

목차

2020년 11월 18일 수요일 오전 9:46

1. 인덱스 SQL 튜닝
 - a. Index range scan
 - b. Index unique scan
 - c. Index skip scan
 - d. Index full scan
 - e. Index fast full scan
 - f. Index merge scan
 - g. Index bitmap merge scan
 - h. Index join
2. 조인 SQL 튜닝
 - a. Nested loop 조인
 - b. Hash 조인
 - c. Sort merge 조인
 - d. 조인 순서의 중요성
 - e. Outer 조인 튜닝
 - f. 스칼라 서브쿼리를 이용한 조인
 - g. 조인을 내포한 DML문 튜닝
 - h. 고급 조인 문장 튜닝
3. 서브쿼리 문장 튜닝
4. 데이터 분석함수를 이용한 튜닝
5. 자동 SQL 튜닝

20.11.16

2020년 11월 16일 월요일 오전 9:39

데이터 분석가들에게 SQL 작성능력은 필수

데이터 분석가들이 데이터 분석을 하는 회사들의 데이터가 대부분 대용량 데이터이기 때문에 데이터를 검색하는 검색 속도가 많이 느리므로 빠른 데이터 분석을 위해서 SQL튜닝 기술도 잘 알고 있어야 한다.

SQL 튜닝이란 데이터 검색속도를 향상 시키는 기술

인덱스 SQL 튜닝

인덱스 액세스 방법 8가지

| 인덱스 액세스 방법 | 힌트 |
|-------------------------|---------------|
| Index range scan | Index |
| Index unique scan | Index |
| Index skip scan | Index_ss |
| Index full scan | Index_fs |
| Index fast full scan | Index_ffs |
| Index merge scan | And_equal |
| Index bitmap merge scan | Index_combine |
| Index join | Index_join |

오라클 힌트 : 오라클 옵티마이저가 SQL을 수행할 때 실행계획을 만드는데 실행계획을 SQL 사용자가 조정하는 명령어

실행계획은 옵티마이저라는 프로세서가 만든다.

옵티마이저에게 문법에 맞는 적절한 힌트를 주면 옵티마이저는 사용자가 요청한대로 실행계획을 만든다.

실행계획 보는 방법

1. Emp와 dept 테이블 재생성 스크립트를 돌린다.
2. 아래의 SQL의 실행계획을 확인한다.

Explain plan for

```
Select ename, sal  
  From emp  
 Where sal = 1300;
```

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

실행계획이 full table scan으로 나오면 emp 테이블을 처음부터 끝까지 모두 스캔 했다는 뜻이다.

위의 실행계획을 보는 방법은 SQL의 결과는 보지 못하고 단지 실행계획만 확인하는 방법이다.

SQL의 결과도 확인하면서 실행계획을 확인하는 방법

```
Select /*+ gather_plan_statistics */ ename, sal  
  From emp  
 Where sal = 1300;
```

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

맨 뒤에 나오는 buffer 의 갯수가 줄어들수록 튜닝이 잘 된것이다.

Ex)

위의 SQL을 튜닝하시오.

1. Emp 테이블에 sal에 인덱스를 생성한다.

```
Create index emp_sal  
  On emp(sal);
```

```
Select /*+ gather_plan_statistics */ ename, sal  
  From emp  
 Where sal = 1300;
```

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

인덱스를 생성하여도 full scan 할 경우 아래와 같이 SQL을 작성한다.

```
Create index emp_sal  
  On emp(sal);
```

```
Select /*+ gather_plan_statistics index(emp emp_sal) */ ename, sal  
  From emp  
 Where sal = 1300;
```

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

Buffer의 갯수가 7개에서 튜닝 후 2개로 줄어든 것을 확인 할 수 있다.

문제1. 아래의 SQL을 튜닝하시오.

```
Select empno, ename, sal, job  
  From emp  
 Where empno = 7788;
```

튜닝 전 : 실행계획 + buffer의 갯수

튜닝 후 : 실행계획 + buffer의 개수

튜닝 전 :

```
Select /*+ gather_plan_statistics */ empno, ename, sal, job  
  From emp  
 Where empno = 7788;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 후 :

```
Create index emp_empno  
  On emp(empno);
```

```
Select /*+ gather_plan_statistics */ empno, ename, sal, job  
  From emp  
 Where empno = 7788;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

알고리즘 문제22)

SQL로 알고리즘 문제 풀기 15(오일러 상수 자연상수 구하기)

```
With temp_table as (select level n  
                      From dual  
                     Connect by level <= 10000)  
Select power (1+(1/n),n)  
  From temp_table  
 Where n = 10000;
```

20.11.17

2020년 11월 17일 화요일 오전 9:45

SQLD : 개발자 (SQL 수업)

SQLP : SQL 튜너 (SQL 수업 + SQL 튜닝 수업)

ADSP : 데이터 분석가

SQL튜닝 목차

1. 인덱스 SQL 튜닝
2. 조인 문장 튜닝
3. 서브쿼리 문장 튜닝
4. 데이터 분석함수를 이용한 튜닝
5. 자동 SQL 튜닝

복습

Full table scan 과 index scan의 성능상의 차이를 실행계획을 통해서 경험을 했다.

인덱스 액세스 방법 8가지

| 인덱스 액세스 방법 | 힌트 |
|-------------------------|---------------|
| Index range scan | Index |
| Index unique scan | Index |
| Index skip scan | Index_ss |
| Index full scan | Index_fs |
| Index fast full scan | Index_ffs |
| Index merge scan | And_equal |
| Index bitmap merge scan | Index_combine |
| Index join | Index_join |

1. Index range scan

인덱스를 range(부분) 으로 먼저 스캔하고 인덱스에서 얻은 rowid로 테이블의 데이터를 액세스 하는 스캔 방법

Ex)

Index : 목차

Table : 책

Rowid : 책의 페이지 번호

인덱스가 없다면 원하는 데이터를 검색하기 위해서 full table scan을 할 수 밖에 없지만 인덱스가 있다면 인덱스를 통해서 먼저 데이터를 검색하고 인덱스의 rowid를 통해서 테이블을 액세스 할 수 있게 된다.

Ex)

```
Create index emp_ename  
On emp(ename);
```

```
Select /*+ gather_plan_statistics index(emp emp_ename) */ ename, sal  
From emp  
Where ename = 'BLAKE';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

실행계획 2가지

1. 예상 실행계획 : SQL을 실행하기 전에 실행계획을 예상

- Explain plan for
Select ename, sal
From emp
Where ename = 'BLAKE';

```
Select * from table(db_xplan.display);
```

2. 실제 실행계획 : SQL을 직접 실행하면서 나오는 실행계획 (권장)

- gather_plan_statistics
- index(emp emp_ename) : emp 테이블의 emp_ename의 인덱스를 액세스 하시오.
- Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));

문제2. 아래의 SQL을 튜닝하고 튜닝 전과 튜닝 후의 실행계획을 비교 하시오.(buffer 의 개수 확인)

Buffer의 개수 : SQL을 처리하기 위해서 읽어들인 메모리 블럭의 개수

튜닝 전 :

```
Create index emp_job on emp(job);
```

```
Select ename, sal, job  
From emp  
Where substr(job,1,5) = 'SALES';
```

```
Select /*+ gather_plan_statistics */ ename, sal, job
```

```
From emp  
Where substr(job,1,5) = 'SALES';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 후 :

```
Select /*+ gather_plan_statistics index(emp emp_job) */ ename, sal, job  
  From emp  
 Where job like 'SALES%';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

좌변을 가공하면 인덱스 스캔을 할 수 없다.

문제3. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Create index emp_sal on emp(sal);
```

```
Select /*+ gather_plan_statistics */ ename, sal  
  From emp  
 Where sal*12 = 36000;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 후 :

```
Select /*+ gather_plan_statistics */ ename, sal  
  From emp  
 Where sal = 36000/12;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제4. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Create index emp_hiredate on emp(hiredate);
```

```
Select /*+ gather_plan_statistics */ ename, hiredate  
  From emp  
 Where to_char(hiredate, 'RRRR') = '1980';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 후 :

```
Select /*+ gather_plan_statistics */ ename, hiredate  
  From emp  
 Where hiredate between to_date('1980/01/01', 'RRRR/MM/DD')  
       And to_date('1980/12/31', 'RRRR/MM/DD');
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제5. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Create index emp_empno on emp(empno);
```

```
Select /*+ gather_plan_statistics */ empno, ename, sal
  From emp
 Where empno || ename = '7788SCOTT';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 후 :

```
Select /*+ gather_plan_statistics */ empno, ename, sal
  From emp
 Where empno = 7788
   And ename = 'SCOTT';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제6. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Select /*+ gather_plan_statistics */ ename, sal
  From emp
 Where sal = '3000';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 후 :

```
Select /*+ gather_plan_statistics */ ename, sal
  From emp
 Where sal = 3000;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

2 - access("SAL"=3000)

위와 같이 오라클이 암시적으로 문자형을 숫자형으로 변경(숫자형이 문자형보다 우선순위가 더 높기 때문에)해주지만 아래의 SQL문이 올바른 표현이다.

문제7. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Select /*+ gather_plan_statistics */ ename, sal  
  From emp  
 Where sal like '30%';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 후 : 함수기반 인덱스를 생성해서 튜닝해야 한다.

```
Create index emp_sal_func  
  On emp(to_char(sal));
```

```
Select /*+ gather_plan_statistics index(emp emp_sal_func) */ ename, sal  
  From emp  
 Where sal like '30%';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

1 - filter(TO_CHAR("SAL") LIKE '30%')

오라클이 암시적으로 sal(숫자형)을 to_char(문자형)으로 형변환을 해주었기 때문에 좌변을 가공하지 않았지만 full scan한다.

SQL에서 조건절에 like를 자주 사용하는 쿼리의 테이블의 컬럼은 테이블 생성시 부터 숫자형이 아닌 문자형으로 만들어줘야 한다.

알고리즘 문제23)

문제8. 1부터 10000까지의 숫자 중에서 숫자 8이 총 몇 번 나오는가
(8이 포함되어 있는 숫자의 갯수를 카운팅 하는게 아니라 8이라는 숫자를 모두 카운팅 하시오.)

힌트 : 계층형 질의문, regexp_count 이용하면 쉽게 할 수 있다.

```
With temp_table as (select level n  
                      From dual  
                     Connect by level <= 10000)
```

```
Select sum(regexp_count (n, 8))  
  From temp_table;
```

암시적 형변환 사용시 주의 사항

```
Select ename, job, sal  
  From emp  
 Order by sal desc;
```

```
Select decode (job, 'PRESIDENT', null, sal)  
  From emp;
```

문제9. 위의 결과에서 최대값을 출력하시오.

```
Select max(decode(job, 'PRESIDENT', null, sal))
  From emp;
```

위의 SQL에서 결과 값이 3000 이 나오지 않고 950이 나온 이유는 암시적 형변환이 발생했기 때문이다.

Decode(job, 'PRESIDENT', null, sal)을 사용하게 되면 decode의 세번째 인자값이 null이면 네번째 인자값의 출력이 문자형으로 암시적 형변환이 발생한다.

Ex)

```
Select to_char(sal) salary
  From emp
 Order by salary desc;
```

월급을 문자형으로 변환하게 되면 숫자9가 앞에 있는 값이 제일 큰 값이 된다.

문제10. 아래의 SQL을 수정해서 암시적 형변환이 일어나지 않도록 SQL을 수정하시오.

```
Select max(decode(job, 'PRESIDENT', 0, sal))
  From emp;
```

- Index range scan
Non unique 한 인덱스를 엑세스하는 스캔 방법

인덱스의 종류 :

1. Unique 인덱스 : 값이 중복되지 않고 unique해야 생성되는 인덱스

Ex)

```
Create unique index emp_empno on emp(empno);
  - 생성 가능
```

```
Create unique index emp_deptno on emp(deptno);
  - 생성 불가능
```

2. Non unique 인덱스 : 값이 중복되어도 상관없이 생성되는 인덱스

문제11. 사원이름에 non unique index를 걸고 이름이 SCOTT인 사원의 이름과 월급을 조회하는 쿼리는 문의 실제 실행계획을 실행하시오.

```
Create index emp_ename on emp(ename);
```

```
Select /*+ gather_plan_statistics */ ename, sal  
  From emp  
 Where ename = 'SCOTT';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

```
|* 2 | INDEX RANGE SCAN | EMP_ENAME | 1 | 1 | 1 | 00:00:00.01 | 1 |
```

인덱스를 읽을 때 인덱스를 2개 이상 읽으면 index range scan 이 되고
하나만 읽으면 index unique scan이 된다.

문제12. 사원 테이블에 직업에 인덱스를 걸고 아래의 SQL을 수행해서
index range scan으로 실행계획이 나오는지 확인 하시오.

