULL outer JOIN DEPT d
ON(e.DEPTNO=d.deptno);
```

Operation	Object Name	Rows	Bytes
SELECT STATEMENT Optimizer Mode=ALL_ROWS		18	270
VIEW		18	270
UNION-ALL			
HASH JOIN OUTER		14	392
TABLE ACCESS FULL	SCOTT.EMP	14	98
TABLE ACCESS FULL	SCOTT.DEPT	4	84
HASH JOIN ANTI		4	92
TABLE ACCESS FULL	SCOTT.DEPT	4	84
TABLE ACCESS FULL	SCOTT.EMP	14	28

튜닝후

```
SELECT /*+ opt_param('_optimizer_native_full_outer_join', 'FORCE')*/ e.ename, d.loc
FROM EMP e FULL outer JOIN DEPT d
ON(e.DEPTNO=d.deptno);
```

```
%% /*+ opt_param('_optimizer_native_full_outer_join', 'off')*/
```

: 성능이 더 좋은 full outer join 실행계획을 사용하지 않겠다.

```
%% /*+ opt_param('_optimizer_native_full_outer_join', 'FORCE')*/
```

: 성능이 더 좋은 full outer join 실행계획을 사용 하겠다.

문제79. 아래의 SQL을 튜닝하시오.

```
SELECT ename, sal, job
FROM EMP
WHERE ename LIKE '%EN%'
OR ename LIKE '%IN%';
```

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	57

CREATE INDEX EMP_ename ON EMP(ename); 이름의 인덱스를 생성하고,

```
SELECT /*+ leading(e3 e1) use_nl(e1) */ e1.ename, e1.sal, e1.job
FROM emp e1,
( SELECT /*+ index_ffs(e2 emp_ename) no_merge*/ rowid rr
FROM EMP e2
WHERE ename LIKE '%EN%' OR ename LIKE '%IN%') e3
Where e1.rowid = e3.rr;
```

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	8

No_merge : in line view를 해체하지 말아라!

문제80. 우리반 테이블에서 주소가 송파구와 동작구에 사는 학생들의 이름과 주소와 나이를 출력하시오.

튜닝전


```
SELECT ename, address, age
FROM EMP2
WHERE address LIKE '%송파구%'
      OR address LIKE '%동작구%';
```

	ENAME	ADDRESS	AGE
1	유혜린	서울시 송파구 방미동	26
2	이유진	서울시 동작구 사당동	25

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	46

튜닝후

CREATE INDEX emp2_address ON emp2(address); 주소 인덱스를 생성해주고

```
SELECT /*+ leading(e4 e2) use_nl(e2) */ e2.ename, e2.address, e2.age
FROM EMP2 e2,
     (SELECT /*+ index_ffs(e3 emp2_address) no_merge */ ROWID rr
      FROM EMP2 e3
      WHERE address LIKE '%송파구%'
            OR address LIKE '%동작구%') e4
WHERE E2.ROWID = e4.rr;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		2
NESTED LOOPS		2
VIEW		2
INDEX FAST FULL SCAN	SCOTT.EMP2_ADDRESS	2
TABLE ACCESS BY USER ROWID	SCOTT.EMP2	1

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	2

hash join 시에 병렬 작업 하는방법

Ex) 이름과 부서위치를 출력하는데 해쉬 조인으로 수행되게 하시오.

```
SELECT /*+ leading(d e) use_hash(e) */ e.ename, d.loc
FROM EMP e, DEPT d
WHERE e.DEPTNO = d.DEPTNO;
```

Operation	Object Name
SELECT STATEMENT Optimizer Mode=ALL_ROWS	
HASH JOIN	
TABLE ACCESS FULL	SCOTT.DEPT
TABLE ACCESS FULL	SCOTT.EMP

해쉬 조인 할때는 full 스캔을 사용해야한다. 오히려 index 스캔을하면 더 느려진다.

문제81. 위의 해쉬조인 문장의 full table scan이 병렬로 처리되게 하시오.

```
SELECT /*+ leading(d e) use_hash(e)
         full(e) full(d)
```

4 = 병렬도 : 많이 줄 수록 빨라지는데

문제81. 위의 애미조인 환경의 full table scan이 강제로 선택되게 하시오.

```
SELECT /*+ leading(d e) use_hash(e)
        full(e) full(d)
        parallel(e,4) parallel(d,4) */ e.ename, d.loc
FROM EMP e, DEPT d
WHERE e.DEPTNO = d.DEPTNO;
```

4 = 병렬도 : 많이 줄 수록 빨라지는데
적당히줘야함

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
PX COORDINATOR		
PX SEND QC (RANDOM)	SYS.:TQ10002	14
HASH JOIN BUFFERED		14
PX RECEIVE		4
PX SEND HASH	SYS.:TQ10000	4
PX BLOCK ITERATOR		4
TABLE ACCESS FULL	SCOTT.DEPT	4
PX RECEIVE		14
PX SEND HASH	SYS.:TQ10001	14
PX BLOCK ITERATOR		14
TABLE ACCESS FULL	SCOTT.EMP	14

Full ~ parallel~

3. SQL 튜닝(서브쿼리 튜닝)

2018년 4월 24일 화요일 오후 1:57

3.SQL 튜닝(서브쿼리 튜닝)

3-1. 옵티마이저 란?

3-2. 쿼리변환

3-3. 서브쿼리 unnesting

3-4. 뷰 merging

3-5. push_pred, no_push_pred

3-6. 스칼라 서브쿼리를 이용한 튜닝

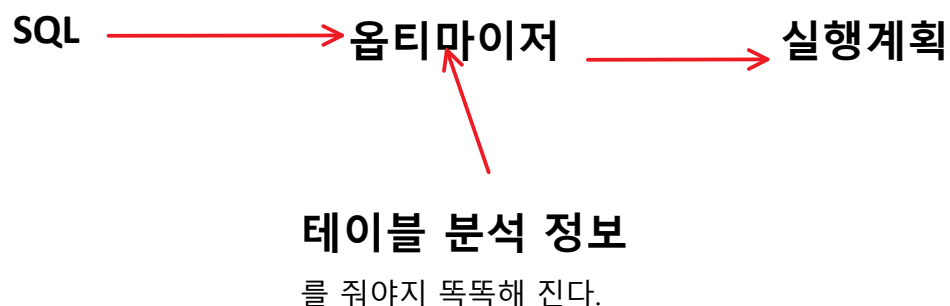
3-7. 수정가능한 조인 뷰

3-8. 분석함수를 이용한 SQL 튜닝

3-9. with절을 이용한 SQL 튜닝

3-1. 옵티마이저 란?

: SQL을 가장 효율적이고 빠르게 수행할 수 있는 최적의 처리 경로를 선택해주는 DBMS의 핵심엔진이다.



3-2. 쿼리변환 이란?

: 옵티마이저가 실행계획을 생성하기 전에 사용자가 작성한 SQL보다 결과

는 똑같은데 비용이 더 적게 발생하는 SQL이 있다면 그 SQL로 알아서 쿼리를
변경을 한다.

Ex) 이름이 EN 또는 IN 을 포함하고 있는 직원들의 이름, 월급, 직업을 출력하시오.

```
SELECT /*+ leading(e3 e1) use_nl(e1) */ e1.ename, e1.sal, e1.job
FROM EMP e1,
      (select /*+ index_ffs(e2 emp_ename) */ rowid rr
       from emp e2
       WHERE ename LIKE '%EN%'
              OR ename LIKE '%IN%') e3
where e1.rowid = e3.rr;
```

No_merge : 쿼리변환을 해체하지마!!

