

5. 조인(JOIN)

한 개 이상의 테이블에서 원하는 결과를 얻기 위한 조인을 학습한다.

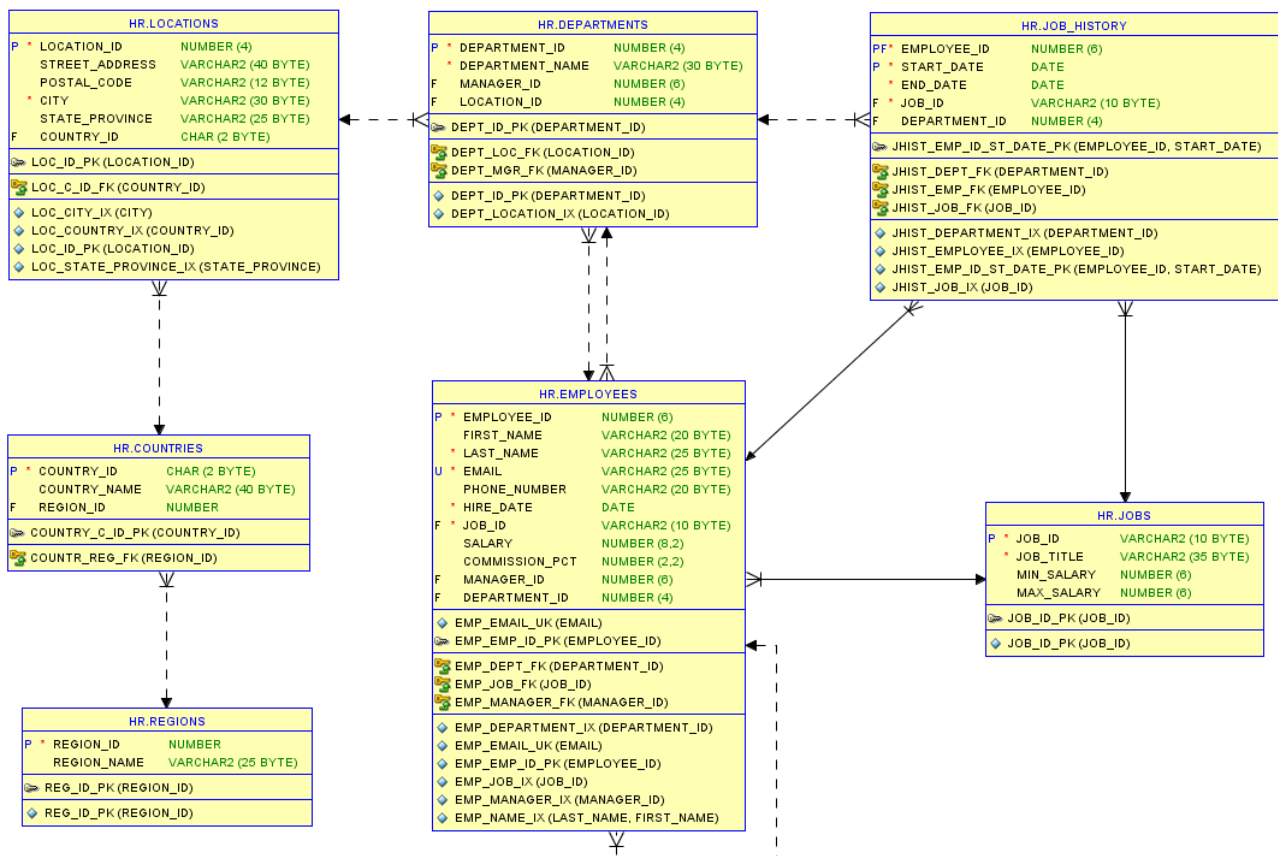
Equi Join, Non-Equi Join, Outer Join, Self Join 방식을 학습한다.

하나의 테이블에 대해서 SQL 명령어를 사용하였다. 하지만 관계형 데이터베이스에서는 테이블간의 관계가 중요하기 때문에 하나 이상의 테이블이 빈번히 결합되어 사용된다. 한 개 이상의 테이블에서 데이터를 조회하기 위해서 사용되는 것이 조인(JOIN) 이다.

다시 말해 SQL에서는 두 개 이상의 테이블을 결합해야만 원하는 결과를 얻을 수 있을 때 한 번의 질의로 원하는 결과를 얻을 수 있는 조인 기능을 제공한다.

종 류	설 명
Equi Join	동일 칼럼을 기준으로 조인한다. (inner join, simple join)
NonEqui Join	동일 칼럼이 없이 다른 조건을 사용하여 조인한다.
Outer Join	조인 조건에 만족하지 않는 행도 나타낸다.
Self Join	한 테이블 내에서 조인한다.

WHERE절에 명시하는 조건이 FROM절에 명시한 여러 Table을 묶는 Join조건이 된다. 이러한 Join조건은 반드시 묶어야 할 Table수보다 하나가 적다. 즉 Table 수가 n개라면 Join 조건은 n-1이 된다.



1) Cartesian Product (카티션 곱) 또는 Cross Join

Cross Join이란 2개 이상의 테이블이 조인될 때 WHERE절에 의해 공통되는 칼럼에 의한 결합이 발생되지 않는 경우를 말한다. 그렇기 때문에 테이블에 존재하는 모든 데이터가 검색 결과로 나타난다.