```
Create index emp_job on emp(job);
```

```
Select /*+ gather_plan_statistics */ ename, job  
  From emp  
 Where job = 'SALESMAN';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

2. Index unique scan

Unique 인덱스를 통해 테이블을 액세스하는 스캔 방법

Unique 인덱스는 중복된 데이터가 없어야지만 인덱스가 걸린다.

문제13. 사원번호가 7788번인 사원의 사원번호와 사원이름을 출력하는 쿼리문의 실행계획을 확인하시오.

```
Create unique index emp_empno on emp(empno);
```

```
Select /*+ gather_plan_statistics */ empno, ename  
  From emp  
 Where empno = 7788;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

```
|* 2 | INDEX UNIQUE SCAN | EMP_EMPNO | 1 | 1 | 1 | 00:00:00.01 | 1 |
```

Unique 인덱스는 인덱스에서 딱 한개의 데이터만 읽고 끝내기 때문에 non unique 인덱스보다
훨씬 좋은 인덱스이다.

SQLD 시험 - primary key제약이나 unique 제약을 컬럼에 걸면 unique인덱스가 자동으로 생성된다.

지금까지 만든 인덱스 리스트를 확인하시오.

```
Select index_name, uniqueness  
  From user_indexes  
 Where table_name = 'EMP';
```

문제14. dept 테이블에 deptno 에 primary key제약을 걸고
Dept 테이블의 deptno에 unique 인덱스가 자동으로 생성되었는지 확인하시오.

```
Alter table dept  
  Add constraint dept_deptno_pk primary key(deptno);
```

```
Select index_name, uniqueness  
  From user_indexes  
 Where table_name = 'DEPT';
```

DEPT_DEPTNO_PK UNIQUE
인덱스 이름이 제약이름으로 생성되었다.

문제15. dept 테이블에 loc에 non unique 인덱스를 생성하시오.

```
Create index dept_loc on dept(loc);
```

문제16. 아래의 SQL을 튜닝하시오.
(unique index를 타도록하시오.)

튜닝 전:

```
Select /*+ gather_plan_statistics */ *  
  From dept  
 Where deptno = 20  
   And loc = 'DALLAS';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제17. 아래의 SQL이 loc에 걸린 non unique 인덱스를 엑세스 하도록 힌트를 주시오.

```
Select /*+ gather_plan_statistics index(dept dept_loc) */ *
```

```
From dept
Where deptno = 20
And loc = 'DALLAS';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

3. Index full scan

인덱스 전체를 full로 읽어서 원하는 데이터를 액세스 하는 방법

Ex)

사원 테이블의 인원수가 전부 몇명인가

```
Select /*+ gather_plan_statistics index_fs(emp emp_empno) */ count(empno)
  From emp;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

사원수가 몇명인지 확인하는 쿼리의 성능을 높이기 위해서 emp_empno 인덱스를 full scan하여 사원수를 카운트 했고 버퍼의 갯수는 1개 이다.

문제18. 위의 SQL이 full table scan이 되도록 하시오.

```
Select /*+ gather_plan_statistics full(emp) */ count(*)
  From emp;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Full table scan은 buffer의 갯수가 7이고 index full scan은 버퍼의 갯수가 1로 7배의 성능이 좋았다는 효과를 볼 수 있다.

문제19. emp12테이블에 ename에 인덱스를 걸고 우리반 학생들의 인원 수를 출력하는 쿼리문의 성능을 높이시오.

```
Alter table emp12
  Drop constraint emp12_ename_un;
```

```
Create unique index emp12_ename on emp12(ename);
```

```
Select /*+ gather_plan_statistics index_fs(emp12 emp12_ename) */ count(ename)
  From emp12;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

4. Index fast full scan

인덱스를 처음부터 끝까지 스캔하는 방법은 index full scan과 같으나 index fast full scan이 index full scan과 다른점은 인덱스를 full로 읽을 때 multi block i/o를 한다.

Multi block i/o

- 책의 목차가 1~100 페이지라고 할 때,

Index full scan은 한 페이지씩 넘길 때 index fast full scan은 한번에 10페이지씩 넘기는 것과 같다.

Ex)

직업, 직별 인원수를 출력하시오.

Create index emp_job on emp(job);

```
Select /*+ gather_plan_statistics index_ffs(emp emp_job) */ job, count(job)
  From emp
 Group by job;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

- 위와 같이 index_ffs 힌트를 줘도 테이블을 full scan 하는 이유는 job 컬럼에 not null 제약이 없기 때문이다.

Index fast full scan을 하려면 not null 제약이 있어야 한다.

- Job에 not null 제약 생성하기

Alter table emp

Modify job constraint emp_job_nn not null;

문제20. 부서번호, 부서번호별 인원수를 출력하는데 index fast full scan이 될 수 있도록 인덱스도 걸고 not null 제약도 걸고 힌트도 부여 하시오.

Create index emp_deptno on emp(deptno);

Alter table emp

Modify deptno constraint emp_deptno_nn not null;

```
Select /*+ gather_plan_statistics index_ffs (emp emp_deptno) */ deptno, count(deptno)
  From emp
 Group by deptno;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Index full scan 과 index fast full scan의 차이점

1. Index full scan 보다는 index fast full scan 이 더 빠르다.
2. Index full scan은 데이터 정렬이 보장 되지만 index fast full scan은 결과로 출력되는 데이터의 레코드가 정렬되지 않는다.

문제21. emp12테이블에서 통신사를 출력하고 통신사별 인원수를 출력하는데 index fast full scan이 될 수 있도록 하시오.

```
Create index emp12_telecom on emp12(telecom);
```

```
Alter table emp12
  Modify telecom constraint emp12_telecom_nn not null;
```

```
Select /*+ gather_plan_statistics index_ffs(emp12 emp12_telecom) */ telecom, count(telecom)
  From emp12
 Group by telecom;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

알고리즘 문제24)

문제22. 2^{100} 의 각 자리수의 합은 ?

```
With temp_table as (select level n
                      From dual
                     Connect by level <= 9)
Select sum(regexp_count(power(2,100),n)*n)
  From temp_table;
```

20.11.18

2020년 11월 18일 수요일 오전 9:49

| 인덱스 액세스 방법 | 힌트 |
|-------------------------|---------------|
| Index range scan | Index |
| Index unique scan | Index |
| Index skip scan | Index_ss |
| Index full scan | Index_fs |
| Index fast full scan | Index_ffs |
| Index merge scan | And_equal |
| Index bitmap merge scan | Index_combine |
| Index join | Index_join |

1. 단일 컬럼 인덱스 : 컬럼을 하나로 해서 만든 인덱스

Ex)

```
Create index emp_deptno  
On emp(deptno);
```

2. 결합 컬럼 인덱스 : 컬럼을 여러개로 해서 만든 인덱스

Ex)

```
Create index emp_deptno_sal  
On emp(deptno, sal);
```

Emp_deptno_sal 결합 컬럼 인덱스의 구조 (컬럼값 + rowid)

```
Select deptno, sal, rowid  
From emp  
Where deptno >= 0;
```

Deptno 를 먼저 ascending 하게 정렬하고 그것을 기준으로 sal을 ascending 하게 정렬되게 인덱스를 구성하고 있다.

결합 컬럼 인덱스가 필요한 이유 :

인덱스에서 데이터를 읽어오면서 테이블 액세스를 줄일 수 있다.

(목차) (책)

검색하고자 하는 데이터가 인덱스에서 찾는다면 보다 빠르게 데이터를 검색 할 수 있다.

인덱스 목록 조회

```
Select index_name
```

```
  From user_indexes
```

```
  Where table_name = 'EMP';
```

단일 컬럼 인덱스를 통해서 테이블 엑세스

```
Select /*+ gather_plan_statistics index (emp emp_deptno) */ deptno, sal
```

```
  From emp
```

```
  Where deptno = 10;
```

```
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 | 2 |
| 1 TABLE ACCESS BY INDEX ROWID BATCHED | EMP | 1 | 3 | 3 | 00:00:00.01 | 2 |
| * 2 INDEX RANGE SCAN | EMP_DEPTNO | 1 | 3 | 3 | 00:00:00.01 | 1 |

결합 컬럼 인덱스를 통해서 테이블 엑세스

```
Select /*+ gather_plan_statistics index (emp emp_deptno_sal) */ deptno, sal
```

```
  From emp
```

```
  Where deptno = 10;
```

```
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 | 3 | 00:00:00.01 | 1 |
| * 1 INDEX RANGE SCAN | EMP_DEPTNO_SAL | 1 | 3 | 3 | 00:00:00.01 | 1 |

단일 컬럼 인덱스를 통해서 테이블을 엑세스 했을 때보다 결합 컬럼 인덱스를 통해서 테이블 엑세스를 했을 때 성능이 2배 좋아졌다.

(현업에서는 주로 단일 컬럼 인덱스 보다는 결합 컬럼 인덱스를 사용한다.)

문제23. 아래의 쿼리가 인덱스를 통해서만 데이터를 가져오고 테이블 엑세스를 하지 않도록 결합컬럼 인덱스를 생성하시오.

```
Select /*+ gather_plan_statistics */ empno, ename, sal, deptno
```

```
  From emp
```

```
  Where empno = 7788;
```

[Create index emp_index1](#)

```
On emp (empno, ename, sal, deptno);
```

| Id | Operation | Name | Starts | E-Rows | A-Rows | A-Time | Buffers |
|------------------------|------------|------|--------|--------|-------------|--------|---------|
| 0 SELECT STATEMENT | | | 1 | 1 | 00:00:00.01 | 1 | |
| * 1 INDEX RANGE SCAN | EMP_INDEX1 | | 1 | 1 | 00:00:00.01 | 1 | |

문제24. @demo 스크립트를 돌리고 아래의 SQL을 튜닝하시오.
(index fast full scan이 되도록 하시오.)

```
Select /*+ gather_plan_statistics */ job, count(job)
  From emp
 Group by job;
```

1. 인덱스 생성

```
Create index emp_job
  On emp(job);
```

2. Not null 제약생성

```
Alter table emp
  Modify job constraint emp_job_nn not null;
```

3. 쿼리문 작성

```
Select /*+ gather_plan_statistics index_ffs (emp emp_job) */ job, count(job)
  From emp
 Group by job;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제25. @demo 스크립트를 돌리고 아래의 SQL이 index fast full scan이 되도록 튜닝하시오.

튜닝 전 :

```
Select /* gather_plan_statisticcs */ deptno, count(empno)
  From emp
 Group by deptno;
```

튜닝 후 :

```
Create index emp_index1
  On emp(deptno, empno);
```

```
Alter table emp
  Modify deptno constraint emp_deptno_nn not null;
```

```
Select /* gather_plan_statistics index_ffs(emp emp_index1) */ deptno, count(empno)
```

```
From emp  
Group by deptno;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제26. @demo 를 돌리고 아래의 인덱스를 생성하고 아래의 SQL 을 튜닝하시오.

(min과 group by를 사용하지 않고 똑같은 결과를 출력하시오.)

튜닝 전 :

```
Select /*+ gather_plan_statistics */ deptno, min(sal)  
  From emp  
 Where deptno = 20  
 Group by deptno;
```

튜닝 후 :

```
Create index emp_deptno_sal  
  On emp(deptno, sal);
```

```
Select /*+ gather_plan_statistics index(emp emp_deptno_sal) */ deptno, sal  
  From emp  
 Where deptno = 20  
   And rownum = 1;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

실행계획에 sort라는 말이 나오면서 데이터를 정렬했다고 나오는 실행계획은 정렬하는데 시간이 걸리기 때문에 좋은 실행계획이 아니다.

