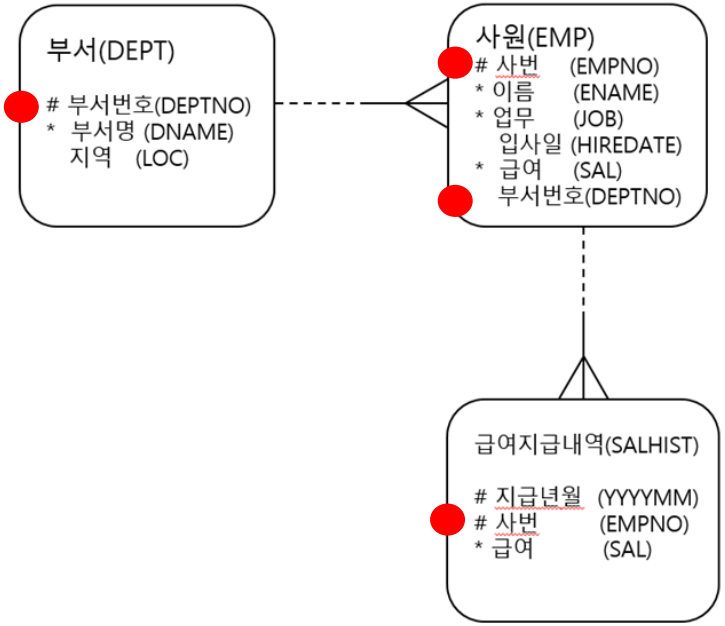
05. Join



#: identifier , * : mandatory

* 튜베로즈: 위험한 관계

1. JOIN

❑ Display data from multiple tables

(한 개 이상의 테이블로 부터 데이터를 읽어야 할때 ➔ 수평적 결합)

- 관계형 데이터베이스는 데이터간 물리적 연결없이 독립적으로 존재 하다가 데이터간 연결이 필요할때 내용에 의한 참조(Content Reference)를 한다. 실시간으로 테이블간 관계 통해 수평적 결합 것이 JOIN.
- JOIN은 관계형 데이터베이스에서 필수적으로 필요한 가장 중요한 연산중 하나.

❑ 필요성

- 관계형 데이터베이스는 **테이블 독립성**, **데이터의 최소 중복성**, **데이터간의 함수적 종속성**을 통해 각각의 데이터를 **각각의 테이블로 나누어(분할) 저장**하도록 설계.
- 조회를 원하는 데이터가 하나의 테이블에 저장되어 있는 경우 SELECT를 통해 조회 하지만 **조회를 원하는 데이터가 여러 테이블에 나누어 저장되어 있는 경우는 JOIN 연산으로 데이터 조회** 한다.

- 사번,이름,급여,부서번호가 필요한 경우

  ➔ SELECT EMPNO,ENAME,SAL,DEPTNO FROM EMP;

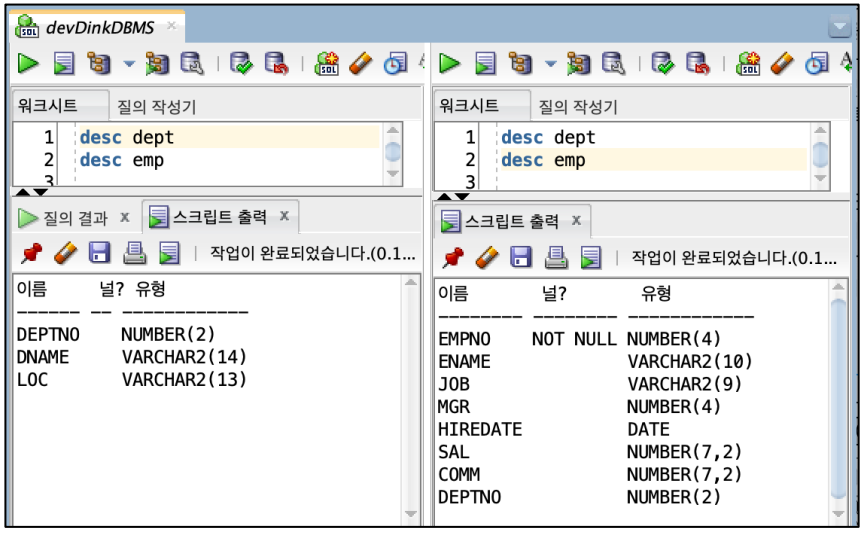
- 부서번호,부서명,부서위치가 필요한 경우

  ➔ SELECT DEPTNO,DNAME,LOC FROM DEPT;

