

## 제 5장. 다중 테이블로부터 데이터 디스플레이

# 1

## ANSI/ISO SQL: 1999 표준 지원

# 개 요

몇가지 새로운 특징들이 **ANSI/ISO SQL: 1999** 표준을 지원하기 위해 **Oracle10G**에 포함되었습니다.

그 중 가장 중요한 몇가지들을 나열하면 다음과 같습니다.

- **SQL: 1999** 조인
- **CASE** 수식
- 스칼라 서브쿼리

# SQL: 1999 지원의 이점

- **Oracle SQL이 ANSI/ISO SQL: 1999** 표준을 따르게 합니다.
- 제 3자 애플리케이션이 기존의 코드에 대한 수정없이 **Oracle**로 쉽게 이전될 수 있도록 합니다.
- **Oracle**이 **ANSI/ISO** 표준 기능을 데이터베이스 내에서 제공할 수 있도록 합니다.
- 다른 데이터베이스 제품의 사용자가 **Oracle**를 보다 쉽게 배울 수 있도록 돕습니다.

# SQL: 1999 조인

**SQL: 1999** 조인 문법은 **Oracle** 조인의 그것과 다음과 같은 점에서 차이가 있습니다.

- 조인의 형태가 **FROM** 절에서 명시적으로 지정됩니다.
- 조인 조건이 **WHERE** 절의 검색 조건과 구별되어 **ON** 절에서 명시됩니다.

# SQL: 1999에 정의된 조인의 형태들

- **Cross** 조인
- **Natural** 조인
- **Using** 절
- **Full** 또는 양면 **outer** 조인
- 임의의 조인 조건

# CROSS JOIN

- CROSS JOIN은 두 테이블의 곱집합을 생성합니다.
- 이는 두 테이블 간의 **Cartesian** 곱과 같습니다.

```
SELECT last_name, department_name  
FROM employees CROSS JOIN departments;
```

LAST_NAME	DEPARTMENT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
...	
King	Marketing
Kochhar	Marketing
De Haan	Marketing
...	

# NATURAL 조인

- **NATURAL** 조인은 두 테이블에서 같은 이름을 가진 모든 컬럼에 기반합니다.
- 두 테이블의 대응되는 모든 컬럼에 대해 같은 값을 가지는 행들을 선택합니다.
- 만일 같은 이름을 가지는 컬럼들이 서로 다른 데이터 형을 가질 때에는 오류가 반환됩니다.
- 만일 **SELECT \*** 문법을 사용한다면, 공통 컬럼들은 결과 집합에서 단 한번만 나타납니다.
- 테이블 이름이나 가명 등의 수식자들은 **NATURAL** 조인에 사용된 컬럼들을 수식할 수 없습니다.



## 예: Natural JOIN

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

DEPARTMENT_ID	LOCATION_ID
20	1800
50	1500
50	1500
50	1500
50	1500

...

## USING 절을 이용한 조인

- 만일 여러 개의 컬럼이 이름은 같지만 데이터 형이 모두 일치되지 않는 때에는, **NATURAL JOIN**은 **USING** 절을 이용하여 동등 조인에 사용될 컬럼들을 명시하도록 수정될 수 있습니다.
- **USING** 절에서 참조되는 컬럼들은 **SQL** 문 어디에서도 수식자 (테이블 이름이나 가명)에 의해 수식될 수 없습니다.
- **NATURAL** 과 **USING**의 두 키워드는 상호 배타적으로 사용됩니다.

## 예: USING 절을 사용한 조인

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

EMPLOYEE_ID	LAST_NAME	LOCATION_ID
-----	-----	-----
200	Whalen	1700
201	Hartstein	1800
202	Goyal	1800
124	Mourgos	1500
141	Rajs	1500
142	Davies	1500
143	Matos	1500
...		

