4주차 5. SQL 질의 & 6. 실체화 뷰

제출일 : 2022년 11월 25일 금요일

제출자 : 박민영

---

1. 시간을 사용한 FLASHBACK

1.1 설명

Flashback : 과거 일정시점의 데이터를 쿼리해 볼 수 있는 기능

1.2 시나리오 수행

시나리오 내용
1. 테이블 생성 및 초기 데이터 삽입
2. 커밋 후 현재 상태의 timestamp 출력
3. 새로운 데이터 삽입 및 커밋
4. 2번 timestamp 시점의 상태 조회

1.3 시나리오 수행 결과

1. 테이블 생성 및 초기 데이터 삽입

```
-- 1. 테이블 생성 및 초기 데이터 삽입
create table todo (item varchar2(20), duedate date);
insert into todo values ('Study', '2022-11-29');
insert into todo values ('Work', '2022-11-26');
```

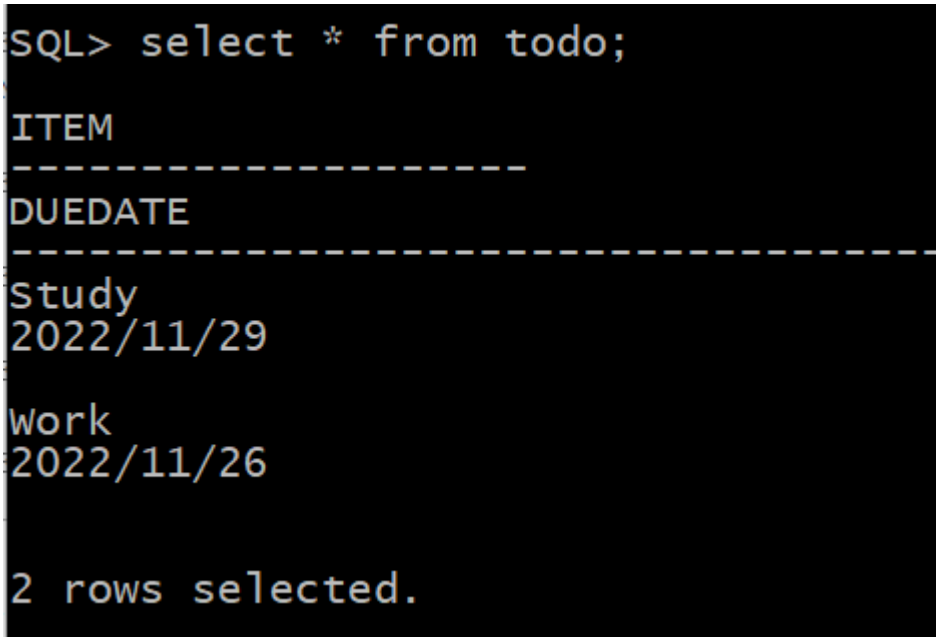
```
SQL> create table todo (item varchar2(20), duedate date);

Table 'TODO' created.
```

```
SQL> insert into todo values ('Study', '2022-11-29');
1 row inserted.
SQL> insert into todo values ('Work', '2022-11-26');
1 row inserted.
```

2. 커밋 후 현재 상태의 timestamp 출력

```
-- 2. 커밋 후 현재 상태의 timestamp 출력
COMMIT;
select systimestamp from dual;
select * from todo;
```

```
SQL> COMMIT;
Commit completed.
SQL> select systimestamp from dual;
SYSTIMESTAMP
---------------------------------------------------------------------
2022/11/23 17:15:12.706616 Asia/Seoul
1 row selected.
```



3. 새로운 데이터 삽입 및 커밋

```
-- 3. 새로운 데이터 삽입 및 커밋
insert into todo values ('Concert', '2022-12-04');
COMMIT;
select * from todo;
```

```
SQL> insert into todo values ('Concert', '2022-12-04');

1 row inserted.

SQL> COMMIT;

Commit completed.

SQL> select * from todo;

ITEM
--------------------
DUEDATE
----------------------------------------------------------------
Study
2022/11/29

Work
2022/11/26

Concert
2022/12/04


3 rows selected.
```

4. 2번 timestamp 시점의 상태 조회

```
-- 4. 2번 timestamp 시점의 상태 조회
-- select * from todo as of timestamp'타임스탬프 값';
```

```
SQL> select * from todo as of timestamp'2022/11/23 17:15:12.706616';

ITEM
--------------------
DUEDATE
----------------------------------------------------------------
Study
2022/11/29

Work
2022/11/26


2 rows selected.
```

1.sh

```
-- 1. 테이블 생성 및 초기 데이터 삽입
create table todo (item varchar2(20), duedate date);
insert into todo values ('Study', '2022-11-29');
insert into todo values ('Work', '2022-11-26');

-- 2. 커밋 후 현재 상태의 timestamp 출력
COMMIT;
select systimestamp from dual;
select * from todo;

-- 3. 새로운 데이터 삽입 및 커밋
insert into todo values ('Concert', '2022-12-04');
COMMIT;
select * from todo;

-- 4. 2번 timestamp 시점의 상태 조회
-- select * from todo as of timestamp'타임스탬프 값';
```

---

2. TSN을 이용한 FLASHBACK

2.1 설명

동적뷰 V$TSN_TIME을 이용하면 TSN과 시간의 맵핑 정보를 알 수 있다.

2.2 시나리오 수행

시나리오 내용
1. 테이블 생성 및 초기 데이터 삽입
2. 커밋 후 현재 상태의 timestamp 출력
3. 새로운 데이터 삽입 및 커밋
4. 3번 커밋 후의 현재 상태의 timestamp 출력
5. 2번과 4번 사이의 상태 시간 조회
6. tsn 값으로 해당 상태 조회

2.3 시나리오 수행 결과

1. 테이블 생성 및 초기 데이터 삽입

```
-- 1. 테이블 생성 및 초기 데이터 삽입
create table todo2 (item varchar2(20), duedate date);
insert into todo2 values ('Study', '2022-11-29');
insert into todo2 values ('Work', '2022-11-26');
```

```
SQL> create table todo2 (item varchar2(20), duedate date);
Table 'TODO2' created.
SQL> insert into todo2 values ('Study', '2022-11-29');
1 row inserted.
SQL> insert into todo2 values ('Work', '2022-11-26');
1 row inserted.
```

2. 커밋 후 현재 상태의 timestamp 출력

```
-- 2. 커밋 후 현재 상태의 timestamp 출력
COMMIT;
select systimestamp from dual;
select * from todo2;
```

