

Introduction to Oracle9i : SQL

Chapter13. 고급 서브쿼리

- ▶ 다중열 서브쿼리를 이해한다.
- ▶ 연쇄 서브쿼리를 사용하여 해결할 수 있는 다양한 상황을 파악한다.

1. 서브쿼리

- 단독 쿼리문만으로는 해결할 수 없는 검색 수행 시 사용

Kochhar보다 많은 봉급을 받는 직원의 이름은?

Main Query

Kochhar보다 많은 봉급을 받는 직원의 이름은?

Subquery


Kochhar의 봉급은?

2. 서브쿼리의 구문

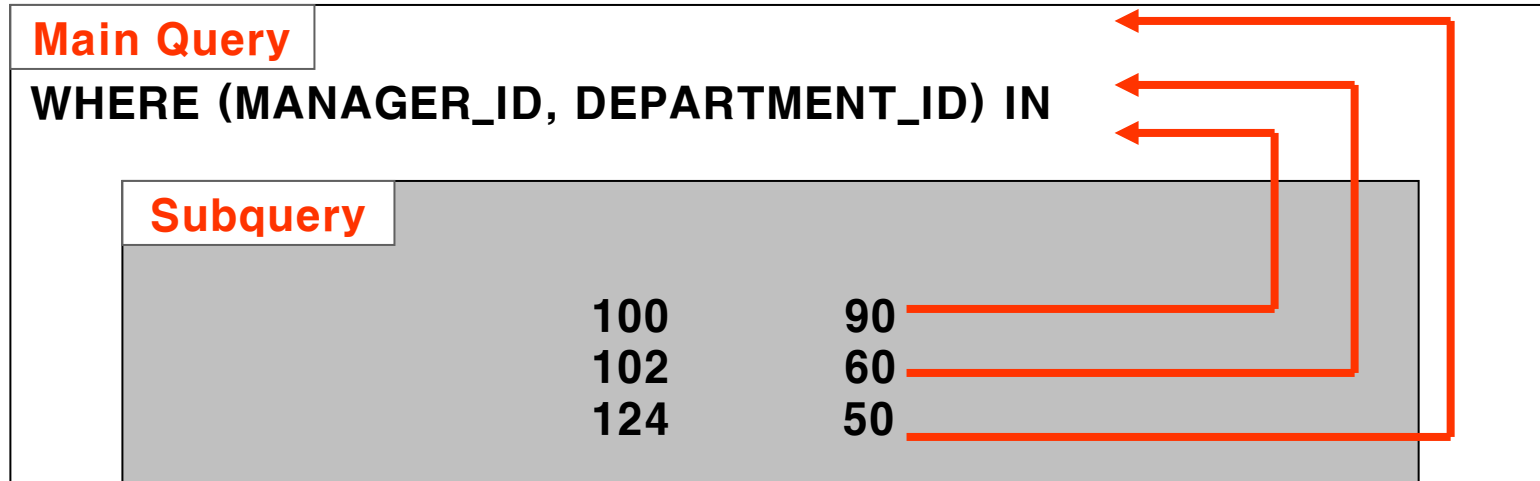
```
SELECT 컬럼명  
FROM 테이블명  
WHERE 표현식 연산자  
                (SELECT 컬럼명  
                  FROM 테이블명) ;
```

- 서브쿼리가 메인쿼리보다 먼저 수행

3. 서브쿼리의 사용

```
SELECT last_name  
FROM employees  
WHERE salary >   
    (SELECT salary  
     FROM employees  
     WHERE last_name = 'Kochhar') ;
```

4. 다중열 서브쿼리



4. 다중열 서브쿼리

(1) 다중열 서브쿼리의 컬럼 비교

- Pairwise 비교
: WHERE절에 제시된 컬럼값을 동시에 만족
- Nonpairwise 비교
: WHERE절에 제시된 컬럼값을 별도로 만족

4. 다중열 서브쿼리

(1) 다중열 서브쿼리의 컬럼 비교

1) Pairwise 비교

174, 178번과 같은 관리자 번호 및 부서번호를 동시에 만족하는
사원들의 정보 검색

```
SELECT employee_id, manager_id, department_id
FROM employees
WHERE (manager_id, department_id) IN
      (SELECT manager_id, department_id
       FROM employees
       WHERE employee_id IN (174, 178))
AND employee_id NOT IN (174, 178) ;
```


4. 다중열 서브쿼리

(1) 다중열 서브쿼리의 컬럼 비교

1) Nonpairwise 비교

174 또는 178번과 같은 관리자 번호, 같은 부서번호를 갖는 직원들의 정보 검색

```
SELECT employee_id, manager_id, department_id
FROM employees
WHERE manager_id IN
      (SELECT manager_id
       FROM employees
       WHERE employee_id IN (174, 178))
AND department_id IN
      (SELECT department_id
       FROM employees
       WHERE employee_id IN (174, 178))
AND employee_id NOT IN (174, 178) ;
```

