

☆ 서브쿼리

단일 행 서브쿼리

다중 행 서브쿼리

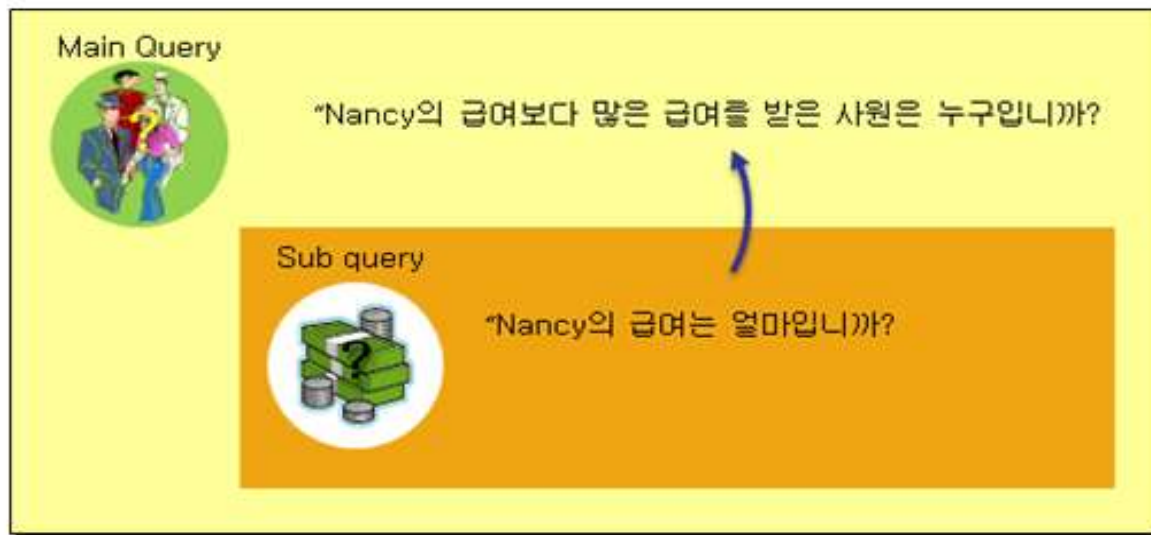
스칼라(Scalar) 서브쿼리

인라인 뷰(Inline View)



“Nancy 보다 많은 급여를 받는 사람은?”

사원 Nancy보다 많은 급여를 받는 사원을 찾기 위한 질의를 작성하고자 합니다. 이 문제를 해결하기 위해서 두개의 질의가 필요합니다. 하나의 질의는 Nancy의 급여를 알기 위한 것이고, 두 번째 질의는 그것보다 많은 급여를 받는 사원을 찾는 것입니다. 하나의 질의를 다른 질의 내부에 두는 방법으로 두 개의 질의를 조합하여 이 문제를 해결할 수 있습니다.



서브쿼리는 다른 SELECT 문장의 절에 내장된 SELECT 문장입니다. 서브쿼리를 사용하여 간단한 문장을 강력한 문장으로 만들 수 있습니다. 테이블 자체의 데이터에 의존하는 조건으로 테이블의 행을 검색할 필요가 있을 때 서브쿼리는 아주 유용합니다.

```
SELECT  select_list
FROM    table
WHERE   expr operator (SELECT  select_list
                        FROM    table);
```

구문형식에서...

- *operator*: >, = 또는 IN 같은 비교 연산자를 포함합니다. 비교 연산자는 단일 행 연산자(>, =, >=, <, <=)와 다중 행 연산자(IN, ANY, ALL)가 있습니다.

Nancy의 급여보다 많은 급여를 받는 사원의 이름과 급여를 출력하는 문제는 서브쿼리를 작성하여 해결할 수 있습니다.

```
SQL> SELECT first_name, salary
2  FROM employees
3  WHERE salary > (SELECT salary
4                  FROM employees
5                  WHERE first_name='Nancy');
```

	FIRST_NAME	SALARY
1	Steven	24000
2	Neena	17000
3	Lex	17000
4	John	14000
5	Karen	13500
6	Michael	13000

다음의 SQL 절에 서브쿼리를 작성할 수 있습니다.