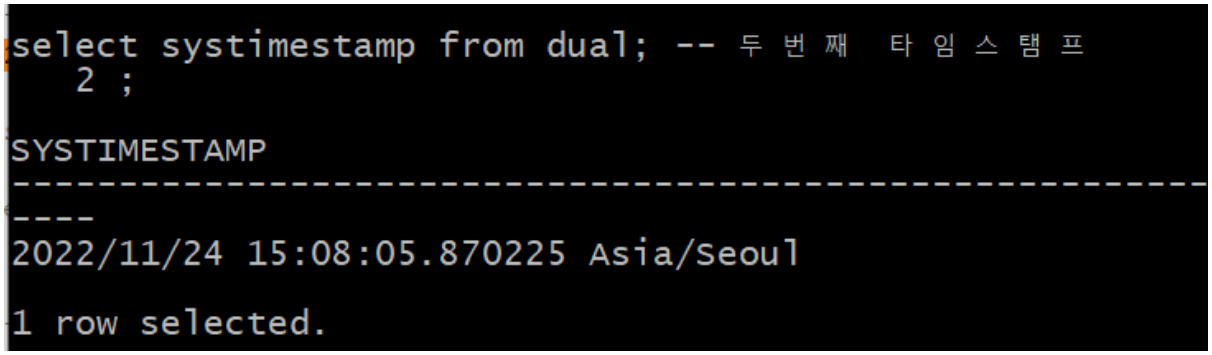
3. 새로운 데이터 삽입 및 커밋

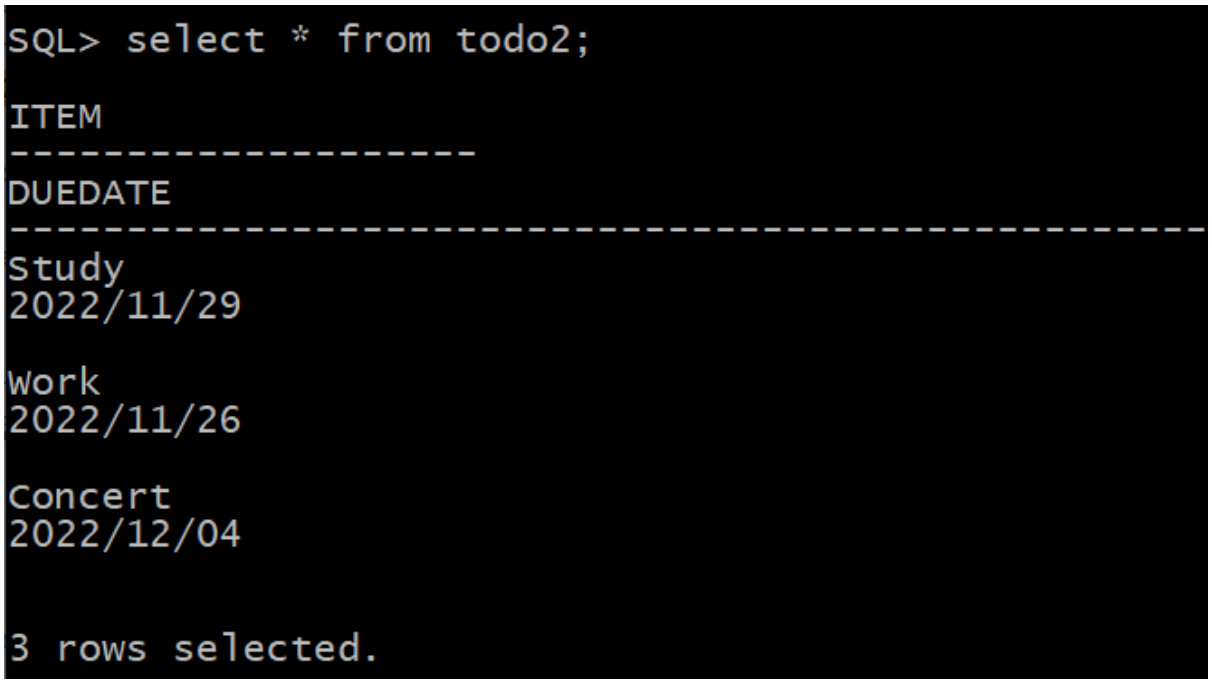
```
-- 3. 새로운 데이터 삽입 및 커밋
insert into todo2 values ('Concert', '2022-12-04');
COMMIT;
```

```
SQL> insert into todo2 values ('Concert', '2022-12-04');
1 row inserted.
SQL> COMMIT;
Commit completed.
```

4. 3번 커밋 후의 현재 상태의 timestamp 출력

```
-- 4. 3번 커밋 후의 현재 상태의 timestamp 출력
select systimestamp from dual;
select * from todo2;
```





5. 2번과 4번 사이의 상태 시간 조회

```
-- 5. 2번과 4번 사이의 상태 시간 조회
--select * from v$tsn_time
--  where time >= '첫번째 타임스탬프값' and time <= '두번째 타임스탬프값';

select * from v$tsn_time
```

```
   where time >= '2022/11/24 15:07:52.068442'
     and time <= '2022/11/24 15:08:05.870225';
```

```
SQL> select * from v$tsn_time where time >= '2022/11/24 15:07:52.068442' and
 time <= '2022/11/24 15:08:05.870225';

       TSN TIME
---------- ---------------------------------------------------------------
    151167 2022/11/24 15:07:52.237905
    151167 2022/11/24 15:07:53.238895
    151167 2022/11/24 15:07:54.239730
    151168 2022/11/24 15:07:55.242142
    151168 2022/11/24 15:07:56.247865
    151168 2022/11/24 15:07:57.248497
    151169 2022/11/24 15:07:58.248977
    151169 2022/11/24 15:07:59.262900
    151171 2022/11/24 15:08:00.270674
    151171 2022/11/24 15:08:01.271889
    151171 2022/11/24 15:08:02.273409
    151172 2022/11/24 15:08:03.276130
    151172 2022/11/24 15:08:04.279221

13 rows selected.
```

6. tsn 값으로 해당 상태 조회

```
-- 6. tsn 값으로 해당 상태 조회
-- select * from todo2 as of scn 숫자;
select * from todo2 as of scn 151168;
```

```
SQL> select * from todo2 as of scn 151168;

ITEM
--------------------
DUEDATE
----------------------------------------
Study
2022/11/29

Work
2022/11/26


2 rows selected.
```

2.sh

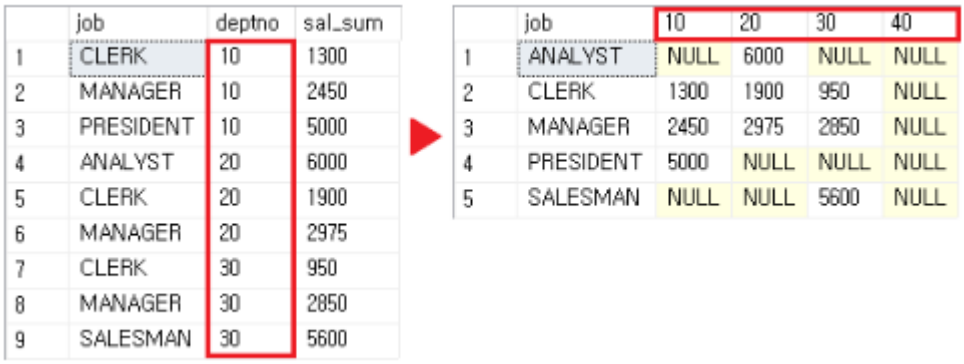
```
-- 1. 테이블 생성 및 초기 데이터 삽입
create table todo2 (item varchar2(20), duedate date);
insert into todo2 values ('Study', '2022-11-29');
insert into todo2 values ('Work', '2022-11-26');

-- 2. 커밋 후 현재 상태의 timestamp 출력
COMMIT;
select systimestamp from dual; -- 첫번째 타임스탬프
;
select * from todo2;

-- 3. 새로운 데이터 삽입 및 커밋
insert into todo2 values ('Concert', '2022-12-04');
COMMIT;

-- 4. 3번 커밋 후의 현재 상태의 timestamp 출력
select systimestamp from dual; -- 두번째 타임스탬프
;
select * from todo2;

-- 5. 2번과 4번 사이의 상태 시간 조회
--select * from v$tsn_time
--  where time >= '첫번째 타임스탬프값' and time <= '두번째 타임스탬프값';


-- 6. tsn 값으로 해당 상태 조회
-- select * from todo2 as of scn 숫자;
```

---

3. PIVOT과 UNPIVOT

3.1 설명

- PIVOT : 행을 열로 변환하는 함수



	job	deptno	sal_sum
1	CLERK	10	1300
2	MANAGER	10	2450
3	PRESIDENT	10	5000
4	ANALYST	20	6000
5	CLERK	20	1900
6	MANAGER	20	2975
7	CLERK	30	950
8	MANAGER	30	2850
9	SALESMAN	30	5600

	job	10	20	30	40
1	ANALYST	NULL	6000	NULL	NULL
2	CLERK	1300	1900	950	NULL
3	MANAGER	2450	2975	2850	NULL
4	PRESIDENT	5000	NULL	NULL	NULL
5	SALESMAN	NULL	NULL	5600	NULL

```
SELECT *
  FROM ( 피벗할 쿼리문 ) AS result
  PIVOT ( 그룹합수(집계컬럼) --오른쪽으로 회전할 컬럼명(같은 그룹끼리 집계)
            FOR [위로 올릴 컬럼]
            IN ([FOR에서 사용한 컬럼명 중 사용할 컬럼 도메인 목록] ... )
AS 피벗Alias
```

- UNPIVOT : 열을 행의 집합으로 보여주는 역할을 함 (PIVOT과 반대)

```
SELECT *
  FROM ( 피벗할 쿼리문 ) AS result
  UNPIVOT ( 컬럼명 --왼쪽으로 회전시킬 컬럼들을 이름 지을 컬럼명
            FOR [오른쪽으로 회전한 후 이름 지을 컬럼명]
            IN (회전 시킬 컬럼들)
AS 피벗Alias
```

출처 : https://nive.tistory.com/172

3.2 시나리오 수행

시나리오 내용
1. 테이블 데이터 조회
2. PIVOT으로 deptno 열을 위로 올림
3. 2번 결과를 view로 만듦
4. 3번 커밋 후의 현재 상태의 timestamp 출력