## ON 절을 사용하는 조인

- **Natural** 조인의 조인 조건은 기본적으로 같은 이름을 가진 모든 컬럼들에 대한 동등 조건입니다.
- 임의의 조인 조건을 지정하거나, 또는 조인할 컬럼을 명시하기 위해서 ON 절이 사용됩니다.
- ON 절은 조인 조건과 다른 조건들을 분리합니다.
- ON 절은 코드를 보다 이해하기 쉽게 합니다.

## 예: ON 절을 이용한 조인

```
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);
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Goyal	20	20	1800
124	Mourgos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500
143	Matos	50	50	1500
...				

# 복잡한 조인

**ON** 절을 사용함으로써 다음과 같은 것들을 이용한 복잡한 조인을 만들 수 있습니다.

- 서브쿼리
- **AND/OR** 연산자
- **[NOT] EXISTS**
- **[NOT] IN**

## 조인 조건과 ON 절

- ON 절을 사용함으로써 다른 조건과 조인 조건을 분리시킬 수 있습니다; 그렇게 하면 코드가 보다 이해하기 쉬워집니다.
- ON 절은 서브쿼리나 논리 연산자 등을 포함한 임의의 조건을 지정할 수 있습니다.

```
SELECT e.manager_id, e.last_name ,  
       d.department_id, d.location_id  
FROM employees e JOIN departments d  
    ON ((e.department_id = d.department_id)  
        AND  
        e.manager_id = 102  
    );
```

## 예: Exists의 사용

```
SELECT department_name, city
FROM locations l JOIN departments d
ON ((l.location_id = d.location_id)
    AND NOT EXISTS (
        SELECT 1 FROM employees e
        WHERE e.department_id = d.department_id
    )
);
```

DEPARTMENT_NAME	CITY
-----	-----
Treasury	Seattle
Corporate Tax	Seattle
Control And Credit	Seattle
...	
Retail Sales	Seattle
Recruiting	Seattle
Payroll	Seattle



## 복수 테이블의 조인

