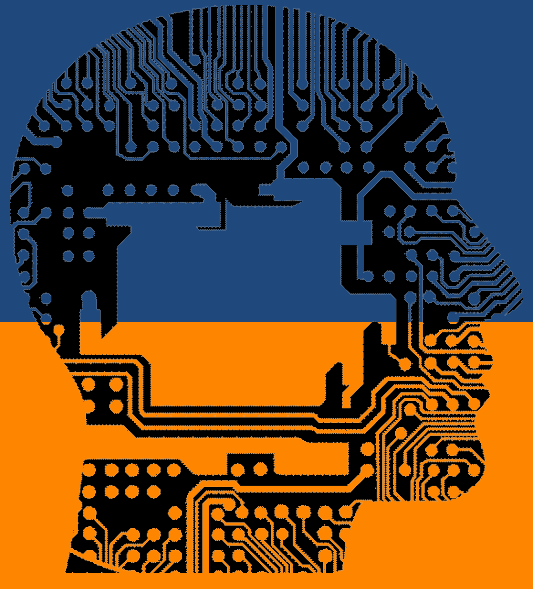
한국IT진흥부설 정보보호교육학원 아이섹

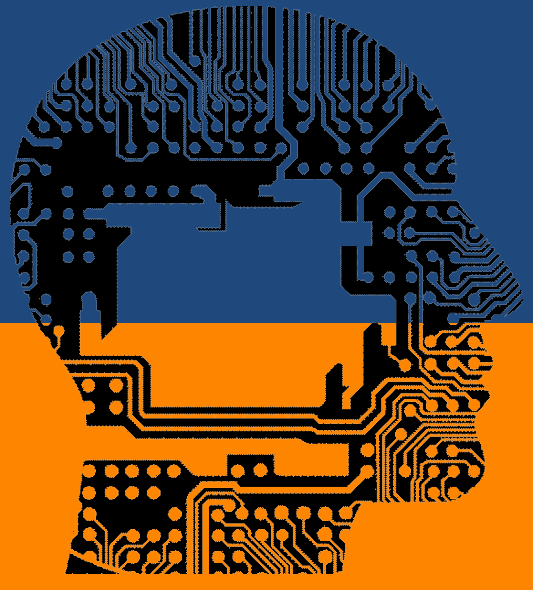


JAVA 웹 개발자 양성과정
DataBase - SQLD

2강 - 조인(JOIN)

By SoonGu Hong

한국IT진흥부설
정보보호교육학원 아이섹



JAVA 웹 개발자 양성과정

DataBase

1. 표준 조인

➢ 일반 집합 연산자와 SQL의 비교

일반 집합 연산자	SQL문	설명
UNION 연산	UNION 기능으로 구현	• UNION 연산은 수학적 합집합을 제공하기 위해, 공통 교집합의 중복을 없애기 위한 사전 작업으로 시스템에 부하를 주는 정렬 작업이 발생한다. • 이후 UNION ALL 기능이 추가되었는데, 특별한 요구 사항이 없다면 공통 집합을 중복해서 그대로 보여 주기 때문에 정렬 작업이 일어나지 않는 장점을 가진다. • 만일 UNION과 UNION ALL의 출력 결과가 같다면 응답 속도 향상이나 자원 효율화 측면에서 데이터 정렬 작업이 발생하지 않는 UNION ALL을 사용하는 것을 권고한다.
INTERSECTION 연산	INTERSECT 기능으로 구현	• INTERSECTION은 수학의 교집합으로써 두 집합의 공통 집합을 추출한다.
DIFFERENCE 연산	EXCEPT(Oracle은 MINUS) 기능으로 구현	• DIFFERENCE는 수학의 차집합으로써 첫 번째 집합에서 두 번째 집합과의 공통 집합을 제외한 부분이다. • 대다수 벤더는 EXCEPT를 Oracle은 MINUS 용어를 사용한다.
PRODUCT 연산	CROSS JOIN 기능으로 구현	• PRODUCT의 경우는 CROSS(ANIS/ISO 표준) PRODUCT라고 불리는 곱집합으로 JOIN 조건이 없는 경우 생길 수 있는 모든 데이터의 조합을 말한다. • 양쪽 집합의 M*N 건의 데이터 조합이 발생하며, CARTESIAN(수학자 이름) PRODUCT라고도 표현한다.

➢ 순수 관계 연산자와 SQL의 비교

일반 집합 연산자	SQL문	설명
SELECT 연산	WHERE 절로 구현	• SELECT 연산은 SQL 문장에서는 WHERE 절 기능으로 구현이 되었다.
PROJECT 연산	SELECT 절로 구현	• PROJECT 연산은 SQL 문장에서는 SELECT 절의 칼럼 선택 기능으로 구현되었다.
(NATURAL) JOIN 연산	다양한 JOIN 기능으로 구현	• JOIN 연산은 WHERE 절의 INNER JOIN 조건과 함께 FROM 절의 NATURAL JOIN, INNER JOIN, OUTER JOIN, USING 조건절, ON 조건절 등으로 가장 다양하게 발전하였다.
DIVIDE 연산	현재 사용되지 않음	• DIVIDE 연산은 나눗셈과 비슷한 개념으로 왼쪽의 집합을 'XZ'로 나누었을 때, 즉 'XZ'를 모두 가지고 있는 'A'가 답이 되는 기능으로 현재 사용되지 않는다.

➢ 조인의 형태

일반 집합 연산자	설명
INNER JOIN	• INNER JOIN은 OUTER(외부) JOIN과 대비하여 내부 JOIN이라고 하며 JOIN 조건에서 동일한 값이 있는 행만 반환
NATURAL JOIN	• NATURAL JOIN은 두 테이블 간의 동일한 이름을 갖는 모든 칼럼들에 대해 EQUI(=) JOIN을 수행
USING 조건절	• NATURAL JOIN에서는 모든 일치되는 칼럼들에 대해 JOIN이 이루어지지만, FROM 절의 USING 조건절을 이용하면 같은 이름을 가진 칼럼들 중에서 원하는 칼럼에 대해서만 선택적으로 EQUI JOIN을 할 수가 있음
ON 조건절	• JOIN 서술부(ON 조건절)와 비 JOIN 서술부(WHERE 조건 절)를 분리하여 이해가 쉬우며, 칼럼 명이 다르더라도 JOIN 조건을 사용할 수 있는 장점이 있음
CROSS JOIN	• CROSS JOIN은 E.F.CODD 박사가 언급한 일반 집합 연산자의 PRODUCT의 개념으로 테이블 간 JOIN 조건이 없는 경우 생길 수 있는 모든 데이터의 조합을 말함
OUTER JOIN	• INNER(내부) JOIN과 대비하여 OUTER(외부) JOIN이라고 불리며 JOIN 조건에서 동일한 값이 없는 행도 반환할 때 사용할 수 있음

➢ INNER JOIN 실습

```
SELECT A.EMP_NO
     , A.EMP_NM
     , A.ADDR
     , B.DEPT_CD
     , B.DEPT_NM
  FROM TB_EMP A, TB_DEPT B
  WHERE A.DEPT_CD = B.DEPT_CD
    AND A.ADDR LIKE '%수원%'
ORDER BY A.EMP_NO
;
```