3.3 시나리오 수행 결과

1. 테이블 데이터 조회

```
-- 1. 테이블 데이터 조회
SELECT deptno, job, sal FROM emp ORDER BY deptno;

SELECT deptno, job, sum(sal) FROM emp
  GROUP BY deptno, job
  ORDER BY deptno;
```

```
SQL> SELECT deptno, job, sal FROM emp ORDER BY deptno;

    DEPTNO JOB              SAL
---------- --------- ----------
        10 PRESIDENT       5000
        10 MANAGER         2450
        10 CLERK           1300
        20 MANAGER         2975
        20 ANALYST         3000
        20 CLERK            800
        20 ANALYST         3000
        20 CLERK           1100
        30 MANAGER         2850
        30 SALESMAN        1250
        30 SALESMAN        1600
        30 SALESMAN        1500
        30 CLERK            950
        30 SALESMAN        1250

14 rows selected.

SQL> SELECT deptno, job, sum(sal) FROM emp
  2  GROUP BY deptno, job
  3  ORDER BY deptno;

    DEPTNO JOB         SUM(SAL)
---------- --------- ----------
        10 CLERK           1300
        10 MANAGER         2450
        10 PRESIDENT       5000
        20 ANALYST         6000
        20 CLERK           1900
        20 MANAGER         2975
        30 CLERK            950
        30 MANAGER         2850
        30 SALESMAN        5600

9 rows selected.
```

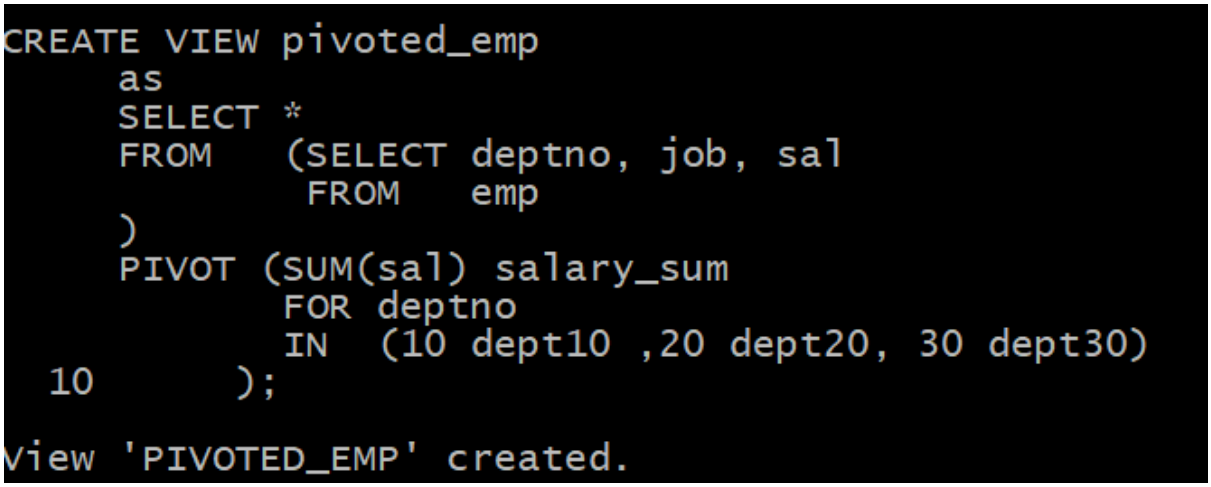
2. PIVOT으로 deptno 열을 위로 올림

```
-- 2. PIVOT으로 deptno 열을 위로 올림
SELECT *
     FROM   (SELECT deptno, job, sal
             FROM   emp
     )
     PIVOT (SUM(sal) salary_sum
            FOR deptno
            IN  (10 dept10 ,20 dept20, 30 dept30)
     );
```

```
SQL> SELECT *
     FROM   (SELECT deptno, job, sal
             FROM   emp
     )
     PIVOT (SUM(sal) salary_sum
            FOR deptno
            IN  (10 dept10 ,20 dept20, 30 dept30)
   8       );

JOB       DEPT10_SALARY_SUM DEPT20_SALARY_SUM DEPT30_SALARY_SUM
--------- ----------------- ----------------- -----------------
CLERK                  1300              1900               950
ANALYST                                  6000
PRESIDENT              5000
SALESMAN                                                   5600
MANAGER                2450              2975              2850

5 rows selected.
```

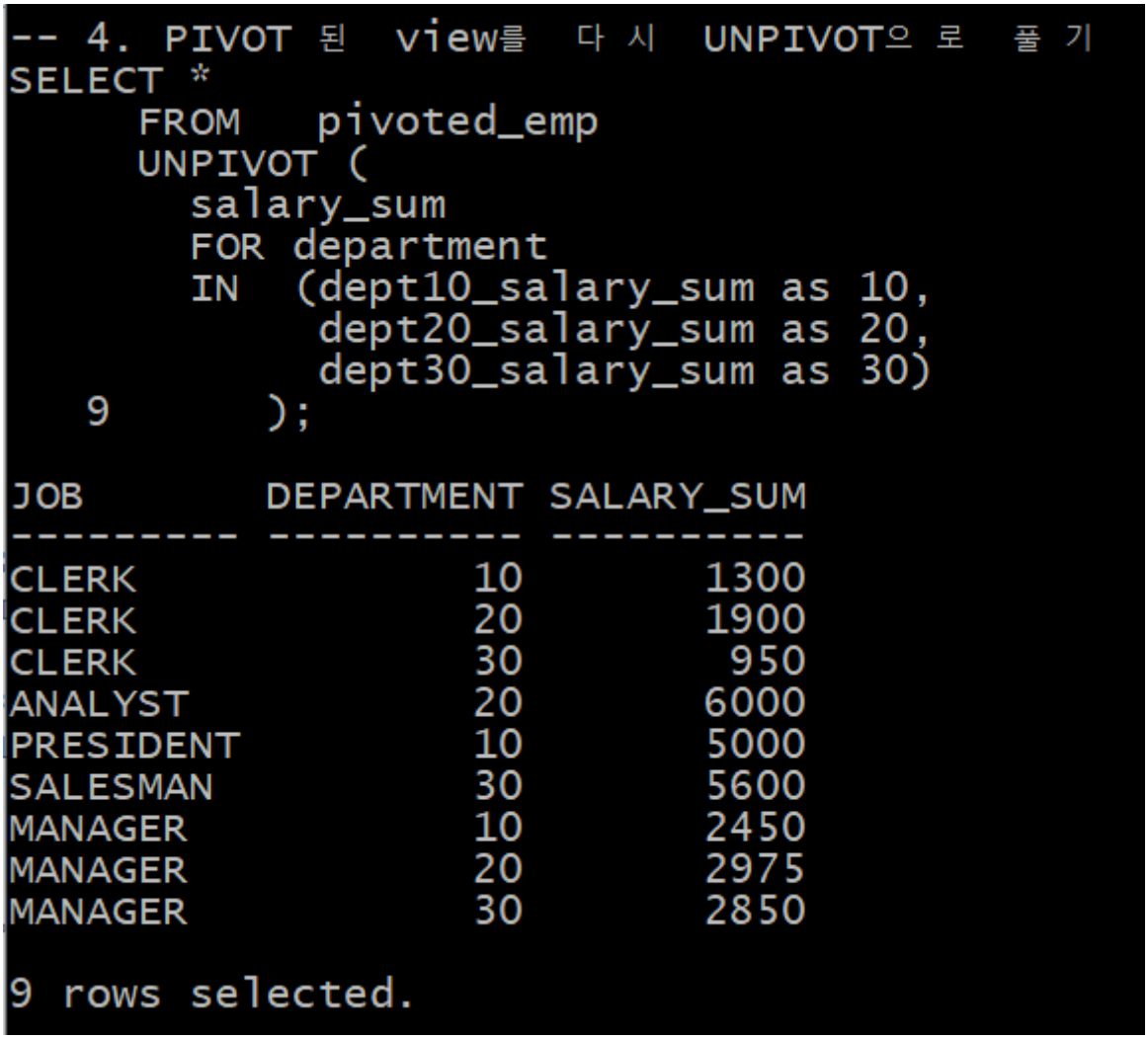
3. 2번 결과를 view로 만듦

```
-- 3. 2번 결과를 view로 만듦
CREATE VIEW pivoted_emp
     as
     SELECT *
     FROM   (SELECT deptno, job, sal
             FROM   emp
     )
     PIVOT (SUM(sal) salary_sum
            FOR deptno
            IN  (10 dept10 ,20 dept20, 30 dept30)
     );
```



