**[문제1] PT4-13, emp, dept 테이블에서 등가조인을 사용하여 오라클 Join, ANSI Join을 적용한 결과를 구하도록 SQL 문을 실습하라. 또한, 2개의 결과 테이블이 동일한지를 확인하라.**

SQL> select \* from emp;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

---------- ---------- --------- ---------- --------- ---------- ---------- ----------

7839 KING PRESIDENT 17-NOV-96 5000 10

7698 BLAKE MANAGER 7839 05-JAN-91 2850 30

7782 CLARK MANAGER 7839 06-SEP-99 2450 10

7566 JONES MANAGER 7839 04-FEB-01 2975 20

7788 SCOTT ANALYST 7566 17-JUN-03 3000 20

7902 FORD ANALYST 7566 12-MAR-81 3000 20

7369 SMITH CLERK 7902 01-DEC-07 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7844 TURNER SALESMAN 7698 09-AUG-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 12-MAR-81 950 30

7934 MILLER CLERK 7782 23-JAN-03 1300 10

1000 Tiger 3600 10

2000 Cat

SQL> select \* from dept;

DEPTNO DNAME LOC

---------- -------------- -------------

90 Hardware Urgench

50 ????

60 DBLAB SOFT212

10 ACCOUNTING BUSAN

20 RESEARCH DALLAS

30 SALES CHICAGO

40 OPERATIONS BOSTON

80 AAA dddd

SQL> select e.empno, e.ename, d.dname from emp e, dept d where e.deptno=d.deptno;

SQL> select e.empno, e.ename, d.dname from emp e join dept d on e.deptno = d.deptno;

EMPNO ENAME DNAME

---------- ---------- --------------

7839 KING ACCOUNTING

7698 BLAKE SALES

7782 CLARK ACCOUNTING

7566 JONES RESEARCH

7788 SCOTT RESEARCH

7902 FORD RESEARCH

7369 SMITH RESEARCH

7499 ALLEN SALES

7521 WARD SALES

7654 MARTIN SALES

7844 TURNER SALES

7876 ADAMS RESEARCH

7900 JAMES SALES

7934 MILLER ACCOUNTING

1000 Tiger ACCOUNTING

2000 Cat SALES

**[실행결과]**

select 절에서 ansi표준으로 했을 시에 [inner] join on 절로 조인의 조건 절을

주지만, 오라클 표준은 from [table], [table] where [on절의 조건] 으로 적어줍니다.

그래서 ansi표준과 join을 쓰지 않은 orcale 표준으로 적은 조인이 같은 결과를 확인한다는 것을 보았습니다.

**[문제2] PT4-15, emp, dept 테이블에서 적용한 조인에서 오류 발생여부를 확인하고 오류가 발생하면 오류를 해결한 결과가 나오도록 SQL 문을 실습하라.**

SQL> select empno, ename, deptno, dname

2 from emp e, dept d

3 where e.deptno = d.deptno;

select empno, ename, deptno, dname

\*

ERROR at line 1:

ORA-00918: column ambiguously defined

**[실행결과]**

단, 위의 문제에서 테이블에 alias 를 주고 e.ename 이런식으로 적어서 dept 테이블과 emp 테이블에 대해서 컬럼 구분을 확실하게 했지만, 조인 되어지는 테이블에 같은 이름이 있을 시에는 꼭 테이블이름.컬럼이름 을 통해서 구분해주어야 합니다.

\* 단, 컬럼명이 겹치지 않을 경우에는 생략이 가능합니다.

**[문제3] PT4-21, 22, Customer, Gift 테이블을 생성하고 고객 테이블에 3명, gift 테이블에 5개의 데이터를 입력한후**

**(1) 오라클 Join, (2) ANSI Join에서 between 절을 사용한 조인적용 결과를 구하도록 SQL 문과 (3) 비등가 조인(PT4-22)의 결과를 구하는 SQL를 실습하라.**

**또한, 3개 테이블의 결과가 동일한지를 확인하라.**

SQL> create table regift(gname varchar(20), g\_start number, g\_end number);

Table created.