❖ 주소가 수원인 직원의 사원번호, 사원명, 주소, 부서코드, 부서명을 출력

EMP_NO	EMP_NM	ADDR	DEPT_CD	DEPT_NM
1000000002	이현승	경기도 수원시 팔달구 매탄동 228	100002	지원팀
1000000006	김혜진	경기도 수원시 팔달구 권선동 887	100003	기획팀
1000000010	박혜령	경기도 수원시 팔달구 매탄동 987	100004	디자인팀
1000000015	이정직	경기도 수원시 팔달구 인계동 124	100006	데이터팀
1000000019	박정혜	경기도 수원시 팔달구 매탄동 987	100007	개발팀
1000000024	박선영	경기도 수원시 팔달구 매탄2동 445	100009	운영팀
1000000033	홍사기	경기도 수원시 팔달구 인계동 778	100012	인공지능팀
1000000037	이현정	경기도 수원시 팔달구 매탄동 225	100013	빅데이터팀

➢ NATURAL JOIN 실습

```
SELECT A.EMP_NO
     , A.EMP_NM
     , A.ADDR
     , DEPT_CD
     , B.DEPT_NM
FROM TB_EMP A NATURAL JOIN TB_DEPT B
  WHERE A.ADDR LIKE '%수원%'
  ;
```

❖ NATURAL조인은 두 테이블이 공통적으로 가지고 있는 DEPT_CD 칼럼으로 자동으로 조인된다.

❖ 주소가 수원인 직원의 사원번호, 사원명, 주소, 부서코드, 부서명을 출력

EMP_NO	EMP_NM	ADDR	DEPT_CD	DEPT_NM
1000000002	이현승	경기도 수원시 팔달구 매탄동 228	100002	지원팀
1000000006	김혜진	경기도 수원시 팔달구 권선동 887	100003	기획팀
1000000010	박혜령	경기도 수원시 팔달구 매탄동 987	100004	디자인팀
1000000015	이정직	경기도 수원시 팔달구 인계동 124	100006	데이터팀
1000000019	박정혜	경기도 수원시 팔달구 매탄동 987	100007	개발팀
1000000024	박선영	경기도 수원시 팔달구 매탄2동 445	100009	운영팀
1000000033	홍사기	경기도 수원시 팔달구 인계동 778	100012	인공지능팀
1000000037	이현정	경기도 수원시 팔달구 매탄동 225	100013	빅데이터팀

➢ NATURAL JOIN 실습 - 조인 칼럼은 앨리어스를 가질 수 없음

```
SELECT A.EMP_NO
     , A.EMP_NM
     , A.ADDR
     , B.DEPT_NM
     , B.DEPT_CD   DEPT_CD
 FROM TB_EMP A NATURAL JOIN TB_DEPT B
WHERE A.ADDR LIKE '%수원%'
 ;
```

- ❖ NATURAL조인은 두 테이블이 공통적으로 가지고 있는 DEPT_CD 칼럼으로 자동으로 조인된다.
- ❖ 하지만 DEPT_CD컬럼을 SELECT절에 기재할 때 앨리어스를 주어서 에러가 났다.
- ❖ NATURAL 조인에서 조인 칼럼은 앨리어스를 주며 안된다.

```
SQL Error [25155] [99999]: ORA-25155: NATURAL 조인에 사용된 열은 식별자를 가질 수 없음
```

➢ USING절 실습

```
SELECT A.EMP_NO
     , A.EMP_NM
     , A.ADDR
     , B.DEPT_NM
     , DEPT_CD
  FROM TB_EMP A JOIN TB_DEPT B USING (DEPT_CD)
 WHERE A.ADDR LIKE '%수원%'
 ;
```

❖ 주소가 수원인 직원의 사원번호, 사원명, 주소, 부서코드, 부서명을 출력

❖ USING절에 두 테이블이 공통적으로 가지고 있는 DEPT_CD 칼럼을 기재한다.
❖ USING절에 들어가는 조인 칼럼에 앨리어스를 쓸 수 없다.

EMP_NO	EMP_NM	ADDR	DEPT_NM	DEPT_CD
1000000002	이현승	경기도 수원시 팔달구 매탄동 228	지원팀	100002
1000000006	김혜진	경기도 수원시 팔달구 권선동 887	기획팀	100003
1000000010	박혜령	경기도 수원시 팔달구 매탄동 987	디자인팀	100004
1000000015	이정직	경기도 수원시 팔달구 인계동 124	데이터팀	100006
1000000019	박정혜	경기도 수원시 팔달구 매탄동 987	개발팀	100007
1000000024	박선영	경기도 수원시 팔달구 매탄2동 445	운영팀	100009
1000000033	홍사기	경기도 수원시 팔달구 인계동 778	인공지능팀	100012
1000000037	이현정	경기도 수원시 팔달구 매탄동 225	빅데이터팀	100013

➢ USING절 실습 - 조인 칼럼에 앨리어스를 쓸수 없음

```
SELECT A.EMP_NO
     , A.EMP_NM
     , A.ADDR
     , B.DEPT_NM
     , DEPT_CD
  FROM TB_EMP A JOIN TB_DEPT B USING (B.DEPT_CD)
 WHERE A.ADDR LIKE '%수원%'
 ;
```

❖ USING절에 두 테이블이 공통적으로 가지고 있는 DEPT_CD 칼럼을 기재한다.
❖ USING절에 들어가는 조인 칼럼에 앨리어스를 쓸 수 없다.

```
SQL Error [1748] [42000]: ORA-01748: 열명 그 자체만 사용할 수 있습니다
```

➢ ON절 실습

```
SELECT A.EMP_NO
     , A.EMP_NM
     , A.ADDR
     , B.DEPT_CD
     , B.DEPT_NM
  FROM TB_EMP A JOIN TB_DEPT B ON (A.DEPT_CD = B.DEPT_CD)
 WHERE A.ADDR LIKE '%수원%'
;
```

❖ 주소가 수원인 직원의 사원번호, 사원명, 주소, 부서코드, 부서명을 출력

❖ ON절에 조인 칼럼인 DEPT_CD칼럼을 기재한다.
❖ ON절 내에 조인 칼럼에 앨리어스를 사용해야 한다.
❖ 앨리어스를 정확히 기재하지 않으면 에러가 발생한다.

EMP_NO	EMP_NM	ADDR	DEPT_CD	DEPT_NM
1000000002	이현승	경기도 수원시 팔달구 매탄동 228	100002	지원팀
1000000006	김혜진	경기도 수원시 팔달구 권선동 887	100003	기획팀
1000000010	박혜령	경기도 수원시 팔달구 매탄동 987	100004	디자인팀
1000000015	이정직	경기도 수원시 팔달구 인계동 124	100006	데이터팀
1000000019	박정혜	경기도 수원시 팔달구 매탄동 987	100007	개발팀
1000000024	박선영	경기도 수원시 팔달구 매탄2동 445	100009	운영팀
1000000033	홍사기	경기도 수원시 팔달구 인계동 778	100012	인공지능팀
1000000037	이현정	경기도 수원시 팔달구 매탄동 225	100013	빅데이터팀

➢ ON절 실습 - 앨리어스를 정확히 기재 하지 않은 경우

```
SELECT A.EMP_NO
       , A.EMP_NM
       , A.ADDR
       , DEPT_CD
       , B.DEPT_NM
    FROM TB_EMP A JOIN TB_DEPT B ON (A.DEPT_CD = B.DEPT_CD)
   WHERE A.ADDR LIKE '%수원%'
 ;
```

❖ ON절에 조인 칼럼인 DEPT_CD칼럼을 기재한다.
❖ ON절 내에 조인 칼럼에 앨리어스를 사용해야 한다.
❖ 앨리어스를 정확히 기재하지 않으면 에러가 발생한다.

```
SQL Error [918] [42000]: ORA-00918: 열의 정의가 애매합니다
```

➢ 3개의 테이블 조인

```
SELECT
       A.EMP_NO
     , A.EMP_NM
     , A.ADDR
     , B.DEPT_CD
     , B.DEPT_NM
     , C.CERTI_CD
  FROM TB_EMP A
     , TB_DEPT B
     , TB_EMP_CERTI C
 WHERE A.DEPT_CD = B.DEPT_CD
   AND A.ADDR LIKE '%수원%'
   AND A.EMP_NO = C.EMP_NO
ORDER BY A.EMP_NO;
```

❖ 3개의 테이블을 조인하는데 조인 조건은 2개가 들어갔다.