4. 3번 커밋 후의 현재 상태의 timestamp 출력

```
-- 4. PIVOT 된 view를 다시 UNPIVOT으로 풀기
SELECT *
     FROM   pivoted_emp
     UNPIVOT (
       salary_sum
       FOR department
       IN  (dept10_salary_sum as 10,
            dept20_salary_sum as 20,
            dept30_salary_sum as 30)
     );
```



```
-- 4. PIVOT 된  view를  다 시  UNPIVOT으 로  풀 기
SELECT *
     FROM   pivoted_emp
     UNPIVOT (
       salary_sum
       FOR department
       IN  (dept10_salary_sum as 10,
            dept20_salary_sum as 20,
            dept30_salary_sum as 30)
   9       );

JOB        DEPARTMENT SALARY_SUM
--------- ---------- ----------
CLERK              10       1300
CLERK              20       1900
CLERK              30        950
ANALYST            20       6000
PRESIDENT          10       5000
SALESMAN           30       5600
MANAGER            10       2450
MANAGER            20       2975
MANAGER            30       2850

9 rows selected.
```

3.sh

```
-- 1. 테이블 데이터 조회
SELECT deptno, job, sal FROM emp ORDER BY deptno;

SELECT deptno, job, sum(sal) FROM emp
  GROUP BY deptno, job
  ORDER BY deptno;


-- 2. PIVOT으로 deptno 열을 위로 올림
SELECT *
     FROM   (SELECT deptno, job, sal
             FROM   emp
     )
     PIVOT (SUM(sal) salary_sum
            FOR deptno
            IN  (10 dept10 ,20 dept20, 30 dept30)
     );
```

```
-- 3. 2번 결과를 view로 만듦
CREATE VIEW pivoted_emp
     as
     SELECT *
     FROM   (SELECT deptno, job, sal
             FROM   emp
     )
     PIVOT (SUM(sal) salary_sum
            FOR deptno
            IN  (10 dept10 ,20 dept20, 30 dept30)
     );

-- 4. PIVOT 된 view를 다시 UNPIVOT으로 풀기
SELECT *
     FROM   pivoted_emp
     UNPIVOT (
       salary_sum
       FOR department
       IN  (dept10_salary_sum as 10,
            dept20_salary_sum as 20,
            dept30_salary_sum as 30)
     );
```

---

4. 동등조인, 자체 조인

4.1 설명

동등조인(Equal Join)

- 동등 연산자(=)로 구성된 조인 조건을 포함한 조인
- 정해진 컬럼에 대해 같은 값을 가지는 로우를 결합하여 결과로 반환

자체 조인(Self Join)

- 하나의 테이블을 사용해서 자신에게 조인하는 것
- 동일한 하나의 테이블이 FROM 절에 두 번 사용되기 때문에 별칭을 사용하여 컬럼을 구분

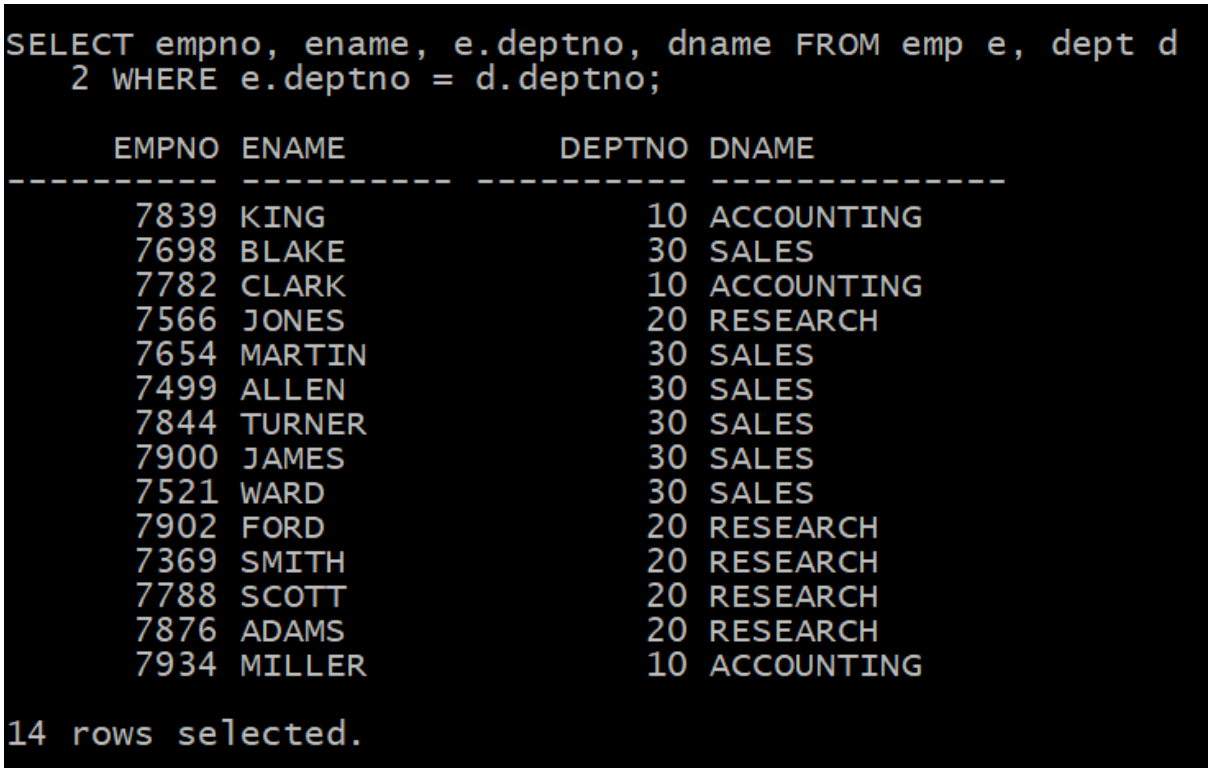
4.2 시나리오 수행

시나리오 내용
1. 각 사원 별 부서 번호와 부서 이름 조회 (동등 조인)
2. 각 사원을 관리하는 상사의 이름을 조회 (자체 조인)

4.3 시나리오 수행 결과

1. 각 사원 별 부서 번호와 부서 이름 조회 (동등 조인)

```
-- 1. 각 사원 별 부서 번호와 부서 이름 조회 (동등 조인)
SELECT empno, ename, deptno, dname FROM emp e, dept d
  WHERE e.deptno = d.deptno;
```



```
SELECT empno, ename, e.deptno, dname FROM emp e, dept d
  2 WHERE e.deptno = d.deptno;

     EMPNO ENAME          DEPTNO DNAME
---------- ---------- ---------- --------------
      7839 KING               10 ACCOUNTING
      7698 BLAKE              30 SALES
      7782 CLARK              10 ACCOUNTING
      7566 JONES              20 RESEARCH
      7654 MARTIN             30 SALES
      7499 ALLEN              30 SALES
      7844 TURNER             30 SALES
      7900 JAMES              30 SALES
      7521 WARD               30 SALES
      7902 FORD               20 RESEARCH
      7369 SMITH              20 RESEARCH
      7788 SCOTT              20 RESEARCH
      7876 ADAMS              20 RESEARCH
      7934 MILLER             10 ACCOUNTING

14 rows selected.
```

2. 각 사원을 관리하는 상사의 이름을 조회 (자체 조인)

```
-- 2. 각 사원을 관리하는 상사의 이름을 조회 (자체 조인)
SELECT e1.empno, e1.ename, e1.mgr, e2.empno, e2.ename
FROM emp e1, emp e2
WHERE e1.mgr = e2.empno;
```

```
SELECT e1.empno, e1.ename, e1.mgr, e2.empno, e2.ename
FROM emp e1, emp e2
   3 WHERE e1.mgr = e2.empno;

     EMPNO ENAME             MGR      EMPNO ENAME
---------- ---------- ---------- ---------- ----------
      7698 BLAKE            7839       7839 KING
      7782 CLARK            7839       7839 KING
      7566 JONES            7839       7839 KING
      7654 MARTIN           7698       7698 BLAKE
      7499 ALLEN            7698       7698 BLAKE
      7844 TURNER           7698       7698 BLAKE
      7900 JAMES            7698       7698 BLAKE
      7521 WARD             7698       7698 BLAKE
      7934 MILLER           7782       7782 CLARK
      7902 FORD             7566       7566 JONES
      7788 SCOTT            7566       7566 JONES
      7369 SMITH            7902       7902 FORD
      7876 ADAMS            7788       7788 SCOTT

13 rows selected.
```