**우선 문제와 상관없이 기존에 있던 테이블들이 이상해서 두 테이블 다 제가 다시만들었습니다.**

SQL> INSERT INTO regift VALUES('ipad', 500000, 1000000);

SQL> INSERT INTO regift VALUES('iphone', 1000000, 1500000);

SQL> INSERT INTO regift VALUES('mac', 1500000, 2000000);

SQL> INSERT INTO regift VALUES('apple watch', 300000, 500000);

SQL> INSERT INTO regift VALUES('apple pencil', 100000, 200000);

**이렇게 상품을 등록해준 후 regift 테이블을 확인해서 결과를 출력하고 정상적으로 입력되었다는 것을 확인합니다.**

**여기서 gname은 상품의 이름 g\_start는 상품의 시작가격 g\_end는 상품의 최고가입니다.**

GNAME G\_START G\_END

-------------------- ---------- ----------

ipad 500000 1000000

iphone 1000000 1500000

mac 1500000 2000000

apple watch 300000 500000

apple pencil 100000 200000

**그리고 고객테이블도 따로 만들어줍니다.**

**고객테이블은 고객의 이름과 point를 나타내는 두개의 열만이 존재합니다.**

SQL> create table recustomer(cname varchar(20), point number);

Table created.

SQL> INSERT INTO recustomer VALUES('Lisa',40000);

SQL> INSERT INTO recustomer VALUES('Stive',190000);

SQL> INSERT INTO recustomer VALUES('Pullman',190000);

SQL> INSERT INTO recustomer VALUES('Pitt',1400000);

SQL> INSERT INTO recustomer VALUES('Jackson',900000);

SQL> INSERT INTO recustomer VALUES('Json',1200000);

SQL> INSERT INTO recustomer VALUES('Gibson',1800000);

SQL> INSERT INTO recustomer VALUES('Doge',1330000);

SQL> INSERT INTO recustomer VALUES('Hanks',490000);

**이렇게 고객이름과 포인트를 적어줍니다.**

CNAME POINT

-------------------- ----------

Lisa 40000

Stive 190000

Pullman 190000

Pitt 1400000

Jackson 900000

Json 1200000

Gibson 1800000

Doge 1330000

Hanks 490000

**여기서 join으로 물건의 가격의 최저가와 최고가사이에 고객이 받을 수 있는 상품을 나열하겠습니다.**

[오라클 문법]

SQL> select c.cname "CUST\_NAME", TO\_CHAR(c.point,'9,999,999') "POINT", g.gname "GIFT\_NAME"

2 from recustomer c , regift g

3 where c.point between g.g\_start and g.g\_end;

[ansi 문법]

SQL> select c.cname "CUST\_NAME", TO\_CHAR(c.point,'9,999,999') "POINT", g.gname "GIFT\_NAME"

2 from recustomer c inner join regift g

3 on c.point between g.g\_start and g.g\_end;

[ansi 문법 : between 제거]

SQL> select c.cname "CUST\_NAME", TO\_CHAR(c.point,'9,999,999') "POINT", g.gname "GIFT\_NAME"

2 from recustomer c inner join regift g

3 on g.g\_end>=c.point and g.g\_start<=c.point;

CUST\_NAME POINT GIFT\_NAME

-------------------- ---------- --------------------

Gibson 1,800,000 mac

Pitt 1,400,000 iphone

Doge 1,330,000 iphone

Json 1,200,000 iphone

Jackson 900,000 ipad

Hanks 490,000 apple watch

Pullman 190,000 apple pencil

Stive 190,000 apple pencil

8 rows selected.

**셋다 같은 결과가 나온 것을 확인 할 수 있습니다.**

**[실행결과]**

문제에 설명 내용을 적는 것이 이해가 빠를 것 같아서 적었음. 이상 없음.

**[문제4] PT4-23, Student, Score, Hakjum 테이블를 생성하고 데이터를 입력한 후 (1) 입력한 데이터를 확인하고, (2) 오라클 비등가 Join, (3) ANSI 비등가 Join 절을 사용한 조인적용 결과를 구하도록 SQL 문을 실습하고, 2개 테이블의 결과가 동일한지를 확인하라.**

이 문제에서 사용할 테이블 목록.