❖ 주소가 수원인 직원의 사원번호, 사원명, 주소, 부서코드, 부서명, 자격증코드를 출력

EMP_NO	EMP_NM	ADDR	DEPT_CD	DEPT_NM	CERTI_CD
1000000002	이현승	경기도 수원시 팔달구 매탄동 228	100002	지원팀	100010
1000000002	이현승	경기도 수원시 팔달구 매탄동 228	100002	지원팀	100009
1000000002	이현승	경기도 수원시 팔달구 매탄동 228	100002	지원팀	100005
1000000006	김혜진	경기도 수원시 팔달구 권선동 887	100003	기획팀	100009
1000000006	김혜진	경기도 수원시 팔달구 권선동 887	100003	기획팀	100012
1000000006	김혜진	경기도 수원시 팔달구 권선동 887	100003	기획팀	100005
1000000010	박혜령	경기도 수원시 팔달구 매탄동 987	100004	디자인팀	100015
1000000010	박혜령	경기도 수원시 팔달구 매탄동 987	100004	디자인팀	100004
1000000010	박혜령	경기도 수원시 팔달구 매탄동 987	100004	디자인팀	100015
1000000015	이정직	경기도 수원시 팔달구 인계동 124	100006	데이터팀	100004
1000000015	이정직	경기도 수원시 팔달구 인계동 124	100006	데이터팀	100018
1000000015	이정직	경기도 수원시 팔달구 인계동 124	100006	데이터팀	100011
1000000019	박정혜	경기도 수원시 팔달구 매탄동 987	100007	개발팀	100005
1000000019	박정혜	경기도 수원시 팔달구 매탄동 987	100007	개발팀	100008
1000000019	박정혜	경기도 수원시 팔달구 매탄동 987	100007	개발팀	100005
1000000024	박선영	경기도 수원시 팔달구 매탄2동 445	100009	운영팀	100014
1000000024	박선영	경기도 수원시 팔달구 매탄2동 445	100009	운영팀	100002
1000000024	박선영	경기도 수원시 팔달구 매탄2동 445	100009	운영팀	100013
1000000033	홍사기	경기도 수원시 팔달구 인계동 778	100012	인공지능팀	100006
1000000033	홍사기	경기도 수원시 팔달구 인계동 778	100012	인공지능팀	100009
1000000033	홍사기	경기도 수원시 팔달구 인계동 778	100012	인공지능팀	100010
1000000037	이현정	경기도 수원시 팔달구 매탄동 225	100013	빅데이터팀	100004
1000000037	이현정	경기도 수원시 팔달구 매탄동 225	100013	빅데이터팀	100017
1000000037	이현정	경기도 수원시 팔달구 매탄동 225	100013	빅데이터팀	100016

➢ 3개의 테이블 조인 - ANSI방식의 조인

```
SELECT A.EMP_NO
     , A.EMP_NM
     , A.ADDR
     , B.DEPT_NM
     , B.DEPT_CD
     , C.CERTI_CD
  FROM TB_EMP A JOIN TB_DEPT B
    ON (A.DEPT_CD = B.DEPT_CD)
  JOIN TB_EMP_CERTI C
    ON (A.EMP_NO = C.EMP_NO)
  WHERE A.ADDR LIKE '%수원%';
```

❖ 3개의 테이블을 조인하는데 조인 조건은 2개가 들어갔다.

❖ 주소가 수원인 직원의 사원번호, 사원명, 주소, 부서코드, 부서명, 자격증코드를 출력

EMP_NO	EMP_NM	ADDR	DEPT_CD	DEPT_NM	CERTI_CD
1000000002	이현승	경기도 수원시 팔달구 매탄동 228	지원팀	100002	100010
1000000002	이현승	경기도 수원시 팔달구 매탄동 228	지원팀	100002	100009
1000000002	이현승	경기도 수원시 팔달구 매탄동 228	지원팀	100002	100005
1000000006	김혜진	경기도 수원시 팔달구 권선동 887	기획팀	100003	100009
1000000006	김혜진	경기도 수원시 팔달구 권선동 887	기획팀	100003	100012
1000000006	김혜진	경기도 수원시 팔달구 권선동 887	기획팀	100003	100005
1000000010	박혜령	경기도 수원시 팔달구 매탄동 987	디자인팀	100004	100015
1000000010	박혜령	경기도 수원시 팔달구 매탄동 987	디자인팀	100004	100004
1000000010	박혜령	경기도 수원시 팔달구 매탄동 987	디자인팀	100004	100015
1000000015	이정직	경기도 수원시 팔달구 인계동 124	데이터팀	100006	100004
1000000015	이정직	경기도 수원시 팔달구 인계동 124	데이터팀	100006	100018
1000000015	이정직	경기도 수원시 팔달구 인계동 124	데이터팀	100006	100011
1000000019	박정혜	경기도 수원시 팔달구 매탄동 987	개발팀	100007	100005
1000000019	박정혜	경기도 수원시 팔달구 매탄동 987	개발팀	100007	100008
1000000019	박정혜	경기도 수원시 팔달구 매탄동 987	개발팀	100007	100005
1000000024	박선영	경기도 수원시 팔달구 매탄2동 445	운영팀	100009	100014
1000000024	박선영	경기도 수원시 팔달구 매탄2동 445	운영팀	100009	100002
1000000024	박선영	경기도 수원시 팔달구 매탄2동 445	운영팀	100009	100013
1000000033	홍사기	경기도 수원시 팔달구 인계동 778	인공지능팀	100012	100006
1000000033	홍사기	경기도 수원시 팔달구 인계동 778	인공지능팀	100012	100009
1000000033	홍사기	경기도 수원시 팔달구 인계동 778	인공지능팀	100012	100010
1000000037	이현정	경기도 수원시 팔달구 매탄동 225	빅데이터팀	100013	100004
1000000037	이현정	경기도 수원시 팔달구 매탄동 225	빅데이터팀	100013	100017
1000000037	이현정	경기도 수원시 팔달구 매탄동 225	빅데이터팀	100013	100016

➢ 아우터 조인 - 실습 환경 구축

```
INSERT INTO TB_DEPT VALUES ('100014', '4차산업혁명팀', '999999');
INSERT INTO TB_DEPT VALUES ('100015', '포스트코로나팀', '999999');

COMMIT;
```