```
SELECT employee_id, city, department_name
FROM locations l
JOIN departments d
    ON (l.location_id = d.location_id)
JOIN employees e
    ON (d.department_id = e.department_id);
```

EMPLOYEE_ID	CITY	DEPARTMENT_NAME
-----	-----	-----
100	Seattle	Executive
101	Seattle	Executive
...		
106	Southlake	IT
107	Southlake	IT
108	Seattle	Finance
...		

## INNER 대 OUTER 조인

- **SQL: 1999** 표준에 의하면 두 테이블을 조인하여 오로지 대응되는 행들만을 반환하는 것을 **INNER** 조인이라 합니다.
- **INNER** 조인의 결과와 함께 왼쪽(오른쪽) 테이블의 대응되지 않는 행들도 반환하는 것을 **LEFT (RIGHT) OUTER** 조인이라 합니다.
- **INNER** 조인의 결과와 함께 **LEFT** 및 **RIGHT OUTER** 조인의 결과까지 모두 반환하는 것을 **FULL OUTER** 조인이라 합니다.

## 예: LEFT OUTER JOIN

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

LAST_NAME	DEPARTMENT_NAME
Faviet	Finance
Greenberg	Finance
Gietz	Accounting
Higgins	Accounting
Grant	
...	

## 예: RIGHT OUTER JOIN

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

LAST_NAME	DEPARTMENT_NAME
...	...
Baer	Public Relations
Higgins	Accounting
Gietz	Accounting
	NOC
	Manufacturing
	Construction
	Control And Credit
...	

## 예: FULL OUTER 조인

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

LAST_NAME	DEPARTMENT_NAME
-----------	-----------------

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

...	
-----	--

