Displaying Data from Multiple Tables

Join

실습

여러 테이블로부터 데이터 추출

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
198	OConnell	50
199	Grant	50
200	Whalen	10
201	Hartstein	20
202	Fay	20
203	Mavris	40
204	Baer	70

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
30	Purchasing	1700
40	Human Resources	2400
50	Shipping	1500
60	IT	1400
70	Public Relations	2700
80	Sales	2500

EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
198	50	Shipping
199	50	Shipping
200	10	Administration
201	20	Marketing
202	20	Marketing
203	40	Human Resources
204	70	Public Relations
205	110	Accounting

Types of Joins

- CROSS join or Cartesian Product
- Equi join [inner join]
  - SELECT e.last_name, d.department_name
    FROM employees e, departments d
    WHERE e.department_id = d.department_id
  - SELECT e.last_name, d.department_name
    FROM employees e INNER JOIN departments d
    ON (e.department_id = d.department_id)
- Natural join
  - 동일한 이름을 가지는 컬럼들을 찾아서 동등한 조건으로 JOIN
  - SELECT e.last_name, d.department_name
    FROM employees e NATURAL JOIN departments d

Types of Joins

- Non-equi join
  - (between and, in, not in, is null, is not null)
- Self join
- Outer join
  - Left Outer join, Right Outer join, Full Outer join

Cartesian Product

- 곱집합을 의미
- Cartesian product 는 다음과 같은 경우에 발생한다.
  - 조인 조건을 생략한 경우
  - 조인 조건이 잘못된 경우
  - 첫 번째 테이블의 모든 행들이 두 번째 테이블의 모든 행과 조인이 되는 경우
- Cartesian product를 피하기 위해서는 WHERE 절에 항상 유효한 조인 조건을 쓰도록 한다.

Cartesian Product

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
198	OConnell	50
199	Grant	50
200	Whalen	10
201	Hartstein	20
202	Fay	20
203	Mavris	40
204	Baer	70

107 rows selected.

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
30	Purchasing	1700
40	Human Resources	2400
50	Shipping	1500
60	IT	1400
70	Public Relations	2700
80	Sales	2500

27 rows selected.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
198	Donald	OConnell	DOCONNEL	650.507.9833	21-JUN-99	SH_CLERK	2600
199	Douglas	Grant	DGRANT	650.507.9844	13-JAN-00	SH_CLERK	2600
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400
• • •							
195	Vance	Jones	VJONES	650.501.4876	17-MAR-99	SH_CLERK	2800
196	Alana	Walsh	AWALSH	650.507.9811	24-APR-98	SH_CLERK	3100
197	Kevin	Feeney	KFEENEY	650.507.9822	23-MAY-98	SH_CLERK	3000

2889 rows selected.

Joining Tables Using Oracle Syntax

```
SELECT      table1.column, table2.column
FROM        table1, table2
WHERE       table1.column1 = table2.column2;
```

- WHERE절에서 join 조건을 작성한다.
- 한 개 이상의 테이블에 같은 컬럼명이 있을 시에는 테이블 명을 컬럼명 앞에 붙여 사용한다.

```
SELECT  employees.employee_id, employees.last_name,
        employees.department_id, departments.department_id,
FROM    employees, departments
WHERE   employees.department_id = departments.department_id;
```

Using Table Aliases

- 테이블 alias를 사용함으로써 조회를 단순화할 수 있다.
- 테이블 alias를 사용함으로써 성능을 향상시킬 수 있다.

```
SELECT   e.employee_id, e.last_name, e.department_id,
         d.department_id, d.location_id
FROM     employees e, departments d
WHERE    e.department_id = d.department_id;
```

AND 연산자를 사용한 추가 검색 조건

- 테이블을 조인할 경우 조인 뿐 아니라 고려 대상인 행을 제한하기 위해 WHERE 절에 조건을 추가해야 하는 경우가 있다.
- 예를 들어 사원 Chen 의 부서번호 및 부서이름을 표시하려면 WHERE 절에 조건을 추가해야 한다.

```
SELECT   last_name, e.department_id, d.department_name
FROM     employees e, departments d
WHERE    e.department_id = d.department_id
AND      e.last_name = 'Chen' ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Chen	100	Finance

Joining More than Two Tables

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
OConnell	50
Grant	50
Whalen	10
Hartstein	20
Fay	20
Mavris	40
Baer	70

DEPARTMENT

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
30	1700
40	2400
50	1500
60	1400
70	2700

LOCATIONS

LOCATION_ID	CITY
1000	Roma
1100	Venice
1200	Tokyo
1300	Hiroshima
1400	Southlake
1500	South San Francisco
1600	South Brunswick

JOIN **JOIN**

```
SELECT   e.last_name, d.department_id, l.city
FROM     employees  e, departments  d , locations l
WHERE    e.department_id = d.department_id
AND      d.location_id = l.location_id ;
```

Creating Natural Joins

- NATURAL JOIN절은 두 테이블에서 동일한 이름을 가진 모든 열을 기준으로 한다.
- 두 개의 테이블에서 모든 컬럼을 비교하여 같은 값을 갖고 있는 행들을 조회한다.
- 같은 이름의 컬럼이지만, 다른 데이터 형식을 갖고 있다면, 에러를 반환한다.

```
SELECT  department_id, department_name, location_id, city
FROM    departments  NATURAL JOIN  locations;
```

Creating Joins with the USING Clause

- 여러 컬럼이 데이터 형식은 같지 않지만, 같은 이름을 가지고 있다면, equi join에 사용된 컬럼을 명시하기 위하여 USING절과 함께 NATURAL JOIN 절을 변형하여 사용할 수 있다.
- USING절은 한 개 이상의 컬럼이 조회될 때, 단지 한 개의 컬럼만을 선택한다.
- 참조되는 컬럼에는 테이블 명이나 alias를 사용할 수 없다.
- NATURAL JOIN과 USING절은 상호 배타적이다.

