SQL #2. QUERY (2)

I. JOIN

조인은 두가지 방법으로 수행이 가능하다.
FROM 절에 테이블을 나열하고 WHERE절에 조인 조건을 주는 방법도 가능하고,
SQL-99에서 사용하는 JOIN 구문을 직접 이용할 수도 있다.
다음 실습을 통하여 두가지 방법을 모두 확인해보자.

SQL*PLUS를 실행하여 scott 계정으로 접속하시오.

```
SQL> conn scott/tiger
```

Cartesian product
두 테이블을 카티전 프로덕트를 하게 되면 모든 가능한 쌍이 나타난다.

```
SQL> SELECT * FROM emp, dept;
SQL> SELECT * FROM emp CROSS JOIN dept;
```

(* SQL:1999 Syntax)

EQUI Join
EQUAL 조건에 의한

```
SQL> SELECT * FROM dept, emp
        WHERE dept.deptno = emp.deptno;
```

 # 자연조인: 위의 결과와 어떠한 점이 다른가?

```
SQL> SELECT * FROM dept NATURAL JOIN emp;
```

(* SQL:1999 Syntax)

```
SQL> SELECT * FROM dept  JOIN emp USING(deptno);
```

(* SQL:1999 Syntax)

```
SQL> SELECT * FROM dept d JOIN emp e ON (d.deptno = e.deptno);
```

(* SQL:1999 Syntax)

 # join condition 이외에 WHERE 절에 추가적인 condition을
 # AND나 OR 연산자를 이용하여 줄 수 있다.

좀더 복잡한 스키마에서 질의연습을 해보기 위해 hr계정으로 접속하자
hr계정이 Locked되어 있는 사람은 지난시간 실습 내용을 바탕으로 Lock을 풀어야 한다.

SQL> conn hr

 # 20번 부서의 이름과 그 부서에 근무하는 사원의 이름을 출력하시오.
SQL> SELECT d.department_name, e.last_name FROM departments d, employees e
WHERE d.department_id = e.department_id AND d.department_id = 20;

 # 1400,1500 번 위치의 도시 이름과 그곳에 있는 부서의 이름을 출력하시오.
SQL> SELECT l.city, d.department_name FROM locations l, departments d
WHERE l.location_id = d.location_id AND (l.location_id =1400 OR l.location_id = 1500);

SQL> SELECT l.city, d.department_name FROM locations l, departments d
WHERE l.location_id = d.location_id AND l.location_id in (1400, 1500);

 # 다시 scott계정으로 접속
SQL> conn SCOTT/TIGER

NON-EQUI Join
조건이 equal이 아닌 조인 theta 조인이라고도 부름

 # 연봉에 따른 등급 표시
SQL> SELECT e.ename, e.job, e.sal, s.grade
FROM emp e, salgrade s
WHERE e.sal BETWEEN s.losal AND s.hisal;

OUTER JOIN
조인에 참가하지 못한 레코드도 결과에 포함한다.
(+)의 위치에 따라 어느쪽의 레코드들이 포함되는지 주의하라
LEFT, RIGHT의 위치에 따라 어느쪽의 레코드들이 포함되는지 주의하라

 # 다음 문장들의 실행 결과를 비교해 보시오.
SQL> SELECT ename, dname FROM emp e, dept d WHERE e.deptno (+)= d.deptno;
SQL> SELECT ename, dname FROM dept LEFT OUTER JOIN emp USING (deptno);

(* SQL:1999 Syntax)

 # 임시 테스트용 : 어떠한 부서에도 속하지 않는 사원을 추가
 # user는 현재 사용자의 id를 가짐
SQL> INSERT INTO emp (empno, ename) VALUES (4444,user);
SQL> SELECT * FROM emp;

SQL> SELECT ename, dname FROM emp e, dept d WHERE e.deptno = d.deptno(+);
SQL> SELECT ename, dname FROM dept RIGHT OUTER JOIN emp USING (deptno);
(* SQL:1999 Syntax)

SQL> SELECT ename, dname FROM emp e, dept d WHERE e.deptno(+) = d.deptno(+);
SQL> SELECT ename, dname FROM dept FULL OUTER JOIN emp USING (deptno);
(* SQL:1999 Syntax)

 # 각 부서별 인원을 출력하시오. (결과를 비교해보시오.)
 # COUNT는 레코드의 개수를 센다.
 # 외부조인(outer join)이 없이는 부서원이 없는 부서는 결과에 나타날 수 없다.

SQL> SELECT dname from dept;

SQL> SELECT dname, count(empno) FROM emp e, dept d WHERE e.deptno = d.deptno group by d.deptno, d.dname;