쿼리변환기

```
SELECT ename, sal, job
FROM EMP
WHERE ename LIKE '%EN%'
      OR ename LIKE '%IN%';
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
TABLE ACCESS FULL	SCOTT.EMP	1

문제82. DALLAS에 있는 부서번호에서 근무하는 직원들의 이름, 월급을 출력하는데 조인을 사용하지 않고,
서브쿼리문으로 작성하시오.

```
SELECT ename, sal
FROM EMP
WHERE deptno = (SELECT deptno
                FROM DEPT
                WHERE loc = 'DALLAS');
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		5
TABLE ACCESS FULL	SCOTT.EMP	5
TABLE ACCESS FULL	SCOTT.DEPT	1

문제83. 문제82번의 '=' 을 'in'으로 바꿔서 출력하시오.

```
SELECT ename, sal
FROM EMP
WHERE deptno in (SELECT deptno
                 FROM DEPT
                 WHERE loc = 'DALLAS');
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		5
HASH JOIN SEMI		5
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.DEPT	1

왜 해쉬조인을 했는가?

In을 써주면 emp먼저 액세스하고 않겠지 하면서 자체적으로 판단해서 hash join을 한다.

문제84. 해쉬 세미(절반)조인 하지말고 서브쿼리로 수행하라고 힌트를 주시오.

```
SELECT ename, sal
FROM EMP
WHERE deptno in (SELECT /*+ no_unnest */ deptno
FROM DEPT
WHERE loc = 'DALLAS');
```

Operation	Object Name
SELECT STATEMENT Optimizer Mode=ALL_ROWS	
FILTER	
TABLE ACCESS FULL	SCOTT.EMP
TABLE ACCESS FULL	SCOTT.DEPT

No_unnest (부정의 부정 = 강한 긍정) : 강하게 감싸라!!

문제85. (마지막문제) 아래의 SQL 실행계획을 dept가 먼저 드라이빙 되면서 해쉬 세미조인 되게하시오.

```
SELECT ename, sal
FROM EMP
WHERE deptno in (SELECT deptno
FROM DEPT
WHERE loc = 'DALLAS');
```

```
SELECT ename, sal
FROM EMP e
WHERE deptno in (SELECT /*+ swap_join_inputs(d) */ deptno
FROM DEPT d
WHERE loc = 'DALLAS');
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		5
HASH JOIN RIGHT SEMI		5
TABLE ACCESS FULL	SCOTT.DEPT	1
TABLE ACCESS FULL	SCOTT.EMP	14

3-3. subquery unnesting

: 서브쿼리문을 실행 할 때 서브쿼리문으로 실행할지 아니면 조인형태로 변경해서 실행할지를 결정하도록 쿼리변환기를 제어하는 기술.

1.unest : 서브쿼리로 수행하지말고 서브쿼리를 풀어해쳐서 수행해라.

2.no_unnest : 반드시 서브쿼리로 실행해라!

2-1. push_subq : 서브쿼리부터 수행해라.

2-2. no_push_subq : 메인쿼리부터 수행해라.

No_unnest 랑 짝궁처럼 같이써줘야함!!

Ex) 아래의 SQL의 실행계획이 서브쿼리로 실행되는지 세미조인형태로 실행되는지 확인.

```
SELECT ename, sal
FROM EMP
WHERE deptno in (SELECT deptno
FROM DEPT
WHERE loc = 'DALLAS');
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		5
HASH JOIN SEMI		5
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.DEPT	1

문제86. 아래의 SQL이 세미조인으로 수행되지 않고 서브쿼리 문으로 수행되도록 힌트를 주시오.

```
SELECT ename, sal
FROM EMP
WHERE deptno in (SELECT deptno
                  FROM DEPT
                  WHERE loc = 'DALLAS');
```

FILTER = 메인쿼리먼저 실행

```
SELECT ename, sal
FROM EMP
WHERE deptno in (SELECT /*+ no_unnest */ deptno
                  FROM DEPT
                  WHERE loc = 'DALLAS');
```

Operation	Object Name	Rows
4 SELECT STATEMENT Optimizer Mode=ALL_ROWS		5
3 FILTER		
1 TABLE ACCESS FULL	SCOTT.EMP	14
2 TABLE ACCESS FULL	SCOTT.DEPT	1

문제87. 문제86번의 실행계획이 서브쿼리부터 수행 될 수 있도록 힌트를 주시오.

```
SELECT ename, sal
FROM EMP
WHERE deptno in (SELECT /*+ no_unnest push_subq */ deptno
                  FROM DEPT
                  WHERE loc = 'DALLAS');
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
TABLE ACCESS FULL	SCOTT.EMP	1
TABLE ACCESS FULL	SCOTT.DEPT	1

QB_NAME : Query Block 에 이름을 지어주는 힌트

문제88. 아래의 SQL을 QB_NAME 힌트를 이용해서 다시 작성하시오.

```
SELECT ename, sal
FROM EMP
WHERE deptno in (SELECT deptno
                  FROM DEPT
                  WHERE loc = 'DALLAS');
```

```
SELECT /*+ QB_NAME(main) */ ename, sal
FROM EMP
WHERE deptno in (SELECT /*+ QB_NAME(sub) no_unnest push_subq */ deptno
                  FROM DEPT
                  WHERE loc = 'DALLAS');
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
TABLE ACCESS FULL	SCOTT.EMP	1
TABLE ACCESS FULL	SCOTT.DEPT	1

실행계획은 별차이가없지만,

문제89. 관리자가 직원들의 이름을 출력하시오. (자기 밑에 직속부하가 한명이라도 있는 직원들)

```
SELECT ename
FROM EMP
WHERE empno in (SELECT mgr
                FROM EMP);
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		6
HASH JOIN SEMI		6
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.EMP	13

문제90. 문제89 SQL 실행계획이 hash semi 조인 되지않고 서브쿼리로 수행되게 하시오.

```
SELECT ename
FROM EMP
WHERE empno in (SELECT /*+ no_unnest*/ mgr
                FROM EMP);
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
FILTER		
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.EMP	2

문제91. 문제90 SQL 실행계획이 서브쿼리 먼저 수행되게 하시오.



```
SELECT ename
FROM EMP
WHERE empno in (SELECT /*+ no_unnest push_subq*/ mgr
                FROM EMP);
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
TABLE ACCESS FULL	SCOTT.EMP	1
TABLE ACCESS FULL	SCOTT.EMP	2

==> 문제89~91번을 보면 실행계획에 보면 둘다 EMP 테이블이라 메인쿼리부터 수행했는지,
서브쿼리 부터 수행했는지 확인하기 어려우니 확인할 수 있도록 힌트를 주는게 QB_NAME이다.

문제92. 위의 SQL 을 QB_NAME 힌트를 주고 실행하시오.

```
SELECT /*+ QB_NAME(main) */ename
FROM EMP
WHERE empno in (SELECT /*+ QB_NAME(sub) no_unnest push_subq */ mgr
                FROM EMP);
```

Format: **ADVANCED**  



Plan hash value: 587534197

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT				6 (100)	
* 1	TABLE ACCESS FULL	EMP	1	8	3 (0)	00:00:01
* 2	TABLE ACCESS FULL	EMP	2	6	3 (0)	00:00:01

Predicate Information (identified by operation id):

1 - filter(IS NOT NULL)
2 - filter("MGR"=:B1)

만약 filter 가 나오면 format 을 바꿔준다.

