ORACLE

Introduction to Oracle9i
: SQL

DBGUIDE.net
데이터베이스 가이드 네트워크

ORACLE

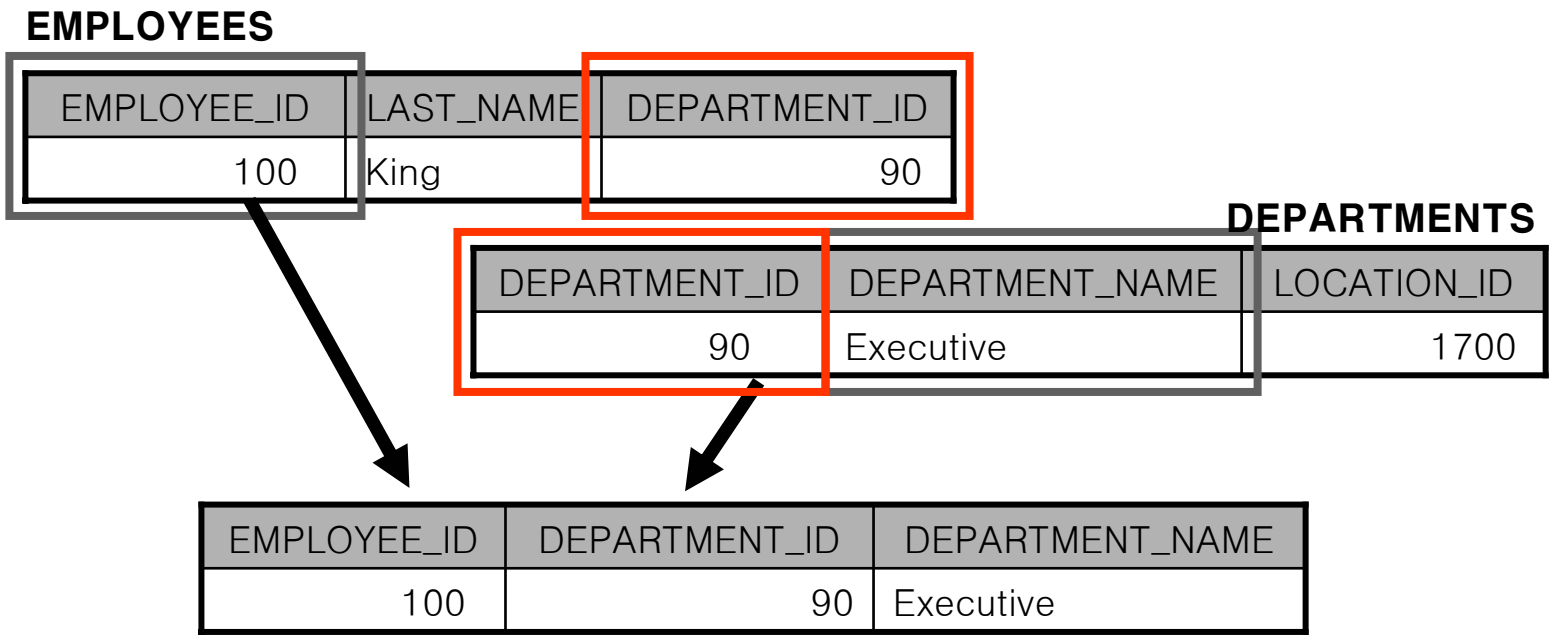
Chapter4.
JOIN문 작성

- JOIN문의 다양한 유형을 이해한다.
- 오라클에서의 JOIN문법과 표준 JOIN문법의 차이점을 이해한다.

