ORACLE®

DB Management

서브쿼리

서브쿼리

- 서브쿼리
  - ✓ Sub Query
  - ✓ 어떤 SQL문의 결과를 다른 SQL문에게 전달하기 위해 두 개 이상의 SQL문을 하나의 SQL문으로 연결하는 방법
  - ✓ 메인쿼리(Main Query)에 포함되는 쿼리를 서브쿼리(Sub Query)라고 함
  - ✓ 서브쿼리를 먼저 실행한 뒤 그 결과를 메인쿼리로 전달하는 방식으로 동작

- 서브쿼리 종류
  - ① 일반 서브쿼리
    - ➢ 단일 행 서브쿼리
    - ➢ 다중 행 서브쿼리
  - ② 스칼라 서브쿼리
  - ③ 인라인 뷰

서브쿼리가 필요한 상황

❖ **제품번호 '1001'인 제품과 동일한 공장에서 생산되는 제품들의 목록을 조회하시오.**

- 처리과정1 - 서브쿼리 작성
  - ✓ 제품번호 '1001'인 제품이 생산되는 공장을 조회한다.

```
SELECT 공장 FROM 제품 WHERE 제품번호 = '1001'
```

- 처리과정2 - 메인쿼리 작성
  - ✓ 제품번호 '1001'인 제품과 동일한 공장에서 생산되는 제품들의 목록을 조회한다.

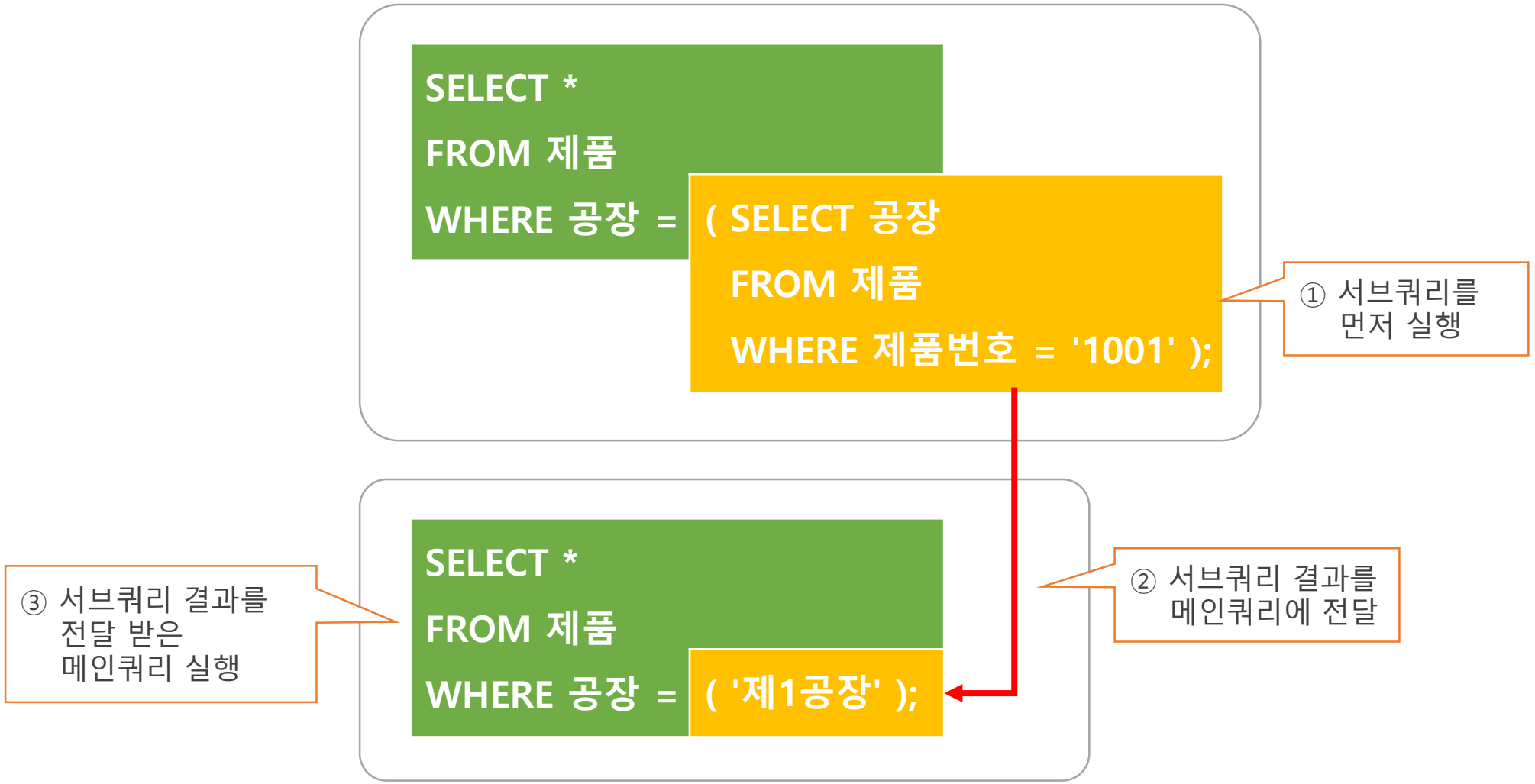
```
SELECT * FROM 제품 WHERE 공장 = (제품번호 1001인 제품이 생산되는 공장)
```