Greenberg	Finance
-----------	---------

Gietz	Accounting
-------	------------

Higgins	Accounting
---------	------------

Grant	
-------	--

--	--

	NOC
--	-----

	Manufacturing
--	---------------

	Construction
--	--------------

...	
-----	--

## 제 5장. 다중 테이블로부터 데이터 디스플레이

- 1) 목 적
- 2) 다중 테이블로부터 데이터 획득
- 3) **JOIN**이란?
- 4) **JOIN**의 유형
- 5) **Equijoin**(자연 조인, 내부조인)
- 6) **Non-Equijoin**
- 7) **Outer Join**
- 8) **Self Join**
- 9) 요 약

# 1) 목 적

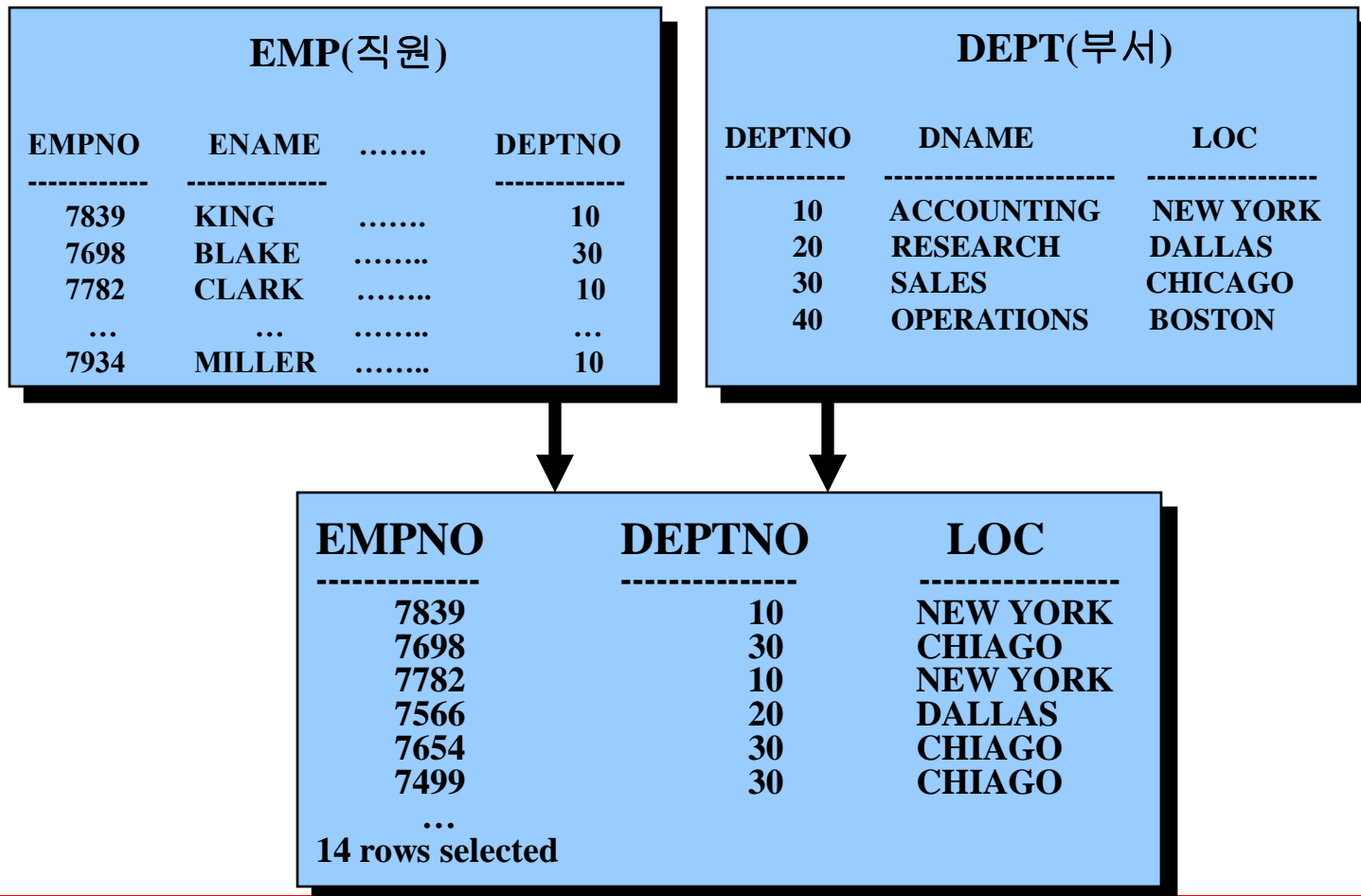
- 이용 가능한 각기 다른 방법들을 사용하여(여러 가지 JOIN의 유형에 따라), 하나 이상의 테이블로부터 데이터를 구하는 방법을 다룸.

□ **equality Join**과 **nonequality Join**을 사용하여,  
하나 이상의 테이블로부터 데이터를 액세스하는  
**SELECT** 문장을 작성함.

□ **Outer Join**을 사용하여  
일반적으로 조인 조건을 만족하지 않는 데이터를 출력 함.

□ 자체적으로 테이블을 Join함.

## 2) 다중 테이블로부터 데이터 획득





### 3) JOIN 이란?

- 하나 이상의 테이블로부터 데이터를 질의하기 위해서 JOIN을 사용함.
- 테이블의 행은  
관련되는 열의 공통 값(**Primary Key**와 **Foreign Key**)열에 따라서  
다른 하나의 테이블의 행과 Join할 수 있음.

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

- WHERE절에 조인 조건(**Join Condition**)을 작성함.
- 하나 이상의 테이블에 똑같은 열 이름이 있을 때,  
열 이름 앞에 테이블 이름을 붙임.

## 4) JOIN의 유형

❖ **Equijoin**



❖ **Non-equijoin**



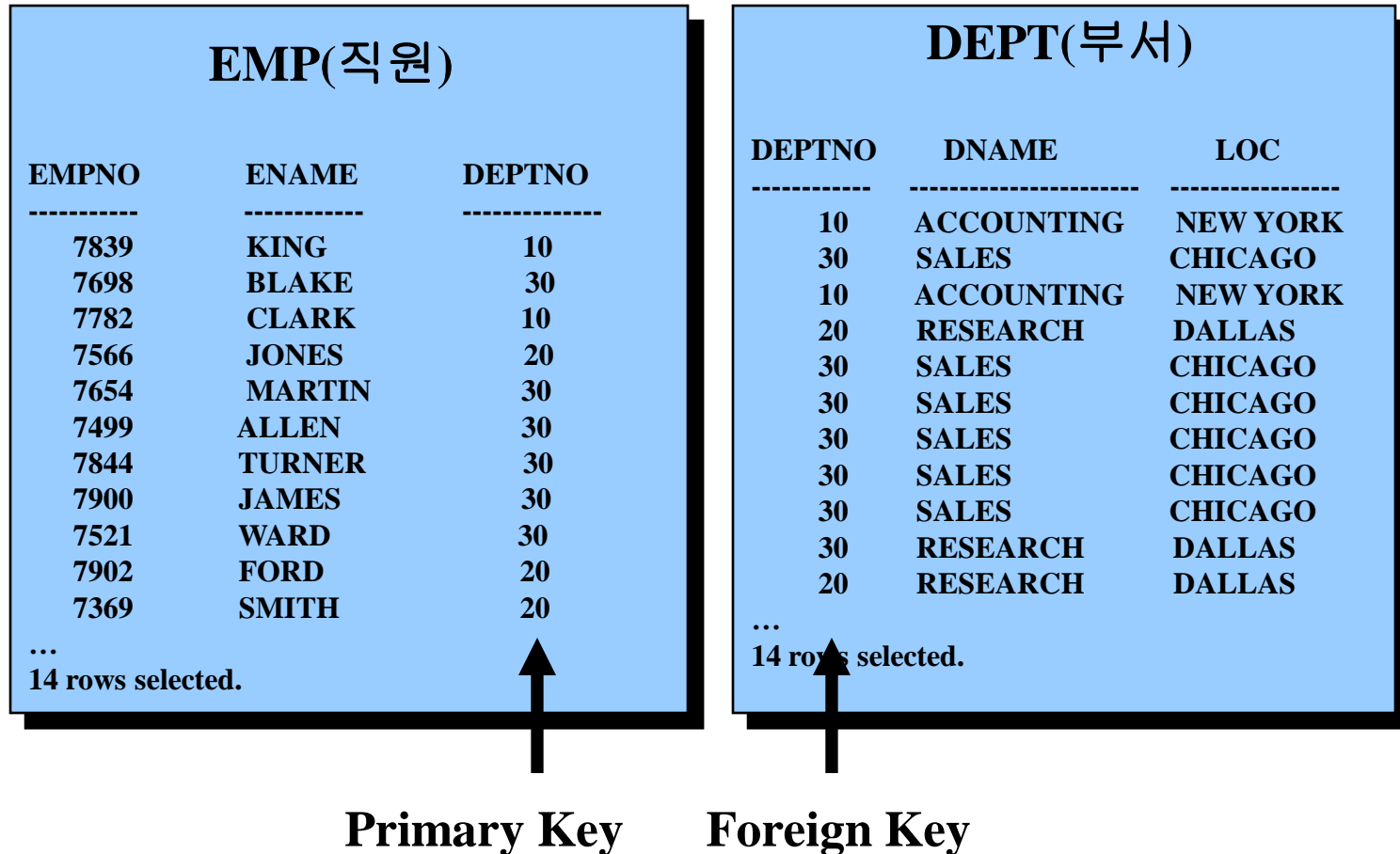
❖ **Outer join**



❖ **Self join**



## 5) Equijoin



## 5-1) Equijoin으로 레코드 검색