1. JOIN의 정의

(1) JOIN의 정의

- 연관이 있는 여러 테이블로부터 데이터를 검색하는 방법



2. JOIN의 유형

- 오라클에서는 오라클 자체 문법을 사용한 조인 이외에 표준에서 정의하는 조인기법을 동시에 적용 가능

오라클	표준
EQUIJOIN	CROSS JOINS
NON-EQUIJOIN	NATURAL JOINS
OUTER JOIN	USING 절
SELF JOIN	OUTER JOINS

2. 오라클 JOIN 구문

(1) Equijoin

- Join 하고자 하는 테이블 양쪽에 같은 특성을 갖는 컬럼이 있는 경우 수행하는 조인

EMPLOYEES

EMPLOYEE_ID	DEPARTMENT_ID
100	90

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
90	Executive

2. 오라클 JOIN 구문

(1) Equijoin

1) Equijoin을 통한 검색

• 어느 직원이 어떤 부서에서 근무하는 지 검색하는 문장

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

2. 오라클 JOIN 구문

(1) Equijoin

2) 테이블 별칭을 사용한 검색

• 어느 직원이 어떤 부서에서 근무하는 지 검색하는 문장

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

2. 오라클 JOIN 구문

(1) Equijoin

3) 조인 조건 외의 검색 조건을 추가할 경우

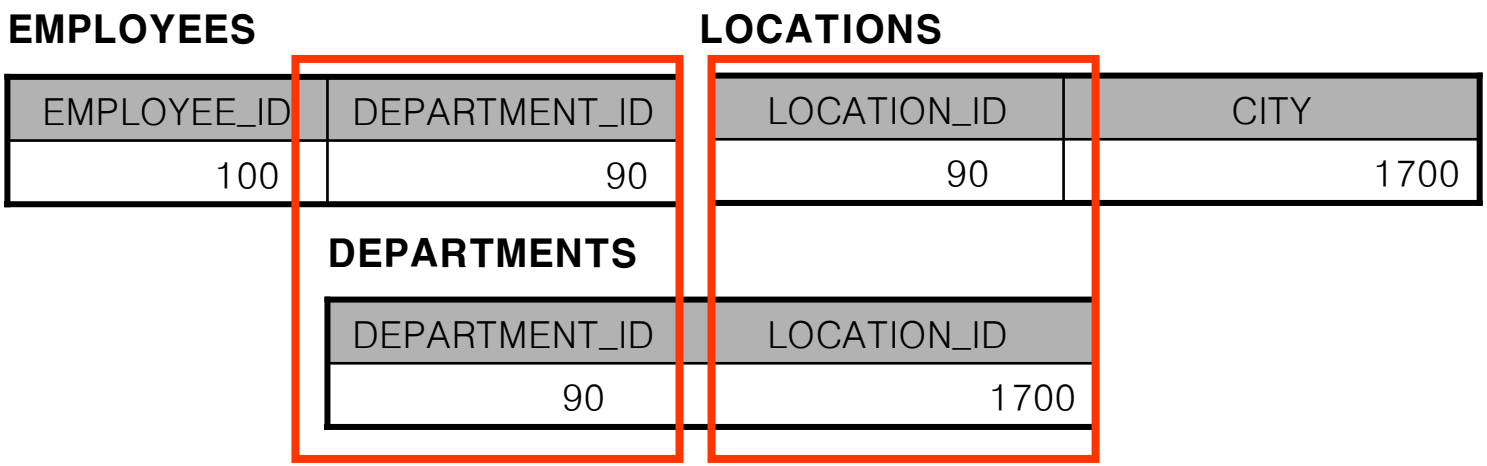
- 이름이 King인 직원이 어떤 부서에서 근무하는 지 검색

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

2. 오라클 JOIN 구문

(1) Equijoin

4) 3개 이상의 테이블을 조인할 경우



- N개의 테이블을 조인할 경우 최소 N-1개의 조인조건 필요

2. 오라클 JOIN 구문

(1) Equijoin

4) 3개 이상의 테이블을 조인할 경우

· 어느 직원이 어떤 부서, 어떤 도시에서 근무하는 지 검색하는 문장

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

2. 오라클 JOIN 구문

(2) Non-Equijoin

- Join 하고자 하는 테이블 양쪽에 같은 특성을 갖는 컬럼이 없는 경우 수행하는 조인

EMPLOYEES

LAST_NAME	SALARY
King	24000

JOB_GRADES

GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

2. 오라클 JOIN 구문

(2) Non-Equijoin

1) Non-Equijoin을 통한 검색

• 직원들의 봉급 레벨을 검색하는 문장

```
SELECT  e.last_name, e.salary, j.grade_level
FROM    employees e, job_grades j
WHERE   e.salary BETWEEN j.lowest_sal AND j.highest_sal ;
```