Format: **ADVANCED ALLSTATS LAST**  

Id	Operation	Name	E-Rows	E-Bytes	Cost (%CPU)	E-Time
0	SELECT STATEMENT				6 (100)	
* 1	TABLE ACCESS FULL	EMP	1	8	3 (0)	00:00:01
* 2	TABLE ACCESS FULL	EMP	2	6	3 (0)	00:00:01

Query Block Name / Object Alias (identified by operation id):

1 - MAIN EMP@MAIN
2 - SUB EMP@SUB

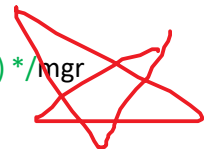
문제93. 관리자가 아닌 사원들의 이름을 출력하시오.

```
SELECT ename
FROM EMP
WHERE empno NOT IN (SELECT mgr
                     FROM EMP
                     WHERE mgr IS NOT null);
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		8
HASH JOIN ANTI SJA		8
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.EMP	13

문제94. 문제93번 실행계획이 hash right anti 조인이 되게하시오.

```
SELECT /*+ QB_NAME(main) */ename
FROM EMP
WHERE empno NOT IN (SELECT /*+ QB_NAME(sub) swap_join_inputs(emp) */mgr
                     FROM EMP
                     WHERE mgr IS NOT null);
```



Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
HASH JOIN RIGHT ANTI		14
TABLE ACCESS FULL	SCOTT.EMP	13
TABLE ACCESS FULL	SCOTT.EMP	14

3-4. view merging

: view 나 in line view 를 해체할지 말지를 결정하도록 쿼리 변환기를 제어하는 방법.

3-4-1. merge : 뷰나 인라인뷰 를 해체하라~

3-4-2. no_merge : 뷰나 인라인 뷰를 해체하지 마라!

문제95. 직업이 SALESMAN인 사람들이 직원번호, 이름, 직업, 관리자 번호, 입사일, 월급, 커미션, 부서번호를

출력하는 view를 생성하시오.(view 이름 : emp_salesman)

```
CREATE VIEW emp_salesman
AS
SELECT empno, ename, job, mgr, hiredate, sal, comm, deptno
FROM EMP
WHERE job = 'SALESMAN';
```

문제96. emp_salesman 뷰와 dept와 조인해서 이름, 직업, 관리자번호, 월급, 부서위치를 출력하시오.

```
SELECT v.ename, v.job, v.mgr, v.sal, d.loc
FROM emp_salesman v, DEPT d
WHERE v.deptno = d.DEPTNO;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		3
HASH JOIN		3
TABLE ACCESS FULL	SCOTT.EMP	3
TABLE ACCESS FULL	SCOTT.DEPT	4

실행계획에 보면 view를 해체하고 있다.(view가 안보임)

문제97. 문제96번을 다시 수행하고 view 를 해체 하지않도록 힌트를 주고 실행하시오.

```
SELECT /*+ no_merge(v) */ v.ename, v.job, v.mgr, v.sal, d.loc
FROM emp_salesman v, DEPT d
WHERE v.deptno = d.DEPTNO;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		3
HASH JOIN		3
VIEW	SCOTT.EMP_SALESMAN	3
TABLE ACCESS FULL	SCOTT.EMP	3
TABLE ACCESS FULL	SCOTT.DEPT	4

문제98. 문제97번의 view와 dept 테이블과의 조인순서와 조인방법이 아래와 같이 수행되게 하시오.

조인 순서 : emp_salesman --> dept

조인 방법 : nested loop join

```
SELECT /*+ no_merge(v) leading(v d) use_nl(d) */ v.ename, v.job, v.mgr, v.sal, d.loc
FROM emp_salesman v, DEPT d
WHERE v.deptno = d.DEPTNO;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		3
NESTED LOOPS		3
VIEW	SCOTT.EMP_SALESMAN	3
TABLE ACCESS FULL	SCOTT.EMP	3
TABLE ACCESS FULL	SCOTT.DEPT	1

Merge 사용 시기 : emp 테이블이 1억건이고, 그중에 salesman이 5건만 있다고 가정하면, View를 해체하지 않고 dept와 조인을 하는게 더 효율적이다.

문제99. 직원번호, 이름, 부서위치, 부서번호를 출력하는 뷰를 생성하시오(view 이름 : emp_9000)

```
CREATE VIEW emp9000
AS
SELECT e.empno, e.ename, d.loc, d.deptno
FROM EMP e, DEPT d
WHERE e.DEPTNO = d.DEPTNO;
```

	EMPNO	ENAME	LOC	DEPTNO
1	7369	SMITH	DALLAS	20
2	7499	ALLEN	CHICAGO	30
3	7521	WARD	CHICAGO	30
4	7566	JONES	DALLAS	20
5	7654	MARTIN	CHICAGO	30
6	7698	BLAKE	CHICAGO	30
7	7782	CLARK	NEW YOR	10
8	7788	SCOTT	DALLAS	20
9	7839	KING	NEW YOR	10
10	7844	TURNER	CHICAGO	30
11	7876	ADAMS	DALLAS	20
12	7900	JAMES	CHICAGO	30
13	7902	FORD	DALLAS	20
14	7934	MILLER	NEW YOR	10

문제100. 문제99번에서 만든 emp9000과 보너스 테이블을 조인해서 이름, 부서위치, 보너스를 출력하시오.

```
SELECT v.ename, v.loc, B.BONUS
FROM emp9000 v, BONUS b
WHERE v.EMPNO = b.EMPNO;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
HASH JOIN		14
HASH JOIN		14
TABLE ACCESS FULL	SCOTT.DEPT	4
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.BONUS	14

실행계획에 보면 view가 해체 되어있다.

문제101. 문제100번의 view 를 해체하지말고 bonus 테이블과 조인되게 하시오.

```
SELECT /*+ no_merge(v) */ v.ename, v.loc, B.BONUS
FROM emp9000 v, BONUS b
WHERE v.EMPNO = b.EMPNO;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
HASH JOIN		14
VIEW	SCOTT.EMP9000	14
HASH JOIN		14
TABLE ACCESS FULL	SCOTT.DEPT	4
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.BONUS	14

문제102. (점심시간 문제) 아래와 같이 실행계획이 나오게하시오.

```
SELECT /*+ no_merge(v) leading(v.e v.d) use_nl(v.d) */ v.ename, v.loc, B.BONUS
FROM emp9000 v, BONUS b
WHERE v.EMPNO = b.EMPNO;
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
HASH JOIN		14
VIEW	SCOTT.EMP9000	14
NESTED LOOPS		14
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.DEPT	1
TABLE ACCESS FULL	SCOTT.BONUS	14

==> v.e 의 뜻 v 안에있는 emp를 가져온다.

서브쿼리 사용사 힌트 총정리

1.세미/안티 형태로 조인해라. no_unnest

- 서브쿼리부터 수행해라 - push_subq
- 메인쿼리부터 수행해라 - no_push_subq

2.세미/안티 조인형태로 변경해서 수행해라. - unnest

- 세미코인 (in 연산자)
 1. nested loop semi 조인 - nl_sj
 - ★ 2. hash semi 조인 - hash_sj
 3. sort merge semi 조인 - merge_sj
- 안티조인 (not in 연산자)
 1. nested loopp anti 조인 - nl_aj
 - ★ 2. hash anti 조인 - hash_aj
 3. sort merge anti 조인 - merge_aj

문제103. 관리자인 직원들의 이름을 출력하는데 실행계획이 nested loop semi 조인이 나오게하시오.