4.sh

```
-- 1. 각 사원 별 부서 번호와 부서 이름 조회 (동등 조인)
SELECT empno, ename, e.deptno, dname FROM emp e, dept d
  WHERE e.deptno = d.deptno;

-- 2. 각 사원을 관리하는 상사의 이름을 조회 (자체 조인)
SELECT e1.empno, e1.ename, e1.mgr, e2.empno, e2.ename
FROM emp e1, emp e2
WHERE e1.mgr = e2.empno;
```

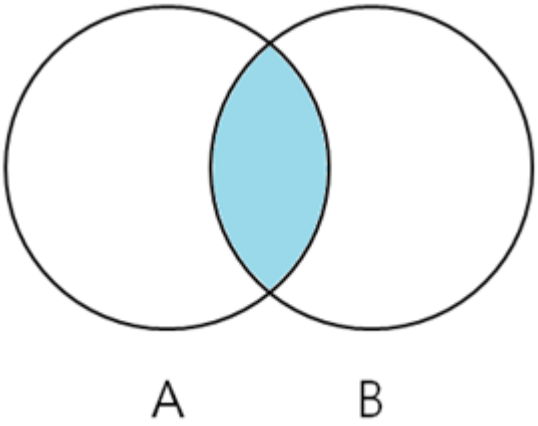
---

5. 내부 조인, 왼쪽 외부 조인

5.1 설명

1) 내부 조인(Inner Join, Simple Join)

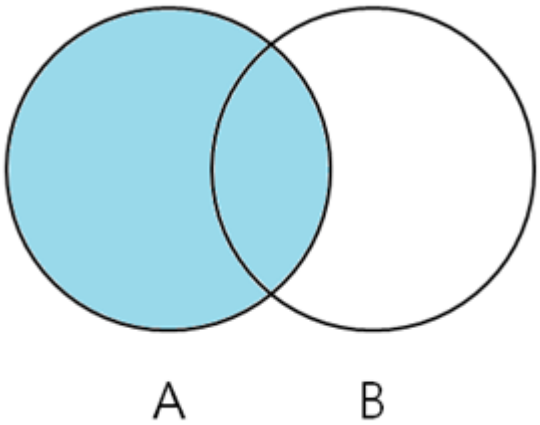
- 조인 조건을 만족하는 로우만 반환하는 2개 이상의 테이블에 대한 조인



2) 외부 조인(Outer Join)

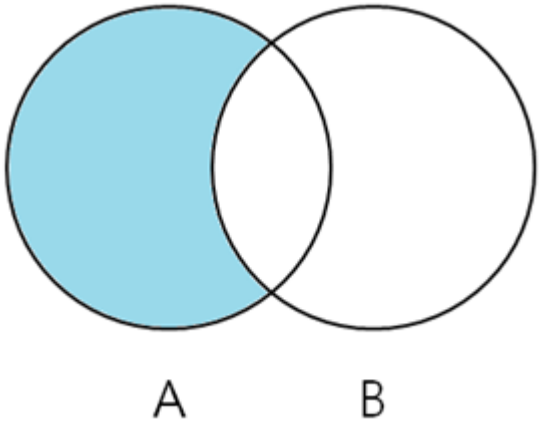
- 조인 조건을 만족하는 로우뿐만 아니라, 한 테이블의 어떤 로우에 대해 반대편 테이블의 모든 로우가 조인 조건을 만족하지 못하는 경우에도 그 로우를 출력

2-1) 왼쪽 외부 조인(left outer join)



- A는 모두 다 추출하고, B는 A에 존재하는(매핑되는) 행들만 추출
- A가 기준이 됨
- A에는 있는데 B에는 없는 경우 NULL 처리

2-2) 왼쪽 외부 조인 (LEFT ONLY)



- A의 전체를 추출하되, 만일 B와 매핑되는 것이 있다면 그거는 제외하고 추출하는 것.

5.2 시나리오 수행

시나리오 내용
1. INNER JOIN
2. LEFT OUTER JOIN
2.1. LEFT OUTER JOIN (LEFT ONLY)

5.3 시나리오 수행 결과

1. INNER JOIN

```
-- 1. INNER JOIN
SELECT * FROM izone z INNER JOIN ive v ON z.name = v.name;
```

```
-- 1. INNER JOIN
SQL> SELECT * FROM izone z INNER JOIN ive v ON z.name = v.name;

NAME       DEBUT_ORDER BIRTH_YEAR NAME       BIRTH_YEAR
---------- ----------- ---------- ---------- ----------
yujin                5       2003 yujin            2003
wonyoung             1       2004 wonyoung         2004

2 rows selected.
```

2. LEFT OUTER JOIN

```
-- 2. LEFT OUTER JOIN
SELECT * FROM izone z LEFT OUTER JOIN ive v ON z.name = v.name;
```

```
-- 2. LEFT OUTER JOIN
SQL> SELECT * FROM izone z LEFT OUTER JOIN ive v ON z.name = v.name;

NAME       DEBUT_ORDER BIRTH_YEAR NAME       BIRTH_YEAR
---------- ----------- ---------- ---------- ----------
eunbi                7       1995
sakura               2       1998
hyewon               8       1999
yena                 4       1999
chaeyeon            12       2000
chaewon             10       2000
minju               11       2001
nako                 6       2001
hitomi               9       2001
yuri                 3       2001
yujin                5       2003 yujin            2003
wonyoung             1       2004 wonyoung         2004

12 rows selected.
```

izone 테이블(LEFT 테이블) 에 있는 모든 내용이 출력되고, ive 테이블(RIGHT 테이블)에서는 izone 테이블과 겹치는 부분만 출력된다.

2.1. LEFT OUTER JOIN (LEFT ONLY)

```
-- 2.1. LEFT OUTER JOIN (LEFT ONLY)
SELECT * FROM izone z LEFT OUTER JOIN ive v ON z.name = v.name
  WHERE v.name is null;
```

```
-- 2.1. LEFT OUTER JOIN (LEFT ONLY)
SQL> SELECT * FROM izone z LEFT OUTER JOIN ive v ON z.name = v.name WHERE v.
name is null;

NAME       DEBUT_ORDER BIRTH_YEAR NAME       BIRTH_YEAR
---------- ----------- ---------- ---------- ----------
eunbi                7       1995
sakura               2       1998
hyewon               8       1999
yena                 4       1999
chaeyeon            12       2000
chaewon             10       2000
minju               11       2001
nako                 6       2001
hitomi               9       2001
yuri                 3       2001

10 rows selected.
```

izone 테이블에서 ive 테이블과 겹치는 부분은 제외하고 출력된다.

5.sh

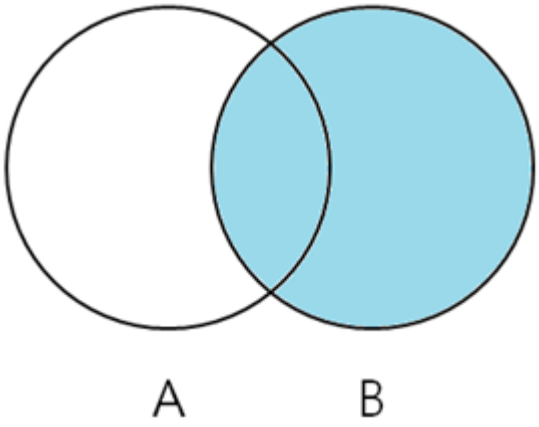
```
-- 1. INNER JOIN
SELECT * FROM izone z INNER JOIN ive v ON z.name = v.name;

-- 2. LEFT OUTER JOIN
SELECT * FROM izone z LEFT OUTER JOIN ive v ON z.name = v.name;

-- 2.1. LEFT OUTER JOIN (LEFT ONLY)
SELECT * FROM izone z LEFT OUTER JOIN ive v ON z.name = v.name
  WHERE v.name is null;
```

---

6. 오른쪽 외부 조인

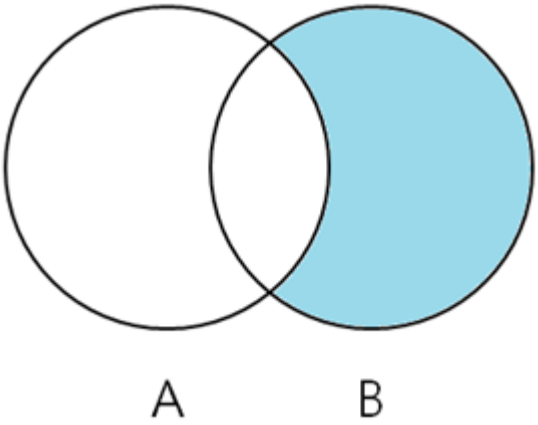
6.1 설명

1) 오른쪽 외부 조인(right outer join)



- B는 모두 다 추출하고, A는 B에 존재하는(매핑되는) 행들만 추출
- B가 기준이 됨
- B에는 있는데 A에는 없는 경우 NULL 처리

1-1) 오른쪽 외부 조인 (LEFT ONLY)



- B의 전체를 추출하되, 만일 A와 매핑되는 것이 있다면 그거는 제외하고 추출하는 것.