- SELECT 절(스칼라 서브쿼리)
- FROM 절(인라인 뷰)
- WHERE 절
- HAVING 절
- ORDER BY 절
- INSERT 문의 VALUES 절
- UPDATE 문의 SET 절
- CREATE TABLE 문의 AS 절

다음은 서브쿼리를 사용할 때 지켜야 할 사항들입니다.

- 서브쿼리는 괄호로 둘러싸야 합니다.
- 서브쿼리는 비교 연산자의 오른쪽에 있어야 합니다.
- 서브쿼리는 ORDER BY 절을 포함할 수 없습니다. SELECT 문장에 대해서는 오직 하나의 ORDER BY 절을 가질 수 있으며, SELECT 문장의 제일 마지막에 있어야 합니다.
- 서브쿼리에서는 두 종류의 비교 연산자를 사용합니다. 단일 행 서브쿼리에는 단일 행 연산자를 사용해야 하며, 다중 행 서브쿼리에는 다중 행 연산자를 사용해야 합니다.

서브쿼리

단일 행 서브쿼리

다중 행 서브쿼리

스칼라(Scalar) 서브쿼리

인라인 뷰(Inline View)



단일 행 서브쿼리는 내부 SELECT 문장으로부터 하나의 행을 리턴하는 질의입니다. 이런 유형의 서브쿼리는 단일 행 연산자를 사용합니다. 다음 표는 단일 행 연산자의 목록을 보여줍니다.

연산자	설명
=	같다
>	보다 크다
>=	보다 크거나 같다
<	보다 작다
<=	보다 작거나 같다
<>	같지 않다.

다음 구문은 103번 사원의 직무와 같은 사원의 이름과 직무, 그리고 입사일을 출력합니다.
103번 사원은 오직 한명입니다. 그러므로 서브쿼리는 최대 한 개 행을 리턴합니다.

```
SQL> SELECT first_name, job_id, hire_date
2  FROM employees IT_PROG
3  WHERE job_id = (SELECT job_id
4                  FROM employees
5                  WHERE employee_id=103);
```

서브쿼리 절이 1행 이하의 결과를 리턴해야 합니다

	↕ FIRST_NAME	↕ JOB_ID	↕ HIRE_DATE
1	Alexander	IT_PROG	06/01/03
2	Bruce	IT_PROG	07/05/21
3	David	IT_PROG	05/06/25
4	Valli	IT_PROG	06/02/05
5	Diana	IT_PROG	07/02/07

단일행 연산자를 사용하는 서브쿼리는 반드시 한 개 행 또는 0개 행을 반환해야 합니다. WHERE 절에서 서브쿼리가 두 개 행 이상 한다면 다음 절에서 설명하는 다중 행 서브쿼리 연산자를 이용해야 합니다.

서브쿼리

단일 행 서브쿼리

다중 행 서브쿼리

스칼라(Scalar) 서브쿼리

인라인 뷰(Inline View)



Nancy의 급여보다 많은 급여를 받는 사원의 이름과 급여를 출력하는 문제에서 다른 사원의 이름을 이용하여 같은 문제를 해결하려 할 때 질의가 정상 실행되지 않을 수 있습니다.

다음 구문은 David 보다 많은 급여를 받는 사원의 이름과 급여를 출력하기 위해 작성했습니다.

SQL> SELECT first_name, salary	ORA-01427: single-row subquery returns more than one row
2 FROM employees	01427, 00000 - "single-row subquery returns more than one row"
3 WHERE salary > (SELECT salary	*Cause:
4 FROM employees	*Action:
5 WHERE first_name='David');	

위 구문을 실행시키면 다음과 같은 에러가 발생합니다. 에러의 의미는 단일 행 서브쿼리가 한 개 행보다 많은 행을 리턴했다는 의미입니다. 위 구문에서 WHERE절의 비교 연산자는 한 번에 여러 개 값과 비교할 수 없습니다.

first_name = David가 여러 행이 존재하기 때문에 단일행 서브쿼리가 실행 되지 않는다

서브쿼리 결과가 2개 행 이상일 경우 다중 행 서브쿼리라고 부릅니다. 서브쿼리가 다중 행 서브쿼리일 경우 사용되는 연산자가 다릅니다.