5. 스칼라 서브쿼리

(1) 스칼라 서브쿼리의 정의

- 정확하게 한 컬럼 한 행의 결과를 리턴하는 서브쿼리
- SELECT문의 GROUP BY 절을 제외한 모든 절에 사용 가능
- DECODE 및 CASE 표현식에서 사용 가능

5. 스칼라 서브쿼리

(2) 스칼라 서브쿼리의 사용

1) CASE 표현식

```
SELECT  employee_id, last_name,  
        (CASE  
          WHEN department_id =  
                (SELECT  department_id  
                   FROM    departments  
                   WHERE    location_id = 1800 )  
          THEN 'Canada' ELSE 'USA' END ) location  
FROM    employees ;
```

5. 스칼라 서브쿼리

(2) 스칼라 서브쿼리의 사용

2) ORDER BY 절

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

6. 연쇄 서브쿼리

(1) 연쇄 서브쿼리의 구문

```
SELECT 컬럼명1, 컬럼명2, ...  
FROM 테이블명 테이블별칭  
WHERE 컬럼명 연산자  
          (SELECT 컬럼명1, 컬럼명2, ...  
            FROM 테이블명  
            WHERE 표현식 = 테이블별칭.표현식) ;
```

6. 연쇄 서브쿼리

(2) 연쇄 서브쿼리의 사용

소속된 부서의 평균 봉급보다 많은 봉급을 받는 직원들의 정보 검색

```
SELECT last_name, salary, department_id
FROM   employees outer
WHERE  salary >
        ( SELECT AVG (salary)
          FROM   employees
          WHERE  department_id = outer.department_id) ;
```

6. 연쇄 서브쿼리

(2) 연쇄 서브쿼리의 사용

최소 2회 이상 직종을 변경한 경험이 있는 직원들의 정보 검색

```
SELECT  e.employee_id, last_name, e.job_id
FROM    employees e
WHERE   2 <=
        ( SELECT COUNT (*)
          FROM    job_history
          WHERE   employee_id = e.employee_id ) ;
```

6. 연쇄 서브쿼리

(3) 연쇄 UPDATE

1) 구문

```
UPDATE   테이블명 테이블별칭
SET      컬럼명 =
          (SELECT   표현식
           FROM      테이블명
           WHERE     컬럼명 = 테이블별칭.컬럼명) ;
```


6. 연쇄 서브쿼리

(3) 연쇄 UPDATE

2) 사용

EMPLOYEES 테이블에 부서명을 저장할 컬럼을 추가한 후 각 사원에게 적합한 부서명을 UPDATE

```
ALTER TABLE employees  
ADD ( department_name VARCHAR2(15) );
```

```
UPDATE employees e  
SET department_name =  
    ( SELECT department_name  
      FROM departments d  
      WHERE e.department_id = d.department_id );
```

6. 연쇄 서브쿼리

(4) 연쇄 DELETE

1) 구문

```
DELETE FROM 테이블명 테이블별칭
WHERE 컬럼명 연산자
      (SELECT 표현식
        FROM 테이블명
        WHERE 컬럼명 = 테이블별칭.컬럼명) ;
```

6. 연쇄 서브쿼리

(3) 연쇄 UPDATE

2) 사용

EMPLOYEES 테이블에도 존재하고 EMP_HISTORY 테이블에도 존재하는 사원에 대한 정보를 EMPLOYEES 테이블로부터 삭제

```
DELETE FROM employees e
SET      employee_id =
          ( SELECT employee_id
            FROM    emp_history
            WHERE   employee_id = e.employee_id) ;
```

7. WITH 절

(1) 정의 및 특징

- 복잡한 서브쿼리를 포함하고 있는 SQL문을 보기 편하고 사용하기 편하게 하는 기능을 제공
- WITH 절을 사용해서 수행한 서브쿼리의 결과를 사용자 임시 테이블스페이스에 저장
- 일반적으로 성능이 향상

7. WITH 절

(2) 사용

```
WITH
dept_costs AS (
    SELECT d.department_name, SUM(e.salary) AS dept_total
    FROM   employees e, departments d
    WHERE  e.department_id = d.department_id
    GROUP BY d.department_name) ,
avg_cost AS (
    SELECT SUM(dept_total) / COUNT(*) AS dept_avg
    FROM   dept_costs )
SELECT *
FROM   dept_costs
WHERE  dept_total >
      (SELECT dept_avg
      FROM avg_cost )
ORDER BY department_name ;
```