문제27. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Select /*+ gather_plan_statistics */ deptno, max(sal)  
  From emp  
 Where deptno = 20  
 Group by deptno;
```

튜닝 후 :

```
Select /*+ gather_plan_statistics index_desc(emp emp_deptno_sal) */ deptno, sal  
  From emp  
 Where deptno = 20  
   And rownum = 1;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제28. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Select job, max(hiredate)
  From emp
 Where job = 'SALESMAN'
 Group by job;
```

튜닝 후 :

```
Create index emp_index2
  On emp(job, hiredate);
```

```
Select /*+ gather_plan_statistics index_desc(emp emp_index2) */ job, hiredate
  From emp
 Where job = 'SALESMAN'
   And rownum = 1;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

5. Index skip scan

인덱스를 full scan 또는 fast full scan 으로 전체 스캔하는것이 아니라 중간 중간 skip 해서 원하는 데이터를 검색하는 스캔 방법

검색할 필요가 없는 부분을 skip 하는 scan 방법

Emp_deptno_sal 인덱스의 구조

```
Create index emp_deptno_sal
  On emp(deptno, sal);
```

```
Select deptno, sal, rowid
  From emp
 Where deptno >= 0;
```

쿼리문에서 결합 컬럼 인덱스를 사용하려면 쿼리문의 where 절에 결합 컬럼 인덱스의 첫번째 컬럼이 where 절에 반드시 있어야 한다.

Emp_deptno_sal 에서 deptno가 반드시 where 절에 있어야 한다.

1. Emp_deptno_sal 결합 컬럼 인덱스의 첫번째 컬럼인 deptno 가 where 절에 있을 경우
Index range scan 으로 실행된다.

```
Select /*+ gather_plan_statistics */ ename, deptno, sal
  From emp
 Where deptno = 20;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

2. Emp_deptno_sal 결합 컬럼 인덱스의 두번째 컬럼이 where 절에 있으면 table full scan, index full scan 으로 실행된다. (성능이 느려진다.)

```
Select /*+ gather_plan_statistics */ ename, deptno, sal  
From emp  
Where sal = 3000;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

위와 같이 index range scan 으로 출력되지 않고 index full scan으로 출력될 경우 성능이 좋지 않으므로 이를 해결하기 위해서 index skip scan을 사용한다.

Index skip scan 의 원리

```
Select /*+ gather_plan_statistics */ ename, deptno, sal  
From emp  
Where sal = 3000;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Deptno와는 다르게 sal은 emp_deptno_sal 인덱스에서 deptno 처럼 정렬되어 있지 않기 때문에 월급이 3000 조회하기 위해 인덱스를 처음부터 끝까지 full 로 scan 해야 한다.

그러면서 불필요한 데이터까지 다 스캔을 해야하는 일이 벌어진다.

결합 컬럼 인덱스를 index range scan 할 수 있는 방법은 ?

- Deptno 를 선두 컬럼으로 두는게 아니고 sal을 선두 컬럼에 두고 인덱스를 다시 생성하면 된다.

```
Drop index emp_deptno_sal;
```

```
Create index emp_sal_deptno  
On emp(sal, deptno);
```

그러나 위와 같은 방법을 사용하면 다른 SQL문에 영향을 끼치기 때문에 index skip scan을 사용한다.

Index skip scan은 emp_deptno_sal 인덱스를 처음부터 끝까지 읽는것은 index full scan과 같지만 중간중간 검색을 skip 할 수 있다.

```
Select /*+ gather_plan_statistics index_ss(emp emp_deptno_sal) */ ename, deptno, sal  
  From emp  
 Where sal = 2975;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

예를 들어 deptno 10번 부터 조회해서 월급이 2975가 있는지 찾고 20번을 조회해서 월급이 2975까지만 검색하고 나머지는 skip 한다. 그리고 30번에 월급이 2975가 있는지 확인한다.

통계를 코드로 구현

이항분포1)

한개의 주사위를 360번 던져서 3의 배수의 눈이 나올 확률을 구하시오.

```
Select count(*)/360  
  From (select round(dbms_random.value(0.5, 6.5)) a  
        From dual  
       Connect by level <= 360)  
 Where a in (3, 6);
```

문제29. demo 스크립트를 돌리고 아래의 SQL이 index skip scan하도록 튜닝 하시오.

튜닝 전 :

```
Select /*+ gather_plan_statistics */ empno, ename, job, sal  
  From emp  
 Where sal = 1250;
```

튜닝 후 :

```
Create index emp_job_sal  
  On emp(job, sal);
```

```
Select /*+ gather_plan_statistics index_ss(emp emp_job_sal) */ empno, ename, job, sal  
  From emp  
 Where sal = 1250;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Index skip scan 이 효과를 보려면 결합 컬럼 인덱스의 첫번째 컬럼의 데이터의 종류가 몇 가지 안되어야 효과를 본다.

예를 들어 부서번호가 10, 20, 30 만 있을 때와

10, 20, 30, 40, ... , 1000 이 있을 때 전자가 skip 할 수 있는 확률이 더 높으므로 선두 컬럼의 데이터의 종류가 적어야 우리하다.

문제30. 직업에 종류가 몇가지 인지 확인하시오.

결과 : 5

```
Select count(distinct(job))
  From emp;
```

Index_asc와 index_desc 힌트를 사용해서 튜닝하는 방법

Order by 절을 너무 자주 사용하면 성능이 떨어진다.

(대용량 데이터 베이스 환경에서는 더욱 그렇다.)

문제 31. demo를 돌리고 아래의 인덱스를 생성한 후 아래의 SQL을 튜닝 하시오.

튜닝 전 :

```
Select /*+ gather_plan_statistics */ ename, sal
  From emp
 Order by sal desc;
```

튜닝 후 :

```
Create index emp_sal
  on emp(sal);
```

```
Select /*+ gather_plan_statistics index_desc (emp emp_sal) */ ename, sal
  From emp
 Where sal >= 0;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Index_desc 힌트가 작동하려면 인덱스 컬럼이 where 절에 검색조건으로 있어야 한다.

문제32. 아래의 SQL을 튜닝하시오. (실행계획에 sort가 있지 않게 하시오.)

튜닝 전 :

```
Select /*+ gather_plan_statistics */ max(sal)
  From emp;
```

튜닝 후 :

```
Create index emp_sal
  On emp(sal);
```

```
Select /*+ gather_plan_statistics index_desc(emp emp_sal) */ sal
  From emp
  Where sal >= 0
    And rownum = 1;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제33. 아래의 SQL을 튜닝하시오. (emp테이블을 한번만 엑세스해서 결과가 나오게 하시오.)

튜닝 전 :

```
Select /*+ gather_plan_statistics */ ename, sal
  From emp
  Where sal = (select max(sal)
    From emp);
```

튜닝 후 :

```
Select /*+ gather_plan_statistics index_desc(emp emp_sal) */ ename, sal
  From emp
  Where sal >= 0
    And rownum = 1;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제34. demo를 돌리고 아래의 SQL을 튜닝 하시오.
(order by 절을 사용하지 않고 출력하시오.)

튜닝 전 :

```
Select /* gather_plan_statistics */ empno, deptno, sal
  From emp
  Where deptno = 20
    Order by sal desc;
```

튜닝 후 :

```
Create index emp_deptno_sal
  On emp(deptno, sal);
```

```
Select /* gather_plan_statistics index_desc(emp emp_deptno_sal) */ empno, deptno, sal
```

```
From emp  
Where deptno = 20;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제35. demo를 돌리고 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Select /*+ gather_plan_statistics */ ename, hiredate  
  From emp  
 Where to_char(hiredate, 'RRRR') = '1981'  
 Order by hiredate desc;
```

튜닝 후 :

```
Create index emp_hiredate  
  On emp(hiredate);
```

```
Select /*+ gather_plan_statistics index_desc(emp emp_hiredate) */ ename, hiredate  
  From emp  
 Where hiredate between to_date('1981/01/01', 'RRRR/MM/DD')  
       And to_date('1981/12/31', 'RRRR/MM/DD');
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제36. demo를 돌리고 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Select /*+ gather_plan_statistics */ ename, job, sal  
  From emp  
 Where substr(job, 1, 5) = 'SALES'  
 Order by sal desc;
```

튜닝 후 :

```
Create index emp_job_sal  
  On emp(job, sal);
```

```
Select /*+ gather_plan_statistics index_desc (emp emp_job_sal) */ ename, job, sal  
  From emp  
 Where job like 'SALES%';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

6. Index merge scan

두개의 인덱스를 동시에 사용하여 하나의 인덱스만 사용했을 때보다 더 좋은 성능을 보이는 스캔 방법

Ex)

```
Create index emp_deptno on emp(deptno);
Create index emp_job on emp(job);
```

아래의 SQL의 실행계획을 확인하시오.

```
Select /*+ gather_plan_statistics */ ename, deptno, job
  From emp
 Where job = 'SALESMAN' and deptno = 30;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제37. 위의 SQL이 deptno에 걸린 인덱스를 타도록 힌트를 주고 실행하시오.

```
Select /*+ gather_plan_statistics index(emp emp_deptno) */ ename, deptno, job
  From emp
 Where job = 'SALESMAN' and deptno = 30;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제38. 이번에는 job에 걸린 인덱스를 타도록 힌트를 주고 실행하시오.

```
Select /*+ gather_plan_statistics index(emp emp_job) */ ename, deptno, job
  From emp
 Where job = 'SALESMAN' and deptno = 30;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Job과 deptno에 단일 컬럼 인덱스를 각각 걸었고 그 위의 SQL의 조건절에서는 두개의 컬럼을 다 사용해서 데이터를 검색하고 있다.

그러면 2개 중에 어느 하나의 인덱스를 사용해야하는데 어떤 컬럼의 인덱스를 사용하는게 더 성능에 좋은가 ?

인덱스를 길게 읽는 것보다 인덱스를 짧게 읽는게 더 좋은 인덱스이다.

```
Select count(*)
  From emp
 Where job = 'SALESMAN';
```

- 4개

```
Select count(*)
  From emp
 Where deptno = 30;
```

- 6개

위와 같은 상황일 때 두개의 인덱스를 모두 사용해서 더 큰 시너지 효과를 일으키는게 index merge scan 이다.

```
Select /*+ gather_plan_statistics and_equal(emp emp_job emp_deptno) */ ename, deptno, job  
From emp  
Where job = 'SALESMAN' and deptno = 30;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

위의 스캔 방법은 옛날 방법이고 현재는 index bitmap merge scan 이라는 더 좋은 스캔 방법이 나왔다.

7. Index bitmap merge scan

두개의 인덱스를 같이 스캔해서 시너지 효과를 보는 방법은 index gerge scan 과 동일 하지만 bitmap을 이용해서 인덱스의 사이즈를 줄인다. 그래서 인덱스를 스캔하는 시간이 짧아진다.

```
Select /*+ gather_plan_statistics index_combine(emp) */ ename, deptno, job  
From emp  
Where job = 'SALESMAN' and deptno = 30;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

8. Index join

인덱스끼리 조인을 해서 바로 결과를 보고 테이블 엑세스는 따로 하지 않는 스캔 방식

힌트 : /*+ index_join(emp emp_deptno dmp_job) */

```
Create index emp_deptno on emp(deptno);  
Create index emp_job on emp(job);
```

```
Select /*+ gather_plan_statistics index_join(emp emp_deptno dmp_job) */ deptno, job  
From emp  
Where job = 'SALESMAN' and deptno = 30;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

통계를 코드로 구현

평균 :

```
Select avg(sal) from emp;
```

분산 : 데이터의 퍼짐 정도

Select variance(sal) from emp;

표준편차 :

Select stddev(sal) from emp;

Select sqrt(variance(sal)) from emp;

루트 : select sqrt(4) from dual;

한개의 주사위를 36000 번 던져서 3의 배수의 눈이 나올 횟수를 X 라고 하자. 이 때 확률변수 X 의 평균값, 분산값, 표준편차를 구하시오.

이항분포의 평균값 : $np = 36000 \times 1/3 = 12000$

이항분포의 분산값 : $npq = 12000 \times 2/3 = 8000$

이항분포의 표준편차 : 이항분포의 분산값에 루트 씌운 값 = $40\sqrt{5}$

문제. 위의 이항분포의 평균값, 분산값, 표준편차를 SQL 코드로 구하시오.

Select avg(X)

From (Select count(*) X

From (select round(dbms_random.value(0.5, 6.5)) a

From dual

Connect by level <= 36000)

Where a in (3, 6))

Connect by level <= 10000;

조인 SQL 튜닝

1. Nested loop 조인
2. Hash 조인
3. Sort merge 조인
4. 조인 순서의 중요성
5. Outer 조인 튜닝
6. 스칼라 서브쿼리를 이용한 조인
7. 조인을 내포한 DML문 튜닝
8. 고급 조인 문장 튜닝

1. Nested loop 조인

SQL 시간에 배웠던 조인 문법 :

1. 1999 ANSI 조인 문법
2. 오라클 조인 문법
 - a. Equi join
 - b. Non equi join
 - c. Outer join
 - d. Self join

SQL 튜닝시간에 배우는 조인 방법 :

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

문제40. 사원 이름, 월급, 부서위치를 출력하는데 조인 순서가 어떻게 되는 것이 더 성능이 좋겠는가.

1. Emp -> dept
2. Dept -> emp
 - 작은 테이블(dept) 테이블을 먼저 읽고 큰 테이블(emp)이랑 조인하는 것이 더 성능이 좋다.