```
SELECT ename
FROM EMP
WHERE empno IN (SELECT /*+ unnest nl_sj */ mgr
FROM EMP);
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		6
NESTED LOOPS SEMI		6
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.EMP	6

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	162

문제104. 관리자인 직원들의 이름을 출력하는데 실행계획이 hash semi 조인이 나오게하시오.

```
SELECT ename
FROM EMP
WHERE empno IN (SELECT /*+ unnest hash_sj */ mgr
FROM EMP);
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		6
HASH JOIN SEMI		6
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.EMP	13

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	64

문제105. 관리자가 아닌 직원들의 이름을 출력하는데 실행계획이 nested loop anti 조인이 되게 하시오.

```
SELECT ename
FROM EMP
WHERE empno
AND empno NOT IN (SELECT /*+ unnest nl_aj */mgr
FROM EMP
WHERE mgr IS NOT null);
```

이렇게하면 될 수도 있고 안될 수도 있는데 이유는 empno에 not null 제약이 있으면 나온다.
그래서 안나오면 따로 걸어줘야한다

```
SELECT ename
FROM EMP
WHERE empno IS NOT null
AND empno NOT IN (SELECT /*+ unnest nl_aj */mgr
FROM EMP
WHERE mgr IS NOT null);
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		8
NESTED LOOPS ANTI		8
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.EMP	6

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	162

그래서 양쪽에 **not null** 을써주는게 좋다.

문제106. 관리자가 아닌 직원들의 이름을 출력하는데 실행계획이 hash anti 조인이 되게 하시오.

```
SELECT ename
FROM EMP
WHERE empno IS NOT NULL
AND empno NOT IN (SELECT /*+ unnest hash_aj */mgr
                  FROM EMP
                  WHERE mgr IS NOT null);
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		8
HASH JOIN ANTI		8
TABLE ACCESS FULL	SCOTT.EMP	14
TABLE ACCESS FULL	SCOTT.EMP	13

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	64

문제107. exists 문을 이용해서 부서 테이블에서 부서위치를 출력하는데 직원테이블에 존재하는 부서번호에 대한 부서위치만 출력하시오.

```
SELECT LOC
FROM DEPT d
WHERE EXISTS (SELECT 'x'
              FROM EMP e
              WHERE e.DEPTNO = d.DEPTNO);
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		3
HASH JOIN SEMI		3
TABLE ACCESS FULL	SCOTT.DEPT	4
TABLE ACCESS FULL	SCOTT.EMP	14

굳이 힌트를 안줘도 hash로 나온다.

문제108. not exists 문을 이용해서 부서 테이블에서 부서위치를 출력하는데 직원테이블에 존재하는 부서번호에 대한 부서위치만 출력하시오.

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		4
HASH JOIN ANTI		4
TABLE ACCESS FULL	SCOTT.DEPT	4
TABLE ACCESS FULL	SCOTT.EMP	14

만약 hash join anti로 만나오면 힌트를 준다.

3-5. push_pred, no_push_pred

1.push_pred : 뷰나 인라인 뷰 바깥에 있는 조인조건을 뷰나 인라인 뷰 안으로 집어넣는 힛트.

2.no_push_pred : 뷰나 인라인 뷰 바깥에 있는 조인조건을 뷰나 인라인 뷰 안으로 집어넣지 않게하는 힌트.

```
CREATE view DEPT_avgsal
AS
    SELECT deptno, avg(sal) as avgсал
    FROM emp
    GROUP BY deptno;
```

부서위치, 그 부서의 평균월급이 출력되게 하시오.

	DEPTNO	LOC	AVGSAL
1	30	CHICAGO	1566.6666666666666666666666666667

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
HASH GROUP BY		1
HASH JOIN		5
TABLE ACCESS FULL	SCOTT.DEPT	1
TABLE ACCESS FULL	SCOTT.EMP	14

문제111. 문제110번을 다시 수행하는데 뷰를 해체하지 말고 수행하게 하시오.

```
SELECT /*+ no_merge(v) */v.deptno, d.loc, v.avgсал
FROM dept_avgсал v, DEPT d
WHERE v.DEPTNO = d.DEPTNO
AND d.loc = 'CHICAGO';
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
HASH JOIN		1
JOIN FILTER CREATE	SYS.:BF0000	1
TABLE ACCESS FULL	SCOTT.DEPT	1
VIEW	SCOTT.DEPT_AVGSAL	3
HASH GROUP BY		3
JOIN FILTER USE	SYS.:BF0000	14
TABLE ACCESS FULL	SCOTT.EMP	14

문제112. 아래의 인덱스 2개를 각각 수행하고 문제 111번의 실행계획을 확인하시오.

```
CREATE INDEX emp_deptno ON EMP(deptno);
CREATE INDEX dept_loc ON DEPT(loc);
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
NESTED LOOPS		1
TABLE ACCESS BY INDEX ROWID	SCOTT.DEPT	1
INDEX RANGE SCAN	SCOTT.DEPT_LOC	1
VIEW PUSHED PREDICATE	SCOTT.DEPT_AVGSAL	1
FILTER		
SORT AGGREGATE		1
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	5
INDEX RANGE SCAN	SCOTT.EMP_DEPTNO	5

Pushed predicate 라는 실행계획이 뷰 밖의 조건을 뷰 안으로 집어 넣어서 group by 되는 건수를 줄이는 부분

문제113. 아래의 뷰 밖의 조건이 뷰 안으로 들어가지 못하도록 힌트를 주시오.(튜닝전으로)

```
SELECT /*+ no_merge(v) */v.deptno, d.loc, v.avgсал
FROM dept_avgсал v, DEPT d
WHERE v.DEPTNO = d.DEPTNO
AND d.loc = 'CHICAGO';
```

```
SELECT /*+ no_merge(v) no_push_pred(v) */v.deptno, d.loc, v.avgсал
FROM dept_avgсал v, DEPT d
WHERE v.DEPTNO = d.DEPTNO
AND d.loc = 'CHICAGO';
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
HASH JOIN		1
JOIN FILTER CREATE	SYS.:BF0000	1
TABLE ACCESS BY INDEX ROWID	SCOTT.DEPT	1
INDEX RANGE SCAN	SCOTT.DEPT_LOC	1
VIEW	SCOTT.DEPT_AVGSAL	3
HASH GROUP BY		3
JOIN FILTER USE	SYS.:BF0000	14
TABLE ACCESS FULL	SCOTT.EMP	14

위에 테이블을 만들때

```
CREATE view DEPT_avgsal
AS
  SELECT deptno, avg(sal) as avgsal
    FROM emp
   GROUP BY deptno;
```

이렇게 만들었는데 **push_pred**를 쓰면 'CHICAGO' 가아니라 deptno = 30 이들어간다.

```
CREATE view DEPT_avgsal
AS
  SELECT deptno, avg(sal)
    FROM emp
   where deptno = '30'
   GROUP BY deptno;
```

Demobld 돌리고 인덱스 생성후

문제114. 아래의 SQL의 실행계획을 인라인 뷰 바깥에 있는 조인 조건을 인라인 뷰 안쪽으로 집어넣지 않도록 하시오.

```
SELECT d.dname, e.*
  FROM DEPT d,
       (SELECT deptno, empno, ename, job, sal, sal*1.1 AS sal2, hiredate
        FROM EMP
        WHERE job = 'CLERK'
       UNION ALL
       SELECT deptno, empno, ename, job, sal, sal*1.1 AS sal2, hiredate
        FROM EMP
        WHERE job = 'SALESMAN') e
 WHERE e.deptno = d.DEPTNO
    AND d.loc = 'CHICAGO';
```

위에 조인조건이 각각 들어가서 인덱스를 탈 수 있다. Where 절에 deptno= 30이 추가된다.