SQL> SELECT dname, count(empno) FROM emp e, dept d WHERE e.deptno (+) = d.deptno group by d.deptno, d.dname;

 # 임시로 넣은 데이터를 삭제하자
SQL> delete from emp where empno = 4444;

SELF-JOIN
자기자신의 테이블에 조인

SQL> select e.ename follower, m.ename manager from emp e, emp m where e.mgr = m.empno;
SQL> select e.ename follower, m.ename manager from emp e, emp m where e.mgr = m.empno (+);
SQL> select e.ename follower, NVL(m.ename, 'I am Boss') manager from emp e, emp m where

e.mgr = m.empno (+);
SQL> select NVL(e.ename, 'I have no follower') follower, NVL(m.ename, 'I am Boss') manager from emp e FULL OUTER JOIN emp m ON e.mgr = m.empno;

```
==============================================
# TREE TRAVERSE
Hierarchical 질의
==============================================
```

 # FROM TOP TO BOTTOM

```
SQL> select empno, ename, job, mgr, level from emp
        START WITH mgr IS NULL
        CONNECT BY PRIOR empno = mgr
        ORDER BY level;
```

 # FROM BOTTOM TO TOM

```
SQL> select empno, ename, job, level from emp
        START WITH ename = 'SMITH'
        CONNECT BY PRIOR mgr = empno;
```

 # 다음 두 문장을 비교해보시오

```
SQL> SELECT LPAD('+' || ename, LENGTH(ename)+(LEVEL*3)-3,'-') AS org_chart
        FROM  emp
        START WITH mgr IS NULL
        CONNECT BY PRIOR  empno = mgr;

SQL> SELECT LPAD('+' || ename, LENGTH(ename)+(LEVEL*3)-3,'-') AS org_chart
        FROM  emp
        START WITH mgr IS NULL
        CONNECT BY PRIOR  empno = mgr AND ename != 'BLAKE';
```

 # (WHERE절에 조건을 준것이랑 어떤 차이가 발생하나?)

```
SQL> SELECT LPAD('+' || ename, LENGTH(ename)+(LEVEL*3)-3,'-') AS org_chart
        FROM  emp
        WHERE ename != 'BLAKE'
        START WITH mgr IS NULL
        CONNECT BY PRIOR  empno = mgr;
```

II. Aggregation

```
========================================
# Aggregation
========================================
```

 # 아래의 문장들을 실행하면서 결과를 확인하시오.
 # 결과가 차이가 나는 이유는 무엇인가?

 # COUNT

```
SQL> SELECT count(*)  FROM emp;
SQL> SELECT count(comm)  FROM emp;
SQL> SELECT count(job)  FROM emp;
SQL> SELECT count(DISTINCT job)  FROM emp;
```

 # SUM

```
SQL> SELECT sum(sal)  FROM emp;
SQL> SELECT sum(DISTINCT sal)  FROM emp;
```

 # AVG

```
SQL> SELECT avg(comm)  FROM emp;
SQL> SELECT sum(comm)/count(*)  FROM emp;
SQL> SELECT sum(comm)/count(comm)  FROM emp;
```

 # 합, 평균, 최소, 최대, 표준편차, 분산

```
SQL> SELECT sum(sal), avg(sal), min(sal), max(sal), stddev(sal), variance(sal)  FROM emp;
```

```
========================================
# Grouping
========================================
```

 # 다음을 실습해 보시오.

```
SQL> SELECT  job, avg(sal), max(sal), min(sal)
        FROM  emp  GROUP BY  job;
```

```
SQL> SELECT   deptno, count(*)
          FROM   emp   GROUP BY   deptno;
```

 # Stock Clerk 직종의 사원들의 부서별 인원수를 출력하시오.

```
SQL> SELECT deptno, COUNT(*) "Number"    FROM emp
          WHERE job = 'CLERK'   GROUP BY deptno;
```

 # 직종별 최고 급여를 급여가 많은 직종부터 출력하시오.

```
SQL> SELECT job, max(sal)   FROM emp
          GROUP BY job   ORDER BY max(sal) DESC;
```

 # 직종별 최고 급여를 급여가 많은 직종부터 출력하시오. (단, 직원이 2명 이상인 경우만)
 # HAVING 사용법

```
SQL> SELECT job, max(sal)   FROM emp
          GROUP BY job
     HAVING count(sal) >= 2
     ORDER BY max(sal) DESC;
```