```
Select e.ename, e.sal, d.loc  
From emp e, dept d  
Where e.deptno = d.deptno;
```

문제41. 위의 조인 문장에 실행계획을 확인해서 어느 테이블을 먼저 읽고 조인 했는지 확인 하시오.

```
Select /*+gather_plan_statistics */ e.ename, e.sal, d.loc  
  From emp e, dept d  
 Where e.deptno = d.deptno;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

| | |
|---|--|
| | 0 SELECT STATEMENT 1 14 00:00:00.01 15 |
| * | 1 HASH JOIN 1 14 14 00:00:00.01 15 1797K 1797K 1112K (0) |
| | 2 TABLE ACCESS FULL DEPT 1 4 4 00:00:00.01 7 |
| | 3 TABLE ACCESS FULL EMP 1 14 14 00:00:00.01 7 |

실행계획을 보면 dept테이블 부터 읽고 emp와 조인한다.

조인 튜닝시 중요한 튜닝방법 2가지

1. 조인 순서를 조정
2. 조인 방법을 조정

문제42. 문제41번의 조인 문장의 조인 순서를 emp -> dept순으로 조인도록 조정하시오.

조인 순서를 조정하는 힌트 : ordered 힌트

Ordered 힌트는 from 절에서 기술한 테이블 순서대로 조인

```
Select /*+gather_plan_statistics ordered */ e.ename, e.sal, d.loc  
  From emp e, dept d  
 Where e.deptno = d.deptno;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

| | |
|---|---|
| | 0 SELECT STATEMENT 1 14 00:00:00.01 15 |
| * | 1 HASH JOIN 1 14 14 00:00:00.01 15 1651K 1651K 980K (0) |
| | 2 TABLE ACCESS FULL EMP 1 14 14 00:00:00.01 7 |
| | 3 TABLE ACCESS FULL DEPT 1 4 4 00:00:00.01 7 |

실행계획을 보면 emp테이블 부터 읽고 dept와 조인한다.

문제43. 다시 위의 SQL의 조인 순서를 dept -> emp 순이 되도록 힌트

를 주고 SQL을 작성하시오.

```
Select /*+gather_plan_statistics ordered */ e.ename, e.sal, d.loc  
  From dept d, emp e  
 Where e.deptno = d.deptno;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

기존 투닝 전 SQL에 힌트가 있으면 힌트를 먼저 빼고 수행해본다.

문제44. emp와 salgrade 테이블을 조인해서 이름과 월급과 급여등급을 출력하시오.

(성능이 좋게 조인 순서를 부여하시오.)

```
Select /*+ gather_plan_statistics ordered */ e.ename, e.sal, s.grade  
  From salgrade s, emp e  
 Where e.sal between s.losal and s.hisal;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

조인 순서를 결정하는 2가지 힌트

1. Ordered : from 절에서 기술한 테이블 순서대로 조인
2. Leading : leading 힌트 안에 사용한 테이블 순서대로 조인

```
Select /*+ gather_plan_statistics leading(s e) */ e.ename, e.sal, s.grade  
  From emp e, salgrade s  
 Where e.sal between s.losal and s.hisal;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제45. emp와 dept테이블을 조인하여 이름, 월급, 직업, 부서위치를 출력하시오.

(leading 힌트를 이용하여 조인순서를 정하시오.

Emp -> dept)

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

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

조인 투닝시 중요한 투닝 방법 2가지

1. 조인순서를 조정

- a. Ordered : from 절에서 기술한 테이블 순서대로 조인
 - b. Leading : leading 힌트 안에 쓴 테이블 순서대로 조인
2. 조인방법을 결정
- a. Nested loop join : 적은량의 데이터를 조인할 때 유리한 조인 방법
 - b. Hash join : 대용량 데이터를 조인할 때 유리한 조인 방법
 - c. Sort merge join : 대용량 데이터를 조인할 때 유리한 조인 방법
- Hash join 으로 수행되지 못하는 SQL에 사용

| | |
|------------------|-----------|
| Nested loop join | Use_nl |
| Hash join | Use_hash |
| Sort merge join | Use_merge |

작은 테이블을 조인할 때 hash join을 하면 메모리를 과다하게 사용하기 때문에 작은 테이블을 조인할 때는 nested loop join을 사용한다.

문제46. 문제45번의 쿼리의 실행계획이 nested loop 조인이 되게 힌트를 주시오.

```
Select /*+ gather_plan_statistics leading(e d) use_nl(e d) */ e.ename, e.sal, e.job, d.loc
  From emp e, dept d
 Where e.deptno = d.deptno;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Hash join 했을 때보다 버퍼가 늘어난 것을 볼 수 있다. 하지만 emp와 dept 테이블이 서로 조인되는 데이터의 양이 작으므로 nested loop join을 사용하는게 더 바람직하다.

문제47. 문제46번 쿼리의 조인문장의 실행계획이 nested loop join 하지만 조인 순서는 dept -> emp 순이 되도록 하시오.

```
Select /*+ gather_plan_statistics leading(d e) use_nl(d e) */ e.ename, e.sal, e.job, d.loc
  From emp e, dept d
 Where e.deptno = d.deptno;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제48. emp와 salgrade 테이블을 조인해서 이름, 월급, 급여등급을 출력하는 조인문장의 조인 순서와 조인방법을 아래와 같이 되게하시오.

Salgrade -> emp

조인 방법 nested loop join

```
Select /*+ gather_plan_statistics leading(s e) use_nl(s e) */ e.ename, e.sal, s.grade  
  From emp e, salgrade s  
 Where e.sal between s.loSal and s.hiSal;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제49. 이름이 SCOTT인 사원의 이름과 월급과 부서위치를 출력하는 조인 문장을 작성하시오.

```
Select e.ename, e.sal, d.loc  
  From emp e, dept d  
 Where e.deptno = d.deptno  
   And e.ename = 'SCOTT';
```

문제50. 위의 SQL은 조인순서가 아래의 2개 중 어떤게 더 좋은 순서인가.

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

Emp 테이블에서 먼저 이름이 SCOTT인 사원의 데이터를 1건만 읽고 dept 테이블과 1번만 조인하면 된다.

문제51. 아래의 조인문장의 조인순서는 emp -> dept로 조인하고 조인 방법은 nested loop join이 되도록 힌트를 주고 튜닝하시오.

```
Select /*+ gather_plan_statistics leading(e d) use_nl(e d) */ e.ename, e.sal, d.loc  
  From emp e, dept d  
 Where e.deptno = d.deptno  
   And e.ename = 'SCOTT';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

조인문장에서 검색조건이 따로 있으면 검색조건으로 검색되는 건수가 몇건인지 먼저 확인을 하고 그 건수가 적은 테이블을 먼저 읽어야 한다.

문제52. 아래의 SQL의 조인방법은 무조건 nested loop join으로 하지 만 조인 순서는 자유롭게 하시오.

```
Select e.ename, e.sal, d.loc  
  From emp e, dept d  
 Where e.deptno = d.deptno  
   And e.job = 'SALESMAN' --4건
```

And d.loc = 'CHICAGO'; --1건

```
Select /*+ gather_plan_statistics leading(d e) use_nl(d 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';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제53. 아래의 SQL이 적절한 실행계획이 나올 수 있도록 튜닝하시오.

튜닝 전 :

```
Select e.ename, e.sal, s.grade
  From emp e, salgrade s
 Where e.sal between s.losal and s.hisal
   And s.grade = 1;
```

튜닝 후 :

```
Select /*+ gather_plan_statistics leading(s e) use_nl(e) */ e.ename, e.sal, s.grade
  From emp e, salgrade s
 Where e.sal between s.losal and s.hisal
   And s.grade = 1;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Leading 힌트에 사용했던 테이블 별칭 중 두번째 별칭을 use_nl에 기술하면 된다.

문제54. emp와 dept와 salgrade를 조인해서 이름, 월급, 부서위치, 급여등급을 출력하시오.

```
Select 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;
```

문제55. 위의 SQL의 조인 순서와 방법을 아래와 같이 되게하시오.

Dept -> emp -> salgrade
Nested loop join

Dept -> salgrade -> dept (dept와 salgrade는 서로 연결고리가 없기 때문에 불가능)

```
Select /*+ gather_plan_statistics leading(d e s) use_nl(e) use_nl(s) */
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.loosal and s.hisal;

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Dept, emp, salgrade 순으로 조인하면서 dept와 emp가 조인 할 때 nested loop join 하고 dept와 emp가 조인한 결과와 salgrade와 조인할 때 nested loop join 한다.

문제56. 아래와 같이 조인 하시오.

Salgrade -> emp -> dept
Nested loop join

```
Select /*+ gather_plan_statistics 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.loosal and s.hisal;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제57. 문제56번의 조인문장의 실행 계획을 분석하시오.

| Id | Operation | Name | Starts | E-Rows | A-Rows | A-Time | Buffers | |
|--|-----------|------|--------|--------|--------|--------|---------|--|
| 0 SELECT STATEMENT 1 14 00:00:00.01 139 - 6 | | | | | | | | |
| 1 NESTED LOOPS 1 14 00:00:00.01 139 - 5 | | | | | | | | |
| 2 NESTED LOOPS 1 14 00:00:00.01 41 - 3 | | | | | | | | |
| 3 TABLE ACCESS FULL SALGRADE 1 5 00:00:00.01 6 - 1 | | | | | | | | |
| * 4 TABLE ACCESS FULL EMP 5 1 00:00:00.01 35 - 2 | | | | | | | | |
| * 5 TABLE ACCESS FULL DEPT 14 1 00:00:00.01 98 - 4 | | | | | | | | |

문제58. 이름이 KING인 사원의 이름, 월급, 부서위치, 급여등급을 출력 하시오.

```
Select /*+ gather_plan_statistics */ 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.loosal and s.hisal  
And e.ename = 'KING';
```

문제59. 위의 SQL의 조인 순서를 아래와 같이 하고 조인방법은 전부 nested loop join 하시오.

조인순서 : emp -> dept -> salgrade

Emp 테이블에서 ename이 KING인 데이터가 1건 이므로 1건만 읽어서 dept 테이블과 조인해서 부서위치를 가져오고 emp와 dept 테이블이 조인한 결과가 salgrade 테이블과 조인한다.

```
Select /*+ gather_plan_statistics leading(e d s) use_nl(d) use_nl(s) */  
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 e.ename = 'KING';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제60. CHICAGO에서 근무하는 사원들의 이름, 부서위치, 월급, 급여 등급을 출력하시오.

```
Select e.ename, d.loc, e.sal, s.grade  
From emp e, dept d, salgrade s  
Where e.deptno = d.deptno  
And e.sal between s.losal and s.hisal  
And d.loc = 'CHICAGO';
```

문제61. 위의 SQL의 조인 방법은 nested loop join으로 하고 조인 순서는 직접 튜닝하시오.