```
(SELECT deptno, empno, ename, job, sal, sal*1.1 AS sal2, hiredate
  FROM EMP
 WHERE job = 'CLERK' and deptno = 30
 UNION ALL
 SELECT deptno, empno, ename, job, sal, sal*1.1 AS sal2, hiredate
  FROM EMP
 WHERE job = 'SALESMAN' and deptno = 30 ) e
```

답)

```
SELECT /*+ no_push_pred(e) */d.dname, e.*
  FROM DEPT d,
       (SELECT deptno, empno, ename, job, sal, sal*1.1 AS sal2, hiredate
        FROM EMP
        WHERE job = 'CLERK'
       UNION ALL
       SELECT deptno, empno, ename, job, sal, sal*1.1 AS sal2, hiredate
        FROM EMP
        WHERE job = 'SALESMAN') e
 WHERE e.deptno = d.DEPTNO
    AND d.loc = 'CHICAGO';
```


Operation	Object Name	Rows	Bytes
SELECT STATEMENT Optimizer Mode=ALL_ROWS		2	208
HASH JOIN		2	208
TABLE ACCESS FULL	SCOTT.DEPT	1	30
VIEW		8	592
UNION-ALL			
TABLE ACCESS FULL	SCOTT.EMP	4	244
TABLE ACCESS FULL	SCOTT.EMP	4	244

현업에서는 '답)' 처럼 나오는데 반대로 push_pred해서 인덱스를 타게 튜닝을해준다.

```
SELECT /*+ push_pred(e) */d.dname, e.*
  FROM DEPT d,
       (SELECT deptno, empno, ename, job, sal, sal*1.1 AS sal2, hiredate
        FROM EMP
        WHERE job = 'CLERK'
       UNION ALL
       SELECT deptno, empno, ename, job, sal, sal*1.1 AS sal2, hiredate
        FROM EMP
        WHERE job = 'SALESMAN') e
 WHERE e.deptno = d.DEPTNO
        AND d.loc = 'CHICAGO';
```

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		1
NESTED LOOPS		1
TABLE ACCESS FULL	SCOTT.DEPT	1
VIEW		1
UNION ALL PUSHED PREDICATE		
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	1
INDEX RANGE SCAN	SCOTT.EMP_DEPTNO_JOB	2
TABLE ACCESS BY INDEX ROWID	SCOTT.EMP	1
INDEX RANGE SCAN	SCOTT.EMP_DEPTNO_JOB	2

3-6. 스칼라 서브쿼리 문장의 튜닝

: select 문에서 서브쿼리를 쓸 수 있는 절

Select --- 스칼라 서브쿼리
 from --- 인라인 뷰
 where --- 서브쿼리
 group by --- 서브쿼리 사용 못함
 having --- 서브쿼리
 order by --- 스칼라 서브쿼리

문제115. 이름, 월급 사원테이블의 최대월급, 최소월급, 평균월급을 출력하시오.

```
SELECT ename, sal, MAX(sal) OVER() AS 최대,
       min(sal) OVER() AS 최소,
       AVG(sal) OVER() AS 평균
  FROM EMP;
```

<input checked="" type="checkbox"/>	ENAME	SAL	최대	최소	평균
-------------------------------------	-------	-----	----	----	----

☐	ENAME	SAL	최대	최소	평균
1	KING	5000	5000	800	2073.21
2	BLAKE	2850	5000	800	2073.21
3	CLARK	2450	5000	800	2073.21
4	JONES	2975	5000	800	2073.21
5	MARTIN	1250	5000	800	2073.21
6	ALLEN	1600	5000	800	2073.21
7	TURNER	1500	5000	800	2073.21
8	JAMES	950	5000	800	2073.21
9	WARD	1250	5000	800	2073.21
10	FORD	3000	5000	800	2073.21
11	SMITH	800	5000	800	2073.21
12	SCOTT	3000	5000	800	2073.21
13	ADAMS	1100	5000	800	2073.21
14	MILLER	1300	5000	800	2073.21

문제116. 문제115번을 분석함수를 쓰지말고 출력하시오.

튜닝 후)

```
SELECT ename, sal, substr(salary,1,10) 최대,
        SUBSTR(salary,11,10) 최소,
        SUBSTR(salary,21,10) 평균
FROM (
    SELECT ename, sal, (SELECT RPAD(MAX(sal),10,' ') ||
                        RPAD(MIN(sal),10,' ') ||
                        ROUND(RPAD(AVG(sal),10,' '))
                      FROM EMP
                      ) AS salary
    FROM EMP
);
```

☐	ENAME	SAL	최대	최소	평균
1	KING	5000	5000	800	2073
2	BLAKE	2850	5000	800	2073
3	CLARK	2450	5000	800	2073
4	JONES	2975	5000	800	2073
5	MARTIN	1250	5000	800	2073
6	ALLEN	1600	5000	800	2073
7	TURNER	1500	5000	800	2073
8	JAMES	950	5000	800	2073
9	WARD	1250	5000	800	2073
10	FORD	3000	5000	800	2073
11	SMITH	800	5000	800	2073
12	SCOTT	3000	5000	800	2073
13	ADAMS	1100	5000	800	2073
14	MILLER	1300	5000	800	2073

Emp 테이블을 4번이나 select 한다

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	14

문제117. 문제116번을 다시 수행하는데 튜닝전 SQL로 작성하시오.

튜닝 전)

```
SELECT ename, sal,
       (SELECT max(sal) FROM emp) 최대,
       (SELECT MIN(sal) FROM emp) 최소,
       (SELECT round(AVG(sal)) FROM EMP) 평균
FROM EMP;
```

Operation	Object Name
SELECT STATEMENT Optimizer Mode=ALL_ROWS	
SORT AGGREGATE	
TABLE ACCESS FULL	SCOTT.EMP
SORT AGGREGATE	
TABLE ACCESS FULL	SCOTT.EMP
SORT AGGREGATE	
TABLE ACCESS FULL	SCOTT.EMP
TABLE ACCESS FULL	SCOTT.EMP

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	67

문제118. 아래의 SQL을 튜닝하시오. (demobld 돌리고)

```
SELECT ename, sal, deptno,
       (SELECT max(sal) FROM emp
        where deptno = outer.deptno) 최대,
       (SELECT MIN(sal) FROM emp
        where deptno = outer.deptno) 최소,
       (SELECT round(AVG(sal)) FROM EMP
        where deptno = outer.deptno) 평균
FROM EMP outer;
```

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	109

-----튜닝 후

```
SELECT /*+ index(e emp_deptno) */e.ename, e.sal, EE.*
FROM EMP e,( SELECT /*+ index(emp emp_deptno) */deptno, MAX(sal), MIN(sal), ROUND(AVG(sal))
              FROM EMP
              WHERE deptno >= 0
              GROUP BY deptno ) ee
WHERE e.deptno >= 0;
```

3-7. 수정 가능한 조인 뷰

: 테이블의 데이터를 갱신할 때 뷰를 이용해서 갱신 하는 방법으로
SQL튜닝을 위해 사용하는 기술.

Ex) 사원 테이블에 loc컬럼을 추가하고 아래의 view를 만드시오.

```
ALTER TABLE EMP
  ADD loc VARCHAR2(20);
```

```
CREATE VIEW emp700
AS
  SELECT e.ename, e.loc AS emp_loc,
         d.loc AS dept_loc
  FROM EMP e, DEPT d
  WHERE e.deptno = d.DEPTNo;
```

```
SELECT ename, emp_loc, dept_loc
FROM emp700;
```

	ENAME	EMP_LOC	DEPT_LOC
1	KING	(null)	NEW YORK
2	BLAKE	(null)	CHICAGO
3	CLARK	(null)	NEW YORK
4	JONES	(null)	DALLAS
5	MARTIN	(null)	CHICAGO
6	ALLEN	(null)	CHICAGO
7	TURNER	(null)	CHICAGO
8	JAMES	(null)	CHICAGO
9	WARD	(null)	CHICAGO
10	FORD	(null)	DALLAS
11	SMITH	(null)	DALLAS
12	SCOTT	(null)	DALLAS
13	ADAMS	(null)	DALLAS
14	MILLER	(null)	NEW YORK

문제119. emp_loc 컬럼의 데이터를 dept_loc 컬럼의 데이터로 변경하시오.