```
SELECT emp.empno, emp.ename emp.deptno dept.deptno, dept.loc  
FROM    emp join dept  
on emp.deptno = dept.deptno;
```

<b>EMPNO</b>	<b>ENAME</b>	<b>DEPTNO</b>	<b>DEPTNO</b>	<b>LOC</b>
-----	-----	-----	-----	-----
<b>7839</b>	<b>KING</b>	<b>10</b>	<b>10</b>	<b>NEW YORK</b>
<b>7698</b>	<b>BLAKE</b>	<b>30</b>	<b>30</b>	<b>CHICAGO</b>
<b>7782</b>	<b>CLARK</b>	<b>10</b>	<b>10</b>	<b>NEW YORK</b>
<b>7566</b>	<b>JONES</b>	<b>20</b>	<b>20</b>	<b>DALLAS</b>

...

**14 rows selected**

## 5-2) 모호한 열 이름 선택

- 여러 테이블에 있는 동일한 열 이름을 선택하기 위해서 테이블 이름을 접두사로 사용함.
- 테이블명 접두사를 사용하여 성능을 향상시킴.
- 동일한 이름을 가지고 있지만, 다른 테이블에 있는 열들은 열 별칭(Column alias)를 사용하여 구분함.

## 5-3) AND 연산자를 사용하는 추가적인 검색 조건

**EMP(직원)**

EMPNO	ENAME	DEPTNO
7839	KING	10
7698	BLAKE	30
7782	CLARK	10
7566	JONES	20
7654	MARTIN	30
7499	ALLEN	30
7844	TURNER	30
7900	JAMES	30
7521	WARD	30
7902	FORD	20
7369	SMITH	20

...  
14 rows selected.

**DEPT(부서)**

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
30	SALES	CHICAGO
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	RESEARCH	DALLAS
20	RESEARCH	DALLAS

...  
14 rows selected.