다음은 Cross Join으로 특별한 키워드 없이 SELECT문의 FROM절에 EMPLOYEES테이블과 DEPARTMENTS테이블을 동시에 기술한다.

```
SELECT *  
FROM EMPLOYEES, DEPARTMENTS;
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID	DEPARTMENT_ID_1	DEPARTMENT_NAME	MANAGER_ID_1	LOCATION_ID
1	100	Steven	King	SKING	515.123.4567	03/06/17	AD_FRES	24000	(null)	(null)	90	10	Administration	200	1700
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	05/09/21	AD_VF	17000	(null)	100	90	10	Administration	200	1700
3	102	Lex	De Haan	LDEHAAN	515.123.4569	01/01/13	AD_VF	17000	(null)	100	90	10	Administration	200	1700
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	06/01/03	IT_PROG	9000	(null)	102	60	10	Administration	200	1700
5	104	Bruce	Ernat	BERNST	590.423.4568	07/05/21	IT_PROG	6000	(null)	103	60	10	Administration	200	1700
6	105	David	Austin	DAUSTIN	590.423.4569	05/06/25	IT_PROG	4800	(null)	103	60	10	Administration	200	1700
7	106	Valli	Pataballa	VPATABAL	590.423.4560	06/02/05	IT_PROG	4800	(null)	103	60	10	Administration	200	1700
8	107	Diana	Lorentz	DLORENTZ	590.423.5567	07/02/07	IT_PROG	4200	(null)	103	60	10	Administration	200	1700
9	108	Nancy	Greenberg	NGREENBE	515.124.4569	02/08/17	FI_MGR	12008	(null)	101	100	10	Administration	200	1700
10	109	Daniel	Faviet	DFAVIET	515.124.4169	02/08/16	FI_ACCOUNT	9000	(null)	108	100	10	Administration	200	1700
11	110	John	Chen	JCHEN	515.124.4269	05/09/28	FI_ACCOUNT	8200	(null)	108	100	10	Administration	200	1700
12	111	Ismael	Sciarra	ISCIARRA	515.124.4369	05/09/30	FI_ACCOUNT	7700	(null)	108	100	10	Administration	200	1700
13	112	Jose Manuel	Urman	JURMAN	515.124.4469	06/03/07	FI_ACCOUNT	7800	(null)	108	100	10	Administration	200	1700
14	113	Luis	Popp	LPOPP	515.124.4567	07/12/07	FI_ACCOUNT	6900	(null)	108	100	10	Administration	200	1700
15	114	Den	Raphaely	DRAPHAEL	515.127.4561	02/12/07	FU_MAN	11000	(null)	100	30	10	Administration	200	1700
16	115	Alexander	Rhoo	AKHOO	515.127.4562	03/05/18	FU_CLERK	3100	(null)	114	30	10	Administration	200	1700
17	116	Shelli	Baida	SBAIDA	515.127.4563	05/12/24	FU_CLERK	2900	(null)	114	30	10	Administration	200	1700
18	117	Sigal	Tobias	STOBIAS	515.127.4564	05/07/24	FU_CLERK	2800	(null)	114	30	10	Administration	200	1700
19	118	Guy	Himuro	GHIIMURO	515.127.4565	06/11/15	FU_CLERK	2600	(null)	114	30	10	Administration	200	1700
20	119	Karen	Colmenares	KCOLMENA	515.127.4566	07/08/10	FU_CLERK	2500	(null)	114	30	10	Administration	200	1700
21	120	Matthew	Weiss	MWEISS	650.123.1234	04/07/18	ST_MAN	8000	(null)	100	50	10	Administration	200	1700
22	121	Adam	Fripp	AFRIPP	650.123.2234	05/04/10	ST_MAN	8200	(null)	100	50	10	Administration	200	1700
23	122	Payan	Kaufling	KKAUFLIN	650.123.3234	03/05/01	ST_MAN	7900	(null)	100	50	10	Administration	200	1700
24	123	Shanta	Vollman	SVOLLMAN	650.123.4234	05/10/10	ST_MAN	6500	(null)	100	50	10	Administration	200	1700
25	124	Kevin	Mourgos	KMORGOS	650.123.5234	07/11/16	ST_MAN	5800	(null)	100	50	10	Administration	200	1700
26	125	Julia	Nayer	JNAYER	650.124.1214	05/07/16	ST_CLERK	3200	(null)	120	50	10	Administration	200	1700
27	126	Irene	Mikkilineni	IMIKKILIN	650.124.1224	06/09/28	ST_CLERK	2700	(null)	120	50	10	Administration	200	1700

조인 대상 테이블들의 조건이 누락되었을 경우 발생하는 현상으로 해당 조인에 참여하는 모든 대상 행을 다 출력한다.

기본적으로 조인은 다음과 같은 규칙을 준수해야 한다.

- ① Primary Key와 Foreign Key 컬럼을 통한 다른 테이블의 행과 연결한다.
- ② 연결 Key 사용으로 테이블과 테이블이 결합한다.
- ③ WHERE 절에서 조인 조건을 사용한다. (조인 조건 개수 = 연결 테이블 수 -1)
- ④ 명확성을 위해 칼럼 이름 앞에 테이블명 또는 테이블 별칭을 붙인다.

조인이 수행될 때는 두 개 이상의 테이블이 사용되는데 이때 둘 중 하나의 테이블을 먼저 읽고 조인 조건 절을 확인하여 나머지 테이블에 가서 데이터를 가져오게 된다 이 때 먼저 읽는 테이블을 선행 테이블 (driving table 또는 Inner table) 이라고 하고 뒤에 읽는 테이블을 후행 테이블 (driven table 또는 Outer table) 이라고 한다. 그리고 선행 테이블은 조회할 데이터가 적은 테이블로 선택해야 속도면에서 유리하다.

2) Equi Join

Equi Join은 가장 많이 사용하는 조인 방법으로서 조인 대상이 되는 두 테이블에서 공통적으로 존재하는 칼럼의 값이 일치되는 행을 연결하여 결과를 생성하는 조인 방법이다.

EMPLOYEES 테이블과 DEPARTMENTS 테이블의 공통 칼럼인 DEPARTMENT_ID의 값이 일치(=)되는 조건을 WHERE절에 사용한다. 두 테이블을 조인하려면 일치되는 공통 칼럼을 사용해야 한다. 칼럼 명이 같게 되면 혼동이 오기 때문에 칼럼명 앞에 테이블명을 점(.)과 함께 기술한다.