```
Select /*+ gather_plan_statistics leading(d e s) use_nl(e) use_nl(s) */  
e.ename, d.loc, e.sal, s.grade  
From emp e, dept d, salgrade s  
Where e.deptno = d.deptno  
And e.sal between s.losal and s.hisal  
And d.loc = 'CHICAGO';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제62. 부서위치, 부서위치별 토탈월급을 출력하시오.

```
Select d.loc, sum(e.sal)  
From emp e, dept d  
Where e.deptno = d.deptno  
Group by d.loc;
```

문제63. 위의 SQL의 조인순서와 조인방법을 기술하고 수행하시오.

```
Select /*+ gather_plan_statistics leadin(d e) use_nl(e) */d.loc, sum(e.sal)  
From emp e, dept d  
Where e.deptno = d.deptno
```

Group by d.loc;

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

2. Hash join

Nested loop 조인은 순차적 반복 조인으로 한 레코드씩 순차적으로 조인을 진행하는 특징이 있다.

```
Select e.ename, d.loc  
  From emp e, dept d  
 Where e.deptno = d.deptno  
   And e.job = 'SALESMAN';
```

| | |
|--------|---------|
| MARTIN | CHICAGO |
| ALLEN | CHICAGO |
| TURNER | CHICAGO |
| WARD | CHICAGO |

MARTIN, ALLEN 순으로 순차적으로 조인을 시도 하기 때문에 조인하려는 데이터의 양이 많으면 성능이 떨어진다.

대량의 데이터를 조인할 때 순차적으로 반복을 여러 번 반복해야 하므로 비효율적이다.

그래서 대량의 데이터를 조인할 때는 nested loop join 보다는 hash join을 사용하는 것이 훨씬 성능이 좋다.

Hash join은 해시 알고리즘을 이용해서 데이터를 메모리에 올려놓고 메모리에서 조인하는 조인 방법이다.

Ex)

```
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 * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

```
|* 1 | HASH JOIN      |  
| 2 | TABLE ACCESS FULL| DEPT | -> hash table (메모리로 올라간 테이블)  
| 3 | TABLE ACCESS FULL| EMP  | -> probe table (디스크에 있는 테이블)
```

Dept테이블과 emp테이블이 둘다 메모리로 올라가는데 메모리 한번에 올라가지 않고 둘 중 하나만 먼저 메모리로 올리고 다른테이블은 데이터의 일부를 조금씩 메모리로 올리면서 조인한다.

해시조인시 주의사항

- 메모리가 공간이 작으므로 emp와 dept 테이블 중 작은 테이블(dept) 를 올려야 한다.

문제64. 아래의 SQL을 튜닝하는데 해시조인되도록 하고 emp 테이블이 메모리로 올라가도록 하시오.

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

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제65. 직업이 SALESMAN인 사원들의 이름, 월급과 comm2를 출력하시오.

```
Select e.ename, e.sal, b.comm2  
  From emp e, bonus b  
 Where e.empno = b.empno  
   And e.job = 'SALESMAN';
```

문제66. 위의 SQL의 실행계획을 아래와 같이 나오게 하시오.

조인순서 : emp -> bonus
Hash join

```
Select /*+ gather_plan_statistics leading(e b) use_hash(b) */ e.ename, e.sal, b.comm2  
  From emp e, bonus b  
 Where e.empno = b.empno  
   And e.job = 'SALESMAN';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

위와 같이 emp와 bonus 둘 다 14건으로 테이블의 크기가 같다면 조건에 의해서 걸러지는 데이터가 작은 것을 메모리로 올리면 된다.

(직업이 SALESMAN인 사원들의 데이터가 4건이므로 4건만 메모리로 올리고 bonus와 조인한다.)

문제67. 아래의 join 문장에 hash join하고 조인순서를 직접 결정하시오.

```
Select e.ename, d.loc  
  From emp e, dept d  
 Where e.deptno = d.deptno  
   And e.job = 'SALESMAN'  
   And d.loc = 'CHICAGO';
```

```
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';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제68. emp와 dept와 bonus를 조인해서 이름, 부서위치, comm2를 출력하시오.

```
Select e.ename, d.loc, b.comm2  
From emp e, dept d, bonus b  
Where e.deptno = d.deptno  
And e.empno = b.empno;
```

문제69. 위의 SQL의 조인순서와 조인방법을 아래와 같이 하시오.

조인순서 : dept -> emp -> bonus
Hash join

```
Select /*+ gather_plan_statistics leading(d e b) use_hash(e) use_hash(b) */  
e.ename, d.loc, b.comm2  
  From emp e, dept d, bonus b  
 Where e.deptno = d.deptno  
   And e.empno = b.empno;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제70. 위의 SQL의 실행계획을 아래와 같이 나오게하시오.

```
Select /*+ gather_plan_statistics leading(d e b)
      use_hash(e) use_hash(b) swap_join_inputs(b)*/
      e.ename, d.loc, b.comm?
```

```

From emp e, dept d, bonus b
Where e.deptno = d.deptno
And e.empno = b.empno;

```

Swap_join_inputs : hash table 을 결정하는 힌트

No_swap_join_inputs : probe table 을 결정하는 힌트

2개의 테이블을 조인할 때 hash table을 결정하는 것은 leading 힌트만으로도 가능하지만 3개 이상의 테이블을 조인할 때 hash table을 결정할 때는 leading 으로는 제어가 되지 않고 swap_join_inputs 를 사용해야 한다.

문제71. 아래와 같이 실행계획이 나오게하시오.

| | Id | Operation | Name | Starts | E-Rows | A-Rows | A-Time | Buffers | OMem | LMem | Used-Mem |
|----|----|-------------------|-------|--------|--------|--------|-------------|---------|-------|-------|-----------|
| ? | 0 | SELECT STATEMENT | | 1 | | 14 | 00:00:00.01 | 18 | | | |
|)* | 1 | HASH JOIN | | 1 | 14 | 14 | 00:00:00.01 | 18 | 1797K | 1797K | 992K (0) |
| ? | 2 | TABLE ACCESS FULL | DEPT | 1 | 4 | 4 | 00:00:00.01 | 7 | | | |
|)* | 3 | HASH JOIN | | 1 | 14 | 14 | 00:00:00.01 | 10 | 1995K | 1995K | 1464K (0) |
| ? | 4 | TABLE ACCESS FULL | BONUS | 1 | 14 | 14 | 00:00:00.01 | 2 | | | |
| ? | 5 | TABLE ACCESS FULL | EMP | 1 | 14 | 14 | 00:00:00.01 | 7 | | | |

```

Select /*+
gather_plan_statistics leading(b e d)
use_hash(e) use_hash(d) swap_join_inputs(d) */
e.ename, d.loc, b.comm2
From emp e, dept d, bonus b
Where e.deptno = d.deptno
And e.empno = b.empno;

```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제72. emp와 salgrade 테이블을 조인해서 이름과 월급과 급여등급을 출력하고 hash조인으로 유도하시오.

```

Select /* gather_plan_statistics 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;

```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

위의 SQL은 실행계획이 hash 조인으로 풀리지 않는다.

왜냐하면 hash 조인이 되려면 where 절의 join 조건이 equal 이어야한다.

Non equal join은 해시조인으로 수행되지 않는다.

따라서 위와 같은 SQL은 sort merge join을 사용하면 된다.

3. Sort merge join

조인하려는 데이터의 양이 많으면서 조인조건이 = 이 아닐 때 검색 성능을 높이기 위한 조인 방법 sort merge 조인을 수행하면 연결고리가 되는 키 컬럼(deptno)을 오라클이 알아서 정렬해놓고 조인을 한다.

Ex)

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

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Order by 절을 사용하지 않았는데 deptno가 정렬되어 있다.

Sort merge join이 deptno를 정렬하고 조인을 수행 한 것을 알 수 있다.
정렬을 해서 데이터를 모아놨기 때문에 조인이 빨라진다.

문제73. 아래의 SQL의 실행계획이 sort merge join이 되도록 하시오.

```
Select /*+ gather_plan_statistics 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;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제. 아래의 문제를 SQL로 구현하여 평균과 분산과 표준편차를 구하시오.

제제 02

두 개의 동전을 300번 던져서 둘 다 앞면이 나오는 횟수를 확률변수 X 라고 할 때, X 의 평균과 분산 및 표준편차를 구하여라.

풀이 두 개의 동전을 한 번 던져서 둘 다 앞면이 나올 확률은 $\frac{1}{4}$ 이고, 매회 시행은 독립시 행이다. 따라서 확률변수 X 는 이항분포 $B(300, \frac{1}{4})$ 을 따르므로

$$E(X) = 300 \times \frac{1}{4} = 75, V(X) = 300 \times \frac{1}{4} \times \frac{3}{4} = \frac{225}{4}$$

$$\sigma(X) = \sqrt{V(X)} = \sqrt{\frac{225}{4}} = \frac{15}{2}$$

답 $E(X) = 75, V(X) = \frac{225}{4}, \sigma(X) = \frac{15}{2}$

Select 300*(count(*)/300) 평균, 300*(count(*)/300)*(1-count(*)/300) 분산,

Sqrt(300*(count(*)/300)*(1-count(*)/300)) 표준편차

From (select round(dbms_random.value(1, 2)) a,

Round(dbms_random.value(1, 2)) b

From dual

Connect by level <= 300)

Where a = 1 and b = 1;

20.11.20

2020년 11월 20일 금요일 오전 9:42

복습

1. 인덱스 튜닝
2. 조인문장 튜닝
3. 서브쿼리문 튜닝
4. 데이터 분석함수를 이용한 튜닝
5. 자동 SQL 튜닝

조인문장 튜닝시 중요한 기술 2가지

1. 조인 순서
 - a. Ordered
 - b. Leading
2. 조인 방법
 - a. Nested loop join : use_nl
 - 적은양의 데이터로 조인
 - b. Hash join : use_hash
 - 대용량의 데이터로 조인
 - c. Sort merge join : use_merge
 - 대용량의 데이터로 조인

OLTP 서버 : 실시간 조회 처리를 위한 데이터가 저장

- Nested loop join을 주로 사용

DW 서버 : 10, 20 년 동안의 정보 저장

- Hash, merge join을 주로 사용
- 데이터 분석함수 주로 사용
- 파티션 테이블, 병렬처리

4. 조인 순서의 중요성

조인순서에 따라서 수행 성능이 달라질 수 있다.

조인순서는 조인하려는 시도 횟수가 적은 테이블을 driving 테이블로 선정해서 조인순서를 고려하는게 중요하다.

Nested loop

Table access full dept 선행 테이블 - driving table

Table access full emp 후행 테이블 - driven table

Hash join

Table access full dept 선행 테이블 - hash table (메모리로 올라가는 테이블)

Table access full emp 후행 테이블 - probe table (디스크에 있는 테이블)

어떤 조인이든 먼저 읽는 테이블의 크기가 작거나 검색조건절에 의해서 액세스되는 건수가 작은것을 먼저 읽는게 조인순서에 있어서 중요하다.

조인문장 튜닝 순서

Ex)

```
Select /*+ gather_plan_statistics leading(d e) use_nl(e) */ e.ename, d.loc  
  From emp e, dept d    -- emp14건 , dept 4건  
 Where e.deptno = d.deptno  
   And e.job = 'SALESMAN' -- 4건  
   And d.loc = 'CHICAGO' -- 1건
```

1. 조인순서를 정한다. - from 절의 테이블의 건수를 확인한다.
 - 검색조건의 건수도 확인한다.
2. 조인방법을 정한다. - 접속한 서버의 종류가 무엇인지 확인한다. (OLTP 서버, DW 서버)
 - 위와 같이 데이터건수를 확인해서 대용량이 아니면 nested loop join을 사용

문제74. 아래의 조인문장의 조인순서와 조인방법을 결정하시오.

```
Select e.ename, d.loc, e.sal  
  From emp e, dept d  
 Where e.deptno = d.deptno  
   And e.ename = 'KING';
```

```
Select /*+ gather_plan_statistics leading(e d) use_nl(d) */ e.ename, d.loc, e.sal  
  From emp e, dept d  
 Where e.deptno = d.deptno  
   And e.ename = 'KING';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제75. emp와 dept가 대용량 테이블이라고 가정하고 아래의 SQL의 조인순서와 조인방법을 결정하시오.

(emp는 12000 건, dept는 5400건)

```
Select e.ename, d.loc  
  From emp e, dept d  
 Where e.deptno = d.deptno;
```

```
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 * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Sort merge 조인은 내부적으로 정렬을 일으키는 단점이 있어서 굳이 deptno를 정렬해서 볼 필요가 없다면 hash조인을 사용한다.

문제76. 아래의 SQL의 조인순서와 조인방법을 결정하시오.

(emp와 salgrade가 대용량 테이블이고 DW서버라고 가정하시오.)