```
SELECT e.employee_id, e.last_name, d.location_id
FROM   employees  e  JOIN  departments  d
       USING (DEPARTMENT_ID)  ;
```

Creating Joins with the ON Clause

- natural join을 위한 join 조건은 같은 이름을 가진 모든 컬럼의 equi join을 기본으로 한다.
- 임의의 조건이나 조인을 위한 컬럼을 기술하기 위하여, ON절을 사용한다.
- join 조건은 outer 검색 조건으로부터 분리시킨다.
- ON 절은 코드를 이해하기 쉽게 만든다.

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id,  d.location_id
FROM employees  e  JOIN  departments  d
     ON (e.department_id = d.department_id);
```

Non-equi join

- 동등 연산자가 아닌 연산자를 포함하는 조인
- 테이블의 어떤 컬럼도 조인할 테이블과 일치하지 않는 경우

```
SELECT e.last_name, d.department_name
FROM employees e ,departments d
WHERE e.department_id=d.department_id
AND e.salary between 10000 and 20000;
```

Self Joins – 1

EMPLOYEES (WORKER)

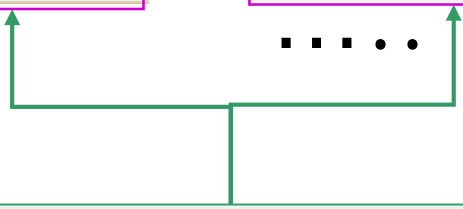
EMPLOYEE_ID	LAST_NAME	MANAGER_ID
198	OConnell	124
199	Grant	124
200	Whalen	101
201	Hartstein	100
202	Fay	201
203	Mavris	101
204	Baer	101

. . . . .

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
198	OConnell
199	Grant
200	Whalen
201	Hartstein
202	Fay
203	Mavris
204	Baer

. . . . .



WORKER 테이블의 MANAGER_ID는

MANAGER 테이블의 EMPLOYEE_ID 와 같다.

Self Joins – 2

```
SELECT worker.last_name || 'works for' || manager.last_name
FROM    employees worker, employees manager
WHERE worker.manager_id = manager.employee_id;
```

WORKER.LAST_NAME||'WORKSFOR'||MANAGER.LAST_NAME
OConnell works for Mourgos
Grant works for Mourgos
Whalen works for Kochhar
Hartstein works for King
Fay works for Hartstein
Mavris works for Kochhar
Baer works for Kochhar

Outer Joins

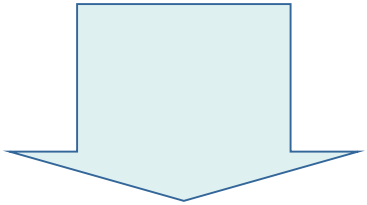
- Outer Join은 조인 조건을 만족하지 않는 행도 볼 수 있다.
- Outer Join 연산자는 (+)이다.
- 일치하는 행이 없는 테이블에서는 열이름 다음에 (+) 기호를 넣는다.

```
SELECT  table1.column, table2.column
FROM    table1, table2
WHERE  table1.column(+) = table2.column;
```

```
SELECT  e.last_name, e.department_id, d.department_name
FROM    employees e, departments d
WHERE   e.department_id(+) = d.department_id;
```

LEFT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM  employees e  LEFT OUTER JOIN  departments d
ON (e.department_id = d.department_id);
```



```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e , departments d
WHERE d.department_id = e.department_id(+) ;
```

RIGHT OUTER JOIN , FULL OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM  employees e RIGHT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

```
SELECT e.last_name, e.department_id, d.department_name
FROM  employees e FULL OUTER JOIN departments d
ON (e.department_id = d.department_id
and e.manager_id=149);
```

실습

1. 모든 사원의 이름, 부서 번호, 부서 이름을 표시하는 질의를 작성하십시오.

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mourgos	50	Shipping
Rajs	50	Shipping

2. 부서 80에 속하는 모든 업무의 고유 목록을 작성하고 출력 결과에 부서의 위치를 포함시키십시오. (중복 제거)

JOB_ID	LOCATION_ID
SA_MAN	2500
SA_REP	2500

3. 커미션을 받는 모든 사원의 이름, 부서 이름, 위치 ID 및 도시를 표시하는 질의를 작성하십시오.

LAST_NAME	DEPARTMENT_NAME	LOCATION_ID	CITY
Zlotkey	Sales	2500	Oxford
Abel	Sales	2500	Oxford

실습

4. 이름에 a(소문자)가 포함된 모든 사원의 이름과 부서 이름을 표시하는 질의를 작성하십시오.

LAST_NAME	DEPARTMENT_NAME
Whalen	Administration
Hartstein	Marketing

5. Toronto에서 근무하는 모든 사원의 이름, 업무, 부서 번호 및 부서 이름을 표시하는 질의를 작성하십시오. (join, on 키워드 사용)

LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
Hartstein	MK_MAN	20	Marketing

6. 사원의 이름 및 사원 번호를 관리자의 이름 및 관리자 번호와 함께 표시하고, 각각의 열 레이블을 Employee, Emp#, Manager, Mgr#로 지정하십시오. (관리자가 없는 사원도 포함)

Employee	EMP#	Manager	Mgr#
Kochhar	101	King	100
De Haan	102	King	100
Mourgos	124	King	100