```
SQL> SELECT empno, ename, emp.deptno, loc
2> FROM emp join dept
3> ON emp.deptno = dept.deptno AND INITCAP(ename) = 'KING';
```

## 5-4) 테이블 별칭(Alias) 사용

```
SQL> SELECT emp.empno, emp.ename emp.deptno dept.deptno, dept.loc  
2> FROM    emp join dept  
3> ON      emp.deptno = dept.deptno;
```

```
SQL> SELECT e.empno, e.ename, e.deptno, d.deptno, d.loc  
2> FORM    emp e join dept d  
3> ON      e.deptno = d.deptno;
```

## 5-5) 두개 이상의 테이블을 JOIN

CUSTOMER	
NAME	CUSTID
-----	-----
JOCKSPORTS	100
TKB SPORT SHOP	101
VOLLYRITE	102
JUST TENNIS	103
K+T SPORTS	105
SHAPE UP	106
WOMENS SPORTS	107
...	...
9 rows selected.	

ORD	
CUSTID	ORDID
-----	-----
101	610
102	611
104	612
106	601
102	602
106	610
106	610
...	...
21 rows selected.	

ITEM	
ORDID	ITEMID
-----	-----
610	3
611	1
612	1
601	1
602	1
...	...
64 rows selected.	



## 6) Non-Equijoin

EMP(직원)		
EMPNO	ENAME	SAL
-----	-----	-----
7839	KING	5000
7698	BLAKE	2850
7782	CLARK	2450
7566	JONES	2975
7654	MARTIN	1250
7499	ALLEN	1600
7844	TURNER	1500
7900	JAMES	950
...		
14 rows selected.		

SALGRADE(급여)		
GRADE	LOSAL	HISAL
-----	-----	-----
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

“EMP 테이블의 SAL은  
SALGRADE 테이블내의  
최저월급(LOSAL)과 최고 월급(HISAL)  
사이의 값이다”

## 6-1) Non-Equijoin으로 레코드 검색

```
SQL> SELECT e.ename, e.sal, s.grade  
2> FROM   emp e join salgrade s  
3> ON e.sal BETWEEN s.losal AND s.hisal;
```


ENAME	SAL	GRADE
JAMES	950	1
SMITH	800	1
ADAMS	1100	1
...		

14 rows selected.

## 7) Outer Join

EMP(직원)	
ENAME	DEPTNO
-----	-----
KING	10
BLAKE	30
CLARK	10
JONES	20
...	

DEPT(부서)	
DEPTNO	DNAME
-----	-----
10	ACCOUNTING
30	SALES
10	ACCOUNTING
20	RESEARCH
40	OPERATIONS
...	
14 rows selected.	



OPERATIONS부서 내에 어떠한 직원도 없다.

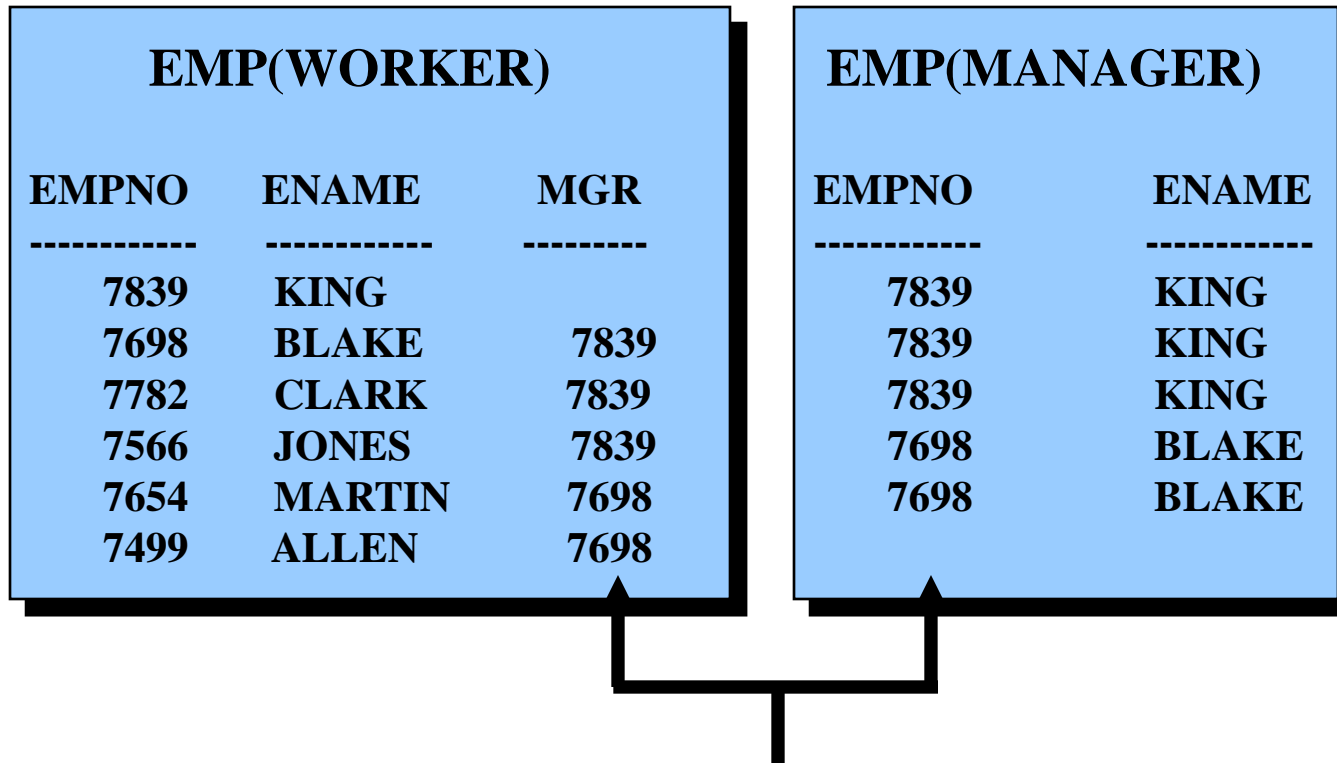
## 7-2) Outer Join의 사용