6.2 시나리오 수행

시나리오 내용
1. RIGHT OUTER JOIN
1.1 RIGHT OUTER JOIN (RIGHT ONLY)

6.3 시나리오 수행 결과

1. RIGHT OUTER JOIN

```
-- 1. RIGHT OUTER JOIN
SELECT * FROM izone z RIGHT OUTER JOIN ive v ON z.name = v.name;
```

```
-- 1. RIGHT OUTER JOIN
SQL> SELECT * FROM izone z RIGHT OUTER JOIN ive v ON z.name = v.name;

NAME       DEBUT_ORDER BIRTH_YEAR NAME       BIRTH_YEAR
---------- ----------- ---------- ---------- ----------
yujin                5       2003 yujin            2003
wonyoung             1       2004 wonyoung         2004
                                  gaeul            2002
                                  rei              2004
                                  liz              2004
                                  leeseo           2007

6 rows selected.
```

ive 테이블(RIGHT 테이블) 에 있는 모든 내용이 출력되고, izone 테이블(LEFT 테이블)에서는 ive 테이블과 겹치는 부분만 출력된다.

1.1 RIGHT OUTER JOIN (RIGHT ONLY)

```
-- 1.1 RIGHT OUTER JOIN (RIGHT ONLY)
SELECT * FROM izone z RIGHT OUTER JOIN ive v ON z.name = v.name
  WHERE z.name is null;
```

```
-- 1.1 RIGHT OUTER JOIN (RIGHT ONLY)
SQL> SELECT * FROM izone z RIGHT OUTER JOIN ive v ON z.name = v.name WHERE z
.name is null;

NAME       DEBUT_ORDER BIRTH_YEAR NAME       BIRTH_YEAR
---------- ----------- ---------- ---------- ----------
                                  gaeul            2002
                                  rei              2004
                                  liz              2004
                                  leeseo           2007

4 rows selected.
```

ive 테이블에서 izone 테이블과 겹치는 부분은 제외하고 출력된다.

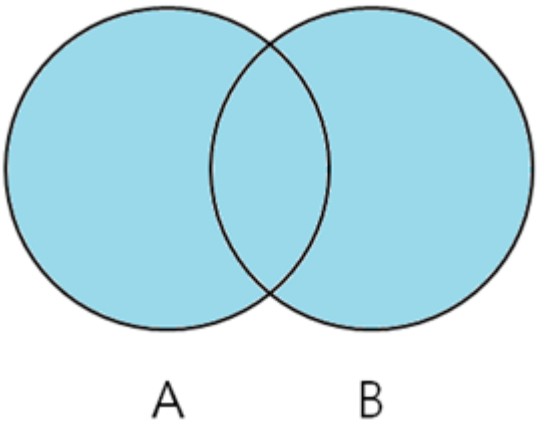
6.sh

```
-- 1. RIGHT OUTER JOIN
SELECT * FROM izone z RIGHT OUTER JOIN ive v ON z.name = v.name;

-- 1.1 RIGHT OUTER JOIN (RIGHT ONLY)
SELECT * FROM izone z RIGHT OUTER JOIN ive v ON z.name = v.name
  WHERE z.name is null;
```

7. 완전 외부 조인

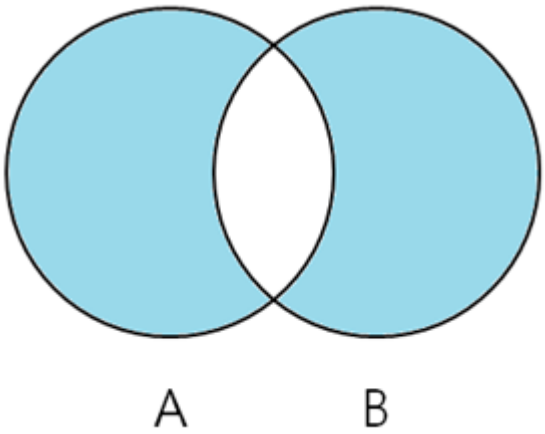
7.1 설명

완전 외부 조인(full outer join)



- A와 B의 합집합을 추출하는 것

완전 외부 조인(full outer join) (Only OUTER 조인)



- FULL OUTER JOIN 한 것 중에서 둘 다 모두 있는 데이터를 제외한, 각각에만 존재하는 데이터를 추출한 것

7.2 시나리오 수행

시나리오 내용
1. FULL OUTER JOIN
1.1 FULL OUTER JOIN (ONLY OUTER)

7.3 시나리오 수행 결과

1. FULL OUTER JOIN

```
-- 1. FULL OUTER JOIN
SELECT * FROM izone z FULL OUTER JOIN ive v ON z.name = v.name;
```

```
-- 1. FULL OUTER JOIN
SQL> SELECT * FROM izone z FULL OUTER JOIN ive v ON z.name = v.name;

NAME       DEBUT_ORDER BIRTH_YEAR NAME       BIRTH_YEAR
---------- ----------- ---------- ---------- ----------
yujin                5       2003 yujin            2003
                                  gaeul           2002
                                  rei             2004
wonyoung             1       2004 wonyoung         2004
                                  liz             2004
                                  leeseo          2007
eunbi                7       1995
sakura               2       1998
hyewon               8       1999
yena                 4       1999
chaeyeon            12       2000
chaewon             10       2000
minju               11       2001
nako                 6       2001
hitomi               9       2001
yuri                 3       2001

16 rows selected.
```

izone테이블과 ive 테이블에 존재하는 모든 합집합 원소들을 출력한다.

1.1 FULL OUTER JOIN (ONLY OUTER)

```
-- 1.1 FULL OUTER JOIN (ONLY OUTER)
SELECT * FROM izone z FULL OUTER JOIN ive v
  ON z.name = v.name
  WHERE z.name is null OR v.name is null;
```

```
-- 1.1 FULL OUTER JOIN (ONLY OUTER)
SELECT * FROM izone z FULL OUTER JOIN ive v
ON z.name = v.name
   3 WHERE z.name is null OR v.name is null;

NAME       DEBUT_ORDER BIRTH_YEAR NAME       BIRTH_YEAR
---------- ----------- ---------- ---------- ----------
                                  gaeul            2002
                                  rei              2004
                                  liz              2004
                                  leeseo           2007
eunbi                7       1995
sakura               2       1998
hyewon               8       1999
yena                 4       1999
chaeyeon            12       2000
chaewon             10       2000
minju               11       2001
nako                 6       2001
hitomi               9       2001
yuri                 3       2001

14 rows selected.
```

izone테이블과 ive 테이블에 동시에 존재하는 원소들을 제외한 원소들을 출력한다.

7.sh

```
-- 1. FULL OUTER JOIN
SELECT * FROM izone z FULL OUTER JOIN ive v ON z.name = v.name;

-- 1.1 FULL OUTER JOIN (ONLY OUTER)
SELECT * FROM izone z FULL OUTER JOIN ive v
  ON z.name = v.name
  WHERE z.name is null OR v.name is null;
```

8. 안티 조인, 세미 조인

8.1 설명

안티 조인

- 기준이 되는 테이블을 중심으로 기준 테이블의 공통 컬럼 값과 다른 값을 가진 또 다른 테이블의 데이터를 추출하는 관계
- Not In 연산자를 사용하는 조인

세미 조인

- EXISTS 연산자를 사용하여 조인
- 서브쿼리내에서 존재하는 데이터만 추출
- 안티조인의 반대

8.2 시나리오 수행

시나리오 내용
1. 안티 조인 (부서 위치가 'NEW YORK'이 아닌 사원 출력)
2. 세미 조인 (sal가 2000이상인 사원이 존재하는 부서의 정보 출력)

8.3 시나리오 수행 결과

1. 안티 조인

```
-- 1. 안티 조인 (부서 위치가 'NEW YORK'이 아닌 사원 출력)
SELECT ename FROM emp
  WHERE emp.deptno NOT IN
    (SELECT dept.deptno FROM dept
      WHERE dept.loc = 'NEW YORK');
```

```
SQL>
SELECT ename FROM emp
WHERE emp.deptno NOT IN
(SELECT dept.deptno FROM dept
   4 WHERE dept.loc = 'NEW YORK');

ENAME
----------
BLAKE
JONES
MARTIN
ALLEN
TURNER
JAMES
WARD
FORD
SMITH
SCOTT
ADAMS

11 rows selected.
```

메인쿼리인 emp의 데이터중에 NOT IN을 사용하여 서브쿼리의 dept 테이블 칼럼 중 dept.loc = 'NEW YORK'인것을 제외한 결과값들을 출력한다.