```
ALTER TABLE DEPT
```

```
  ADD CONSTRAINT dept_deptno_pk PRIMARY KEY(deptno);  <-- pk제약을 먼저 걸어주고
```

```
UPDATE emp700
  SET emp_loc = dept_loc;
```

문제120. 다시 demobld를 돌리고 emp 테이블의 loc 컬럼의 데이터를 dept 테이블의 loc 컬럼의 데이터로

변경 하는데 merge 문을 이용해서 수행하시오.

```
ALTER TABLE EMP
  ADD loc VARCHAR2(20);
```

```
MERGE INTO EMP e
  USING DEPT d
  ON(e.deptno = d.deptno)
  WHEN matched THEN
    UPDATE set e.loc = d.loc;
```

%% emp 테이블에 추가된 loc 컬럼을 dept 테이블의 loc 컬럼의 데이터로 변경하는 3가지 방법

1. 상호관련 서브쿼리를 이용해서 변경 (악성 SQL인데 많이쓴다...)
2. 수정가능한 조인 뷰를 이용해서 변경 (뷰를 생성 해야하고, primary key 제약을 생성해야한다)

3. Merge 문을 이용해서 변경 (뷰 생성 x, 제약 x, 속도가 빠르고 그 자리에서 바로 할 수 있다.)

문제121. emp테이블의 loc 컬럼을 dept 테이블의 loc컬럼의 데이터로 변경하는데 상호관련 서브쿼리를 이용 해서 변경하시오.

```
UPDATE EMP e
    SET loc = (SELECT loc
                FROM DEPT d
                WHERE d.deptno = e.deptno);
```

```
SELECT ename, loc FROM EMP;
```

	ENAME	LOC
1	KING	NEW YORK
2	BLAKE	CHICAGO
3	CLARK	NEW YORK
4	JONES	DALLAS
5	MARTIN	CHICAGO
6	ALLEN	CHICAGO
7	TURNER	CHICAGO
8	JAMES	CHICAGO
9	WARD	CHICAGO
10	FORD	DALLAS
11	SMITH	DALLAS
12	SCOTT	DALLAS
13	ADAMS	DALLAS
14	MILLER	NEW YORK

==> 총 14건이 있어서 14번 갱신을 하는데, 데이터가 많으면 많이 갱신해서 악성 SQL이다.

%%merge문 실습을 위한 스크립트 생성

```
CREATE TABLE sales300
AS
SELECT * FROM sh.SALES;
```

```
CREATE TABLE sales400
AS
SELECT ROWNUM rn, prod_id, cust_id, time_id, channel_id,
       promo_id, quantity_sold, amount_sold
FROM sh.sales;
```

```
ALTER TABLE sales400
ADD date_id DATE;
```

```
CREATE TABLE time2
(rn number(10), date_id date);
```

```
BEGIN FOR i IN 1 .. 918843 LOOP
    INSERT INTO time2
    VALUES(i, TO_DATE('1961/01/02', 'YYYY/MM/DD') +i);
END LOOP;
END;
/
```

```
SELECT * FROM sales400 WHERE ROWNUM < 10;
```

RN	PROD_ID	CUST_ID	TIME_ID	CHANNEL_ID	PROMO_ID	QUANTITY_SOLD	AMOUNT_SOLD	DATE_ID
1	13	987	1998-01-10 오전 12:00:00	3	999	1	1232.16	(null)
2	13	1660	1998-01-10 오전 12:00:00	3	999	1	1232.16	(null)
3	13	1762	1998-01-10 오전 12:00:00	3	999	1	1232.16	(null)
4	13	1843	1998-01-10 오전 12:00:00	3	999	1	1232.16	(null)
5	13	1948	1998-01-10 오전 12:00:00	3	999	1	1232.16	(null)
6	13	2273	1998-01-10 오전 12:00:00	3	999	1	1232.16	(null)
7	13	2380	1998-01-10 오전 12:00:00	3	999	1	1232.16	(null)
8	13	2683	1998-01-10 오전 12:00:00	3	999	1	1232.16	(null)
9	13	2865	1998-01-10 오전 12:00:00	3	999	1	1232.16	(null)

SELECT * FROM time2 WHERE ROWNUM < 10;

RN	DATE_ID
1	1715 1965-09-13 오전 12:00:00
2	1716 1965-09-14 오전 12:00:00
3	1717 1965-09-15 오전 12:00:00
4	1718 1965-09-16 오전 12:00:00
5	1719 1965-09-17 오전 12:00:00
6	1720 1965-09-18 오전 12:00:00
7	1721 1965-09-19 오전 12:00:00
8	1722 1965-09-20 오전 12:00:00
9	1723 1965-09-21 오전 12:00:00

문제122. sales400 의 date_id 컬럼을 time2의 date_id컬럼의 데이터로 변경하는데 상호관련
서브쿼리문으로 수행하시오.

```
UPDATE sales400 s400
  SET date_id = (SELECT date_id
                FROM time2 t2
                WHERE s400.rn = t2.rn);
```

이렇게하면 반나절이 지나도 안끝난다.

문제123. 다시 sales400의 date_id를 time2의 date_id로 변경하는데 수정가능한 조인뷰를 이용해서
변경하시오.

```
ALTER TABLE time2
  ADD CONSTRAINT time2_rn_pk primary KEY(rn);
```

```
CREATE VIEW sales400_view
AS
  SELECT s.rn, s.date_id AS s_date_id,
         t2.date_id AS t2_date_id
  FROM sales400 s, time2 t2
  WHERE s.rn = t2.rn;
```

```
UPDATE sales400_view
  SET s_date_id = t2_date_id;
```

Update	918843	18 초
--------	--------	------

18초 걸린다

이걸 한번에 실행하기위해
컬럼을 삭제하고 다시 추가해준다.

```
ALTER TABLE sales400
DROP COLUMN date_id;
```

```
ALTER TABLE sales400
ADD date_id DATE;
```

답)

```
UPDATE ( SELECT s.rn, s.date_id AS s_date_id,
              t2.date_id AS t2_date_id
        FROM sales400 s, time2 t2
        WHERE s.rn = t2.rn )
SET s_date_id = t2_date_id;
```

문제124. 다시 sales400의 date_id를 time2의 date_id로 변경하는데 merge문을 이용해서 변경하시오.