```
SELECT EMPLOYEE_ID, FIRST_NAME, DEPARTMENT_ID
FROM EMPLOYEES;
```

```
SELECT DEPARTMENT_ID, DEPARTMENT_NAME
FROM DEPARTMENTS;
```

	EMPLOYEE_ID	FIRST_NAME	DEPARTMENT_ID
1	100	Steven	90
2	101	Neena	90
3	102	Lex	90
4	103	Alexander	60
5	104	Bruce	60
6	105	David	60
7	106	Valli	60
8	107	Diana	60
9	108	Nancy	100
10	109	Daniel	100
11	110	John	100
12	111	Ismael	100
13	112	Jose Manuel	100
14	113	Luis	100
15	114	Den	30
16	115	Alexander	30
17	116	Shelli	30
18	117	Sigal	30
19	118	Guy	30
20	119	Karen	30

	DEPARTMENT_ID	DEPARTMENT_NAME
1	10	Administration
2	20	Marketing
3	30	Purchasing
4	40	Human Resources
5	50	Shipping
6	60	IT
7	70	Public Relations
8	80	Sales
9	90	Executive
10	100	Finance
11	110	Accounting
12	120	Treasury
13	130	Corporate Tax
14	140	Control And Credit
15	150	Shareholder Services
16	160	Benefits
17	170	Manufacturing
18	180	Construction
19	190	Contracting
20	200	Operations

```
SELECT FIRST_NAME, DEPARTMENT_NAME
FROM EMPLOYEES, DEPARTMENTS
WHERE EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID;
```

시작 페이지 x PDB_HR x

워크시트 | 질의 작성기

```

11
12 SELECT FIRST_NAME, DEPARTMENT_NAME
13 FROM EMPLOYEES, DEPARTMENTS
14 WHERE EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID;

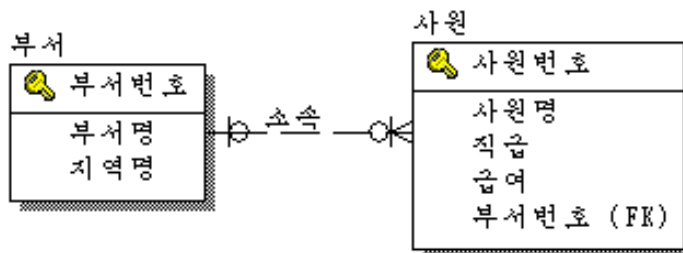
```

스크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.005초)

	FIRST_NAME	DEPARTMENT_NAME
1	Jennifer	Administration
2	Pat	Marketing
3	Michael	Marketing
4	Sigal	Purchasing
5	Karen	Purchasing
6	Shelli	Purchasing
7	Den	Purchasing
8	Alexander	Purchasing
9	Guy	Purchasing
10	Susan	Human Resources
11	Kevin	Shipping
12	Jean	Shipping
13	Adam	Shipping
14	Timothy	Shipping
15	Ki	Shipping

조인한 결과를 살펴보면 부서번호를 기준으로 같은 값을 가진 사원 테이블과 부서 테이블이 결합되었다. 조인은 Primary Key와 Foreign Key를 통한 다른 테이블 행과 연결한다.



부서 테이블의 Primary Key인 부서번호가 사원 테이블의 Foreign Key로 설정되어 있다. 이 연결 Key를 WHERE 절에서 조인 조건에 사용하였다. 비교 연산자로 "="를 사용하였으므로 이를 Equi Join이라고 한다.

WHERE EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID

테이블명이 너무 긴 경우에는 테이블 명에 간단하게 별칭을 부여해서 문장을 간단하게 기술할 수 있다. 테이블 명의 별칭은 FROM 절 다음에 테이블 이름을 명시하고 공백을 둔 다음에 별칭을 지정한다.

FROM EMPLOYEES E, DEPARTMENTS D

테이블명 별칭, 테이블명 별칭

```

SELECT E.FIRST_NAME, D.DEPARTMENT_NAME, E.DEPARTMENT_ID
FROM EMPLOYEES E, DEPARTMENTS D
WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID;

```

The screenshot shows the SQL Developer interface with a query window titled 'PDB_HR'. The query is as follows:

```

SELECT E.FIRST_NAME, D.DEPARTMENT_NAME, E.DEPARTMENT_ID
FROM EMPLOYEES E, DEPARTMENTS D
WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID;

```

The result set is displayed in a table with 15 rows and 3 columns: FIRST_NAME, DEPARTMENT_NAME, and DEPARTMENT_ID.

	FIRST_NAME	DEPARTMENT_NAME	DEPARTMENT_ID
1	Jennifer	Administration	10
2	Pat	Marketing	20
3	Michael	Marketing	20
4	Sigal	Purchasing	30
5	Karen	Purchasing	30
6	Shelli	Purchasing	30
7	Den	Purchasing	30
8	Alexander	Purchasing	30
9	Guy	Purchasing	30
10	Susan	Human Resources	40
11	Kevin	Shipping	50
12	Jean	Shipping	50
13	Adam	Shipping	50
14	Timothy	Shipping	50
15	Ki	Shipping	50

Equi Join에 AND 연산하기

Susan인 직원의 정보만을 출력하기 위해서는 WHERE 절에서 AND 연산자를 추가한다.