❖ 부서 테이블에 부서 데이터를 추가함
❖ 새로 추가한 팀에는 어떠한 사원도 속하지 않은 상태임

```
ALTER TABLE SQLD.TB_EMP DROP CONSTRAINT FK_TB_EMP01;
```

❖ 참조 무결성 제약 조건(FK) 잠시 DROP

❖ 실습을 위해 사원 테이블에 신규 사원 5명을 추가함
❖ 추가되는 5명의 부서 코드는 존재하지 않는 부서인 "000000"로 인서트 시킴

```
INSERT INTO SQLD.TB_EMP T (T.EMP_NO, T.EMP_NM, T.BIRTH_DE, T.SEX_CD, T.ADDR, T.TEL_NO, T.DIRECT_MANAGER_EMP_NO,
T.FINAL_EDU_SE_CD, T.SAL_TRANS_BANK_CD, T.SAL_TRANS_ACCNT_NO, T.DEPT_CD, T.LUNAR_YN )
    VALUES ('1000000041', '이순신', '19811201', '1', '경기도 용인시 수지구 죽전1동 435', '010-5456-7878', NULL, '006', '003', '114-
554-223433', '000000', 'N');
INSERT INTO SQLD.TB_EMP T (T.EMP_NO, T.EMP_NM, T.BIRTH_DE, T.SEX_CD, T.ADDR, T.TEL_NO, T.DIRECT_MANAGER_EMP_NO,
T.FINAL_EDU_SE_CD, T.SAL_TRANS_BANK_CD, T.SAL_TRANS_ACCNT_NO, T.DEPT_CD, T.LUNAR_YN )
    VALUES ('1000000042', '정약용', '19820402', '1', '경기도 고양시 덕양구 화정동 231', '010-4054-6547', NULL, '004', '001', '110-
223-553453', '000000', 'Y');
INSERT INTO SQLD.TB_EMP T (T.EMP_NO, T.EMP_NM, T.BIRTH_DE, T.SEX_CD, T.ADDR, T.TEL_NO, T.DIRECT_MANAGER_EMP_NO,
T.FINAL_EDU_SE_CD, T.SAL_TRANS_BANK_CD, T.SAL_TRANS_ACCNT_NO, T.DEPT_CD, T.LUNAR_YN )
    VALUES ('1000000043', '박지원', '19850611', '1', '경기도 수원시 팔달구 매탄동 553', '010-1254-1116', NULL, '004', '001', '100-
233-664234', '000000', 'N');
INSERT INTO SQLD.TB_EMP T (T.EMP_NO, T.EMP_NM, T.BIRTH_DE, T.SEX_CD, T.ADDR, T.TEL_NO, T.DIRECT_MANAGER_EMP_NO,
T.FINAL_EDU_SE_CD, T.SAL_TRANS_BANK_CD, T.SAL_TRANS_ACCNT_NO, T.DEPT_CD, T.LUNAR_YN )
    VALUES ('1000000044', '장보고', '19870102', '1', '경기도 성남시 분당구 정자동 776', '010-1215-8784', NULL, '004', '002', '180-
345-556634', '000000', 'Y');
INSERT INTO SQLD.TB_EMP T (T.EMP_NO, T.EMP_NM, T.BIRTH_DE, T.SEX_CD, T.ADDR, T.TEL_NO, T.DIRECT_MANAGER_EMP_NO,
T.FINAL_EDU_SE_CD, T.SAL_TRANS_BANK_CD, T.SAL_TRANS_ACCNT_NO, T.DEPT_CD, T.LUNAR_YN )
    VALUES ('1000000045', '김종서', '19880824', '1', '경기도 고양시 일산서구 백석동 474', '010-3687-1245', NULL, '004', '002',
'325-344-45345', '000000', 'Y');
COMMIT;
```

➢ 아우터 조인 - LEFT OUTER 조인

```
SELECT A.EMP_NO
     , A.EMP_NM
     , B.DEPT_CD
     , B.DEPT_NM
  FROM TB_EMP A, TB_DEPT B
 WHERE A.DEPT_CD IN ( '000000', '100001')
   AND A.DEPT_CD = B.DEPT_CD(+)
;
```

❖ 현재 부서에 소속되어 있지 않은 직원들도 모두 집합에 포함됨
❖ 즉 TB_EMP(LEFT)는 다 나오고 TB_DEPT는 매칭되는 것만 나오게 됨

EMP_NO	EMP_NM	DEPT_CD	DEPT_NM
1000000001	이경오	100001	운영본부
1000000041	이순신	(NULL)	(NULL)
1000000042	정약용	(NULL)	(NULL)
1000000043	박지원	(NULL)	(NULL)
1000000044	장보고	(NULL)	(NULL)
1000000045	김종서	(NULL)	(NULL)

```
SELECT A.EMP_NO
     , A.EMP_NM
     , B.DEPT_CD
     , B.DEPT_NM
  FROM TB_EMP A
LEFT OUTER JOIN TB_DEPT B
 ON (A.DEPT_CD = B.DEPT_CD)
WHERE A.DEPT_CD IN ( '000000', '100001')
;
```

❖ ANSI조인 방식의 LEFT OUTER JOIN임

EMP_NO	EMP_NM	DEPT_CD	DEPT_NM
1000000001	이경오	100001	운영본부
1000000041	이순신	(NULL)	(NULL)
1000000042	정약용	(NULL)	(NULL)
1000000043	박지원	(NULL)	(NULL)
1000000044	장보고	(NULL)	(NULL)
1000000045	김종서	(NULL)	(NULL)

➢ 아우터 조인 - RIGTH OUTER 조인

```
SELECT A.EMP_NO
     , A.EMP_NM
     , B.DEPT_CD
     , B.DEPT_NM
  FROM TB_EMP A
  , TB_DEPT B
WHERE B.DEPT_CD IN ('100014', '100015', '100001')
  AND A.DEPT_CD(+) = B.DEPT_CD ;
```

❖ 현재 아무런 사원을 가지고 있지 않은 부서도 모두 출력됨
❖ 즉 TB_DEPT(RIGHT)는 모두 나오고 TB_EMP는 매칭되는 집합만 출력됨

EMP_NO	EMP_NM	DEPT_CD	DEPT_NM
1000000001	이경오	100001	운영본부
(NULL)	(NULL)	100015	포스트코로나팀
(NULL)	(NULL)	100014	4차산업혁명팀

```
SELECT A.EMP_NO
     , A.EMP_NM
     , B.DEPT_CD
     , B.DEPT_NM
  FROM TB_EMP A
  RIGHT OUTER JOIN TB_DEPT B
  ON (A.DEPT_CD = B.DEPT_CD)
WHERE B.DEPT_CD IN ( '100014', '100015', '100001');
```

❖ ANSI조인 방식의 RIGHT OUTER JOIN임

EMP_NO	EMP_NM	DEPT_CD	DEPT_NM
1000000001	이경오	100001	운영본부
(NULL)	(NULL)	100015	포스트코로나팀
(NULL)	(NULL)	100014	4차산업혁명팀

➢ 아우터 조인 - FULL OUTER 조인

```
SELECT
       A.EMP_NO
     , A.EMP_NM
     , B.DEPT_CD
     , B.DEPT_NM
 FROM TB_EMP A
 FULL OUTER JOIN TB_DEPT B ON (A.DEPT_CD = B.DEPT_CD)
WHERE 1 = 1
  AND (   A.EMP_NO IS NULL
       OR B.DEPT_CD IS NULL
      )
ORDER BY B.DEPT_CD DESC, A.EMP_NO DESC
;
```