연산자	설명
IN	목록의 어떤 값과 같은지 확인합니다. in 가장 많이 쓰여!
ANY, SOME	값을 서브쿼리에 의해 리턴된 각각의 값과 비교합니다. 하나라도 만족하면 됩니다.
ALL	값을 서브쿼리에 의해 리턴된 모든 값과 비교합니다. 모든 값과 비교해서 만족해야 합니다.
EXISTS	결과를 만족하는 값이 존재하는지 여부를 확인합니다.
all과 any의 차이점	<ul style="list-style-type: none"> < ANY : 가장 큰 값보다 작으면 됩니다. > ANY : 가장 작은 값보다 크면 됩니다. < ALL : 가장 작은 값보다 작아야 합니다. > ALL : 가장 큰 값보다 커야 합니다. = ANY : IN과 같은 역할을 합니다.



다음 구문은 이름이 David인 사원들 중에서 어느 한사람의 급여보다 많은 급여를 받는 사원의 이름과 급여를 출력합니다.

```
SQL> SELECT first_name, salary
2 FROM employees
3 WHERE salary > ANY (SELECT salary
4 FROM employees
5 WHERE first_name='David');
```

David의 급여

	SALARY	
1	4800	SALARY > ANY
2	9500	
3	6800	SALARY < ANY

SQL | 인출된 모든 행: 58(0.003초)

	FIRST_N...	SALARY
51	Shanta	6500
52	Sundar	6400
53	Charles	6200
54	Amit	6200
55	Sundita	6100
56	Pat	6000
57	Bruce	6000
58	Kevin	5800

- > ANY : 최솟값보다 커야 합니다. 4800보다 많은 급여를 받는 사람의 정보를 출력합니다.
- < ANY : 최댓값보다 작아야 합니다. 9500보다 적은 급여를 받는 사람의 정보를 출력합니다.
- > ALL : 최댓값보다 커야 합니다. 9500보다 많은 급여를 받는 사람의 정보를 출력합니다.
- < ALL : 최솟값보다 작아야 합니다. 4800보다 적은 급여를 받는 사람의 정보를 출력합니다.

다음 구문은 David와 같은 부서에 근무하는 사원의 이름(First Name)과 부서번호, 직무를 출력합니다. IN 연산자와 서브쿼리를 이용하였습니다.

```
SQL> SELECT first_name, department_id, job_id
2   FROM employees
3  WHERE department_id IN (SELECT department_id
4                           FROM employees
5                           WHERE first_name='David');
```

위 구문은 39개 행을 출력합니다.

	FIRST_N...	DEPARTMENT_ID	JOB_ID
1	Alexander	60	IT_PROG
2	Bruce	60	IT_PROG
3	David	60	IT_PROG
4	Valli	60	IT_PROG
5	Diana	60	IT_PROG
6	John	80	SA_MAN
7	Karen	80	SA_MAN
8	Alberto	80	SA_MAN
9	Gerald	80	SA_MAN
10	Eleni	80	SA_MAN

...

서브쿼리

단일 행 서브쿼리

다중 행 서브쿼리

스칼라(Scalar) 서브쿼리

인라인 뷰(Inline View)



스칼라(Scalar) 서브쿼리는 SELECT 절에 사용하는 서브쿼리입니다. 스칼라 서브쿼리를 이용하면 다양한 결과를 도출할 수 있으며, 특히 조인을 수행할 시 조인할 행의 수를 줄여 성능을 향상시킬 수 있습니다.