```

SELECT E.FIRST_NAME, D.DEPARTMENT_NAME
FROM EMPLOYEES E, DEPARTMENTS D
WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID AND E.FIRST_NAME='Susan';

```

The screenshot shows the SQL Developer interface with the same query window. The query is updated to include the AND condition:

```

SELECT E.FIRST_NAME, D.DEPARTMENT_NAME
FROM EMPLOYEES E, DEPARTMENTS D
WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID AND E.FIRST_NAME='Susan';

```

The result set now only contains one row, for Susan in the Human Resources department.

	FIRST_NAME	DEPARTMENT_NAME
1	Susan	Human Resources

3) Non-Equi Join

Non-Equi 조인은 조인할 테이블 사이에 칼럼의 값이 직접적으로 일치하지 않을 시 사용하는 조인으로 '='를 제외한 연산자를 사용한다.

예제를 수행하기 위해 급여 등급 테이블(SALARYGRADE)을 생성하여 레코드를 입력한다.

```

CREATE TABLE SALARYGRADE (
  GRADE NUMBER,
  MINSALARY NUMBER,

```

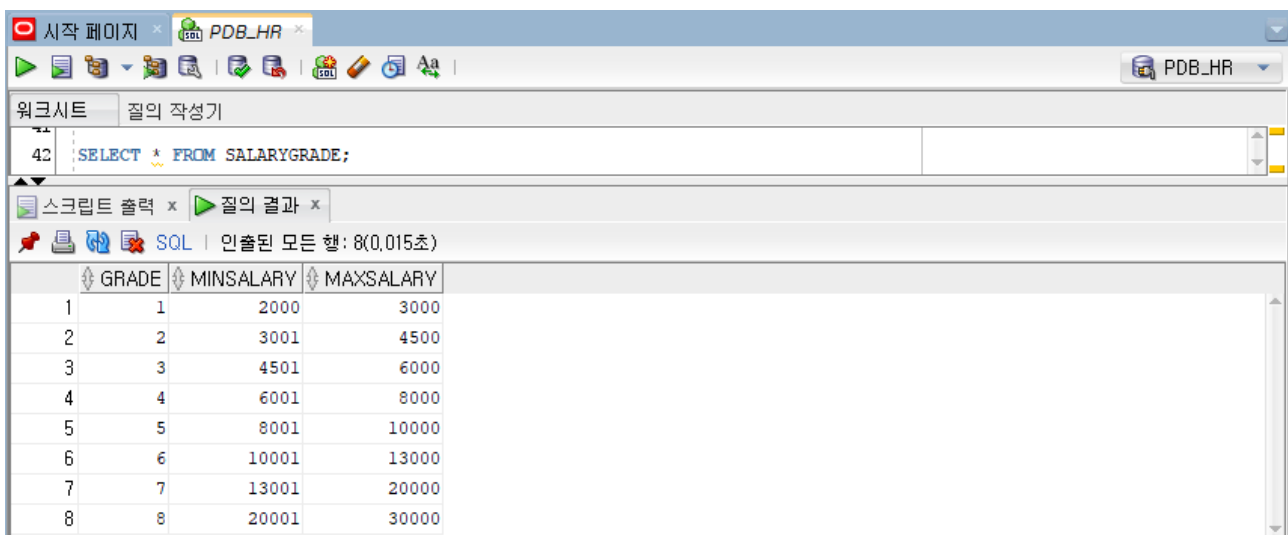
MAXSALARY NUMBER

);

```
INSERT INTO SALARYGRADE (GRADE, MINSALARY, MAXSALARY) VALUES(1, 2000, 3000);
INSERT INTO SALARYGRADE (GRADE, MINSALARY, MAXSALARY) VALUES(2, 3001, 4500);
INSERT INTO SALARYGRADE (GRADE, MINSALARY, MAXSALARY) VALUES(3, 4501, 6000);
INSERT INTO SALARYGRADE (GRADE, MINSALARY, MAXSALARY) VALUES(4, 6001, 8000);
INSERT INTO SALARYGRADE (GRADE, MINSALARY, MAXSALARY) VALUES(5, 8001, 10000);
INSERT INTO SALARYGRADE (GRADE, MINSALARY, MAXSALARY) VALUES(6, 10001, 13000);
INSERT INTO SALARYGRADE (GRADE, MINSALARY, MAXSALARY) VALUES(7, 13001, 20000);
INSERT INTO SALARYGRADE (GRADE, MINSALARY, MAXSALARY) VALUES(8, 20001, 30000);
```

commit;

```
SELECT * FROM SALARYGRADE;
```



	GRADE	MINSALARY	MAXSALARY
1	1	2000	3000
2	2	3001	4500
3	3	4501	6000
4	4	6001	8000
5	5	8001	10000
6	6	10001	13000
7	7	13001	20000
8	8	20001	30000

급여 등급을 8개로 나누어 놓은 SALARYGRADE 테이블에서 정보를 얻어 와서 각 사원의 급여 등급을 지정한다. 이를 위해서는 EMPLOYEES 테이블과 SALARYGRADE 테이블을 조인해야 한다.

```
SELECT E.FIRST_NAME, E.SALARY, S.GRADE
FROM EMPLOYEES E, SALARYGRADE S
WHERE E.SALARY BETWEEN S.MINSALARY AND S.MAXSALARY;
```

```
SELECT E.FIRST_NAME, E.SALARY, S.GRADE
FROM EMPLOYEES E, SALARYGRADE S
WHERE E.SALARY >= S.MINSALARY AND E.SALARY <= S.MAXSALARY;
```

시작 페이지 x PDB_HR x

워크시트 | 질의 작성기

```

44 SELECT E.FIRST_NAME, E.SALARY, S.GRADE
45 FROM EMPLOYEES E, SALARYGRADE S
46 WHERE E.SALARY BETWEEN S.MINSALARY AND S.MAXSALARY;

```

스크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.004초)

	FIRST_NAME	SALARY	GRADE
1	TJ	2100	1
2	Steven	2200	1
3	Hazel	2200	1
4	James	2400	1
5	Ki	2400	1
6	Karen	2500	1
7	James	2500	1
8	Joshua	2500	1
9	Peter	2500	1
10	Martha	2500	1
11	Randall	2500	1
12	Guy	2600	1
13	Randall	2600	1
14	Donald	2600	1
15	Douglas	2600	1
16	Irene	2700	1
17	John	2700	1
18	Sigal	2800	1
19	Mozhe	2800	1
20	Girard	2800	1
21	Vance	2800	1
22	Shelli	2900	1
23	Michael	2900	1
24	Timothy	2900	1
25	Anthony	3000	1

4) Outer Join

행이 조인 조건에 만족하지 않을 경우 그 행은 결과에 나타나지 않게 된다. 이때 조인 조건에 만족하지 않는 행들도 나타내기 위해 Outer Join이 사용된다.

사원 테이블과 부서 테이블을 조인하여 사원 이름과 부서번호와 부서명을 출력한다.