❖ EMP_NO가 널이거나 DEPT_CD가 널인 것에 대한 조건을 주었다.
❖ 즉 EQUI 조인에 실패한 것들만을 추출하였음

EMP_NO	EMP_NM	DEPT_CD	DEPT_NM
1000000045	김종서	(NULL)	(NULL)
1000000044	장보고	(NULL)	(NULL)
1000000043	박지원	(NULL)	(NULL)
1000000042	정약용	(NULL)	(NULL)
1000000041	이순신	(NULL)	(NULL)
(NULL)	(NULL)	100015	포스트코로나팀
(NULL)	(NULL)	100014	4차산업혁명팀

➢ 아우터 조인 – 실습 종료 후 데이터 삭제 및 제약 조건 재설정

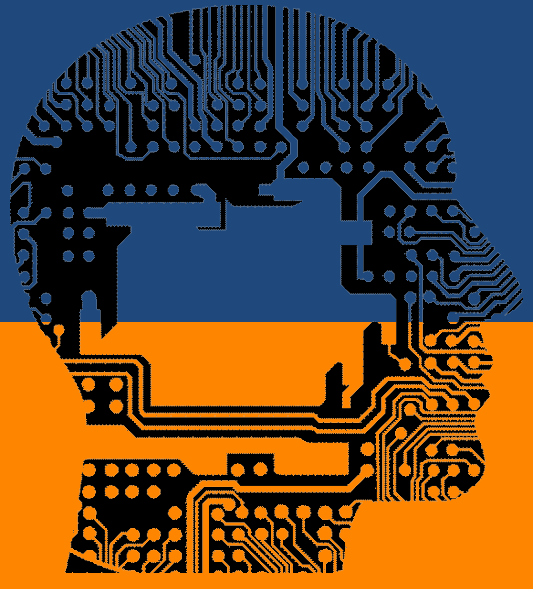
```
DELETE
  FROM TB_DEPT
 WHERE DEPT_CD IN ('100014', '100015');

DELETE FROM TB_EMP
WHERE EMP_NO IN ('1000000041','1000000042','1000000043','1000000044','1000000045');

COMMIT;

ALTER TABLE SQLD.TB_EMP ADD CONSTRAINT FK_TB_EMP01 FOREIGN KEY (DEPT_CD) REFERENCES SQLD.TB_DEPT (DEPT_CD);
```

한국IT진흥부설
정보보호교육학원 아이섹



JAVA 웹 개발자 양성과정
DataBase

2. 집합 연산자

➢ 집합연산자의 종류

종류	설명
UNION	• 여러 개의 SQL문의 결과에 대한 합집합 • 중복된 행은 한개의 행으로 출력됨
UNION ALL	• 여러 개의 SQL문의 결과에 대한 합집합 • 중복된 행도 그대로 결과로 표시한다.
INTERSECT	• 여러 개의 SQL문의 대한 교집합 중복된 행은 하나로 표시한다.
EXCEPT	• 위의 SQL문의 집합에서 아래의 SQL문의 집합을 뺀 결과를 표시한다.

➢ UNION 실습

```
SELECT A.EMP_NO, A.EMP_NM, A.BIRTH_DE
FROM TB_EMP A
WHERE A.BIRTH_DE BETWEEN '19600101' AND '19691231'
UNION
SELECT A.EMP_NO, A.EMP_NM, A.BIRTH_DE
FROM TB_EMP A
WHERE A.BIRTH_DE BETWEEN '19700101' AND '197901231'
;
```

❖ 생년월일이 1960년대 및 1970년대에 태어난 직원의 사원번호, 사원명, 생년월일을 출력함
❖ 중복되는 행에 대해서는 한 건 만 출력

EMP_NO	EMP_NM	BIRTH_DE
1000000014	이관심	19780213
1000000023	이관심	19780213
1000000025	박호진	19720128
1000000026	김길정	19690524
1000000032	이준표	19771202
1000000034	김익정	19720128
1000000035	최창수	19690524
9999999999	김회장	19651105

❖ 총 8건의 레코드 출력
❖ 결과 집합이 정렬되어 있음

➢ UNION ALL 실습

```
SELECT A.EMP_NO, A.EMP_NM, A.BIRTH_DE
FROM TB_EMP A
WHERE A.BIRTH_DE BETWEEN '19600101' AND '19691231'
UNION ALL
SELECT A.EMP_NO, A.EMP_NM, A.BIRTH_DE
FROM TB_EMP A
WHERE A.BIRTH_DE BETWEEN '19700101' AND '197901231'
;
```

❖ 생년월일이 1960년대 및 1970년대에 태어난 직원의 사원번호, 사원명, 생년월일을 출력함
❖ 중복되는 행을 모두 출력(즉 중복 된 것도 그대로 보여줌)

EMP_NO	EMP_NM	BIRTH_DE
9999999999	김회장	19651105
1000000026	김길정	19690524
1000000035	최창수	19690524
1000000014	이관심	19780213
1000000023	이관심	19780213
1000000025	박호진	19720128
1000000032	이준표	19771202
1000000034	김익정	19720128

❖ 총 8건의 레코드 출력
❖ 결과 집합이 정렬되어 있지 않음

➢ UNION & UNION ALL 중복 행 실습

```
SELECT A.EMP_NM, A.BIRTH_DE
FROM TB_EMP A
WHERE A.BIRTH_DE BETWEEN '19600101' AND '19691231'
UNION ALL
SELECT A.EMP_NM, A.BIRTH_DE
FROM TB_EMP A
WHERE A.BIRTH_DE BETWEEN '19700101' AND '197901231'
;
```

❖ 생년월일이 1960년대 및 1970년대에 태어난 직원의 사원명, 생년월일을 출력함
❖ 동명이인으로 "이관심"이라는 직원이 중복된 것을 알 수 있음("이관심"이라는 2명의 직원은 이름과 생년월일까지 모두 같음)

EMP_NM	BIRTH_DE
김회장	19651105
김길정	19690524
최창수	19690524
이관심	19780213
이관심	19780213
박호진	19720128
이준표	19771202
김익정	19720128

❖ 총 8건의 레코드 출력
❖ 결과 집합이 정렬되어 있지 않음

```
SELECT A.EMP_NM, A.BIRTH_DE
FROM TB_EMP A
WHERE A.BIRTH_DE BETWEEN '19600101' AND '19691231'
UNION
SELECT A.EMP_NM, A.BIRTH_DE
FROM TB_EMP A
WHERE A.BIRTH_DE BETWEEN '19700101' AND '197901231'
;
```

❖ 생년월일이 1960년대 및 1970년대에 태어난 직원의 사원명, 생년월일을 출력함
❖ 동명이인인 "이관심" 직원은 중복된 행이 제거되는 과정에서 한 건으로만 보여짐

EMP_NM	BIRTH_DE
김길정	19690524
김익정	19720128
김회장	19651105
박호진	19720128
이관심	19780213
이준표	19771202
최창수	19690524