다음 구문은 모든 사원의 이름과 부서이름을 출력합니다.

```
SQL> SELECT first_name, (SELECT department_name
2 FROM departments d
3 WHERE d.department_id=e.department_id) department_name
4 FROM employees e
5 ORDER BY first_name;
```

OPERATION	OBJECT_NAME	CARDINALITY	COST
SELECT STATEMENT		107	4
TABLE ACCESS (BY INDEX ROWID)	DEPARTMENTS	1	1
INDEX (UNIQUE SCAN)	DEPT_ID_PK	1	0
Access Predicates			
D,DEPARTMENT_ID=:B1			
SORT (ORDER BY)		107	4
TABLE ACCESS (FULL)	EMPLOYEES	107	3

다음 구문은 모든 사원의 이름과 부서이름을 출력하기 위해 조인을 사용했습니다.

```
SQL> SELECT first_name, department_name
2  FROM employees e
3  JOIN departments d ON (e.department_id=d.department_id)
4  ORDER BY first_name;
```

OPERATION	OBJECT_NAME	CARDINALITY	COST
SELECT STATEMENT		106	7
SORT (ORDER BY)		106	7
MERGE JOIN		106	6
TABLE ACCESS (BY INDEX ROWID) DEPARTMENTS		27	2
INDEX (FULL SCAN) DEPT_ID_PK	DEPT_ID_PK	27	1
SORT (JOIN)		107	4
Access Predicates			
E,DEPARTMENT_ID=D,DEP#			
Filter Predicates			
E,DEPARTMENT_ID=D,DEP#			
VIEW	index\$_join\$_001	107	3
HASH JOIN			

SQL Developer에서 위 두 구문을 각각 실행한 다음 [F10] 키를 누르면 실행계획을 볼 수 있습니다.

서브쿼리

단일 행 서브쿼리


다중 행 서브쿼리

스칼라(Scalar) 서브쿼리

인라인 뷰(Inline View)



인라인 뷰는 FROM 절에 서브쿼리가 온 것을 말합니다. FROM 절에는 테이블 또는 뷰가 올 수 있습니다. 그런데 서브쿼리를 FROM 절에 사용해 하나의 테이블 또는 뷰처럼 사용할 수 있습니다. 뷰도 하나의 독립적인 SELECT문이므로 FROM 절에 사용하는 서브쿼리도 하나의 뷰로 볼 수 있습니다. 그래서 FROM 절에 오는 뷰를 인라인 뷰라고 부릅니다.



```
SQL> SELECT rownum, first_name, salary
2 FROM (SELECT first_name, salary
3 FROM employees
4 ORDER BY salary DESC)
5 WHERE rownum between 1 and 10;
```

위의 구문의 실행 결과는 급여액이 큰 직원부터
상위 10명의 정보를 출력합니다.

	ROWNUM	FIRST_NAME	SALARY
1	1	Steven	24000
2	2	Neena	17000
3	3	Lex	17000
4	4	John	14000
5	5	Karen	13500
6	6	Michael	13000
7	7	Nancy	12008
8	8	Shelley	12008
9	9	Alberto	12000
10	10	Lisa	11500

```
SQL> SELECT rownum, first_name, salary
2 FROM (SELECT first_name, salary
3        FROM employees
4        ORDER BY salary DESC)
5 WHERE rownum between 1 and 10;
```

위의 구문의 실행 결과는 급여액이 큰 직원부터
상위 10명의 정보를 출력합니다.

	ROWNUM	FIRST_NAME	SALARY
1	1	Steven	24000
2	2	Neena	17000
3	3	Lex	17000
4	4	John	14000
5	5	Karen	13500
6	6	Michael	13000
7	7	Nancy	12008
8	8	Shelley	12008
9	9	Alberto	12000
10	10	Lisa	11500

ROWNUM은 첫 번째 행부터 조회할 수 있습니다.

```
SQL> SELECT first_name, salary
2 FROM (SELECT first_name, salary
3        FROM employees
4        ORDER BY salary DESC)
5 WHERE rownum between 11 and 20;
```

?

어떻게 행 번호를 카운트 해야할까?

SQL | 인출된 모든 행: 0(0초)
FIRST_N... SALARY

```
SQL> SELECT rnum, first name, salary
2 FROM (SELECT first name, salary, rownum AS rnum
3       FROM (SELECT first_name, salary
4             FROM employees
5             ORDER BY salary DESC)
6       )
7 WHERE rnum between 11 and 20;
```

	RNUM	FIRST_NAME	SALARY
1	11	Den	11000
2	12	Gerald	11000
3	13	Ellen	11000
4	14	Eleni	10500
5	15	Clara	10500
6	16	Janette	10000
7	17	Peter	10000
8	18	Hermann	10000
9	19	Harrison	10000
10	20	Tayler	9600

3중 쿼리

추후 게시판 페이지에 이용가능 하므로 알아야 하는 쿼리중 하나입니다

문제 1.