```

SELECT E.FIRST_NAME, D.DEPARTMENT_ID , D.DEPARTMENT_NAME
FROM EMPLOYEES E, DEPARTMENTS D
WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID
ORDER BY D.DEPARTMENT_ID;

```

시작 페이지 x PDB_HR x

워크시트 | 질의 작성기

```

52 SELECT E.FIRST_NAME, D.DEPARTMENT_ID, D.DEPARTMENT_NAME
53 FROM EMPLOYEES E, DEPARTMENTS D
54 WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID
55 ORDER BY D.DEPARTMENT_ID;

```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 106(0,034초)

	FIRST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Jennifer	10	Administration
2	Pat	20	Marketing
3	Michael	20	Marketing
4	Sigal	30	Purchasing
5	Karen	30	Purchasing
6	Shelli	30	Purchasing
7	Den	30	Purchasing
8	Alexander	30	Purchasing
9	Guy	30	Purchasing
10	Susan	40	Human Resources
11	Kevin	50	Shipping
12	Jean	50	Shipping
13	Adam	50	Shipping
14	Timothy	50	Shipping
15	Ki	50	Shipping

부서번호가 110까지만 존재한다.

SELECT * FROM DEPARTMENTS;

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	100	Finance	100	1700
11	110	Accounting	205	1700
12	120	Treasury	(null)	1700
13	130	Corporate Tax	(null)	1700
14	140	Control And Credit	(null)	1700
15	150	Shareholder Services	(null)	1700
16	160	Benefits	(null)	1700
17	170	Manufacturing	(null)	1700
18	180	Construction	(null)	1700
19	190	Contracting	(null)	1700
20	200	Operations	(null)	1700
21	210	IT Support	(null)	1700
22	220	NOC	(null)	1700
23	230	IT Helpdesk	(null)	1700
24	240	Government Sales	(null)	1700
25	250	Retail Sales	(null)	1700
26	260	Recruiting	(null)	1700
27	270	Payroll	(null)	1700

부서 테이블을 조회하면 번호가 110번 이상 부서가 존재한다. 하지만, 조인 결과를 보면 10번부터 110번 부서번호만 출력되고 120번부터는 출력되지 않는다. 이는 직원 테이블의 부서번호에는 110번 보다 큰 번호가 존재하지 않기 때문이다.

부서 테이블의 120번 부서와 조인할 직원 테이블의 부서번호가 없지만, 120번 이상의 부서도 출력되도록 하려면 Outer Join을 사용해야 한다. **Outer Join**을 하기 위해서 사용하는 기호는 (+)이며 조인 조건에서 정보가 부족한 칼럼 명 뒤에 위치하게 하면 된다.

즉, 사원 테이블에 부서번호 120번 이상의 부서번호가 없기 때문에 E.DEPARTMENT_ID(+)쪽에 + 기호를 덧붙이면 된다.

```
SELECT E.FIRST_NAME, D.DEPARTMENT_ID , D.DEPARTMENT_NAME
FROM EMPLOYEES E, DEPARTMENTS D
WHERE E.DEPARTMENT_ID(+) = D.DEPARTMENT_ID;
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 122(0.008초)

FIRST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
99 Luis	100	Finance
100 Jose Manuel	100	Finance
101 John	100	Finance
102 Daniel	100	Finance
103 Ismael	100	Finance
104 Nancy	100	Finance
105 William	110	Accounting
106 Shelley	110	Accounting
107 (null)	120	Treasury
108 (null)	130	Corporate Tax
109 (null)	140	Control And Credit
110 (null)	150	Shareholder Services
111 (null)	160	Benefits
112 (null)	170	Manufacturing
113 (null)	180	Construction
114 (null)	190	Contracting
115 (null)	200	Operations
116 (null)	210	IT Support
117 (null)	220	NOC
118 (null)	230	IT Helpdesk
119 (null)	240	Government Sales
120 (null)	250	Retail Sales
121 (null)	260	Recruiting
122 (null)	270	Payroll

2007년도 상반기에 입사한 사원을 구해보자.

```
SELECT EMPLOYEE_ID, FIRST_NAME, HIRE_DATE, DEPARTMENT_ID
FROM EMPLOYEES
WHERE HIRE_DATE >= '2007/01/01' AND HIRE_DATE <= '2007/06/30';
```

	EMPLOYEE_ID	FIRST_NAME	HIRE_DATE	DEPARTMENT_ID
1	104	Bruce	07/05/21	60
2	107	Diana	07/02/07	60
3	127	James	07/01/14	50
4	132	TJ	07/04/10	50
5	163	Danielle	07/03/19	80
6	171	William	07/02/23	80
7	172	Elizabeth	07/03/24	80
8	178	Kimberely	07/05/24	(null)
9	182	Martha	07/06/21	50
10	187	Anthony	07/02/07	50
11	195	Vance	07/03/17	50
12	198	Donald	07/06/21	50

