Chapter 18. 서브쿼리 II

이번 장에서는 앞서 설명한 기본적인 서브쿼리의 사용방법에 이어 서브쿼리의 고급 활용 기법인 복수 컬럼 서브쿼리, 스칼라 서브쿼리, 상관관계 서브쿼리와 Oracle 9i부터 추가된 WITH 구문에 대하여 설명한다.

복수 컬럼 서브쿼리

복수 컬럼 서브쿼리란 서브쿼리에서 리턴되는 행들이 두개 이상의 컬럼을 가지는 경우이다. 즉, 다음과 같은 형식의 서브쿼리를 복수 컬럼 서브쿼리라고 한다.

```
SELECT column, column, ...
FROM table
WHERE (column, column, ... ) IN
                              (SELECT column, column, ...
                               FROM table
                               WHERE condition);
```

복수 컬럼 서브쿼리는 비교조건의 형태의 따라 Pairwise 비교와 Nonpairwise 비교로 구분된다. Pairwise 비교조건은 다음과 같이 사원 테이블에서 사번이 7369 또는 7499번인 직원과 직급 및 부서코드가 동일한 사원을 검색하는 경우이다.

```
SQL> SELECT EMPNO, JOB, DEPTNO
  2  FROM EMP
  3  WHERE (JOB, DEPTNO) IN
  4                       (SELECT JOB, DEPTNO
  5                        FROM EMP
  6                        WHERE EMPNO IN (7369, 7499))
  7  AND EMPNO NOT IN (7369, 7499);

     EMPNO JOB           DEPTNO
---------- --------- ----------
      7876 CLERK             20
      7521 SALESMAN          30
      7654 SALESMAN          30
      7844 SALESMAN          30
```

반면, Nonpairwise 비교조건은 다음과 같이 사원 테이블에서 사번이 7369 또는 7499번인 직원과 관리자사번이 같거나 직급이 같은 사원을 검색하는 경우이다.

```
SQL> SELECT EMPNO, JOB, DEPTNO
  2  FROM EMP
  3  WHERE JOB IN
  4             (SELECT JOB
  5              FROM EMP
  6              WHERE EMPNO IN (7369, 7499))
  7  AND DEPTNO IN
  8             (SELECT DEPTNO
  9              FROM EMP
 10              WHERE EMPNO IN (7369, 7499))
 11  AND EMPNO NOT IN (7369, 7499);

     EMPNO JOB           DEPTNO
---------- --------- ----------
      7876 CLERK             20
      7900 CLERK             30
      7521 SALESMAN          30
      7654 SALESMAN          30
      7844 SALESMAN          30
```

스칼라 서브쿼리

서브쿼리가 1개의 행에 1개의 컬럼값을 리턴하는 경우를 스칼라 서브쿼리라고 한다. 스칼라 서브쿼리는 Oracle 8i부터 지원되었지만 SELECT 문장과 INSERT 문장의 VALUES 구문에서와 같이 제한된 경우에서만 사용 할 수 있었다. 그러나 Oracle 9i부터는 스칼라 서브쿼리의 적용 범위가 훨씬 확대 되었으며, 다음과 같은 경우에도 사용 할 수 있게 되었다.

· DECODE와 CASE 함수내의 조건식 또는 연산식
· GROUP BY를 제외한 SELECT 문장의 모든 구문
· UPDATE 문장에서 SET 및 WHERE 구문에서 연산자의 좌변

반면, 다음과 같은 경우에는 스칼라 서브쿼리를 사용 할 수 없다.
· 컬럼에 대한 디폴트 값
· DML 명령의 RETURNING 구문
· 함수기반 인덱스
· GROUP BY 절, CHECK 제약조건, WHEN 조건
· HAVING 절
· START WITH, CONNECT BY 절
· CREATE PROFILE과 같이 쿼리와 관련 없는 문장