```
ALTER TABLE sales400
DROP COLUMN date_id;
```

```
ALTER TABLE sales400
add date_id DATE;
```

다시 컬럼을 삭제, 추가하고

답)

```
MERGE INTO SALES400 s400
USING time2 t2
ON (s400.rn = t2.rn)
WHEN matched THEN
UPDATE SET s400.date_id = t2.date_id;
```

Merge	918843	15 초
-------	--------	------

15초가 걸린다

3-8. 데이터 분석함수를 이용한 SQL 튜닝

: 데이터 분석함수를 이용하면 복잡한 쿼리를 간단하게 수행을 할 수 있다.
(코딩도 간단해지고 성능도 좋아짐)

Ex) 튜닝 전 :

```
SELECT deptno, SUM(sal)
FROM EMP
GROUP BY deptno
UNION
SELECT NULL, SUM(sal)
FROM EMP;
```

	DEPTNO	SUM(SAL)
1	10	8750
2	20	10875
3	30	9400
4	(null)	29025

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	55

튜닝 후 :

```
SELECT deptno, SUM(sal)
FROM EMP
GROUP BY ROLLUP(deptno);
```

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	48

문제125. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
SELECT deptno, null as job, SUM(sal)
FROM EMP
GROUP BY deptno
UNION
SELECT NULL as deptno, job, SUM(sal)
FROM EMP
GROUP BY job;
```

	DEPTNO	JOB	SUM(SAL)
1	10	(null)	8750
2	20	(null)	10875
3	30	(null)	9400
4	(null)	ANALYS	6000
5	(null)	CLERK	4150
6	(null)	MANAGE	8275
7	(null)	PRESIDE	5000
8	(null)	SALESM	5600

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	55


튜닝 후 :

```
SELECT deptno, job, SUM(sal)
FROM EMP
GROUP BY grouping sets(deptno, job);
```

자동 추적		
	속성	값
1	recursive calls	304
2	db block gets	24
3	consistent gets	144

이렇게 하면 쿼리는 짧은데 성능이 안좋아져서 쿼리 변환 힌트를 준다.


```
SELECT /*+ expand_gset_to_union */ deptno, job, SUM(sal)
FROM EMP
GROUP BY grouping sets(deptno, job); <-- grouping sets를 다시 union 으로 해라
```


 자동 추적

	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	55

문제126. (점심시간문제)아래의 SQL을 튜닝하시오.


튜닝 전 :

```
SELECT deptno, ename, sum(sal)
FROM EMP
GROUP BY deptno, ename
UNION ALL
SELECT null AS deptno, NULL AS ename, SUM(sal)
FROM EMP
ORDER BY deptno;
```

 자동 추적

	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	55


	DEPTNO	ENAME	SUM(SAL)
2	10	KING	5000
3	10	CLARK	2450
4	20	ADAMS	1100
5	20	SMITH	800
6	20	FORD	3000
7	20	JONES	2975
8	20	SCOTT	3000
9	30	MARTIN	1250
10	30	BLAKE	2850
11	30	JAMES	950
12	30	WARD	1250
13	30	ALLEN	1600
14	30	TURNER	1500
15	(null)	(null)	29025

 자동 추적

	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	55

튜닝 후 :

```
SELECT deptno, ename, sum(sal)
FROM EMP
GROUP BY ROLLUP((deptno, ename))
ORDER BY deptno;
```

 자동 추적

	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	48

문제127. 아래의 SQL을 튜닝하시오.

튜닝 전:

```
SELECT /*+leading(d e) use_nl(e) */e.ename, e.sal, e.deptno, d.avgsal
FROM EMP e, ( SELECT /*+ no_merge */deptno, AVG(sal) avgsal
FROM EMP
GROUP BY deptno) d
WHERE e.deptno = d.deptno;
```

튜닝 후 :

```
SELECT ename, sal, deptno, AVG(sal) OVER(PARTITION BY deptno) avgsal
FROM EMP;
```

문제128. 아래의SQL을 튜닝하시오.

튜닝 전 :

```
SELECT e.empno, e.ename, e2.job, e2.max_sal
FROM EMP e, ( SELECT job,MAX(sal) AS max_sal
FROM EMP
GROUP BY job ) e2
WHERE e.job=e2.job
AND e.sal = e2.max_sal;
```

튜닝 후 :

```
SELECT *
FROM (
SELECT empno, ename, job, sal, RANK() OVER(PARTITION BY job ORDER by sal desc) 순위
FROM EMP
)
WHERE 순위 = 1;
```

문제129. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
SELECT empno, ename, sal,
( SELECT sum(sal) FROM EMP e
WHERE e.empno <= b.empno) sumsal
FROM EMP b
ORDER BY empno;
```

	EMPNO	ENAME	SAL	SUMSAL
1	7369	SMITH	800	800
2	7499	ALLEN	1600	2400
3	7521	WARD	1250	3650
4	7566	JONES	2975	6625
5	7654	MARTIN	1250	7875
6	7698	BLAKE	2850	10725
7	7782	CLARK	2450	13175
8	7788	SCOTT	3000	16175
9	7839	KING	5000	21175
10	7844	TURNER	1500	22675
11	7876	ADAMS	1100	23775
12	7900	JAMES	950	24725
13	7902	FORD	3000	27725
14	7934	MILLER	1300	29025

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	144

자신의 사원번호가 이전의 사원번호보다 크면 더한다.

튜닝 후 :

```
SELECT empno, ename, sal,
       SUM(sal) over( ORDER BY empno ROWS BETWEEN unbounded preceding
                      AND CURRENT row) sumsal
FROM EMP;
```

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	46

Unbounded preceding : 맨 첫 번째 행

Unbounded following : 맨 마지막 행

Current row : 현재 행

Ex) unbounded preceding and unbounded following 을쓰면 로우의 총합이 나온다.

	EMPNO	ENAME	SAL	SUMSAL
1	7369	SMITH	800	29025
2	7499	ALLEN	1600	29025
3	7521	WARD	1250	29025
4	7566	JONES	2975	29025
5	7654	MARTIN	1250	29025
6	7698	BLAKE	2850	29025
7	7782	CLARK	2450	29025
8	7788	SCOTT	3000	29025
9	7839	KING	5000	29025
10	7844	TURNER	1500	29025
11	7876	ADAMS	1100	29025
12	7900	JAMES	950	29025
13	7902	FORD	3000	29025
14	7934	MILLER	1300	29025

문제130. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
SELECT deptno, empno, ename, sal,
       ( SELECT SUM(sal) FROM EMP e
         WHERE e.empno <= b.empno
         AND e.DEPTNO = b.deptno ) sumsal
FROM EMP b
ORDER BY deptno, empno;
```

<input type="checkbox"/>	DEPTNO	EMPNO	ENAME	SAL	SUMSAL
1	10	7782	CLARK	2450	2450
2	10	7839	KING	5000	7450
3	10	7934	MILLER	1300	8750
4	20	7369	SMITH	800	800
5	20	7566	JONES	2975	3775
6	20	7788	SCOTT	3000	6775
7	20	7876	ADAMS	1100	7875
8	20	7902	FORD	3000	10875
9	30	7499	ALLEN	1600	1600
10	30	7521	WARD	1250	2850
11	30	7654	MARTIN	1250	4100
12	30	7698	BLAKE	2850	6950
13	30	7844	TURNER	1500	8450
14	30	7900	JAMES	950	9400

튜닝 후 :

```
SELECT deptno, empno, ename, sal,
       SUM(sal) OVER (PARTITION BY deptno ORDER BY empno ROWS BETWEEN unbounded preceding
                      AND CURRENT row) sumsal
FROM EMP
ORDER BY deptno, empno;
```

문제131. 아래의 SQL을 튜닝하시오.