 # 다음은 무엇이 잘못되었는가? 어떤 차이가 있는가?

```
SQL> SELECT d.deptno, d.dname, count(e.ename) cnt
      from emp e, dept d
      where e.deptno = d.deptno;

SQL> SELECT d.deptno, d.dname, count(e.ename) cnt
      from emp e, dept d
      where e.deptno = d.deptno
      group by d.deptno;

SQL> SELECT d.deptno, d.dname, count(e.ename) cnt
      from emp e, dept d
      where e.deptno = d.deptno
      group by d.deptno, d.dname;

SQL> SELECT d.deptno, d.dname, count(e.ename) cnt
      from emp e, dept d
```

```
    where e.deptno = d.deptno
    group by d.deptno, d.dname;

SQL> SELECT d.deptno, d.dname, count(e.ename) cnt
    from emp e, dept d
    where e.deptno (+) = d.deptno
    group by d.deptno, d.dname;
```

```
========================================
# ROLLUP & CUBE
========================================
```

```
SQL> SELECT deptno, job, sum(sal) FROM emp
    GROUP BY ROLLUP (deptno, job);

SQL> SELECT deptno, job, sum(sal) FROM emp
    GROUP BY CUBE (deptno, job);

SQL> SELECT DECODE(GROUPING(deptno), 1, '--ALL--', deptno) "DEPT NO",
          DECODE(GROUPING(job), 1, '--ALL--',job) "JOB",
    sum(sal)
    FROM emp
    GROUP BY CUBE (deptno, job);

SQL> SELECT deptno, job, sum(sal)
          FROM emp
          GROUP BY GROUPING SETS ((deptno,job),(job),())
          ORDER BY 1,2;
```

III. Subquery

```
================================================================
# Single-row Subquery vs. Multi-Row Subquery
================================================================
```

 # 다음 중 에러가 나는 것은 무엇이며 이유는 무엇인가?

```
SQL> SELECT   ename,  sal,  deptno
          FROM   emp
```

```
        WHERE ename =  (SELECT  MIN(ename) FROM  emp);

SQL> SELECT  ename,  sal,  deptno
        FROM  emp
        WHERE ename =  (SELECT  MIN(ename) FROM  emp GROUP BY deptno);

SQL> SELECT  ename,  sal,  deptno
        FROM  emp
        WHERE ename IN  (SELECT  MIN(ename) FROM  emp GROUP BY deptno);
```

```
================================================================
# EXISTS 연산자
================================================================
```

 # 사원이 한명이라도 있는 부서명을 출력하시오.

```
SQL> SELECT dname FROM dept d
        WHERE EXISTS ( SELECT 1 FROM emp WHERE deptno = d.deptno);
```

```
================================================================
# Subquery 의 종류
================================================================
```

 # 각 부서별로 최고급여를 받는 사원을 출력하시오.

 # Nested Query

```
SQL> SELECT deptno, empno, ename, sal
        FROM emp
        WHERE (deptno,sal) IN  (SELECT deptno, max(sal)
                                FROM emp GROUP BY deptno);
```

 # Inline View

```
SQL> SELECT e.deptno, e.empno, e.ename, e.sal
        FROM emp e,  (SELECT s.deptno, max(s.sal)  msal
                              FROM emp s GROUP BY deptno) m
         WHERE e.deptno = m.deptno AND e.sal = m.msal;
```

 # Correlated Query

```
SQL> SELECT deptno, empno, ename, sal
        FROM emp  e
```

```
        WHERE e.sal = (SELECT max(sal) FROM emp WHERE deptno = e.deptno);
```

```
================================================================
# TOP-K 질의
================================================================
```

 # 급여를 많이 받는 순서대로 상위 3명을 출력하시오.

```
SQL> SELECT rownum, ename, sal
        FROM (SELECT * FROM emp  ORDER BY sal DESC)
        WHERE rownum < 4;
```

```
================================================================
# ROWNUM (PSEUDO 컬럼)과 RANK 관련 함수들
================================================================
```

 # 아래 결과가 예상과 동일하게 나오는가?

```
SQL> SELECT sal, ename, rownum
        FROM emp
        ORDER BY sal DESC;
```

 # RANK 관련 함수들은 각각 어떤 차이가 존재하는가?

```
SQL> SELECT sal, ename,
          RANK() OVER (ORDER BY sal DESC) AS rank,
          DENSE_RANK() OVER (ORDER BY sal DESC) AS dense_rank,
          ROW_NUMBER() OVER (ORDER BY sal DESC) AS row_number,
          rownum AS "rownum"
       FROM emp;

SQL> SELECT sal, ename,
          RANK() OVER (ORDER BY sal DESC) AS rank,
          DENSE_RANK() OVER (ORDER BY sal DESC) AS dense_rank,
          ROW_NUMBER() OVER (ORDER BY sal DESC) AS row_number
        FROM emp
        ORDER BY ename;
```