스칼라 서브쿼리를 CASE 함수에 사용하면 다음과 같다.

```
SQL> SELECT EMPNO, ENAME,
  2  (CASE
  3   WHEN DEPTNO = (SELECT DEPTNO FROM DEPT
  4                  WHERE LOC = 'CHICAGO')
  5   THEN '뉴욕' ELSE '기타' END) LOCATION
  6  FROM EMP;

     EMPNO ENAME      LOCA
---------- ---------- ----
      7369 SMITH      기타
      7499 ALLEN      뉴욕
      7521 WARD       뉴욕
      7566 JONES      기타
      7654 MARTIN     뉴욕
      7698 BLAKE      뉴욕
      7782 CLARK      기타
      7788 SCOTT      기타
      7839 KING       기타
      7844 TURNER     뉴욕
      7876 ADAMS      기타
      7900 JAMES      뉴욕
      7902 FORD       기타
      7934 MILLER     기타

14 개의 행이 선택되었습니다.
```

스칼라 서브쿼리를 ORDER BY 구문에 사용하면 다음과 같다.

```
SQL> SELECT EMPNO, ENAME
  2  FROM EMP E
  3  ORDER BY (SELECT DNAME
  4            FROM DEPT D
  5            WHERE E.DEPTNO = D.DEPTNO);

     EMPNO ENAME
---------- ----------
      7782 CLARK
      7839 KING
      7934 MILLER
      7369 SMITH
      7876 ADAMS
      7902 FORD
      7788 SCOTT
      7566 JONES
      7499 ALLEN
      7698 BLAKE
      7654 MARTIN
      7900 JAMES
      7844 TURNER
      7521 WARD

14 개의 행이 선택되었습니다.
```

상관관계 서브쿼리

상관관계 서브쿼리는 서브쿼리가 메인쿼리의 컬럼을 참조하도록 되어 있는 서브쿼리로서 단

독으로 실행이 불가능하기 때문에 메인쿼리에서 조건에 맞는 행을 차례로 서브쿼리에 넘겨주면 서브쿼리에서 참조하는 메인쿼리의 컬럼이 상수로 교체되어 서브쿼리가 실행되고 그 결과가 메인쿼리에 전달되어 처리된다. 즉, 상관관계 서브쿼리는 메인쿼리가 넘겨주는 행의 개수 만큼 반복 실행되게 된다.

상관관계 서브쿼리의 작성 문법은 다음과 같다.

```
SELECT column1, column2, ...
FROM table1 outer
WHERE column1 operator
              (SELECT column1, column2
               FROM table2
               WHERE expr1 = outer.expr2);
```

다음은 사원 테이블에서 자기 부서의 평균 급여보다 많은 급여를 받는 사원을 검색하기 위해 상관관계 서브쿼리를 사용한 것이다.

```
SQL> SELECT ENAME, SAL
  2  FROM EMP E
  3  WHERE SAL > (SELECT AVG(SAL)
  4               FROM EMP
  5               WHERE E.DEPTNO = DEPTNO);

ENAME             SAL
---------- ----------
ALLEN            1600
JONES            2975
BLAKE            2850
KING             5000
FORD             3000
SCOTT            3000

6 개의 행이 선택되었습니다.
```

위의 쿼리 문장에서는 메인쿼리의 첫 번째 행이 서브쿼리로 전달되어 E.DEPTNO가 상수로 교체되면 서브쿼리가 실행된 후, 그 결과가 다시 메인쿼리로 전달되어 처리된다.

EXISTS

EXISTS는 서브쿼리의 결과가 한 개의 행이라도 존재하면 조건은 참이 되며 반대로 한 개의 행도 존재하지 않으면 조건은 거짓이 된다. 또한, EXISTS는 서브쿼리에서 첫 번째 행이 검색되면 두 번째 행을 검색하지 않고 바로 메인 쿼리에 참을 리턴하기 때문에 검색 속도도 상당히 빠른 편이다. 다음은 사원 테이블에서 부하 직원을 한 사람이상 데리고 있는 사원을 검색하는 방법이다.

```
SQL> SELECT EMPNO, ENAME
  2      FROM EMP E
  3      WHERE EXISTS (SELECT 'X'
  4                    FROM EMP
  5                    WHERE E.EMPNO = MGR);

     EMPNO ENAME
---------- ----------
      7566 JONES
      7698 BLAKE
      7782 CLARK
      7839 KING
      7902 FORD
      7788 SCOTT

6 개의 행이 선택되었습니다.
```