```
Select e.ename, e.sal, s.grade  
  From emp e, salgrade s  
 Where e.sal between s.losal and s.hisal  
 Order by e.sal asc;
```

```
Select /*+ gather_plan_statistics 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;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

5. Outer 조인 튜닝

Outer join의 조인 순서는 outer join sign이 없는 쪽에서 있는 쪽으로 순서가 고정이 되기 때문에 조인 순서를 변경하기가 어려워 튜닝이 힘든데 이를 개선할 수 있는 힌트가 있다.

Ex)

```
Insert into emp(empno, ename, sal, deptno)  
 Values (2921, 'JACK', 4500, 70);
```

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

결과에 JACK이 출력되면서 부서위치가 비어있는게 확인이 된다.

JACK이 결과로 출력되게 하려면 equi join으로 불가능하고 outer join을 사용해야 한다.

위의 실행계획을 확인해보면 emp테이블을 선행테이블로 읽는다.

왜냐하면 outer join은 outer 사인(sign)이 없는 쪽에서 있는 쪽으로 조인한다.

문제77. 아래의 조인순서를 dept -> emp 순이 되게 하시오.

```
Select /*+ gather_plan_statistics */ e.ename, d.loc  
  From emp e, dept d
```

```
Where e.deptno = d.deptno (+);
```

```
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 * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

조인순서를 변경하기 위해 leading 힌트를 사용하였지만 조인순서가 변경되지 않았다.
왜냐하면 outer join은 조인순서가 outer join 사인이 없는 쪽에서 있는 쪽으로 고정이 되기 때문이다.

문제78. 위의 outer 조인의 조인순서를 dept -> emp 순이 되게 하시오.

```
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 (+);
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Swap_join_inputs 를 사용해서 dept 테이블을 hash 테이블로 구성해서 dept 테이블을 먼저 드라빙 할 수 있도록 한다.

Outer join을 튜닝하려면 swap_join_inputs 은 해시조인시에만 사용할 수 있기 때문에 해시조인으로 수행해야 한다.

문제79. emp와 bonus를 조인해서 이름, comm2 를 출력하는데
outer join을 이용하여 JACK도 출력되게 하시오.

```
Select e.ename, b.comm2  
From emp e, bonus b  
Where e.empno = b.empno(+);
```

문제80. 위의 조인순서가 bonus -> emp 순이 되게 하시오.

```
Select /*+ gather_plan_statistics leading(b e) use_hash(e) swap_join_inputs(b) */  
e.ename, b.comm2  
From emp e, bonus b  
Where e.empno = b.empno(+);
```

6. 스칼라 서브쿼리를 이용한 조인

Select 문에서 서브쿼리를 쓸 수 있는 절

Select -> 서브쿼리 가능 : 스칼라 서브쿼리(scalar subquery)

From -> 서브쿼리 가능 : in line view

Where -> 서브쿼리 가능

Group by

Having -> 서브쿼리 가능

Order by -> 서브쿼리 가능 : 스칼라 서브쿼리(scalar subquery)

문제81. 이름, 월급, 사원 테이블에서의 최대월급을 출력하시오.

튜닝 전 :

```
Select /*+ gather_plan_statistics */
ename, sal, (select max(sal)
              From emp) 최대월급
      From emp;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 후 :

```
Select /*+ gather_plan_statistics */
ename, sal, max(sal) over() 최대월급
      From emp;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Emp테이블을 튜닝전에는 두번 엑세스 했는데 emp 테이블을 한번만 엑세스 해도 똑같은 결과를 출력할 수 있게 되었다.

문제82. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Select ename, sal,
       (select max(sal) from emp) 최대월급,
       (select min(sal) from emp) 최소월급
      From emp;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 후 :

```
Select /*+ gather_plan_statistics */
ename, sal,
max(sal) over() 최대월급,
min(sal) over() 최소월급
```

```
From emp;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제83. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Select /*+ gather_plan_statistics */
    e.deptno, e.ename, e.sal, v.부서평균
        From emp e, (select deptno, avg(sal) 부서평균
                        From emp
                        Group by deptno) v
    Where e.deptno = v.deptno and e.sal > v.부서평균;
```

튜닝 후 : emp테이블을 한번만 엑세스 해서 같은 결과가 출력되게 하시오.

```
Select /*+ gather_plan_statistics */
    Deptno, ename, sal, avg(sal) over (partition by deptno) 부서평균
        From emp
    Where sal > 부서평균;
```

위의 SQL을 수행하려면 from 절의 서브쿼리인 in line view를 사용해야 한다.

```
Select *
    From (Select /*+ gather_plan_statistics */
        Deptno, ename, sal, avg(sal) over (partition by deptno) 부서평균
            From emp)
    Where sal > 부서평균;
```

문제84. 부서번호, 이름, 월급, 순위를 출력하는데 순위가 부서번호별로 각각 월급이 높은 순서대로 순위를 부여해서 출력하고 순위가 1등인 사원들만 출력하시오.

```
Select *
    From (Select deptno, ename, sal, dense_rank() over(partition by deptno
                                                    order by sal desc) 순위
        From emp)
    Where 순위 = 1;
```

7. 조인을 내포한 DML문 튜닝

DML문을 이용한 서브쿼리문의 튜닝

Ex)

ALLEN의 월급을 KING의 월급으로 변경하시오.

Update emp

```
Set sal = (select sal
            From emp
            Where ename = 'KING')
Where ename = 'ALLEN';
```

문제85. 사원 테이블에 loc컬럼을 추가하고 해당 사원이 속한 부서위치로 값을 갱신하시오.

Alter table emp

```
Add loc varchar2(20);
```

튜닝 전 :

Update emp e

```
Set loc = (select loc
            From dept d
            Where e.deptno = d.deptno);
```

Emp 테이블의 컬럼이 서브쿼리 안으로 들어가게 되면 서브쿼리부터 실행되지 않고 메인 쿼리부터 실행 된다.

튜닝 후 :

Merge into emp e

Using dept d

On (e.deptno = d.deptno)

When matched then

Update set e.loc = d.loc;

튜닝 전과는 다르게 튜닝 후는 데이터를 한번에 갱신한다.

문제86. 사원 테이블에 sal2라는 컬럼을 추가하시오.

Alter table emp

```
Add sal2 number(10);
```

문제87. 지금 추가한 sal2에 해당 사원의 월급으로 값을 갱신하시오.

Update emp

```
Set sal2 = sal;
```

문제88. 아래의 복합뷰를 생성하고 뷰를 쿼리하시오.

Create view emp_dept

As

```
Select e.ename, e.loc emp_loc, d.loc dept_loc  
From emp e, dept d  
Where e.deptno = d.deptno;
```

```
Select * from emp_dept;
```

문제89. emp_dept 뷰를 수정해서 emp_loc의 값을 dept_loc의 값으로 갱신하시오.

```
Update emp_dept  
Set emp_loc = dept_loc;
```

문제90. 위의 update문이 수행되게 설정하시오.

위의 복합뷰를 갱신하려면 dept테이블에 deptno에 primary key 제약을 걸어주면 갱신될 수 있다.

```
Alter table dept  
Add constraint dept_deptno_pk primary key(deptno);
```

문제91. view를 만들 수 없다면 어떻게 해야하는가.

Update 문에서 서브쿼리를 쓸 수 있는 절

Update - 서브쿼리 사용가능
Set - 서브쿼리 사용가능
Where - 서브쿼리 사용가능

```
Update (Select e.ename, e.loc emp_loc, d.loc dept_loc  
From emp e, dept d  
Where e.deptno = d.deptno)  
Set emp_loc = dept_loc;
```

View를 만들 수 없는 상황이라면 update에 서브쿼리를 사용해서 뷰의 쿼리문을 직접 작성하고 set 절에서 update 하면 된다.

문제92. emp테이블에 dname 컬럼을 추가하고 해당 사원의 부서명으로 값을 갱신하시오.

공식처럼 외우기

```
Alter table emp  
Add dname varchar2(10);
```

```
Update (select e.dname emp_dname, d.dname dept_dname
        From emp e, dept d
        Where e.deptno = d.deptno)
Set emp_dname = dept_dname;
```

```
Merge into emp e
Using dept d
On (e.deptno = d.deptno)
When matched then
Update set e.dname = d.dname;
```

8. 고급 조인 문장 튜닝

뷰 안의 조인문장의 조인순서를 변경하고자 할때 사용하는 튜닝방법

Ex)

```
Drop view emp_dept;
```

```
Create view emp_dept
```

As

```
  Select e.empno, e.ename, e.sal, e.deptno, d.loc
    From emp e, dept d
   Where e.deptno = d.deptno;
```

```
Select * from emp_dept;
```

문제93. emp_dept view과 salgrade 테이블을 서로 조인해서 이름과 월급과 부서위치와 급여등급을 출력하시오.

```
Select v.ename, v.sal, v.loc, s.grade
  From emp_dept v, salgrade s
 Where v.sal between s.losal and s.hisal;
```

문제94. 위의 SQL의 실제 실행계획을 확인하시오.

```
Select /*+ gather_plan_statistics leading(s v) use_nl(v) */ v.ename, v.sal, v.loc, s.grade
  From emp_dept v, salgrade s
 Where v.sal between s.losal and s.hisal;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

위의 힌트에서는 nested loop join을 하도록 하였지만 실행계획에서는 nested loop join이 없

다.

View 해체해버렸기 때문에 힌트를 무시한다. View가 해체되지 않았다면 실행계획에 view 가 보인다.

그래서 view를 해체하지 못하도록 힌트를 줘야 하는데 그 힌트는 no_merge 이다.

```
Select /*+ gather_plan_statistics no_merge(v) leading(s v) use_nl(v) */  
v.ename, v.sal, v.loc, s.grade  
From emp_dept v, salgrade s  
Where v.sal between s.loosal and s.hisal;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

| | | |
|-----|-------------------|----------|
| 0 | SELECT STATEMENT | |
| 1 | NESTED LOOPS | |
| 2 | TABLE ACCESS FULL | SALGRADE |
| * 3 | VIEW | EMP_DEPT |
| * 4 | HASH JOIN | |
| 5 | TABLE ACCESS FULL | DEPT |
| 6 | TABLE ACCESS FULL | EMP |

힌트 :

No_merge : view를 해체하지 못하게 한다.

Merge : view를 해체한다.

실행계획 해석 : 먼저 salgrade를 풀 스캔하고 view를 스캔하면서 nested loop join 을 수행했다.

뷰안의 조인 문장 테이블의 조인순서 변경

```
Create view emp_dept  
As  
Select e.empno, e.ename, e.sal, e.deptno, d.loc  
From emp e, dept d  
Where e.deptno = d.deptno;
```

```
Select /*+ gather_plan_statistics no_merge(v) leading(s v) use_nl(v)  
leading(v.e v.d) use_hash(v.d) */  
v.ename, v.sal, v.loc, s.grade  
From emp_dept v, salgrade s  
Where v.sal between s.loosal and s.hisal;
```

문제95. 위의 실행계획이 아래와 같이 나오게 하시오.

| | | | | |
|---|---|-------------------|----------|--|
| | 0 | SELECT STATEMENT | | |
| | 1 | NESTED LOOPS | | |
| | 2 | TABLE ACCESS FULL | SALGRADE | |
| * | 3 | VIEW | EMP_DEPT | |
| | 4 | NESTED LOOPS | | |
| | 5 | TABLE ACCESS FULL | EMP | |
| * | 6 | TABLE ACCESS FULL | DEPT | |

```
Select /*+ gather_plan_statistics no_merge(v) leading(s v) use_nl(v)
           leading(v.e v.d) use_nl(v.d) */
      v.ename, v.sal, v.loc, s.grade
   From emp_dept v, salgrade s
 Where v.sal between s.loosal and s.hisal;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제96. 위의 실행계획을 아래와 같이 출력되게 하시오.

| | | | |
|--|-------------------|----------|--|
| | SELECT STATEMENT | | |
| | NESTED LOOPS | | |
| | VIEW | EMP_DEPT | |
| | HASH JOIN | | |
| | TABLE ACCESS FULL | EMP | |
| | TABLE ACCESS FULL | DEPT | |
| | TABLE ACCESS FULL | SALGRADE | |

```
Select /*+ gather_plan_statistics no_merge(v) leading(v s) use_nl(s)
           leading(v.e v.d) use_hash(v.d) */
      v.ename, v.sal, v.loc, s.grade
   From emp_dept v, salgrade s
 Where v.sal between s.loosal and s.hisal;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제97. 아래와 같이 실행계획이 나오게 하시오.

| | | | |
|--|-------------------|----------|--|
| | SELECT STATEMENT | | |
| | NESTED LOOPS | | |
| | TABLE ACCESS FULL | SALGRADE | |
| | VIEW | EMP_DEPT | |
| | HASH JOIN | | |
| | TABLE ACCESS FULL | DEPT | |
| | TABLE ACCESS FULL | EMP | |

```
Select /*+ gather_plan_statistics no_merge(v) leading(s v) use_nl(v)
      leading(v.d v.e) use_hash(v.e) */
      v.ename, v.sal, v.loc, s.grade
      From emp_dept v, salgrade s
      Where v.sal between s.losal and s.hisal;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

View 를 사용하여 조인하는 어떤 SQL 문장의 성능이 느리다면 위와 같은 방법으로 튜닝을 하면 효과적으로 튜닝을 할 수 있다.

3. 서브쿼리 문장 튜닝

```
Select ename, sal
  From emp
 Where sal > (select sal
                  From emp
                  Where ename = 'JONES');
```

서브쿼리 문장 튜닝의 기술은 크게 2가지가 있다.

1. 순수하게 서브쿼리문으로 수행되게 하는 방법
 - 힌트 : no_unnest
2. 서브쿼리를 조인으로 변경해서 수행되게 하는 방법
 - 힌트 : unnest

문제98. 1과 10 사이의 숫자 중에서 소수만 출력하시오.