2. 오라클 JOIN 구문

(3) Outer Join

- Join 조건에 매칭되지 않는 값도 출력하고자 할 때 수행하는 조인
- (+) 기호를 사용하여 수행
- (+) 기호를 양쪽조건에 모두 사용하는 것은 불가

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
King	90
Kochhar	90
Hunold	60
…	…
Zlotket	80

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
50	Shipping
…	
190	Contracting

190번 부서에 근무하는 직원이 없다

2. 오라클 JOIN 구문

(3) Outer Join

1) Outer Join을 통한 검색

- 어느 직원이 어떤 부서에서 근무하는 지 검색하되 부서를 배정받지 않은 직원도 출력하는 문장

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

- 어느 직원이 어떤 부서에서 근무하는 지 검색하되 직원이 배정되지 않은 부서도 출력하는 문장

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

2. 오라클 JOIN 구문

(4) Self Join

- 하나의 테이블을 두개인 것처럼 두고 수행하는 조인
- 반드시 테이블의 별칭을 사용

EMPLOYEES(E)

EMPLOYEE_ID	MANAGER_ID
100	
101	100
102	100

EMPLOYEES(M)

EMPLOYEE_ID	LAST_NAME
100	King

2. 오라클 JOIN 구문

(4) Self Join

1) Self Join을 통한 검색

• 전체 직원 이름 및 직원의 관리자 이름을 출력하는 문장

```
SELECT  e.last_name, m.last_name
FROM    employees e, employees m
WHERE   e.manager_id = m.employee_id ;
```

3. 표준 JOIN 구문

(1) Cross Join

- 두 테이블의 값을 무조건 조인

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

3. 표준 JOIN 구문

(2) Natural Join (≒Equi Join)

- Join 하고자 하는 테이블 양쪽에 같은 특성을 갖는 컬럼이 있고 그 컬럼의 이름이 같은 경우 수행하는 조인

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

- NATURAL JOIN 대신 JOIN으로 표기해도 무관
- 비교하고자 하는 컬럼의 이름이 같더라도 데이터 타입이 다르면 JOIN 수행 불가능

3. 표준 JOIN 구문

(2) Natural Join (≒Equi Join)

1) USING절 사용

- Join 하고자 하는 테이블 양쪽에 같은 이름을 갖는 컬럼이 여러 개일 경우 수행하는 조인

```
SELECT    last_name, department_name
FROM      employees  JOIN departments
USING     (department_id) ;
```

- USING절의 컬럼 이름 앞에 테이블 명을 붙이지 말 것
- 비교하고자 하는 컬럼의 이름이 같더라도 데이터 타입이 다르면 JOIN 수행 불가능

3. 표준 JOIN 구문

(2) Natural Join (≒Equi Join)

2) ON절 사용

- Join 하고자 하는 테이블 양쪽에 같은 특성을 갖는 컬럼이 있지만 컬럼의 이름이 다른 경우 수행하는 조인

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

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

3. 표준 JOIN 구문

(3) Outer Join

1) 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 ) ;
```

2) RIGHT 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 ) ;
```

3. 표준 JOIN 구문

(3) Outer Join

3) FULL OUTER JOIN

```
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 ) ;
```

3. 표준 JOIN 구문

(4) Join 조건 외의 조건 추가

3) FULL OUTER JOIN

```
SELECT    e.last_name, e.department_id, d.department_name
FROM      employees e  JOIN  departments d
ON        (e.department_id = d.department_id )
AND       e.employee_id = 101 ;
```