위 문장은 IN을 사용하여 일반 서브쿼리 문장으로 변환이 가능하다.

```
SELECT EMPNO, ENAME
FROM EMP
WHERE EMPNO IN (SELECT MGR
                FROM EMP
                WHERE MGR IS NOT NULL);
```

부서 테이블에서 사원들이 배정되지 않은 부서명을 검색하려면 NOT EXISTS를 사용하면 된다.

```
SQL> SELECT DEPTNO, DNAME
  2   FROM DEPT D
  3   WHERE NOT EXISTS (SELECT 'X'
  4                     FROM EMP
  5                     WHERE DEPTNO = D.DEPTNO);

    DEPTNO DNAME
---------- --------------
        40 OPERATIONS
```

마찬가지로, 위의 문장을 IN으로 표현하면 다음과 같다.

```
SELECT DEPTNO, DNAME
FROM DEPT
WHERE DEPTNO NOT IN (SELECT DEPTNO
                     FROM EMP);
```

상관관계 UPDATE

UPDATE 명령에도 상관관계 서브쿼리를 사용하여 다른 테이블의 행을 기반으로 테이블의 행을 변경 할 수 있다. 상관관계 UPDATE 명령의 문법은 다음과 같다.

```
UPDATE table1 alias1
SET column = (SELECT expression
              FROM table2 alias2
              WHERE alias1.column = alias2.column);
```

상관관계 UPDATE 명령을 연습하기 위해 다음과 같이 테이블에 DNAME 컬럼을 추가한다.

```
ALTER TABLE EMP
ADD DNAME VARCHAR2(14);
```

부서 테이블을 기반으로 사원 테이블의 DNAME 컬럼에 부서명을 입력하는 방법은 다음과 같다.

```
SQL> UPDATE EMP E
  2  SET E.DNAME = (SELECT DNAME
  3                 FROM DEPT
  4                 WHERE DEPTNO = E.DEPTNO);

14 행이 갱신되었습니다.
```

상관관계 DELETE

DELETE 명령에도 상관관계 서브쿼리를 사용하여 다른 테이블의 행을 기반으로 테이블의 행을 삭제 할 수 있다. 상관관계 DELETE 명령의 문법은 다음과 같다.

```
DELETE FROM table1 alias1
WHERE column operator
             (SELECT expression
              FROM table2 alias2
              WHERE alias1.column = alias2.column);
```

상관관계 DELETE 명령을 연습하기 위해 다음과 같이 테이블을 추가한다. 추가되는 테이블에는 사원 테이블에서 삭제할 직원의 사번이 입력되어 있다고 가정한다.

```
CREATE TABLE DEL_LIST
(EMPNO NUMBER(4));

INSERT INTO DEL_LIST VALUES(7934);
INSERT INTO DEL_LIST VALUES(7788);
```

사원 테이블에서 DEL_LIST 테이블에 입력되어 있는 사원들을 삭제하는 방법은 다음과 같다.

```
SQL> DELETE FROM EMP E
  2  WHERE E.EMPNO = (SELECT EMPNO
  3                   FROM DEL_LIST
  4                   WHERE EMPNO = E.EMPNO);

2 행이 삭제되었습니다.
```

WITH

WITH는 쿼리 문장안에 반복해서 포함되는 서브쿼리를 블럭으로 선언하여 문장의 다양한 위치에서 활용하는 방법이다. WITH 구문에 기술되어 있는 서브쿼리의 실행 결과는 사용자의 임시 테이블스페이스에 저장되며 쿼리의 수행속도를 향상시켜 준다.

다음은 전체 부서별 총 급여의 평균보다 높은 부서의 총 급여를 검색하는 방법이다.

```
SQL> WITH
  2  DEPT_SAL AS (
  3       SELECT D.DNAME, SUM(E.SAL) AS SAL_SUM
  4       FROM DEPT D, EMP E
  5       WHERE D.DEPTNO = E.DEPTNO
  6       GROUP BY D.DNAME),
  7  DEPT_AVG AS (
  8       SELECT AVG(SAL_SUM) AS SAL_AVG
  9       FROM DEPT_SAL)
 10  SELECT *
 11  FROM DEPT_SAL
 12  WHERE SAL_SUM > (SELECT SAL_AVG
 13                   FROM DEPT_AVG)
 14  ORDER BY DNAME;

DNAME             SAL_SUM
-------------- ----------
RESEARCH            10875
```