튜닝 전:

```
SELECT a.deptno, a.empno, a.ename, a.sal, b.sal
FROM ( SELECT ROWNUM no1, deptno, empno, ename, sal
      FROM (SELECT deptno, empno, ename, sal
            FROM EMP
            ORDER BY deptno, empno) ) a,
     (SELECT rownum+1 no2, deptno, empno, sal
      FROM (SELECT deptno, empno, ename, sal
            FROM EMP
            ORDER BY deptno, empno) ) b
WHERE a.no1 = b.no2(+)
ORDER BY no1;
```

<input type="checkbox"/>	DEPTNO	EMPNO	ENAME	SAL	SAL_1
1	10	7782	CLARK	2450	(null)
2	10	7839	KING	5000	2450
3	10	7934	MILLER	1300	5000
4	20	7369	SMITH	800	1300
5	20	7566	JONES	2975	800
6	20	7788	SCOTT	3000	2975
7	20	7876	ADAMS	1100	3000
8	20	7902	FORD	3000	1100
9	30	7499	ALLEN	1600	3000
10	30	7521	WARD	1250	1600
11	30	7654	MARTIN	1250	1250
12	30	7698	BLAKE	2850	1250
13	30	7844	TURNER	1500	2850
14	30	7900	JAMES	950	1500

튜닝 후 :

```
SELECT deptno, empno, ename, sal,
```

```
Lag(sal,1) OVER(ORDER BY deptno,empno) lag_sal
FROM EMP;
```

	DEPTNO	EMPNO	ENAME	SAL	SAL_1
1	10	7782	CLARK	2450	(null)
2	10	7839	KING	5000	2450
3	10	7934	MILLER	1300	5000
4	20	7369	SMITH	800	1300
5	20	7566	JONES	2975	800
6	20	7788	SCOTT	3000	2975
7	20	7876	ADAMS	1100	3000
8	20	7902	FORD	3000	1100
9	30	7499	ALLEN	1600	3000
10	30	7521	WARD	1250	1600
11	30	7654	MARTIN	1250	1250
12	30	7698	BLAKE	2850	1250
13	30	7844	TURNER	1500	2850
14	30	7900	JAMES	950	1500

문제132. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
SELECT DECODE(NO,1,e.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,e.deptno,2,null)
order BY DECODE(NO,1,e.deptno,2,null);
```

	DEPTNO	SUM(SAL)
1	10	8750
2	20	10875
3	30	9400
4	(null)	29025

튜닝 후 :

```
SELECT deptno, SUM(sal)
FROM EMP e
GROUP BY ROLLUP(deptno);
```

	DEPTNO	SUM(SAL)
1	10	8750
2	20	10875
3	30	9400
4	(null)	29025

문제133. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
SELECT ename, job
FROM EMP e1
WHERE 4 <= (SELECT count(*)
            FROM EMP e2
            WHERE e2.job = e1.job);
```

	ENAME	JOB
1	MARTIN	SALESMAN
2	ALLEN	SALESMAN
3	TURNER	SALESMAN
4	JAMES	CLERK
5	WARD	SALESMAN
6	SMITH	CLERK
7	ADAMS	CLERK
8	MILLER	CLERK

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	81

튜닝 후 :

```
SELECT ename, job
FROM(
    SELECT ename, job,
           COUNT(*) over(PARTITION BY job) cnt
    FROM EMP
)
WHERE cnt >= 4;
```

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	7

문제134. 아래의 SQL을 튜닝하시오.

```
CREATE TABLE times100
AS
SELECT * FROM sh.TIMES;
```

```
CREATE TABLE sales100
AS
SELECT * FROM sh.sales;
```

테이블을 생성 하고,

튜닝 전 :

```
SELECT * FROM times100 t
WHERE 20 < (SELECT COUNT(*)
            FROM sales100 s
            WHERE s.time_id = t.TIME_ID);
```

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	333182

튜닝 후 :

```
select *
```

```
FROM
    (select time_id,count(*) cnt
     from sales100 group by time_id) s, times100 t
where t.time_id=s.time_id and cnt >20;
```

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	4595

문제135. 문제134에서 튜닝후 SQL의 결과 데이터와 튜닝전 SQL의 결과 데이터가 같은지 확인.

```
select t.*
from
    (select time_id,count(*) cnt
     from sales100 group by time_id) s, times100 t
where t.time_id=s.time_id and cnt >20
minus
select *
from times100 t
where 20< (select count(*)
from sales100 s
where s.time_id=t.time_id);
```

3-9. WITH 절을 이용한 튜닝

: 비슷한 패턴의 SQL 이 하나의 SQL에서 여러 개로 사용되면서 성능이 느릴 때 유용한 튜닝방법.

Ex) 직업과 직업별 토탈월급을 출력하시오.

```
SELECT job, SUM(sal)
FROM EMP
GROUP BY job;
```

문제136. 직업별 토탈 월급들의 평균값을 출력하시오.

```
SELECT AVG(SUM(sal))
FROM EMP
GROUP BY job;
```

문제137. 직업과 직업별 토탈월급을 출력하는데, 직업별 토탈월급들에 대한 평균 값보다 더 큰것만 출력하시오.

```
SELECT job, SUM(sal)
FROM EMP
GROUP BY job
HAVING SUM(sal) > (SELECT AVG(SUM(sal))
FROM EMP
GROUP BY job);
```

문제138. 아래의 SQL을 WITH절로 수행하시오.

```
SELECT job, SUM(sal)
FROM EMP
```

```

GROUP BY job
HAVING SUM(sal) > (SELECT AVG(SUM(sal))
                   FROM EMP
                   GROUP BY job);

```

답)

```

WITH job_sum AS (SELECT job, SUM(sal) sumsal
                  FROM EMP
                  GROUP BY job)
SELECT job, sumsal
FROM job_sum
WHERE sumsal > (SELECT AVG(sumsal)
                FROM job_sum);

```

자동 추적		
	속성	값
1	recursive calls	2
2	db block gets	8
3	consistent gets	54

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
TEMP TABLE TRANSFORMATION		
LOAD AS SELECT	.SYS_TEMP_0FD9D6614_2EA4147	
HASH GROUP BY		14
TABLE ACCESS FULL	SCOTT.EMP	14
VIEW		14
TABLE ACCESS FULL	SYS.SYS_TEMP_0FD9D6614_2EA4147	14
SORT AGGREGATE		1
VIEW		14
TABLE ACCESS FULL	SYS.SYS_TEMP_0FD9D6614_2EA4147	14

문제139. 아래의 SQL을 with 절이 temp 테이블을 사용하지 않도록 힌트를 주시오.

```

WITH job_sum AS (SELECT /* inline */ job, SUM(sal) sumsal
                  FROM EMP
                  GROUP BY job)
SELECT job, sumsal
FROM job_sum
WHERE sumsal > (SELECT AVG(sumsal)
                FROM job_sum);

```

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	53

Operation	Object Name	Rows
SELECT STATEMENT Optimizer Mode=ALL_ROWS		14
FILTER		
HASH GROUP BY		14
TABLE ACCESS FULL	SCOTT.EMP	14
SORT AGGREGATE		1
VIEW		14
SORT GROUP BY		14
TABLE ACCESS FULL	SCOTT.EMP	14

문제140. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
WITH v AS (SELECT /*+ materialize */ deptno, SUM(sal) AS sumsal
            FROM EMP
            WHERE job = 'SALESMAN'
            GROUP BY deptno)
SELECT d.deptno, d.dname, v.sumsal
FROM DEPT d, v
WHERE d.DEPTNO = v.deptno
and v.sumsal = (SELECT max(v.sumsal) FROM v);
```

자동 추적		
	속성	값
1	recursive calls	2
2	db block gets	8
3	consistent gets	61

튜닝 후 :

```
WITH v AS (SELECT /*+ inline */ deptno, SUM(sal) AS sumsal
            FROM EMP
            WHERE job = 'SALESMAN'
            GROUP BY deptno)
SELECT d.deptno, d.dname, v.sumsal
FROM DEPT d, v
WHERE d.DEPTNO = v.deptno
and v.sumsal = (SELECT max(v.sumsal) FROM v);
```

자동 추적		
	속성	값
1	recursive calls	0
2	db block gets	0
3	consistent gets	53