2007년도 상반기에 입사한 사원번호, 사원명, 입사일, 부서명을 구한다.

```

SELECT EMPLOYEE_ID, FIRST_NAME, HIRE_DATE, D.DEPARTMENT_NAME
FROM EMPLOYEES E, DEPARTMENTS D
WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID
      AND HIRE_DATE >= '2007/01/01' AND HIRE_DATE <= '2007/06/30';

```

	EMPLOYEE_ID	FIRST_NAME	HIRE_DATE	DEPARTMENT_NAME
1	187	Anthony	07/02/07	Shipping
2	182	Martha	07/06/21	Shipping
3	127	James	07/01/14	Shipping
4	132	TJ	07/04/10	Shipping
5	198	Donald	07/06/21	Shipping
6	195	Vance	07/03/17	Shipping
7	104	Bruce	07/05/21	IT
8	107	Diana	07/02/07	IT
9	172	Elizabeth	07/03/24	Sales
10	163	Danielle	07/03/19	Sales
11	171	William	07/02/23	Sales

```

SELECT EMPLOYEE_ID, FIRST_NAME, HIRE_DATE, D.DEPARTMENT_NAME
FROM EMPLOYEES E, DEPARTMENTS D
WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID(+)
      AND HIRE_DATE >= '2007/01/01' AND HIRE_DATE <= '2007/06/30';

```

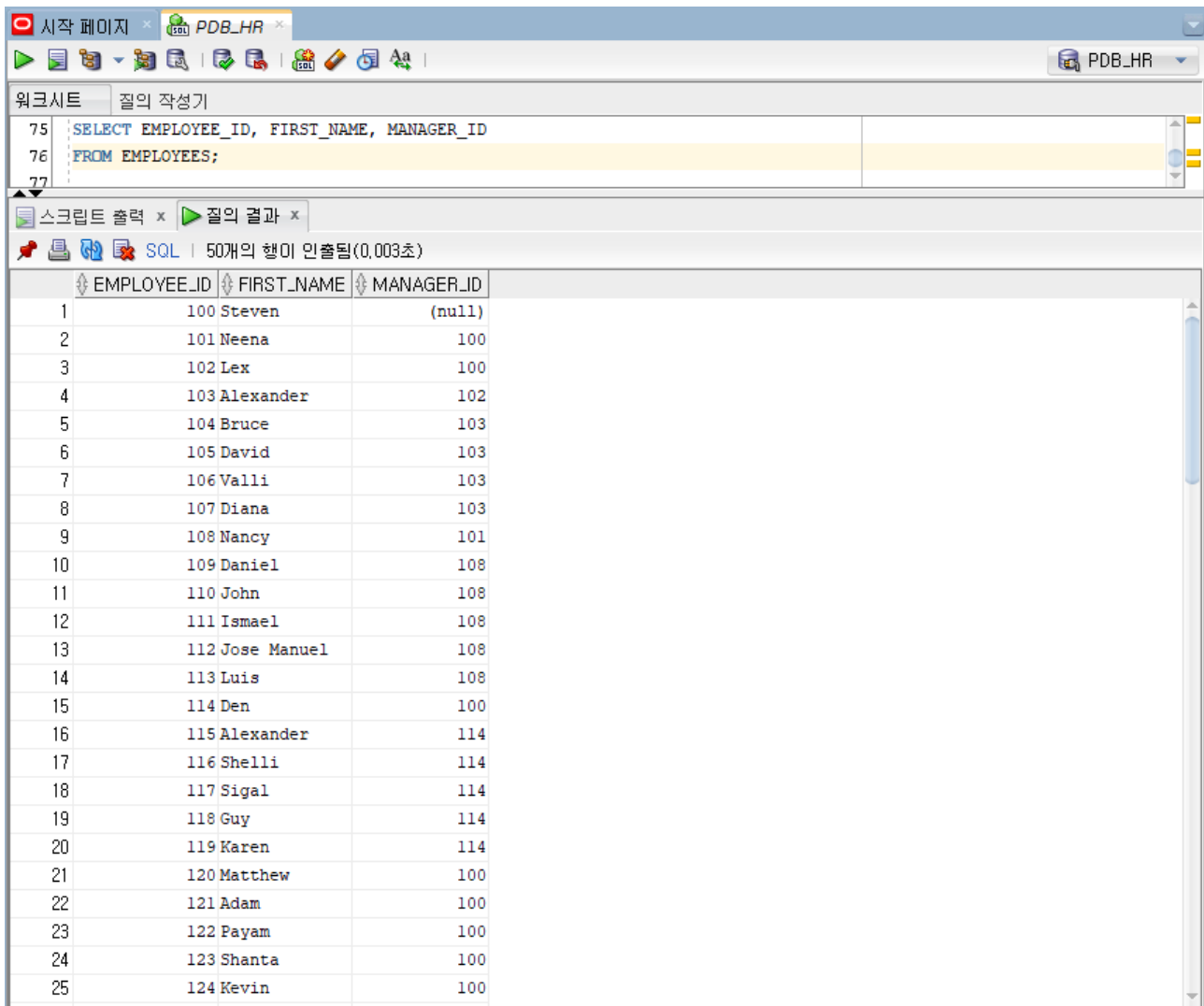
	EMPLOYEE_ID	FIRST_NAME	HIRE_DATE	DEPARTMENT_NAME
1	127	James	07/01/14	Shipping
2	132	TJ	07/04/10	Shipping
3	182	Martha	07/06/21	Shipping
4	187	Anthony	07/02/07	Shipping
5	195	Vance	07/03/17	Shipping
6	198	Donald	07/06/21	Shipping
7	104	Bruce	07/05/21	IT
8	107	Diana	07/02/07	IT
9	163	Danielle	07/03/19	Sales
10	171	William	07/02/23	Sales
11	172	Elizabeth	07/03/24	Sales
12	178	Kimberely	07/05/24	(null)

5) Self Join

Self Join이란 말 그대로 자기 자신과 조인을 맺는 것을 말한다. FROM 절 다음에 동일한 테이블명을 2번 기술하고 WHERE 절에도 조인 조건을 주어야 하는데 이때 서로 다른 테이블인 것처럼 인식할 수 있도록 하기 위해서 별칭을 사용한다.