SQL> select \* from student;

STUDONO NAME PROFNO

---------- ---------- ----------

1 mina 1

2 junha 2

3 hamin 3

4 jin 4

5 ga 5

SQL>

SQL> select \* from score;

STUDNO TOTAL

---------- ----------

1 99

2 79

3 65

4 95

5 81

SQL> select \* from hakjum;

GR MIN\_POINT MAX\_POINT

-- ---------- ----------

A+ 96 100

A0 90 95

B+ 85 89

B0 80 84

C+ 75 79

C0 70 74

D 0 69

**[오라클 표준]**

SQL> select s.name "STU\_NAME", o.total "SCORE", h.grade "CREDIT"

2 from student s, score o, hakjum h

3 where s.studono = o.studno

4 and o.total >= h.min\_point

5 and o.total <= h.max\_point;

STU\_NAME SCORE CR

---------- ---------- --

mina 99 A+

jin 95 A0

ga 81 B0

junha 79 C+

hamin 65 D

**학생들의 pk 값과 score에 점수에 해당하는 fk의 같은 것들을 join 해주고**

**그 다음으로 점수 열의 total이 성적기준인 학점 테이블의 사이값을 where에 놓고 해당되는 학점을 출력했습니다.**

**이를 오라클 표준이 아니라 ansi 표준으로 적게되면**

**[ansi 표준]**

SQL> select s.name "STU\_NAME", o.total "SCORE", h.grade "CREDIT"

2 from student s inner join score o on s.studono = o.studno inner join hakjum h

3 on h.min\_point <= o.total and h.max\_point >= o.total;

STU\_NAME SCORE CR

---------- ---------- --

mina 99 A+

jin 95 A0

ga 81 B0

junha 79 C+

hamin 65 D

**[실행결과]**

문제에 설명 내용을 적는 것이 이해가 빠를 것 같아서 적었음. 이상 없음.

**[문제5] PT4-24, Student, Professor 테이블를 생성하고 (1) 오라클 Outer 조인, (2) ANSI Outer Join 절을 사용한 조인 적용결과를 구하도록**

**SQL 문을 실습하고, 2개 테이블의 결과가 동일한지를 확인하라.**

**+ 추가된 테이블**

SQL> select \* from Professor;

PROFNO NAME

---------- ----------

1 james

2 soo

3 nicole

4 danny

[**오라클 join]**

SQL> select s.name "STU\_NAME", p.name "PROF\_NAME"

2 from student s , professor p

3 where s.profno = p.profno;

담당교수와 그 교수의 담당학생을 출력하는 오라클 join

**[ansi 표준]**

SQL> select s.name "STU\_NAME", p.name "PROF\_NAME"

2 from student s inner join professor p on s.profno = p.profno;

STU\_NAME PROF\_NAME

---------- ----------

mina james

junha soo

hamin nicole

jin danny

**둘의 결과가 동일하다는 것을 확인했습니다.**

**[실행결과]**

문제에 설명 내용을 적는 것이 이해가 빠를 것 같아서 적었음. 이상 없음.

**====================================OUTER JOIN====================================**

**[문제6] PT4-25, Student, Professor 테이블에서 (1) 오라클 Outer 조인, (2) ANSI Outer Join 절을 사용한 조인 적용결과를 구하도록 SQL 문을 실습하고,**

**2개 테이블의 결과가 동일한지를 확인하라.**

**outer join 에서는 (+)를 붙여줌으로서 (+)가 조건절에서 붙지 않은 테이블을 기준으로 right outer join을 수행합니다.**

**즉 B -(INTERSECTION) A(+) 이 됩니다.**

텍스트, 클립아트이(가) 표시된 사진

자동 생성된 설명

**확인을 위해 professor table 에**

SQL> INSERT INTO professor VALUES(6,'json');

SQL> INSERT INTO professor VALUES(7,'jon');

SQL> INSERT INTO professor VALUES(8,'cal');