❖ 총 7건의 레코드 출력
❖ 결과 집합이 정렬되어 있음

➢ INTERSECT 실습

```
SELECT A.EMP_NO, A.EMP_NM, A.ADDR, B.CERTI_CD, C.CERTI_NM
FROM TB_EMP A , TB_EMP_CERTI B, TB_CERTI C
WHERE A.EMP_NO = B.EMP_NO
AND B.CERTI_CD = C.CERTI_CD
AND C.CERTI_NM = 'SQLD'
INTERSECT
SELECT A.EMP_NO, A.EMP_NM, A.ADDR, B.CERTI_CD, C.CERTI_NM
FROM TB_EMP A , TB_EMP_CERTI B, TB_CERTI C
WHERE A.EMP_NO = B.EMP_NO
AND B.CERTI_CD = C.CERTI_CD
AND A.ADDR LIKE '%용인%';
```

❖ SQLD자격증을 보유하면서 용인시에 사는 직원을 추출함

EMP_NO	EMP_NM	ADDR	CERTI_CD	CERTI_NM
1000000013	이나라	경기도 용인시 수지구 풍덕천동 124	100001	SQLD

❖ INTERSECT 연산은 아래의 SQL문과 결과 집합이 동일함

```
SELECT A.EMP_NO, A.EMP_NM, A.ADDR, B.CERTI_CD, C.CERTI_NM
FROM TB_EMP A , TB_EMP_CERTI B, TB_CERTI C
WHERE A.EMP_NO = B.EMP_NO
AND B.CERTI_CD = C.CERTI_CD
AND C.CERTI_NM = 'SQLD'
AND EXISTS ( SELECT 1
               FROM TB_EMP K
               WHERE K.EMP_NO = A.EMP_NO
                 AND K.ADDR LIKE '%용인%')
;
```

```
SELECT A.EMP_NO, A.EMP_NM, A.ADDR, B.CERTI_CD, C.CERTI_NM
  FROM TB_EMP A , TB_EMP_CERTI B, TB_CERTI C
 WHERE A.EMP_NO = B.EMP_NO
   AND B.CERTI_CD = C.CERTI_CD
   AND C.CERTI_NM = 'SQLD'
   AND A.ADDR LIKE '%용인%';
```

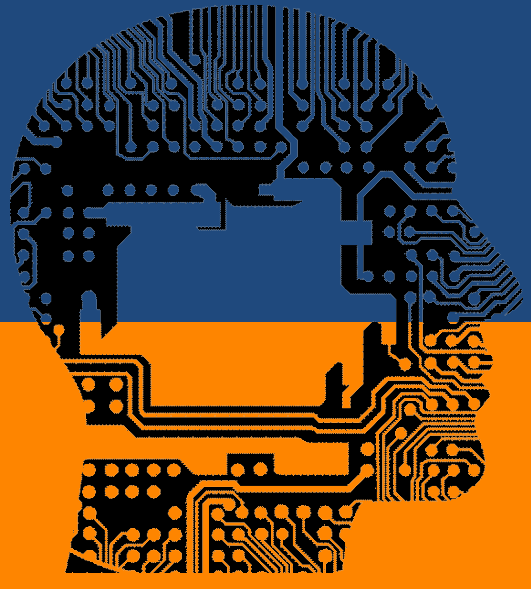
➢ MINUS 연산 실습

```
SELECT A.EMP_NO, A.EMP_NM, A.SEX_CD, A.DEPT_CD FROM TB_EMP A
MINUS
SELECT A.EMP_NO, A.EMP_NM, A.SEX_CD, A.DEPT_CD  FROM TB_EMP A
WHERE A.DEPT_CD = '100001'
MINUS
SELECT A.EMP_NO, A.EMP_NM, A.SEX_CD, A.DEPT_CD  FROM TB_EMP A
WHERE A.DEPT_CD = '100002'
MINUS
SELECT A.EMP_NO, A.EMP_NM, A.SEX_CD, A.DEPT_CD  FROM TB_EMP A
WHERE A.DEPT_CD = '100003'
MINUS
SELECT A.EMP_NO, A.EMP_NM, A.SEX_CD, A.DEPT_CD  FROM TB_EMP A
WHERE A.SEX_CD = '1'
```

❖ 전체 직원에서
❖ 부서 코드가 "100001" 인 직원들을 집합에서 제외
❖ 부서 코드가 "100002" 인 직원들을 집합에서 제외
❖ 부서 코드가 "100003" 인 직원들을 집합에서 제외
❖ 그 상태에서 성별이 남성인 직원들을 집합에서제외

EMP_NO	EMP_NM	SEX_CD	DEPT_CD
0000000010	박혜령	2	000004
0000000011	최수자	2	000004
0000000013	이나라	2	000004
0000000016	이진실	2	000006
0000000018	박바른	2	000006
0000000019	박정혜	2	000007
0000000020	최정진	2	000007
0000000022	김순수	2	000007
0000000025	박호진	2	000009
0000000027	박이수	2	000009
0000000028	김나라	2	000010
0000000029	장나라	2	000010
0000000031	김사랑	2	000010
0000000034	김익정	2	000012
0000000036	박여진	2	000012
0000000037	이현정	2	000013
0000000038	김혜수	2	000013
0000000040	김여진	2	000013

한국IT진흥부설
정보보호교육학원 아이섹

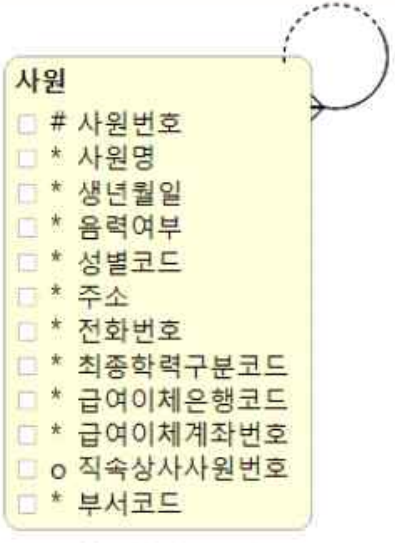


JAVA 웹 개발자 양성과정
DataBase

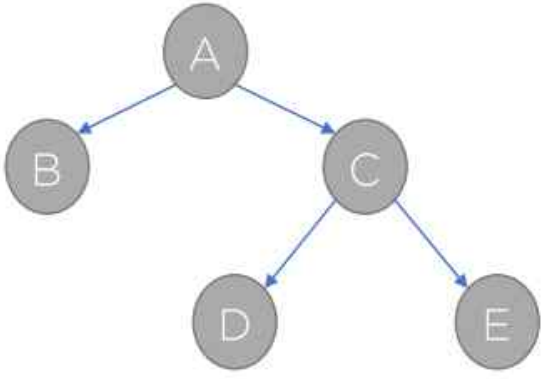
3. 계층형 질의와 SELF 조인

➢ 계층형 질의

① 테이블에 계층 형 데이터가 존재하는 경우 데이터를 조회하기 위해서 계층 형 질의(Hierarchical Query)를 사용
② 계층 형 데이터란 동일 테이블에 계층적으로 상위와 하위 데이터가 포함된 데이터를 말한다.