2. 세미 조인 (sal가 2000이상인 사원이 존재하는 부서의 정보 출력)

```
-- 2. 세미 조인
SELECT dept.* FROM dept
  WHERE EXISTS ( SELECT * FROM emp
                 WHERE emp.deptno = dept.deptno
                   AND emp.sal >= 2000);
```

```
SELECT dept.* FROM dept
WHERE EXISTS ( SELECT * FROM emp
WHERE emp.deptno = dept.deptno
   4 AND emp.sal >= 2000);

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        30 SALES          CHICAGO
        20 RESEARCH       DALLAS

3 rows selected.
```

emp 테이블과 dept 테이블의 deptno가 같은 값들을 뽑아내고, sal이 2000 이상인 직원만 뽑아서 추출한다. EXISTS를 붙였기 때문에 서브쿼리인 emp 테이블 내에 있는 데이터만 추출하고 중복을 제거해서 결과를 띄운다.

8.sh

```
-- 1. 안티 조인 (부서 위치가 'NEW YORK'이 아닌 사원 출력)
SELECT ename FROM emp
  WHERE emp.deptno NOT IN
    (SELECT dept.deptno FROM dept
      WHERE dept.loc = 'NEW YORK');


-- 2. 세미 조인 (sal가 2000이상인 사원이 존재하는 부서의 정보 출력)
SELECT dept.* FROM dept
  WHERE EXISTS ( SELECT * FROM emp
                  WHERE emp.deptno = dept.deptno
                    AND emp.sal >= 2000);
```

---

9. CONNECT BY

9.1 설명

계층 데이터를 위해 CONNECT BY 절을 이용한다.

구문	설명
WHERE	데이터를 가져온 뒤 마지막으로 조건절에 맞게 정리
START WITH	어떤 데이터로 계층구조를 지정하는지 지정
CONNECT BY	각 행들의 연결 관계를 설정

- START WITH 는 가장 처음에 데이터를 거르는 플랜을 타게 되고, 따라서 이 컬럼에는 인덱스가 걸려있어야 성능을 보장받는다.
- CONNECT BY 절의 결과에는 LEVEL 이라는 컬럼이 있으며, 이는 계층의 깊이를 의미한다.

출처 : https://mozi.tistory.com/155

CONNECT_BY_ROOT

제일 레벨이 높은 row를 반환하는 연산자

SYS_CONNECT_BY_PATH

루트 데이터부터 현재 전개할 데이터까지의 경로를 표시한다.

9.2 시나리오 수행

시나리오 내용
1. emp 테이블에 있는 정보 출력
2. JOB = 'PRESIDENT'를 기준으로 (계층의 level 1으로 설정) empno, mgr이 같은 관계를 계층적으로 출력함.
3. CONNECT_BY_ROOT, SYS_CONNECT_BY_PATH

9.3 시나리오 수행 결과

1. emp 테이블에 있는 정보 출력

```
-- 1. emp 테이블에 있는 정보 출력
SELECT empno, ename, mgr, job FROM emp;
```

```
SQL> SELECT empno, ename, mgr, job FROM emp;

     EMPNO ENAME                MGR JOB
---------- ---------- ---------- ---------
      7839 KING                  PRESIDENT
      7698 BLAKE               7839 MANAGER
      7782 CLARK               7839 MANAGER
      7566 JONES               7839 MANAGER
      7654 MARTIN              7698 SALESMAN
      7499 ALLEN               7698 SALESMAN
      7844 TURNER              7698 SALESMAN
      7900 JAMES               7698 CLERK
      7521 WARD                7698 SALESMAN
      7902 FORD                7566 ANALYST
      7369 SMITH               7902 CLERK
      7788 SCOTT               7566 ANALYST
      7876 ADAMS               7788 CLERK
      7934 MILLER              7782 CLERK

14 rows selected.
```

2. JOB = 'PRESIDENT'를 기준으로 (계층의 level 1으로 설정) empno, mgr이 같은 관계를 계층적으로 출력함.

```
-- 2. JOB = 'PRESIDENT'를 기준으로 (계층의 level 1)
--    empno, mgr이 같은 관계를 계층적으로 출력함.
SELECT LEVEL, empno, ename, mgr, job
  FROM emp
  START WITH job = 'PRESIDENT'
  CONNECT BY PRIOR empno = mgr;
```

```
SELECT LEVEL, empno, ename, mgr, job
FROM emp
START WITH job = 'PRESIDENT'
CONNECT BY PRIOR empno = mgr
   5 ORDER BY level;

     LEVEL      EMPNO ENAME                MGR JOB
---------- ---------- ---------- ---------- ---------
         1       7839 KING                   PRESIDENT
         2       7566 JONES              7839 MANAGER
         2       7782 CLARK              7839 MANAGER
         2       7698 BLAKE              7839 MANAGER
         3       7788 SCOTT              7566 ANALYST
         3       7902 FORD               7566 ANALYST
         3       7934 MILLER             7782 CLERK
         3       7521 WARD               7698 SALESMAN
         3       7900 JAMES              7698 CLERK
         3       7844 TURNER             7698 SALESMAN
         3       7499 ALLEN              7698 SALESMAN
         3       7654 MARTIN             7698 SALESMAN
         4       7876 ADAMS              7788 CLERK
         4       7369 SMITH              7902 CLERK

14 rows selected.
```

3. CONNECT_BY_ROOT, SYS_CONNECT_BY_PATH

```
-- 3. CONNECT_BY_ROOT, SYS_CONNECT_BY_PATH
COL PATH FOR A30;

SELECT ENAME, CONNECT_BY_ROOT ENAME mgr,
            SYS_CONNECT_BY_PATH(ENAME, '-') PATH
     FROM EMP
     WHERE LEVEL > 1
       CONNECT BY PRIOR EMPNO = mgr
       START WITH ENAME = 'BLAKE';
```

```
SQL> COL PATH FOR A30;
SELECT ENAME, CONNECT_BY_ROOT ENAME mgr,
            SYS_CONNECT_BY_PATH(ENAME, '-') PATH
     FROM EMP
     WHERE LEVEL > 1
       CONNECT BY PRIOR EMPNO = mgr
   6         START WITH ENAME = 'BLAKE';

ENAME      MGR        PATH
---------- ---------- ------------------------------
WARD       BLAKE      -BLAKE-WARD
JAMES      BLAKE      -BLAKE-JAMES
TURNER     BLAKE      -BLAKE-TURNER
ALLEN      BLAKE      -BLAKE-ALLEN
MARTIN     BLAKE      -BLAKE-MARTIN

5 rows selected.
```

9.sh

```
-- 1. emp 테이블에 있는 정보 출력
SELECT empno, ename, mgr, job FROM emp;

-- 2. JOB = 'PRESIDENT'를 기준으로 (계층의 level 1으로 설정)
--    empno, mgr이 같은 관계를 계층적으로 출력함.
SELECT LEVEL, empno, ename, mgr, job
  FROM emp
  START WITH job = 'PRESIDENT'
  CONNECT BY PRIOR empno = mgr
  ORDER BY level;


-- 3. CONNECT_BY_ROOT, SYS_CONNECT_BY_PATH
COL PATH FOR A30;

SELECT ENAME, CONNECT_BY_ROOT ENAME mgr,
            SYS_CONNECT_BY_PATH(ENAME, '-') PATH
     FROM EMP
     WHERE LEVEL > 1
       CONNECT BY PRIOR EMPNO = mgr
       START WITH ENAME = 'BLAKE';
```

10. 병렬 질의

10.1 설명

병렬 질의

- 하나의 SQL 문장을 여러 워킹 스레드를 사용하여 처리하는 것
- 대용량 테이블을 스캔할 때 여러 워킹 스레드가 테이블의 영역을 나눠서 처리하면 워킹 스레드 하나를 사용했을 때보다 빠르게 작업을 실행할 수 있다.
- 대용량 데이터를 다루는 환경에서 주로 사용

```
SELECT /*+ PARALLEL (4) */ DEPTNO, AVG(SAL) FROM EMP GROUP BY DEPTNO;
```

'PARALLEL (4)'라고 명시한 부분의 숫자 4 : 병렬 질의 처리에서 사용할 워킹 스레드의 개수

→ **DOP** (Degree of parallelism)

→ 4개의 워킹 스레드를 사용해 해당 질의를 처리하도록 지시한다.