SQL> INSERT INTO professor VALUES(9,'cal');

**몇개의 임시 데이터를 student의 학생의 pk보다 더 많게 나열했습니다.**

**[오라클 right outer join]**

SQL> select s.name "STU\_NAME", p.name "PROF\_NAME"

2 from student s, professor p

3 where s.profno(+) = p.profno;

STU\_NAME PROF\_NAME

---------- ----------

mina james

junha soo

hamin nicole

jin danny

cal

jon

json

cal

**[오라클 left outer join]**

SQL> select s.name "STU\_NAME", p.name "PROF\_NAME"

2 from student s, professor p

3 where s.profno = p.profno(+);

STU\_NAME PROF\_NAME

---------- ----------

mina james

junha soo

hamin nicole

jin danny

ga

**양방향으로 했을 때는 에러가 나고 오라클에는 ansi 기준으로 full outer join과 오라클 문법으로 union으로 해결한 조인결과가 있습니다.**

**[오라클 ansi right outer join]**

SQL> select s.name "STU\_NAME", p.name "PROF\_NAME"

2 from student s right outer join professor p

3 on s.profno = p.profno;

**이렇게 professor의 데이터들은 다 출력되고 맞지 않는 student는 null을 확인했습니다.**

**[ansi full outer join]**

SQL> select s.name "STU\_NAME", p.name "PROF\_NAME"

2 from student s full outer join professor p

3 on s.profno = p.profno;

STU\_NAME PROF\_NAME

---------- ----------

mina james

junha soo

hamin nicole

jin danny

ga

cal

jon

json

cal

**[실행결과]**

문제에 설명 내용을 적는 것이 이해가 빠를 것 같아서 적었음. 이상 없음.

**[문제7] PT4-26, Student, Professor 테이블에서 (1) 오라클 Outer 조인, (2) ANSI Outer Join (3) 오라클 Outer 조인에서  
   Where 절의 s.profno(+) = p.profno(+) 를 사용한 조인 적용결과를 구하도록 SQL 문을 실습하고, 3개 테이블의 결과가 동일한지를 확인하라.**

SQL> select s.name "STU\_NAME", p.name "PROF\_NAME"

2 from student s, professor p

3 where s.profno(+)=p.profno

4 union

5 select s.name "STU\_NAME", p.name "PROF\_NAME"

6 from student s, professor p

7 where s.profno = p.profno(+);

STU\_NAME PROF\_NAME

---------- ----------

ga

hamin nicole

jin danny

junha soo

mina james

cal

jon

json

8 rows selected.

**[ansi full outer join]**

SQL> select s.name "STU\_NAME", p.name "PROF\_NAME"

2 from student s full outer join professor p

3 on s.profno = p.profno;

STU\_NAME PROF\_NAME

---------- ----------

mina james

junha soo

hamin nicole

jin danny

ga

cal

jon

json

cal

**[실행결과]**

Ansi 형식과 (+)를 양쪽으로 이용해서 중복을 없애주는 union을 이용하면 full outer join과 같은 결과를 얻을 수 있다.

**[문제8] PT4-37, emp 테이블에서 사원번호, 사원이름, 입사일, 자신보다 먼저 입사한 사람 인원수를 출력하세요. 단 자신보다**

**입사일이 빠른 사람 수를 오름차순으로 출력하세요.**

**(Oracle Join 구문과 ANSI Join 구문으로 각각 SQL을 작성하세요)**

위 내용들을 응용해서 풀어보겠습니다.

[오라클]

SQL> select e.empno, e.ename, e.hiredate, COUNT(NVL2(e2.empno, e.empno, NULL)) COUNT

2 from emp e, emp e2

3 where e.hiredate>e2.hiredate(+)

4 group by e.empno, e.ename, e.hiredate

5 order by count;

EMPNO ENAME HIREDATE COUNT

---------- ---------- --------- ----------

1000 Tiger 0

2000 Cat 0

7499 ALLEN 20-FEB-81 0

7521 WARD 22-FEB-81 1

7900 JAMES 12-MAR-81 2

7902 FORD 12-MAR-81 2

7844 TURNER 09-AUG-81 4

7654 MARTIN 28-SEP-81 5

7698 BLAKE 05-JAN-91 6

7839 KING 17-NOV-96 7

7782 CLARK 06-SEP-99 8

7566 JONES 04-FEB-01 9

7934 MILLER 23-JAN-03 10

7788 SCOTT 17-JUN-03 11

7369 SMITH 01-DEC-07 12

7876 ADAMS 23-MAY-87 13

16 rows selected.