순환 관계 모델



계층 형 구조

사원	관리자
A	
B	A
C	A
D	C
E	C

샘플 데이터

➢ 오라클 계층 형 SQL

구분	설명
SELECT	• 조회하고자 하는 칼럼을 지정한다.
FROM TABLE	• 대상 테이블을 지정한다.
WHERE	• 모든 전개를 수행한 후에 지정된 조건을 만족하는 데이터만 추출한다.(필터링)
START WITH 조건	• 계층 구조 전개의 시작 위치를 지정하는 구문이다. 즉, 루트 데이터를 지정한다.
CONNECT BY [NOCYCLE] [PRIOR] A AND B	• CONNECT BY절은 다음에 전개될 자식 데이터를 지정하는 구문이다. • PRIOR 자식 = 부모 형태를 사용하면 계층구조에서 자식 데이터에서 부모 데이터(자식 → 부모) 방향으로 전개하는 순방향 전개를 한다. • PRIOR 부모 = 자식 형태를 사용하면 반대로 부모 데이터에서 자식 데이터(부모 → 자식) 방향으로 전개하는 역방향 전개를 한다. • NOCYCLE를 추가하면 사이클이 발생한 이후의 데이터는 전개하지 않는다.
ORDER SIBLINGS BY 칼럼	• 형제 노드(동일 LEVEL) 사이에서 정렬을 수행한다.

➢ 계층 형 질의에서 사용되는 가상 칼럼

구분	설명
LEVEL	• 루트데이터면 1 • 그 하위 데이터 면 2 • 하위데이터가 있을 때마다 1씩 증가
CONNECT_BY_ISLEAF	• 전개과정에서 해당 데이터가 리프 데이터 이면 1 그렇지 않으면 0
CONNECT_BY_ISCYCLE	• 전개과정에서 자식을 갖는데 해당 데이터가 조상으로서 존재하면 1 그렇지않으면 0

➢ 계층 형 SQL 실습

```
SELECT LEVEL LVL
     , LPAD(' ', 4*(LEVEL-1))|| EMP_NO || '(' || EMP_NM || ')' AS "조직인원"
     , A.DEPT_CD
     , B.DEPT_NM
     , CONNECT_BY_ISLEAF
FROM TB_EMP A, TB_DEPT B
WHERE A.DEPT_CD = B.DEPT_CD
START WITH A.DIRECT_MANAGER_EMP_NO IS NULL
CONNECT BY
PRIOR A.EMP_NO = A.DIRECT_MANAGER_EMP_NO
;
```

❖ 관리자사원번호가 널인 값을 첫 시작 값으로 하였음
❖ PRIOR 자식 = 부모 순으로 순방향 전개를 하였음
❖ CONNECT _BY_ISLEAFT를 이용해서 LEAF노드인 경우 1 을 출력함

LVL	조직인원	DEPT_CD	DEPT_NM	CONNECT_BY_ISLEAF
1	9999999999(김회장)	999999	회장실	0
2	1000000001(이경오)	100001	운영본부	0
3	1000000002(이현승)	100002	지원팀	0
4	1000000003(이정수)	100002	지원팀	1
4	1000000004(이승준)	100002	지원팀	1
4	1000000005(김현희)	100002	지원팀	1
중간생략…				
3	1000000037(이현정)	100013	빅데이터팀	0
4	1000000038(김혜수)	100013	빅데이터팀	1
4	1000000039(이박력)	100013	빅데이터팀	1
4	1000000040(김여진)	100013	빅데이터팀	1

➢ 계층 형 SQL 실습 – START WITH의 변경

```
SELECT LEVEL LVL
     , LPAD(' ', 4*(LEVEL-1))|| EMP_NO || '(' || EMP_NM || ')' AS "조직인원"
     , A.DEPT_CD
     , B.DEPT_NM
     , CONNECT_BY_ISLEAF
FROM TB_EMP A, TB_DEPT B
WHERE A.DEPT_CD = B.DEPT_CD
START WITH A.EMP_NM = '이경오'
CONNECT BY PRIOR A.EMP_NO = A.DIRECT_MANAGER_EMP_NO
;
```

- ❖ 사원명이 이경오인 행을 처음으로 시작함
- ❖ PRIOR 자식 = 부모 순으로 순방향 전개를 하였음
- ❖ CONNECT _BY_ISLEAFT를 이용해서 LEAF노드인 경우 1 을 출력함

LVL	조직인원	DEPT_CD	DEPT_NM	CONNECT_BY_ISLEAF
1	1000000001(이경오)	100001	운영본부	0
2	1000000002(이현승)	100002	지원팀	0
3	1000000003(이정수)	100002	지원팀	1
3	1000000004(이승준)	100002	지원팀	1
3	1000000005(김현희)	100002	지원팀	1
2	1000000006(김혜진)	100003	기획팀	0
3	1000000007(이순자)	100003	기획팀	1
3	1000000008(김려원)	100003	기획팀	1
3	1000000009(박태범)	100003	기획팀	1
2	1000000010(박혜령)	100004	디자인팀	0
3	1000000011(최수자)	100004	디자인팀	1
3	1000000012(김호형)	100004	디자인팀	1
3	1000000013(이나라)	100004	디자인팀	1

➢ 계층형 SQL 실습 – CONNECT_BY_ROOT의 사용

```
SELECT LEVEL LVL
     , LPAD(' ', 4*(LEVEL-1))|| EMP_NO || '(' || EMP_NM || ')' AS "조직인원"
     , A.DEPT_CD
     , B.DEPT_NM
     , CONNECT_BY_ISLEAF
     , CONNECT_BY_ROOT A.EMP_NO AS "최상위관리자"
FROM TB_EMP A, TB_DEPT B
WHERE A.DEPT_CD = B.DEPT_CD
START WITH A.DIRECT_MANAGER_EMP_NO IS NULL
CONNECT BY PRIOR A.EMP_NO = A.DIRECT_MANAGER_EMP_NO;
```

- ❖ 최고관리자를 START WITH로 시작함
- ❖ PRIOR 자식 = 부모 순으로 순방향 전개를 하였음
- ❖ CONNECT _BY_ISLEAFT를 이용해서 LEAF노드인 경우 1을 출력함
- ❖ CONNECT_BY_ROOT를 이용하여 최상위 관리자를 출력함

LVL	조직인원	DEPT_CD	DEPT_NM	CONNECT_BY_ISLEAF	최상위관리자
1	9999999999(김회장)	999999	회장실	0	9999999999
2	1000000001(이경오)	100001	운영본부	0	9999999999
3	1000000002(이현승)	100002	지원팀	0	9999999999
4	1000000003(이정수)	100002	지원팀	1	9999999999
4	1000000004(이승준)	100002	지원팀	1	9999999999
4	1000000005(김현희)	100002	지원팀	1	9999999999
중간생략...					
2	1000000032(이준표)	100011	신사업본부	0	9999999999
3	1000000033(홍사기)	100012	인공지능팀	0	9999999999
4	1000000034(김익정)	100012	인공지능팀	1	9999999999
4	1000000035(최창수)	100012	인공지능팀	1	9999999999
4	1000000036(박여진)	100012	인공지능팀	1	9999999999
3	1000000037(이현정)	100013	빅데이터팀	0	9999999999
4	1000000038(김혜수)	100013	빅데이터팀	1	9999999999
4	1000000039(이박력)	100013	빅데이터팀	1	9999999999
4	1000000040(김여진)	100013	빅데이터팀	1	9999999999