10.2 시나리오 수행

시나리오 내용
1. SQL을 실제 수행하고 그 결과와 함께 실행계획을 출력하도록 설정 및 EMP 테이블 정보 출력
2. PARALLEL 힌트 사용

10.3 시나리오 수행 결과

1. SQL을 실제 수행하고 그 결과와 함께 실행계획을 출력하도록 설정 및 EMP 테이블 정보 출력

```
-- 1. SQL을 실제 수행하고 그 결과와 함께 실행계획을 출력하도록 설정
--    및 EMP 테이블 정보 출력
SET AUTOT TRACE EXPLAIN
SELECT EMPNO, ENAME, SAL FROM EMP ORDER BY SAL;
```

```
-- 1. SQL을 실제 수행하고 그 결과와 함께 실행계획을 출력하도록 설정
--    및 EMP 테이블 정보 출력
SET AUTOT TRACE EXPLAIN
SQL> SELECT EMPNO, ENAME, SAL FROM EMP ORDER BY SAL;

SQL ID: 7xbx9dk5txhgt
Child number: 147
Plan hash value: 4214489257

Execution Plan
----------------------------------------------------------------------------
--------
   1  ORDER BY (SORT) (Cost:23, %%CPU:0, Rows:14)
   2    TABLE ACCESS (FULL): EMP (Cost:23, %%CPU:0, Rows:14)
```

2. PARALLEL 힌트 사용

```
-- 2. PARALLEL 힌트 사용
SELECT /*+ PARALLEL (4) */ EMPNO, ENAME, SAL FROM EMP ORDER BY SAL;
```

```
-- 2. PARALLEL 힌트 사용
SQL> SELECT /*+ PARALLEL (4) */ EMPNO, ENAME, SAL FROM EMP ORDER BY SAL;

SQL ID: 8j5fdj2568ajx
Child number: 153
Plan hash value: 188909282

Execution Plan
----------------------------------------------------------------------------
--------
   1  PE MANAGER (Cost:0, %%CPU:0, Rows:14)
   2    PE SEND QC (ORDER) (Cost:0, %%CPU:0, Rows:14)
   3      ORDER BY (SORT) (Cost:23, %%CPU:0, Rows:14)
   4        PE RECV (Cost:0, %%CPU:0, Rows:14)
   5          PE SEND (RANGE) (Cost:0, %%CPU:0, Rows:14)
   6            PE BLOCK ITERATOR (Cost:23, %%CPU:0, Rows:14)
   7              TABLE ACCESS (FULL): EMP (Cost:23, %%CPU:0, Rows:14)
```

PARALLEL 힌트를 사용한 병렬 질의의 실행 계획에 기존에는 없던 새로운 연산 노드가 추가가 된 것을 확인 할 수 있다. 이때 새로 추가된 노드는 병렬 질의 실행에 필요한 작업을 하게 된다.

PE RECV, PE SEND 노드가 추가된 경우에는 Tibero는 병렬 질의를 실행하기 위해 2-set 모델을 사용한다.

이 경우 실제 사용하는 워킹 스레드의 개수는 DOP의 2배가 되며 2개의 set 중에서 한 곳의 워킹 스레드는 consumer의 역할을, 다른 set의 워킹 스레드는 producer의 역할을 하게 된다. 이렇게 2-set 모델을 사용하는 이유는 두 연산 노드를 동시에 병렬로 실행하여 파이프라이닝(Pipelining) 효과를 보기 위해서이다.

10.sh

```
-- 1. SQL을 실제 수행하고 그 결과와 함께 실행계획을 출력하도록 설정
--    및 EMP 테이블 정보 출력
SET AUTOT TRACE EXPLAIN
SELECT EMPNO, ENAME, SAL FROM EMP ORDER BY SAL;

-- 2. PARALLEL 힌트 사용
SELECT /*+ PARALLEL (4) */ EMPNO, ENAME, SAL FROM EMP ORDER BY SAL;
```

---

11. 실체화 뷰 - 완전 문자열 비교

11.1 설명

실체화 뷰

- DB엔진이 쿼리가 실행될 때마다 뷰를 동적으로 재구성하지 않도록 실체화된 뷰를 만들 수 있다.
- 뷰가 가상 테이블이 아닌 실제 물리적인 테이블이 되는 것이다.

ENABLE QUERY REWRITE : query rewrite 활성화

출처 : http://wiki.gurubee.net/pages/viewpage.action?pageId=26742369

11.2 시나리오 수행

시나리오 내용
1. 실체화 뷰 생성
2. 질의 실행
3. 질의 다시 쓰기가 동작함

11.3 시나리오 수행 결과

1. 실체화 뷰 생성

```
-- 1. 실체화 뷰 생성
CREATE MATERIALIZED VIEW MV_SUM_SALARY ENABLE QUERY REWRITE AS
    SELECT DNAME, SUM(SAL) FROM DEPT, EMP
    WHERE DEPT.DEPTNO = EMP.DEPTNO
    GROUP BY DNAME;
```

```
CREATE MATERIALIZED VIEW MV_SUM_SALARY ENABLE QUERY REWRITE AS
    SELECT DNAME, SUM(SAL) FROM DEPT, EMP
WHERE DEPT.DEPTNO = EMP.DEPTNO
   4     GROUP BY DNAME;

Materialized View 'MV_SUM_SALARY' created.
```

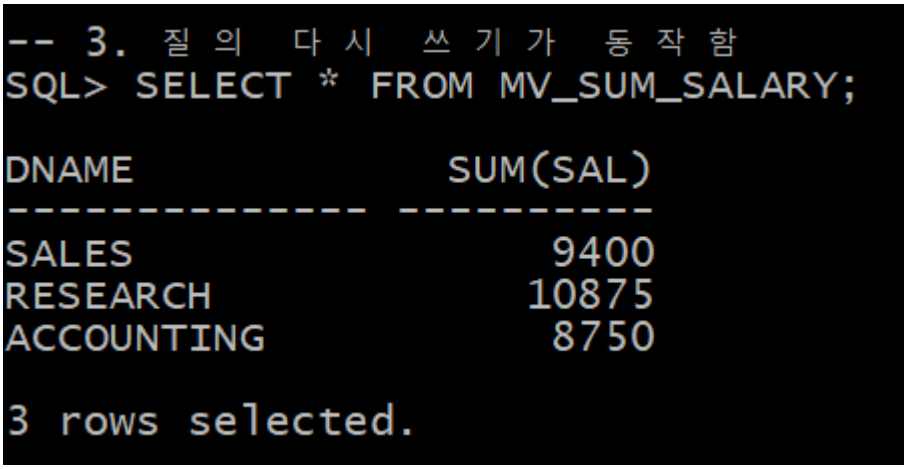
2. 질의 실행

```
-- 2. 질의 실행
SELECT dname, SUM(sal) FROM dept,emp WHERE DEPT.DEPTNO = EMP.DEPTNO
    GROUP BY dname;
```

```
-- 2. 질 의  실 행
SELECT dname, SUM(sal) FROM dept,emp WHERE DEPT.DEPTNO = EMP.DEPTNO
   2     GROUP BY dname;

DNAME            SUM(SAL)
-------------- ----------
SALES                9400
RESEARCH            10875
ACCOUNTING           8750

3 rows selected.
```

3. 질의 다시 쓰기가 동작함

```
-- 3. 질의 다시 쓰기가 동작함
SELECT * FROM MV_SUM_SALARY;
```



11.sh

```
-- 1. 실체화 뷰 생성
CREATE MATERIALIZED VIEW MV_SUM_SALARY ENABLE QUERY REWRITE AS
    SELECT DNAME, SUM(SAL) FROM DEPT, EMP
    WHERE DEPT.DEPTNO = EMP.DEPTNO
    GROUP BY DNAME;

-- 2. 질의 실행
SELECT dname, SUM(sal) FROM dept,emp WHERE DEPT.DEPTNO = EMP.DEPTNO
    GROUP BY dname;

-- 3. 질의 다시 쓰기가 동작함
SELECT * FROM MV_SUM_SALARY;
```