특정 사원을 담당하는 매니저 사원의 이름을 출력하자.

```
SELECT EMPLOYEE_ID, FIRST_NAME, MANAGER_ID
FROM EMPLOYEES;
```



EMPLOYEE_ID	FIRST_NAME	MANAGER_ID
1	Steven	(null)
2	Neena	100
3	Lex	100
4	Alexander	102
5	Bruce	103
6	David	103
7	Valli	103
8	Diana	103
9	Nancy	101
10	Daniel	108
11	John	108
12	Ismael	108
13	Jose Manuel	108
14	Luis	108
15	Den	100
16	Alexander	114
17	Shelli	114
18	Sigal	114
19	Guy	114
20	Karen	114
21	Matthew	100
22	Adam	100
23	Payam	100
24	Shanta	100
25	Kevin	100

EMPLOYEES 테이블에 별칭을 사용하여 하나의 테이블을 두 개의 테이블인 것처럼 사용하려면 WORK(사원 테이블)과 MANAGER(매니저 테이블)로 별칭을 부여한다.

```
SELECT WORK.FIRST_NAME 사원명, MANAGER.FIRST_NAME 매니저명
FROM EMPLOYEES WORK, EMPLOYEES MANAGER
WHERE WORK.MANAGER_ID = MANAGER.EMPLOYEE_ID;
```

```
SELECT EMPLOYEE_ID, FIRST_NAME, MANAGER_ID
FROM EMPLOYEES;
```

	EMPLOYEE_ID	FIRST_NAME	MANAGER_ID
1	100	Steven	(null)
2	101	Neena	100
3	102	Lex	100
4	103	Alexander	102
5	104	Bruce	103
6	105	David	103
7	106	Valli	103
8	107	Diana	103
9	108	Nancy	101
10	109	Daniel	108
11	110	John	108
12	111	Ismael	108
13	112	Jose Manuel	108
14	113	Luis	108
15	114	Den	100

```
SELECT EMPLOYEE_ID, FIRST_NAME
FROM EMPLOYEES ORDER BY EMPLOYEE_ID;
```

	EMPLOYEE_ID	FIRST_NAME
1	100	Steven
2	101	Neena
3	102	Lex
4	103	Alexander
5	104	Bruce
6	105	David
7	106	Valli
8	107	Diana
9	108	Nancy
10	109	Daniel
11	110	John
12	111	Ismael
13	112	Jose Manuel
14	113	Luis
15	114	Den

시작 페이지 x PDB_HR x

워크시트 | 질의 작성기

```
78 SELECT WORK.FIRST_NAME 사원명, MANAGER.FIRST_NAME 매니저명
79 FROM EMPLOYEES WORK, EMPLOYEES MANAGER
80 WHERE WORK.MANAGER_ID = MANAGER.EMPLOYEE_ID;
```

스크립트 출력 x | 질의 결과 x

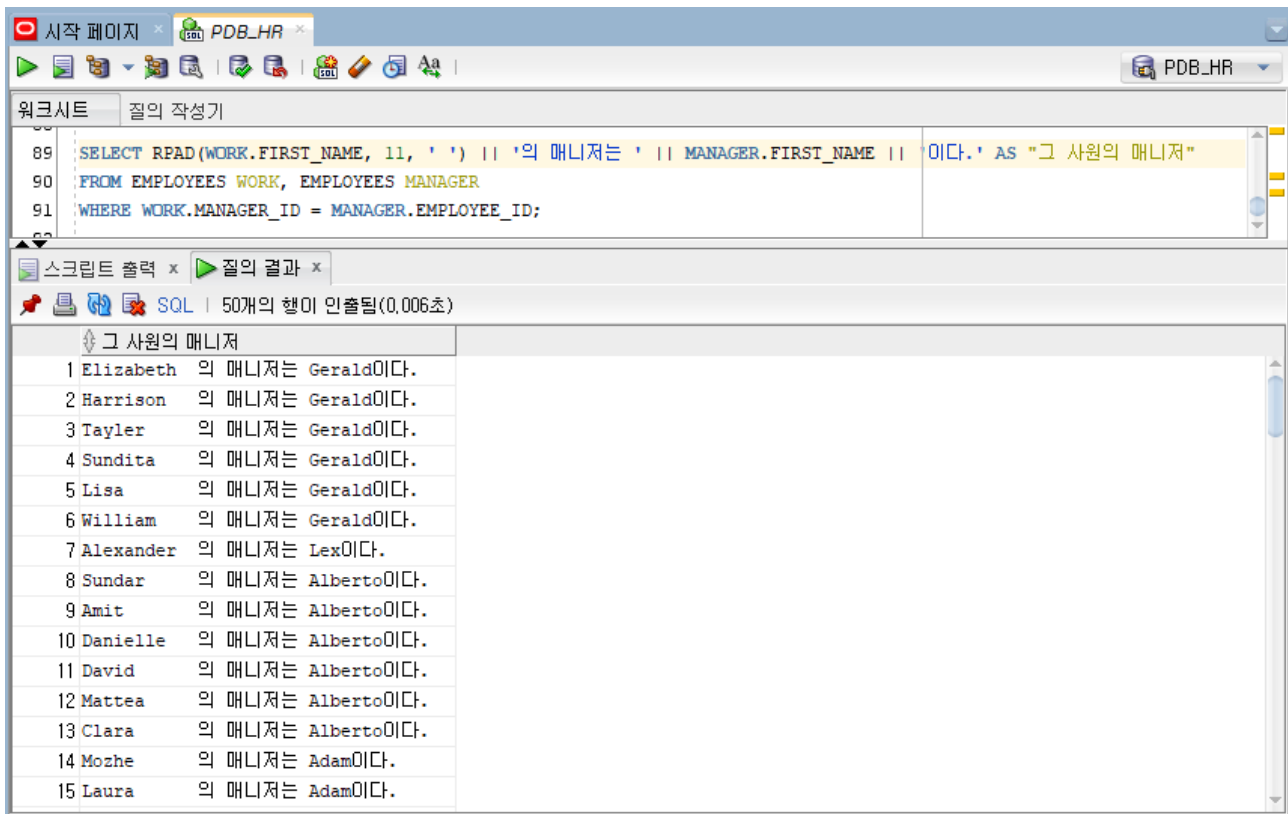
SQL | 50개의 행이 인출됨(0.004초)