➢ 계층형 SQL 실습 – SYS_CONNECT_BY_PATH의 사용

```
SELECT LEVEL LVL
     , LPAD(' ', 4*(LEVEL-1))|| EMP_NO || '(' || EMP_NM || ')' AS "조직인원"
     , A.DEPT_CD
     , B.DEPT_NM
     , CONNECT_BY_ISLEAF
     , CONNECT_BY_ROOT A.EMP_NO AS "최상위관리자"
     , SYS_CONNECT_BY_PATH(EMP_NO || '(' || EMP_NM || ')', '/') AS "조직인원경로"
FROM TB_EMP A, TB_DEPT B
WHERE A.DEPT_CD = B.DEPT_CD
START WITH A.EMP_NM = '이경오'
CONNECT BY PRIOR A.EMP_NO = A.DIRECT_MANAGER_EMP_NO;
```

- ❖ 사원명이 이경오인 행을 START WITH로 시작함
- ❖ PRIOR 자식 = 부모 순으로 순방향 전개를 하였음
- ❖ CONNECT _BY_ISLEAFT를 이용해서 LEAF노드인 경우 1을 출력함
- ❖ CONNECT_BY_ROOT를 이용하여 최상위 관리자를 출력함
- ❖ SYS_CONNECT_BY_PATH 함수를 이용하여 조직인원경로를 출력함

LVL	조직인원	DEPT_CD	DEPT_NM	CONNECT_BY_ISLEAF	최상위관리자	조직인원경로
1	1000000001(이경오)	100001	운영본부	0	1000000001	/1000000001(이경오)
2	1000000002(이현승)	100002	지원팀	0	1000000001	/1000000001(이경오)/1000000002(이현승)
3	1000000003(이정수)	100002	지원팀	1	1000000001	/1000000001(이경오)/1000000002(이현승)/1000000003(이정수)
3	1000000004(이승준)	100002	지원팀	1	1000000001	/1000000001(이경오)/1000000002(이현승)/1000000004(이승준)
3	1000000005(김현희)	100002	지원팀	1	1000000001	/1000000001(이경오)/1000000002(이현승)/1000000005(김현희)
2	1000000006(김혜진)	100003	기획팀	0	1000000001	/1000000001(이경오)/1000000006(김혜진)
3	1000000007(이순자)	100003	기획팀	1	1000000001	/1000000001(이경오)/1000000006(김혜진)/1000000007(이순자)
3	1000000008(김려원)	100003	기획팀	1	1000000001	/1000000001(이경오)/1000000006(김혜진)/1000000008(김려원)
3	1000000009(박태범)	100003	기획팀	1	1000000001	/1000000001(이경오)/1000000006(김혜진)/1000000009(박태범)
2	1000000010(박혜령)	100004	디자인팀	0	1000000001	/1000000001(이경오)/1000000010(박혜령)
3	1000000011(최수자)	100004	디자인팀	1	1000000001	/1000000001(이경오)/1000000010(박혜령)/1000000011(최수자)
3	1000000012(김호형)	100004	디자인팀	1	1000000001	/1000000001(이경오)/1000000010(박혜령)/1000000012(김호형)
3	1000000013(이나라)	100004	디자인팀	1	1000000001	/1000000001(이경오)/1000000010(박혜령)/1000000013(이나라)

➢ 계층형 SQL 실습 – SELF 조인의 활용

```
SELECT A.EMP_NO "사원번호"
     , A.EMP_NM "사원번호"
     , A.DIRECT_MANAGER_EMP_NO "관리자사원번호"
     , (SELECT L.EMP_NM FROM TB_EMP L WHERE L.EMP_NO = A.DIRECT_MANAGER_EMP_NO) AS "관리자사원명"
     , B.DIRECT_MANAGER_EMP_NO AS "차상위관리자사원번호"
     , (SELECT L.EMP_NM FROM TB_EMP L WHERE L.EMP_NO = B.DIRECT_MANAGER_EMP_NO) AS "차상위관리자사원명"
FROM TB_EMP A INNER JOIN TB_EMP B
ON(A.DIRECT_MANAGER_EMP_NO = B.EMP_NO)
JOIN TB_DEPT C
ON (A.DEPT_CD = C.DEPT_CD)
;
```

사원번호	사원번호	관리자사원번호	관리자사원명	차상위관리자사원번호	차상위관리자사원명
1000000006	김혜진	1000000001	이경오	9999999999	김회장
1000000010	박혜령	1000000001	이경오	9999999999	김회장
1000000002	이현승	1000000001	이경오	9999999999	김회장
1000000004	이승준	1000000002	이현승	1000000001	이경오
1000000003	이정수	1000000002	이현승	1000000001	이경오
1000000005	김현희	1000000002	이현승	1000000001	이경오
1000000007	이순자	1000000006	김혜진	1000000001	이경오
1000000008	김려원	1000000006	김혜진	1000000001	이경오
1000000009	박태범	1000000006	김혜진	1000000001	이경오
1000000012	김호형	1000000010	박혜령	1000000001	이경오
1000000013	이나라	1000000010	박혜령	1000000001	이경오
1000000011	최수자	1000000010	박혜령	1000000001	이경오
중간생략...					
1000000001	이경오	9999999999	김회장	(NULL)	(NULL)
1000000014	이관심	9999999999	김회장	(NULL)	(NULL)
1000000023	이관심	9999999999	김회장	(NULL)	(NULL)
1000000032	이준표	9999999999	김회장	(NULL)	(NULL)

- ❖ SELF 조인은 동일한 테이블끼리의 조인을 의미함
- ❖ SELF 조인을 이용해서 계층 형의 데이터를 출력할 수 있음
- ❖ SELF 조인 시 INNER 조인을 하여 관리자가 존재하지 않는 김회장의 레코드는 결과 집합에 포함되지 않음

➢ 계층형 SQL 실습 – SELF 조인 및 OUTER 조인의 활용

```
SELECT A.EMP_NO "사원번호"
     , A.EMP_NM "사원번호"
     , A.DIRECT_MANAGER_EMP_NO "관리자사원번호"
     , (SELECT L.EMP_NM FROM TB_EMP L WHERE L.EMP_NO = A.DIRECT_MANAGER_EMP_NO) AS "관리자사원명"
     , B.DIRECT_MANAGER_EMP_NO AS "차상위관리자사원번호"
     , (SELECT L.EMP_NM FROM TB_EMP L WHERE L.EMP_NO = B.DIRECT_MANAGER_EMP_NO) AS "차상위관리자사원명"
FROM TB_EMP A LEFT OUTER JOIN TB_EMP B
ON(A.DIRECT_MANAGER_EMP_NO = B.EMP_NO)
JOIN TB_DEPT C
ON (A.DEPT_CD = C.DEPT_CD)
;
```

❖ SELF 조인을 하면서 OUTER 조인까지 하여 상위 관리자가 없어서 결과 집합에 나오지 않았던 김회장의 행까지 출력 하게 됨

사원번호	사원번호	관리자사원번호	관리자사원명	차상위관리자사원번호	차상위관리자사원명
1000000001	이경오	9999999999	김회장	(NULL)	(NULL)
1000000014	이관심	9999999999	김회장	(NULL)	(NULL)
1000000023	이관심	9999999999	김회장	(NULL)	(NULL)
1000000032	이준표	9999999999	김회장	(NULL)	(NULL)
1000000002	이현승	1000000001	이경오	9999999999	김회장
1000000006	김혜진	1000000001	이경오	9999999999	김회장
1000000010	박혜령	1000000001	이경오	9999999999	김회장
중간생략...					
9999999999	김회장	(NULL)	(NULL)	(NULL)	(NULL)

감사합니다
THANK YOU