```
SQL> SELECT e.ename, d.deptno, d.dname  
2> FROM emp e right outer join dept d  
3> ON e.deptno = d.deptno ORDER BY e.deptno;
```

ENAME	DEPTNO	DNAME
KING	10	ACCOUNTING
CLARK	10	ACCOUNTING
...		
	40	OPERATIONS

15 rows selected.

## 8) Self Join



“WORKER 테이블에 있는 MGR은  
MANAGER 테이블에 있는 EMPNO와 동일 함.”

## 참고1)) 자체적으로 테이블 조인

```
SQL> SELECT worker.ename || ' works for ' || manager.ename  
2> FROM    emp worker join emp manager  
3> ON      worker.mgr = manager.empno;
```

```
WORKER.ENAME || 'WORKS FOR' || MANAG
```

---

```
BLAKE   works for KING  
CLARK   works for KING  
JONES   works for KING  
MARTIN  works for BLAKE  
...  
13 rows selected.
```

## 3-1) Cartesian Product

### 1) Cartesian Product 발생

- JOIN 조건이 생략된 경우
- JOIN 조건이 잘못된 경우
- 첫 번째 테이블의 모든 행이  
두 번째 테이블의 모든행과 JOIN되는 경우

### 2) Cartesian Product 발생 피하기

- 항상 WHERE절에 올바른 JOIN 조건문을 사용할 것.

## 3-2) Cartesian Product 생성

EMP(직원) – 14 Rows			
EMPNO	ENAME	.....	DEPTNO
7839	KING	.....	10
7698	BLAKE	.....	30
7782	CLARK	.....	10
...	...	.....	...
7934	MILLER	.....	10

DEPT(부서) – 4 Rows		
DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

“Cartesian Product :  
14\*4 = 56 rows”

ENAME	DNAME
-----	-----
KING	ACCOUNTING
BLAKE	ACCOUNTING
...	
KING	RESEARCH
BLAKE	RESEARCH
...	
56 rows selected.	



## 9) 요약

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

- 1) Equijoin    2) Non-equijoin    3) Outer Join    4) Self Join