- EMPLOYEES 테이블에서 모든 직원들의 평균급여보다 높은 직원들을 데이터를 출력 하세요 (AVG(컬럼) 사용)
- EMPLOYEES 테이블에서 모든 직원들의 평균급여보다 높은 직원들을 수를 출력하세요
- EMPLOYEES 테이블에서 job_id가 IT_PFOG인 직원들의 평균급여보다 높은 직원들을 데이터를 출력하세요

문제 2.

- DEPARTMENTS테이블에서 manager_id가 100인 사람의 department_id와 EMPLOYEES테이블에서 department_id가 일치하는 **모든 사원의 정보**를 검색하세요.

문제 3.

- EMPLOYEES테이블에서 “Pat”의 manager_id보다 높은 manager_id를 갖는 모든 사원의 데이터를 출력하세요
- EMPLOYEES테이블에서 “James”(2명)들의 manager_id와 갖는 모든 사원의 데이터를 출력하세요.

문제 4.

- EMPLOYEES테이블 에서 first_name기준으로 내림차순 정렬하고, 41~50번째 데이터의 행 번호, 이름을 출력하세요

문제 5.

- EMPLOYEES테이블에서 hire_date기준으로 오름차순 정렬하고, 31~40번째 데이터의 행 번호, 직원id, 이름, 번호, 입사일을 출력하세요.

문제 6.

employees테이블 departments테이블을 left 조인하세요

조건) 직원아이디, 이름(성, 이름), 부서아이디, 부서명 만 출력합니다.

조건) 직원아이디 기준 오름차순 정렬

문제 7.

문제 6의 결과를 (스칼라 쿼리)로 동일하게 조회하세요

문제 8.

departments테이블 locations테이블을 left 조인하세요

조건) 부서아이디, 부서이름, 매니저아이디, 로케이션아이디, 스트리트_어드레스, 포스트 코드, 시티 만 출력합니다

조건) 부서아이디 기준 오름차순 정렬

문제 9.

문제 8의 결과를 (스칼라 쿼리)로 동일하게 조회하세요

문제 10.

locations테이블 countries 테이블을 left 조인하세요

조건) 로케이션아이디, 주소, 시티, country_id, country_name 만 출력합니다

조건) country_name기준 오름차순 정렬

문제 11.

문제 10의 결과를 (스칼라 쿼리)로 동일하게 조회하세요

조인과 서브쿼리

문제 12.

employees테이블, departments테이블을 left조인 hire_date를 오름차순 기준으로 1-10번째 데이터만 출력합니다
조건) rownum을 적용하여 번호, 직원아이디, 이름, 전화번호, 입사일, 부서아이디, 부서이름 을 출력합니다.
조건) hire_date를 기준으로 오름차순 정렬 되어야 합니다. rownum이 들어지면 안됩니다.

문제 13.

--EMPLOYEES 과 DEPARTMENTS 테이블에서 JOB_ID가 SA_MAN 사원의 정보의 LAST_NAME, JOB_ID,
DEPARTMENT_ID,DEPARTMENT_NAME을 출력하세요

문제 14

--DEPARTMENT테이블에서 각 부서의 ID, NAME, MANAGER_ID와 부서에 속한 인원수를 출력하세요.
--인원수 기준 내림차순 정렬하세요.
--사람이 없는 부서는 출력하지 뽑지 않습니다

문제 15

--부서에 대한 정보 전부와, 주소, 우편번호, 부서별 평균 연봉을 구해서 출력하세요
--부서별 평균이 없으면 0으로 출력하세요

문제 16

-문제 15결과에 대해 DEPARTMENT_ID기준으로 내림차순 정렬해서 ROWNUM을 붙여 1-10데이터 까지만
출력하세요