[ansi]

SQL> select e.empno, e.ename, e.hiredate, COUNT(NVL2(e2.empno, e.empno, NULL)) COUNT

2 from emp e

3 left join emp e2

4 on e.hiredate > e2.hiredate

5 group by e.empno, e.ename, e.hiredate

6 order by count;

EMPNO ENAME HIREDATE COUNT

---------- ---------- --------- ----------

1000 Tiger 0

2000 Cat 0

7499 ALLEN 20-FEB-81 0

7521 WARD 22-FEB-81 1

7900 JAMES 12-MAR-81 2

7902 FORD 12-MAR-81 2

7844 TURNER 09-AUG-81 4

7654 MARTIN 28-SEP-81 5

7698 BLAKE 05-JAN-91 6

7839 KING 17-NOV-96 7

7782 CLARK 06-SEP-99 8

7566 JONES 04-FEB-01 9

7934 MILLER 23-JAN-03 10

7788 SCOTT 17-JUN-03 11

7369 SMITH 01-DEC-07 12

7876 ADAMS 23-MAY-87 13

16 rows selected.

**[실행결과]**

응용해서 푼 결과 이상 없음.

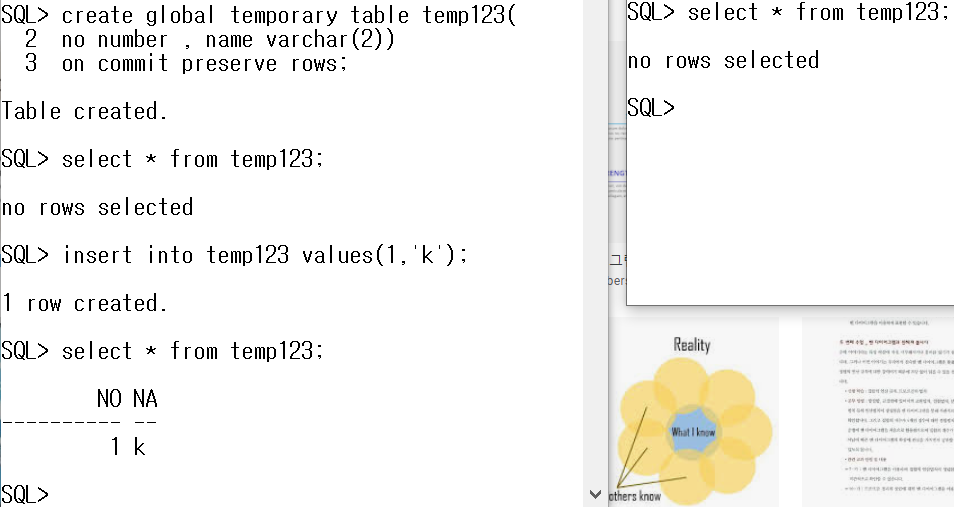
**[문제9] PT5-9, 10, ON COMMIT delete ROWS, ON COMMIT preserve ROWS 2가지 유형의 테이블을 생성하고,**

**서로 다른 세션과 동일한 세션에서 테이블이 세션과 commit 적용시에 서로 다름을 SQL문을 통해 실습하고 결과를 확인하라.**

텍스트이(가) 표시된 사진

자동 생성된 설명

**[ ON COMMIT perserve ROWS]**



**[ ON COMMIT delete ROWS]**

SQL> create global temporary table temp123123(

2 no number, name varchar2(2))

3 on commit delete rows;

SQL> insert into temp123123 values(1,'aa');

SQL> select \* from temp123123;

NO NA

---------- --

1 aa

SQL> commit;

Commit complete.