- **부서명**,**이름,급여** 정보가 필요한 경우

  ➔ SELECT **DNAME**, **ENAME,SAL** FROM **DEPT** , **EMP**

  WHERE **DEPT.DEPTNO** = **EMP.DEPTNO** ;



1. JOIN

❑ JOIN 종류

종 류	내 용
Equi-Join (Simple Join , **Inner Join**)	가장 일반적인 형태의 조인으로 조인 대상이 되는 두 테이블간에 공통적으로 존재하는 데이터의 값이 일치되는(Equal)행을 연결하여 결과 집합 생성. 동등 연산자(=)를 사용하여 데이터 일치를 찾는다.
Non Equi-Join	조인 대상이 되는 두 테이블간에 공통적으로 존재하는 데이터의 값을 기준으로 동등 연산자 이외의 비교 연산자 사용을 통해 조건을 만족하는 행을 연결하여 결과 집합 생성.
Outer-Join	조인 조건에 직접적으로 만족되지 않는 정보도 연결하여 결과 집합 생성.
Self-Join	하나의 테이블이 자기 자신(Self)과 조인을 하여 결과 집합 생성.

1. JOIN

❑ Equi-Join

* JOIN에 사용되는 테이블의 컬럼간에 정확이 일치(EQUAL)하는 데이터를 RETURN
* EQUAL(=) 연산자를 사용하여 JOIN

① SELECT DNAME,ENAME,JOB,SAL FROM EMP, DEPT WHERE DEPTNO = DEPTNO;

➔ ORA-00918: 열의 정의가 애매합니다.. 이유는 ?

② SELECT DNAME,ENAME,JOB,SAL FROM SCOTT.EMP, SCOTT.DEPT WHERE **EMP.DEPTNO = DEPT.DEPTNO**;

- OBJECT NAME 표기법 : [SCHEMA.]OBJECT_NAME EX) SCOTT.EMP , EMP
- COLUMN NAME 표기법 : [TABLE_NAME.]COLUMN_NAME EX) EMP.EMPNO, EMPNO

③ SELECT DNAME,ENAME,JOB,SAL FROM EMP, DEPT
WHERE **EMP.DEPTNO = DEPT.DEPTNO** AND **EMP.JOB IN ('MANAGER','CLERK')**
ORDER BY DNAME;

- 조건절 처리 순서?? (a) JOIN 처리 **(EMP.DEPTNO = DEPT.DEPTNO)** → 필터링 (**EMP.JOB IN ('MANAGER','CLERK')**)
  (b) 필터링 → JOIN 처리

④ SELECT D.DNAME,E.ENAME,E.JOB,E.SAL FROM EMP **E**, DEPT **D** WHERE E.DEPTNO = D.DEPTNO;

- TABLE ALIAS *(a)* 편의성 (b) 가독성(의미있는 이름사용) EX) EMP E , EMP A (c) Self Join시 필수 사용 (d) 동일 컬럼명이 존재하는 경우

⑤ SELECT D.DNAME,E.ENAME,E.JOB,E.SAL FROM EMP E **INNER JOIN** DEPT D
ON E.DEPTNO = D.DEPTNO // ANSI-SQL , ON 조인조건

⑥ SELECT D.DNAME,E.ENAME,E.JOB,E.SAL FROM EMP E INNER JOIN DEPT D
ON E.DEPTNO = D.DEPTNO // 조인 조건
WHERE E.DEPTNO IN (10,20) AND D.DNAME = 'RESEARCH' // 필터링 조건

1. JOIN

❏ Non Equi-Join

* EQUAL(=) 이외의 연산자를 사용하여 JOIN

① SELECT E.ENAME, E.JOB,E.SAL,S.GRADE FROM EMP E, SALGRADE S
WHERE E.SAL **BETWEEN** S.LOSAL **AND** S.HISAL; // 범위 연산자

② SELECT DNAME,ENAME,JOB,SAL,GRADE
FROM **EMP E, DEPT D, SALGRADE S**
WHERE E.DEPTNO = D.DEPTNO **AND** E.SAL BETWEEN S.LOSAL AND S.HISAL;

- 3개 테이블 JOIN , 최소 JOIN조건: N(테이블개수) – 1

③ SELECT E.ENAME, E.JOB,E.SAL,E.GRADE FROM EMP E, SALGRADE S
WHERE E.SAL BETWEEN S.LOSAL AND S.HISAL AND E.DEPTNO IN (10,30);
ORDER BY E.ENAME;

- 조건절 처리 순서?? (a) JOIN 처리 (**E.SAL BETWEEN S.LOSAL AND S.HISAL**) → 필터링 (**E.DEPTNO IN (10,30)**)
(b) 필터링 → JOIN 처리