```
Select n1
  From (select level n1
          From dual
          Connect by level <= 10) a,
  (Select level n2
      From dual
      Connect by level <= 10) b
  Where mod(n1, n2) = 0
  Group by n1
  Having count(n1) = 2;
```

20.11.23

2020년 11월 23일 월요일 오전 9:36

3. 서브쿼리 문장 튜닝

Ex)

DALLAS에서 근무하는 사원들의 이름과 월급을 출력하시오.

```
Select ename, sal  
From emp  
Where deptno in (select deptno  
                    From dept  
                    Where loc = 'DALLAS');
```

문제99. 위의 서브쿼리문을 조인으로 수행해서 같은 결과를 보시오.

DALLAS에서 근무하는 사원들의 이름과 월급을 출력하시오.

```
Select e.ename, e.sal  
      From emp e, dept d  
     Where e.deptno = d.deptno  
       And d.loc = 'DALLAS';
```

아래의 서브쿼리문의 실행계획을 보시오.

```
Select /*+ gather_plan_statistics */ ename, sal  
From emp  
Where deptno in (select deptno  
                    From dept  
                    Where loc = 'DALLAS');
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

| <hr/> | | | | | | |
|---|--|--|--|--|--|--|
| Id Operation | Name Starts E-Rows A-Rows A-Time Buffers | | | | | |
| <hr/> | | | | | | |
| 0 SELECT STATEMENT | 1 5 00:00:00.01 14 | | | | | |
| * 1 HASH JOIN SEMI | 1 5 5 00:00:00.01 14 | | | | | |
| 2 TABLE ACCESS FULL EMP 1 14 14 00:00:00.01 7 | | | | | | |
| * 3 TABLE ACCESS FULL DEPT 1 1 1 00:00:00.01 7 | | | | | | |

위의 실행계획에서 filter가 보이면 서브쿼리문으로 수행한 것이고 hash join이 보이면 조인문장으로 변경해서 수행한 것이다. SQL문은 서브쿼리문으로 작성하였지만 옵티마이저가 조인으로 변경해서 수행한 것이다.

서브쿼리의 실행계획 2가지

1. 순수하게 서브쿼리문으로 수행되게 하는 방법
 - 힌트 : no_unnest
 - 서브쿼리부터 수행 : push_subq
 - 메인쿼리부터 수행 : no_push_subq
2. 서브쿼리를 조인으로 변경해서 수행되게 하는 방법
 - 힌트 : unnest
 - 세미조인(semi join)
 - o Nested loop semi join : nl_sj
 - o Hash semi join : hash_sj
 - o Merge semi join : merge_sj
 - 안티조인(anti join)
 - o Nested loop anti join : nl_aj
 - o Hash anti join : hash_aj
 - o Merge anti join : merge_aj

문제100. 아래의 SQL의 실행계획이 조인으로 풀리게 하시오.

```
Select /*+ gather_plan_statistics */ ename, sal
From emp
Where deptno in (select /*+ unnest */ deptno
                   From dept
                   Where loc = 'DALLAS');
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

서브쿼리문의 실행계획이 filter가 나오면서 성능이 너무 느릴 때 조인의 방법 중 가장 강력한 hash join 으로 수행되게 하면서 성능을 높이고 싶을 때 unnest 힌트가 유리하다.

문제101. 아래의 실행계획이 조인으로 풀리지 않고 filter가 실행계획에 나오는 순수한 서브쿼리문으로 실행되게 하시오.

```
Select /*+ gather_plan_statistics */ ename, sal
From emp
Where deptno in (select /*+ no_unnest */ deptno
                   From dept
                   Where loc = 'DALLAS');
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

No_unnest를 사용하여 순수하게 서브쿼리문으로 수행되었으나 버퍼의 갯수는 hash join semi 로 수행 되었을 때보다 성능이 느려진 것을 확인 할 수 있다.

위의 서브쿼리의 테이블 2개가 대용량 테이블인 경우는 unnest를 사용하여 hash join으로 수행되는 것이 유리하지만 테이블이 작다면 굳이 메모리를 사용하는 hash join으로 유도하지 말고 서브쿼리문의 실행계획(filter)로 수행하는 것이 유리하다.

문제102. 아래의 SQL의 실행계획을 확인하고 조인으로 변경되어서 수행되게 하시오.

```
Select /*+ gather_plan_statistics */ ename, sal
  From emp
 Where deptno = (select deptno
                   From dept
                  Where deptno = 10);
```

위의 SQL은 순수하게 서브쿼리문으로 수행한 SQL 인데 서브쿼리와 메인쿼리 중 서브쿼리문부터 수행하였다.

```
Select /*+ gather_plan_statistics */ ename, sal
  From emp
 Where deptno in (select /*+ unnest */ deptno
                   From dept
                  Where deptno = 10);
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

위의 서브쿼리문과 메인 쿼리문 사이의 연산자를 = 로 하면 서브쿼리에서 메인쿼리로 한건만 리턴된다면 굳이 hash join으로 풀지 않아도 되기 때문에 unnest힌트가 먹히지 않는다.
Hash join 으로 수행하고 싶다면 연산자를 = 에서 in으로 변경해줘야 한다.

문제103. 아래의 SQL의 실행계획이 서브쿼리문으로 수행되게 하는데 서브쿼리문부터 수행되게 하시오.

```
Select /*+ gather_plan_statistics */ ename, sal
  From emp
 Where deptno in (select /*+ no_unnest */ deptno
                   From dept
                  Where deptno = 10);
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

```
Select /*+ gather_plan_statistics */ ename, sal
  From emp
 Where deptno = (select /*+ no_unnest push_subq */ deptno
                   From dept
                  Where deptno = 10);
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제104. 위의 SQL 실행계획이 메인쿼리부터 수행되게 하시오.
(순수하게 서브쿼리로 수행되면서 메인쿼리부터 수행되게 하시오.)

```
Select /*+ gather_plan_statistics */ ename, sal  
  From emp  
 Where deptno = (select /*+ no_unnest no_push_subq */ deptno  
                   From dept  
                  Where deptno = 10);
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

No_unnest와 no_push_subq는 서로 짹꿍 힌트이다.

No_unnest는 서브쿼리문으로 수행하게 하는 힌트이고 이 힌트를 먼저 써줘야 서브쿼리문으로 수행될 수 있기 때문에 no_push_subq 힌트가 수행될 수 있다.

대체로 push_subq 힌트가 서브쿼리문으로 수행되었을 때 더 유리한 힌트이다. 왜냐하면 서브쿼리문부터 수행하면서 데이터를 검색해 메인쿼리로 넘겨주기만 하면 되기 때문이다.

문제105. 아래의 SQL이 순수하게 서브쿼리문으로 수행되게하고 서브쿼리부터 수행되게 하시오.

```
Select /*+ gather_plan_statistics */ ename, sal  
  From emp  
 Where deptno in (select deptno  
                   From dept);
```

```
Select /*+ gather_plan_statistics */ ename, sal  
  From emp  
 Where deptno in (select /*+ no_unnest push_subq */ deptno  
                   From dept);
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제106. 위의 SQL의 실행계획이 메인쿼리 부터 수행되게 하시오.

```
Select /*+ gather_plan_statistics */ ename, sal  
  From emp  
 Where deptno in (select /*+ no_unnest no_push_subq */ deptno  
                   From dept);
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제107. 위의 SQL이 조인으로 수행되게 하시오.

(hash semi join으로 수행되게 하시오.)

```
Select /*+ gather_plan_statistics */ ename, sal  
  From emp  
 Where deptno in (select /*+ unnest hash_sj */ deptno  
                   From dept);
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Semi는 절반이라는 뜻으로 완전한 조인을 하지 않고 절반만 조인한다.

위의 SQL이 조인문장이 아니라 서브쿼리 문장이기 때문에 완전한 조인을 하지 못하고 절반만 조인한다.

문제108. 위의 SQL의 실행계획에서 dept가 메모리로 올라가게 하시오. (dept가 hash table이 되게 하시오.)

```
Select /*+ gather_plan_statistics */ ename, sal  
  From emp  
 Where deptno in (select /*+ unnest hash_sj swap_join_inputs(dept) */ deptno  
                   From dept);
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Emp와 dept가 대용량 테이블이고 위와 같은 SQL이면 해쉬 세미조인으로 수행하되 작은 테이블이 메모리로 올라가게 힌트를 준 위의 힌트가 가장 모범적인 힌트이다.

문제109. 위의 실행계획이 nested loop semi join이 되게하시오.

```
Select /*+ gather_plan_statistics */ ename, sal  
  From emp  
 Where deptno in (select /*+ unnest nl_sj */ deptno  
                   From dept);
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제110. 관리자인 사원들의 이름을 출력하시오.

(자기 밑에 직속부하가 한명이라도 있는 사원들)

```
Select /*+ gather_plan_statistics */ ename  
  From emp  
 Where empno in (select mgr  
                  From emp);
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

실행계획을 보면 둘다 emp여서 메인쿼리부터 수행했는지 서브쿼리부터 수행했는지 확실히 알기가 어렵다.

문제111. QB_NAME 힌트를 사용하여 다시 실행하시오.

실행계획에서 서브쿼리의 emp와 메인쿼리의 emp 중 어떤것을 먼저 읽었는지 확인

```
Select /*+ gather_plan_statistics QB_NAME(main) */ ename
  From emp
 Where empno in (select /*+ QB_NAME(sub) */ mgr
                  From emp);
```

```
Select * from table(dbms_xplan.display_cursor(format=>'advanced'));
```

Format=>'advanced' 를 사용하게 되면 좀 더 정보가 많은 실행계획을 확인 할 수 있다.

QB_NAME힌트를 사용하면 그 해당 쿼리에 이름을 부여할 수 있다.

| Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time | |
|--|-----------|------|------|-------|-------------|------|--|
| 0 SELECT STATEMENT 6 (100) | | | | | | | |
| * 1 HASH JOIN SEMI 6 198 6 (0) 00:00:01 | | | | | | | |
| 2 TABLE ACCESS FULL EMP 14 280 3 (0) 00:00:01 | | | | | | | |
| 3 TABLE ACCESS FULL EMP 14 182 3 (0) 00:00:01 | | | | | | | |

| Query Block Name / Object Alias (identified by operation id): | | | | | | | |
|---|--|--|--|--|--|--|--|
| ----- | | | | | | | |
| 1 - SEL\$526A7031 | | | | | | | |
| 2 - SEL\$526A7031 / EMP@MAIN | | | | | | | |
| 3 - SEL\$526A7031 / EMP@SUB | | | | | | | |

문제112. 주사위 한개를 288회 던져서 5이상의 눈이 나올 확률을 구하시오.

```
Select count(*)/288
  From (select round(dbms_random.value(0.5,6.5)) a
        From dual
       Connect by level <= 288)
 Where a >= 5;
```

문제113. 아래의 쿼리의 실행계획이 hash semi join으로 수행되게 하시오.

```
Select /*+ gather_plan_statistics */ ename
  From emp
```

```
Where empno in (select mgr  
From emp);
```

```
Select * from table(dbms_xplan.display_cursor(format=>'advanced'));
```

```
Select /*+ gather_plan_statistics */ ename  
From emp  
Where empno in (select /*+ unnest hash_sj */ mgr  
From emp);
```

```
Select * from table(dbms_xplan.display_cursor(format=>'advanced'));
```

문제114. 위의 SQL의 실행계획에서 메인쿼리와 서브쿼리 중 어떤 쿼리부터 수행했는지 확인하기 위해 QB_NAME 힌트를 써서 수행하시오.

```
Select /*+ gather_plan_statistics QB_NAME(main) */ ename  
From emp  
Where empno in (select /*+ QB_NAME(sub) unnest hash_sj */ mgr  
From emp);
```

```
Select * from table(dbms_xplan.display_cursor(format=>'advanced'));
```

Semi join은 무조건 메인쿼리부터 수행된다.