SQL> select \* from temp123123;

no rows selected

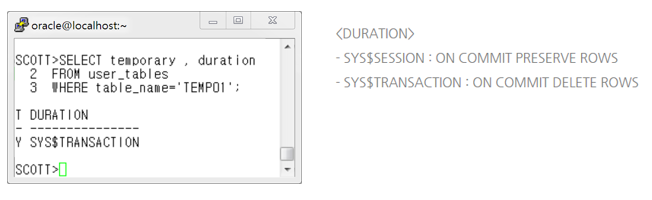
**[실행결과]**

트랜잭션 단위에서 이루어지는 ON COMMIT delete ROWS 는 커밋을 치는 순간 데이터가 없어져버린다.

반대로 세션단위로 이루어지는 ON COMMIT preserve ROWS 는 세션간에 확인이 불가능하며 세션단위로 이루어져 세션이 끊어지는 순간 소멸된다.

**[문제10] PT5-11, 기존에 생성된 임시 테이블의 상태를 확인하도록 SQL문을 실습하라.**

SQL> select temporary, duration from user\_tables where table\_name ='testtable';



**[실행결과]**

테이블의 상태를 예제와 같이 duration 으로 확인해주면서 확인할 수 있습니다.

**[문제11] PT5-12, Emp 테이블에서 Emp2로 Empno, ename, sal 컬럼과 데이터를 복사하도록 SQL 문을 실습하라.**

**[전체 컬럼 복사하기]**

SQL> create table deptcopy

2 as

3 select \* from dept;

Table created.

SQL> select \* from deptcopy;

DEPTNO DNAME LOC

---------- -------------- -------------

90 Hardware Urgench

50 ????

60 DBLAB SOFT212

10 ACCOUNTING BUSAN

20 RESEARCH DALLAS

30 SALES CHICAGO

40 OPERATIONS BOSTON

80 AAA dddd

**[부분 복사하기]**

SQL> create table deptpartcopy

2 as

3 select deptno, loc from dept;

Table created.

SQL> select \* from deptpartcopy;

DEPTNO LOC

---------- -------------

90 Urgench

50

60 SOFT212

10 BUSAN

20 DALLAS

30 CHICAGO

40 BOSTON

80 dddd

**[실행결과]**

As 다음에 가져오고 싶은 테이블을 적어주게 되면 그 테이블의 커스텀화 한만큼 create table 로 새 테이블을 생성할 수 있습니다.

**[문제12] PT5-13, Dept 테이블에서 Dept2로 테이블의 구조만 복사하도록 SQL 문을 실습하라.**

SQL> create table deptnodatacopy

2 as select \* from dept

3 where 1=2;

Table created.

SQL> select \* from deptnodatacopy;

no rows selected

SQL> desc deptnodatacopy;

Name

Null? Type

----------------------------------------------------------------------------------------------------------------- -------- ----------------------------------------------------------------------------

DEPTNO

NUMBER(2)

DNAME

VARCHAR2(14)

LOC

VARCHAR2(13)

SQL>

**[실행결과]**

테이블의 구조(컬럼)만 가져오고 데이터 안 가져오기

주로 데이터는 필요 없이 테이블 구조만 가져올 때 많이 사용하는 방식이며 문법은 거의 동일한데 3번 라인에 보면 WHERE 절에 틀린 조건을 적어 주시는 부분만 다릅니다.

전체적으로 오늘 가벼웠던 내용은 table에 관한 내용이였지만, join을 다뤄봐서 재미있었습니다. 원래는 표준인 ansi를 기준으로만 사용해 왔는데 from 과 where에 join을 대신함으로서 단순하다고 느끼기도 했지만 뭔가 join쿼리가 복잡해졌을 때 단순하다고만 생각했던것들이 ansi 표준보다 더 복잡해질 것같다는 생각을 했습니다.

Jpa는 표준 방언을 지원하는데 그중 oracle도 지원하니까 한번 방언을 바꿔서 쿼리를 날리는 공부를 해야겠다는 생각이 들었습니다.