④ SELECT E.ENAME, E.JOB,E.SAL,S.GRADE FROM EMP E, SALGRADE S
WHERE E.SAL < S.LOSAL AND E.DEPTNO IN (10,30)
ORDER BY E.ENAME;

- **원하는 결과가 아닌 무의미(불필요)한 다량의 곱집합 결과 , 이유는?**

1. JOIN

❏ Outer-Join

* JOIN 조건에 직접 만족되지 않는 정보도 조회 (!= inner Join)

① SELECT D.DNAME,E.ENAME,E.JOB,E.SAL FROM EMP E,DEPT D WHERE E.DEPTNO = D.DEPTNO
ORDER BY D.DNAME;

- 40번 부서에 근무하는 직원이 없기 때문에 Equi Join에서는 40번 부서관련 정보 조회가 안된다.

② SELECT D.DNAME,E.ENAME,E.JOB,E.SAL FROM EMP E,DEPT D WHERE **E.DEPTNO(+)** = **D.DEPTNO**
ORDER BY D.DNAME;

- 기준되는 테이블(DEPT)과 조인되는 반대편 테이블(EMP)의 조인 조건에 (+) 표시 , 직접 매핑되지 않는 컬럼에는 NULL

④ SELECT D.DNAME,E.ENAME,E.JOB,E.SAL FROM EMP E,DEPT D WHERE E.DEPTNO = **D.DEPTNO(+)**
ORDER BY D.DNAME;

- Equi-Join과 동일한 결과 출력, 불필요한 아웃터 조인은 비효율적인 자원 사용 가능

⑤ SELECT D.DNAME,**NVL(E.ENAME,'비상근 부서')**,E.JOB,E.SAL FROM EMP E,DEPT D
WHERE E.DEPTNO(+) = D.DEPTNO
ORDER BY D.DNAME;

- 직접 매핑되지 않는 컬럼에는 NULL

⑥ SELECT D.DNAME,E.ENAME,E.JOB,E.SAL FROM EMP E,DEPT D WHERE E.DEPTNO**(+)** = D.DEPTNO**(+)**
ORDER BY D.DNAME;

- ORACLE SQL은 양방향 OUTER JOIN을 허용하지 않는다.
- ANSI-SQL 1999 에서는 양방향 OUTER JOIN을 허용

1. JOIN

❏ Ansi Outer-Join

① SELECT E.DEPTNO,D.DNAME,E.ENAME FROM SCOTT.EMP E **LEFT OUTER JOIN** SCOTT.DEPT D
ON E.DEPTNO = D.DEPTNO
ORDER BY E.DEPTNO;

② SELECT E.DEPTNO,D.DNAME,E.ENAME FROM SCOTT.EMP E **RIGHT OUTER JOIN** SCOTT.DEPT D
ON E.DEPTNO = D.DEPTNO
ORDER BY E.DEPTNO;

- DEPT을 기준 테이블(Driving Table)하여 Join 연산 수행 , 40번 부서의 정보 표기 !!!!

③ SELECT D.DEPTNO,D.DNAME,E.ENAME FROM SCOTT.EMP E **FULL OUTER JOIN** SCOTT.DEPT D
ON E.DEPTNO = D.DEPTNO
ORDER BY E.DEPTNO;

- 양방향 아웃터 조인(Full Outer Join) 실행

1. JOIN

❏ Self-Join

① SELECT E.ENAME||' ''S MANAGER IS '||M.ENAME
FROM **EMP E, EMP M**
WHERE E.MGR = M.EMPNO
ORDER BY M.ENAME;

- 같은 테이블 끼리 조인
- 테이블 개체내 참조 무결성 관계시 ex) empno vs mgr
- 테이블 Alias 필수 사용

❏ 과제

1) 위의 ① SQL에서는 회사 대표(job='president') 정보가 누락 되었다.
- SQL을 수정하여 누락된 정보가 조회 되도록 작성
- 매니저가 없는 경우 매니저의 이름은 NOBODY로 표기

2) 위의 ① SQL을 ANSI-SQL로 변환.