하지만 swap_join_inputs를 사용하면 서브쿼리부터 수행되게 할 수 있다.

문제115. 위의 SQL이 subquery부터 수행되는 hash semi join 되게 하시오.

```
Select /*+ gather_plan_statistics QB_NAME(main) */ ename  
From emp  
Where empno in (select /*+ QB_NAME(sub) unnest hash_sj  
swap_join_inputs(emp@sub) */ mgr  
From emp);
```

```
Select * from table(dbms_xplan.display_cursor(format=>'advanced'));
```

위와 같이 메인쿼리의 테이블과 서브쿼리의 테이블이 서로 같을 때 hash semi join을 하는 경우 서브쿼리의 테이블부터 수행되게 하고 싶다면 QB_NAME(sub) 힌트를 이용해서 쿼리블럭의 이름을 부여하고 swap_join_inputs 괄호안에 테이블 별칭으로 emp@sub 을 사용하면 된다. 그러면 실행계획이 서브쿼리의 테이블 부터 수행되면서 hash right semi join으로 수행된다.

문제116. 관리자가 아닌사원들의 이름을 출력하시오.

```
Select ename
  From emp
  Where empno not in (select mgr
    From emp
    Where mgr is not null);
```

문제117. 위의 실행계획이 조인으로 풀리게하시오.

(서브쿼리의 테이블과 메인쿼리의 테이블이 서로 대용량이면 hash join으로 풀리는게 훨씬 성능이 좋다.)

```
Select /*+ gather_plan_statistics QB_NAME(main) */ ename
  From emp
  Where empno not in (select /*+ QB_NAME(sub) unnest hash_aj */ mgr
    From emp
    Where mgr is not null);
```

```
Select * from table(dbms_xplan.display_cursor(format=>'advanced'));
```

Not in 을 사용한 서브쿼리 문장의 성능을 높이기 위해서는 hash anti join을 사용하면 된다.

문제118. 위의 해쉬조인 실행계획의 조인순서가 서브쿼리부터 수행되게 하시오.

```
Select /*+ gather_plan_statistics QB_NAME(main) */ ename
  From emp
  Where empno not in (select /*+ QB_NAME(sub) unnest hash_aj
    Swap_join_inputs(emp@sub) */ mgr
    From emp
    Where mgr is not null)
  And empno is not null; (생략가능)
```

```
Select * from table(dbms_xplan.display_cursor(format=>'advanced'));
```

Not in을 사용한 서브쿼리문장의 성능을 높이기 위해서는 hash anti join으로 수행되게 하면 되는데 서브쿼리부터 수행되게 하려면 swap_join_inputs 을 사용하여 hash right anti join으로 수행되게 하면된다.

서브쿼리 튜닝 정리

서브쿼리와 메인쿼리의 테이블이 대용량이 아닐 때

- 순수하게 서브쿼리 실행계획으로 실행되게 한다. 힌트 : no_unnest
- No_unnest와 짹꿍 힌트
 - o Push_subq
 - o No_push_subq

서브쿼리와 메인쿼리의 테이블이 대용량 일 때

- Hash semi join, hash anti join으로 실행되게 한다.

- Select ename

```
From emp
Where emp in (select /*+ unnest hash_sj */ mgr
                From emp);
```

- Select ename

```
From emp
Where emp not in (select /*+ unnest hash_aj */ mgr
                    From emp);
```

4. 데이터 분석함수를 이용한 튜닝

앞에서는 SQL튜닝을 할 때 힌트를 이용해서 튜닝을 했지만 지금부터는 SQL을 완전히 다른 SQL로 변경해서 튜닝한다.

문제119. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Select /*+ gather_plan_statistics */ deptno, sum(sal)
      From emp
      Group by deptno
Union all
Select null as deptno, sum(sal)
      From emp
      Order by deptno asc;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 후 :

```
Select /*+ gather_plan_statistics */ deptno, sum(sal)
      From emp
      Group by rollup(deptno);
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제120. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Select /*+ gather_plan_statistics */ deptno, null as job, sum(sal)
      From emp
      Group by deptno
Union all
Select null as deptno, job, sum(sal)
      From emp
```

```
Group by job  
Order by deptno asc, job asc;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 후 :

```
Select /*+ gather_plan_statistics */ deptno, job, sum(sal)  
  From emp  
 Group by grouping sets ((deptno), (job))  
 Order by deptno, job;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제121. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Select /*+ gather_plan_statistics */ deptno, null as job, sum(sal)  
  From emp  
 Group by deptno  
Union all  
Select null as deptno, job, sum(sal)  
  From emp  
 Group by job  
Union all  
Select null deptno, null job, sum(sal)  
  From emp  
 Order by deptno, job
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 후 :

```
Select /*+ gather_plan_statistics */ deptno, job, sum(sal)  
  From emp  
 Group by grouping sets ((deptno), (job), ())  
 Order by deptno, job;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제122. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Select /*+ gather_plan_statistics */ empno, ename, sal,  
      (select sum(sal)  
       From emp s  
      Where s.empno <= m.empno) 누적치  
  From emp m  
 Order by empno;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 후 :

```
Select /*+ gather_plan_statistics */ empno, ename, sal,  
      Sum(sal) over(order by empno) 누적치  
   From emp;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 전은 emp 테이블을 2번 엑세스 했으나 튜닝 후는 한번만 엑세스 한다.

문제123. 아래의 SQL을 튜닝하시오.

튜닝 전 :

```
Select /*+ gather_plan_statistics */ deptno, empno, ename, sal,  
      (select sum(sal)  
       From emp s  
      Where s.empno <= m.empno  
        And s.deptno = m.deptno) 누적치  
   From emp m  
 Order by deptno, empno;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

튜닝 후 :

```
Select /*+ gather_plan_statistics */ deptno, empno, ename, sal,  
      Sum(sal) over(partition by deptno order by empno) 누적치  
   From emp;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

문제124. 아래의 SQL을 튜닝하시오. (SQLP 시험 주관식)

```
Create index emp_ename on emp(ename);
```

튜닝 전 :

```
Select /*+ gather_plan_statistics */ ename, sal, job  
   From emp  
  Where ename like '%EN%'  
    Or ename like '%IN%';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Ename에 인덱스가 있어도 like연산자를 사용할 때 와일드 카드가 앞에 나오면 인덱스를 엑세스 하지 못하고 full table scan 한다.

튜닝 후 :

1. 이름에 EN 또는 IN이 포함되어져 있는 사원의 ROWID를 emp_ename 인덱스를 통해서 알아낸다.

Alter table emp

 Modify ename constraint emp_ename_nn not null;

```
Select /*+ gather_plan_statistics index_ffs(emp emp_ename) */ rowid
  From emp
 Where ename like '%IN%'
   Or ename like '%EN%';
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

2. 알아낸 ROWID를 통해서 테이블에서 해당 데이터를 검색하는데 nested loop join으로 검색한다.

```
Select /*+ leading(v e) use_nl(e) */ e.ename, e.sal, e.job
  From emp e,
       (Select /*+ gather_plan_statistics index_ffs(emp emp_ename) no_merge */ rowid rn
        From emp
         Where ename like '%IN%'
           Or ename like '%EN%') v
 Where e.rowid = v.rn;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

From 절의 서브쿼리인 인라인 뷰에서 emp_ename의 인덱스를 빠르게 스캔해서 rowid 3개를 알아낸 다음 emp테이블과 조인했고 인라인 뷰를 해체하지 못하도록 no_merge 힌트를 부여했다.

문제125. emp12테이블에서 성과 이름 중에 '정'자 와 '준'자를 포함하고 있는 학생들의 이름과 나이를 출력하시오.

```
create index emp12_ename on emp12(ename);
```

튜닝 전 :

```
Select ename, age
  From emp12
 Where ename like '%정%' or ename like '%준%';
```

```
Select /*+ gather_plan_statistics */ ename, age
  From emp12
 Where regexp_like(ename,'정|준');
```

튜닝 후 :

```
Select /*+ gather_plan_statistics no_merge(v) leading(v e) use_nl(e) */ e.ename, e.age
  From emp12 e,
       (Select /*+ index_ffs(emp12 emp12_ename) */ rowid rn
        From emp12
       Where ename like '%정%'
             Or ename like '%준%')
   Where e.rowid = v.rn;
```

```
Select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

5. 자동 SQL 튜닝

SQL튜닝 어드바이저(advisor)

```
-- scott으로 test 테이블 생성
```

```
SQL> connect scott/tiger

SQL> create table test (n number);

SQL> declare
begin
for i in 1 .. 10000 loop
insert into test values(i);
commit;
end loop;
end;
/
```

```
-- 인덱스 생성
```

```
SQL> create index test_idx on test(n);
```

```
-- test 테이블 통계정보 분석
```

```
SQL> analyze table test estimate statistics;
```

- Test 테이블 통계정보 분석

(test 테이블에 대한 정보를 분석해서 오라클에게 알려주는 명령어)

- 테이블의 건수
- 테이블의 크기 등의 정보

-- NO_INDEX 힌트를 주어 플레이블 스캔으로 수행되는 SQL확인

```
SQL> set autot traceonly explain  
SQL> select /*+ NO_INDEX(test test_idx) */ * from test where n = 1 ;
```

- test 테이블에 n컬럼의 데이터가 1인 데이터를 검색하는데 no_index 힌트를 이용해서 test 테이블의 test_idx 인덱스를 타지 말고 full table scan 하게 실행한다.
(악성 SQL을 만들기 위해 일부러 no_index 힌트를 줬다.)

-- 튜닝 TASK생성 후, SQL Tuning Advisor를 실행

```
SQL> connect / as sysdba  
  
SQL> declare  
my_task_name VARCHAR2(30);  
my_sqltext CLOB;  
begin  
my_sqltext := 'select /*+ no_index(test test_idx) */ *  
from test where n = 1';  
my_task_name := DBMS_SQLTUNE.CREATE_TUNING_TASK (  
sql_text => my_sqltext,  
user_name => 'SCOTT',  
scope => 'COMPREHENSIVE',  
time_limit => 60,  
task_name => 'my_sql_tuning_task_1',  
description => 'Task to tune a query on a specified table' );  
end;  
/  
  
SQL> begin  
DBMS_SQLTUNE.EXECUTE_TUNING_TASK (  
task_name => 'my_sql_tuning_task_1');  
end;  
/
```

-- SQL Tuning Advisor를 통해 얻은 결과(튜닝 레포트)를 확인하고 SQL Profile을 적용

```
SQL> SET LONG 70000
SQL> SET LONGCHUNKSIZE 1000
SQL> SET LINESIZE 100

SQL> select DBMS_SQLTUNE.REPORT_TUNING_TASK( 'my_sql_tuning_task_1') from DUAL;
```

```
SQL> DECLARE
my_sqlprofile_name VARCHAR2(30);
BEGIN
my_sqlprofile_name := DBMS_SQLTUNE.ACCEPT_SQL_PROFILE (
task_name => 'my_sql_tuning_task_1',
name => 'my_sql_profile' );
END;
/
```

문제126. 숫자를 물어보게 하고 숫자를 입력하면 해당 숫자가 정수인지 소수인지 출력되게 하시오.

숫자를 입력하시오.

결과 값

합성수 입니다.

소수 입니다.

Accept n prompt '숫자를 입력하시오.'

Select case

```
From (select count(mod(&n,level))
      From dual
      Connect by level <= 10)
```