WITH에 기술된 첫 번째 서브쿼리는 부서별 총 급여를 산출하여 DEPT_SAL이라는 별칭이 부여되고, 두 번째 서브쿼리는 첫 번째 서브쿼리에서 생성된 DEPT_SAL을 이용하여 부서별 총 급여의 평균을 산출하여 DEPT_AVG라는 별칭이 부여되었다. 이 후, 두 개의 서브쿼리 결과를 이용하여 최종 결과를 산출하게 된다.

복습

1. 다음 JOIN 문장을 상관관계 서브쿼리로 변환하시오.

```
SELECT E.ENAME
FROM DEPT D, EMP E
WHERE D.DEPTNO = E.DEPTNO;
```

2. 사원 테이블에서 같은 업무를 수행하는 직원들 중, 해당 업무별 평균 급여보다 많은 급여를 받는 직원의 사원명, 업무, 급여를 검색하시오.(상관관계 서브쿼리로 작성할 것)

3. 사원 테이블에서 같은 부서에 근무하는 직원들 중, 해당 부서별 최대 급여를 받는 직원의 사원명, 부서번호, 급여를 검색하시오.(상관관계 서브쿼리로 작성할 것)

[4-6] 다음과 같은 MASTER 테이블과 TEMP 테이블을 작성하시오

```
CREATE TABLE MASTER
(ID NUMBER(3),
 NAME VARCHAR2(10));

INSERT INTO MASTER VALUES (1, 'NORWAY');
INSERT INTO MASTER VALUES (2, 'SPAIN');
INSERT INTO MASTER VALUES (3, 'SWEDEN');
INSERT INTO MASTER VALUES (4, 'ASIA');
INSERT INTO MASTER VALUES (5, 'CHINA');
INSERT INTO MASTER VALUES (6, 'INDIA');

CREATE TABLE TEMP
(ID NUMBER(3),
 NAME VARCHAR2(10));

INSERT INTO TEMP VALUES (1, 'DENMARK');
INSERT INTO TEMP VALUES (2,  NULL);
INSERT INTO TEMP VALUES (3, 'FRANCE');
INSERT INTO TEMP VALUES (4,  NULL);
INSERT INTO TEMP VALUES (5, 'GERMANY');
INSERT INTO TEMP VALUES (7, 'BRAZIL');
INSERT INTO TEMP VALUES (8, 'CANADA');
```

MASTER 테이블에는 데이터의 원본이며, TEMP 테이블은 MASTER 테이블에 변경할 데이터가 저장되는 테이블이다. TEMP 테이블을 기반으로 MASTER 테이블을 변경하는 규칙은 다음과 같다.

- MASTER.ID = TEMP.ID 이면 MASTER 테이블에서 해당 행을 변경
- MASTER.ID에 TEMP.ID가 없는 경우는 MASTER 테이블에서 해당 행을 추가
- TEMP.NAME이 NULL인 경우는 MASTER 테이블에서 해당 행을 삭제

4. TEMP 테이블을 기반으로 MASTER 테이블의 해당 행을 변경하시오. 변경 후의 MASTER 테이블은 다음과 같아야 한다.

```
        ID NAME
---------- ----------
         1 DENMARK
         2 SPAIN
         3 FRANCE
         4 ASIA
         5 GERMANY
         6 INDIA
```

5. TEMP 테이블을 기반으로 MASTER 테이블의 해당 행을 추가하시오. 변경 후의 MASTER 테이블은 다음과 같아야 한다.

```
        ID NAME
---------- ----------
         1 DENMARK
         2 SPAIN
         3 FRANCE
         4 ASIA
         5 GERMANY
         6 INDIA
         7 BRAZIL
         8 CANADA
```

6. TEMP 테이블을 기반으로 MASTER 테이블의 해당 행을 삭제하시오. 변경 후의 MASTER 테이블은 다음과 같아야 한다.

```
        ID NAME
---------- ----------
         1 DENMARK
         3 FRANCE
         5 GERMANY
         6 INDIA
         7 BRAZIL
         8 CANADA
```