	사원명	매니저명
1	Elizabeth	Gerald
2	Harrison	Gerald
3	Tayler	Gerald
4	Sundita	Gerald
5	Lisa	Gerald
6	William	Gerald
7	Alexander	Lex
8	Sundar	Alberto
9	Amit	Alberto
10	Danielle	Alberto
11	David	Alberto
12	Mattea	Alberto
13	Clara	Alberto
14	Mozhe	Adam
15	Laura	Adam
16	Alexis	Adam
17	Anthony	Adam
18	Julia	Adam
19	James	Adam
20	TJ	Adam

사원의 이름과 그의 매니저 이름을 출력하는 쿼리문

```
SELECT RPAD(WORK.FIRST_NAME, 11, ' ') || '의 매니저는 ' || MANAGER.FIRST_NAME || '이다.' AS "그 사
원의 매니저"
FROM EMPLOYEES WORK, EMPLOYEES MANAGER
```

```
WHERE WORK.MANAGER_ID = MANAGER.EMPLOYEE_ID;
```



6) ANSI Join

ANSI(미국표준연구소) SQL은 대부분의 상용 데이터베이스 시스템에서 표준 언어이다.

다른 DBMS와의 호환성을 위해서는 ANSI 조인을 사용하는 것이 좋다.

ANSI 표준 SQL 조인 구문은 몇 가지 새로운 키워드와 절을 제공하여, SELECT 문의 FROM 절에서 조인을 완벽하게 지정할 수 있다.

① ANSI Cross Join

이전에는 쉼표(,)로 테이블 명을 구분하였으나 쉼표 대신 CROSS JOIN이라고 명확하게 지정한다.

```
SELECT * FROM EMPLOYEES CROSS JOIN DEPARTMENTS;
```

② ANSI Inner Join

앞서 배운 조인 구문 중 공통 칼럼을 '=' (equal) 비교연산자를 통해 같은 값을 가지는 로우를 연결하는 형태이나 ANSI Inner Join 은 다음과 같은 형식으로 작성한다.

```
SELECT * FROM table1 INNER JOIN table2
ON table1.column1 = table2.column2
```

```
SELECT FIRST_NAME, DEPARTMENT_NAME
FROM EMPLOYEES INNER JOIN DEPARTMENTS
ON EMPLOYEES.DEPARTMENT_ID=DEPARTMENTS.DEPARTMENT_ID;
```

```

SELECT FIRST_NAME, DEPARTMENT_NAME
FROM EMPLOYEES INNER JOIN DEPARTMENTS
ON EMPLOYEES.DEPARTMENT_ID=DEPARTMENTS.DEPARTMENT_ID
WHERE FIRST_NAME='Susan';

```

	FIRST_NAME	DEPARTMENT_NAME
1	Susan	Human Resources

- USING을 이용한 조인 조건 지정

두 테이블 간의 조인 조건에 사용되는 칼럼이 같다면 ON 대신 USING을 사용할 수 있다.

```

SELECT * FROM table1 INNER JOIN table2
USING (공통칼럼)

```

```

SELECT EMPLOYEES.FIRST_NAME, DEPARTMENTS.DEPARTMENT_NAME
FROM EMPLOYEES INNER JOIN DEPARTMENTS
USING(DEPARTMENT_ID)
WHERE FIRST_NAME='Susan';

```

④ ANSI Outer Join

기존 조인에서는 반드시 모든 레코드가 출력되어야 되는 경우 '(+)' 표시를 했다.

하지만 ANSI구문의 OUTER JOIN에서는 이전에 지원하지 않았던 FULL까지 지원한다.

```

SELECT * FROM table1 [LEFT | RIGHT | FULL] OUTER JOIN table2

```

```

SELECT E.FIRST_NAME, D.DEPARTMENT_ID, D.DEPARTMENT_NAME
FROM EMPLOYEES E RIGHT OUTER JOIN DEPARTMENTS D
ON E.DEPARTMENT_ID = D.DEPARTMENT_ID;

```

	FIRST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
102	Daniel	100	Finance
103	Ismael	100	Finance
104	Nancy	100	Finance
105	William	110	Accounting
106	Shelley	110	Accounting
107	(null)	120	Treasury
108	(null)	130	Corporate Tax
109	(null)	140	Control And Credit
110	(null)	150	Shareholder Services
111	(null)	160	Benefits
112	(null)	170	Manufacturing

```

SELECT EMPLOYEE_ID, FIRST_NAME, HIRE_DATE, D.DEPARTMENT_NAME
FROM EMPLOYEES E LEFT OUTER JOIN DEPARTMENTS D
ON E.DEPARTMENT_ID = D.DEPARTMENT_ID
WHERE HIRE_DATE >= '2007/01/01' AND HIRE_DATE <= '2007/06/30';

```

[예제]

1. Sales 부서 소속 사원의 이름과 입사일을 출력하라.
2. 커미션을 받는 직원의 이름과 그가 속한 부서명을 출력하라.
3. Guy과 동일한 부서에서 근무하는 직원의 이름과 부서번호를 출력하라.