1. JOIN

❑카티션곱 (CARTESIAN PRODUCT)

* Join시 두 테이블(집합)간에 곱집합 연산으로 유용하지 않은 대량의 데이터를 생성하는 현상
  - 데카르트의 곱집합
  - 발생원인 (a) Join 조건 생략시 (b) 잘못된 JOIN 조건
  - 용도 (a) 테스트용 샘플데이타 생성 (b) 곱집합 기능을 이용한 빠른 연산 응용

① SELECT ENAME,JOB,DNAME FROM EMP, DEPT;
- Join 조건 생략시 발생 , 데이터 건수 체크

② SELECT ENAME,JOB,DNAME FROM EMP , DEPT
WHERE EMP.SAL > 2000 **and** DEPT.DEPTNO IN (10,20);
- 필터링 조건(O) , Join 조건(X) , and 와 데이터 건수 체크

③ SELECT ENAME,JOB,DNAME FROM EMP , DEPT
WHERE EMP.SAL > 2000 **or** DEPT.DEPTNO IN (10,20);
- 필터링 조건(O) , Join 조건(X) , or 와 데이터 건수 체크

④ SELECT E.ENAME, E.JOB,E.SAL,S.GRADE FROM EMP E, SALGRADE S
WHERE E.SAL < S.LOSAL AND E.DEPTNO IN (10,30)
ORDER BY E.ENAME;

- 잘못된 Join 조건

1. JOIN

❏ 과제

1) 아래의 2개 SQL을 실행하여 데이터를 비교한후 원하는 결과 집합이 나오도록 2번째 SQL 수정

```
SELECT  D.DEPTNO,D.DNAME,E.ENAME,E.JOB,E.SAL FROM EMP E,DEPT D
WHERE  E.DEPTNO = D.DEPTNO AND  E.SAL > 2000
ORDER BY D.DNAME;

SELECT  D.DEPTNO,D.DNAME,E.ENAME,E.JOB,E.SAL FROM EMP E,DEPT D
WHERE  E.DEPTNO(+) = D.DEPTNO AND E.SAL > 2000
ORDER BY D.DNAME;
```

2) JOIN을 사용하여 부서별 급여 지급 순위 계산

부서번호	이름	직업	급여	급여순위
10	CLARK	MANAGER	5000	1
10	KING	PRESIDENT	3500	2
10	MILLER	CLERK	2750	3
20	SCOTT	ANALYST	3000	1
20	FORD	ANALYST	3000	1
20	JONES	MANAGER	2975	2

3) RANK, DENSE_RANK 함수를 사용하여 2)와 동일한 결과를 나타내는 SQL 작성

❏ 과제

4) MAKE_ENV.SQL을 사용하여 실습 환경을 구성한후 아래의 결과가 출력되는 Join 구문 작성

MAKE_ENV.SQL

```
CREATE  TABLE SYSTEM(   SYSTEM_ID          VARCHAR2(5),
                        SYSTEM_NAME     VARCHAR2(10)
);
INSERT INTO SYSTEM  VALUES('XXX','혜화DB');
INSERT INTO SYSTEM  VALUES('YYY','강남DB');
INSERT INTO SYSTEM  VALUES('ZZZ','영등포DB');

CREATE TABLE RESOURCE_USAGE(SYSTEM_ID                   VARCHAR2(5),
                            RESOURCE_NAME               VARCHAR2(10)
);
INSERT INTO RESOURCE_USAGE  VALUES('XXX','FTP');
INSERT INTO RESOURCE_USAGE  VALUES('YYY','FTP');
INSERT INTO RESOURCE_USAGE  VALUES('YYY','TELNET');
INSERT INTO RESOURCE_USAGE  VALUES('YYY','EMAIL');
COMMIT;
```

SYSTE	SYSTEM_NAME	FTP	TELNET	EMAIL
XXX	혜화DB	사용	미사용	미사용
YYY	강남DB	사용	미사용	사용
ZZZ	영등포DB	미사용	미사용	미사용

```
* SELECT   S.SYSTEM_ID,S.SYSTEM_NAME,R.RESOURCE_NAME
  FROM    SYSTEM S, RESOURCE_USAGE R
  WHERE   S.SYSTEM_ID = R.SYSTEM_ID;
* SELECT S.SYSTEM_ID,S.SYSTEM_NAME,R.RESOURCE_NAME
  FROM    SYSTEM S,RESOURCE_USAGE R
  WHERE   S.SYSTEM_ID = R.SYSTEM_ID(+);
```

1. JOIN

❑ 과제

5) 부서번호, 이름, 급여, 급여비율(소수점이하 2자리)을 출력하는 SQL을 카티션곱을 응용하여 작성

	DEPTNO	ENAME	SAL	SAL_RATE
1	20	SMITH	800	2.76%
2	30	ALLEN	1600	5.51%
3	30	WARD	1250	4.31%
4	20	JONES	2975	10.25%
5	30	MARTIN	1250	4.31%
6	30	BLAKE	2850	9.82%
7	10	CLARK	2450	8.44%
8	20	SCOTT	3000	10.34%
9	10	KING	5000	17.23%
10	30	TURNER	1500	5.17%
11	20	ADAMS	1100	3.79%
12	30	JAMES	950	3.27%
13	20	FORD	3000	10.34%
14	10	MILLER	1300	4.48%