- 처리과정 - 메인쿼리 완성
  - ✓ 처리과정1과 처리과정2의 SQL문을 하나의 SQL문으로 작성한다.

```
SELECT * FROM 제품 WHERE 공장 = (SELECT 공장 FROM 제품 WHERE 제품번호 = '1001')
```

서브쿼리 처리과정

- 서브쿼리가 먼저 처리되고, 메인쿼리가 나중에 처리



단일 행 서브쿼리

- 일반 서브쿼리의 한 종류
  - ✓ WHERE절에서 사용되며 어떤 값을 반환하는 서브쿼리
- 서브쿼리가 단 하나의 행(Row)만을 검색하여 메인쿼리에 반환하는 방식
- 단일 행 서브쿼리가 되기 위한 방법
  - ✓ WHERE절에서 기본키(PK)나 고유키(UNIQUE)를 가진 칼럼과 동등비교(=)한 서브쿼리
  - ✓ 집계(그룹) 함수를 사용하여 어떤 하나의 값을 얻어내는 서브쿼리
- 메인쿼리의 WHERE절에서 단일 행 서브쿼리의 결과와 비교할 때는 반드시 단일 행 비교연산자를 사용해야 함
  - ✓ 단일 행 비교연산자 : =, !=, >, >=, <, <=

단일 행 서브쿼리

- 평균 연봉 이상을 받는 사원 조회하기

EMP_NO	NAME	DEPART	POSITION	GENDER	HIRE_DATE	SALARY
1001	구창민	1	과장	M	95-05-01	5000000
1002	김민서	1	사원	M	17-09-01	2500000
1003	이은영	2	부장	F	90-09-01	5500000
1004	한성일	2	과장	M	93-04-01	5000000

- 메인쿼리 + 서브쿼리

```
SELECT * FROM EMPLOYEE_TBL WHERE SALARY > (SELECT AVG(SALARY) FROM EMPLOYEE_TBL);
```

- 해설

서브쿼리를 먼저 처리하고,
처리 결과 값 1개를
메인쿼리로 전달한다.

```
SELECT * FROM EMPLOYEE_TBL WHERE SALARY > (4500000);
```

단일 행 비교연산자 사용

ORACLE

다중 행 서브쿼리

- 일반 서브쿼리의 한 종류
  - ✓ WHERE절에서 사용되며 어떤 값을 반환하는 서브쿼리

- 서브쿼리가 두 개 이상의 행(Row)을 검색하여 메인쿼리에 반환하는 방식

- 메인쿼리의 WHERE절에서 다중 행 서브쿼리의 결과와 비교할 때는 반드시 다중 행 비교연산자를 사용해야 함

종류	특징
IN	1. 서브쿼리의 결과 중에서 하나라도 일치하면 참 2. 동등비교(=)
ANY	1. 서브쿼리의 결과들과 비교한 결과가 하나라도 일치하면 참 2. 범위비교(<, >)
ALL	1. 서브쿼리의 결과들과 비교한 결과가 모두 일치하면 참 2. 범위비교(<, >)
EXISTS	서브쿼리의 검색 결과가 하나라도 존재하면 참

다중 행 서브쿼리

- 부서번호가 1인 부서에 존재하는 직급과 같은 직급을 가진 사원 조회하기

EMP_NO	NAME	DEPART	POSITION	GENDER	HIRE_DATE	SALARY
1001	구창민	1	과장	M	95-05-01	5000000
1002	김민서	1	사원	M	17-09-01	2500000
1003	이은영	2	부장	F	90-09-01	5500000
1004	한성일	2	과장	M	93-04-01	5000000

- 메인쿼리 + 서브쿼리

```
SELECT * FROM EMPLOYEE_TBL WHERE POSITION IN (SELECT POSITION
                                                FROM EMPLOYEE_TBL
                                               WHERE DEPART = 1);
```

- 해설

서브쿼리를 먼저 처리하고,
처리 결과 값 2개를
메인쿼리로 전달한다.

```
SELECT * FROM EMPLOYEE_TBL WHERE POSITION IN ('과장', '부장');
```

다중 행 비교연산자 사용

스칼라 서브쿼리

- 하나의 칼럼처럼 사용되는 쿼리
  - ✓ SELECT절에서 사용되며 어떤 값을 반환하는 서브쿼리
- 칼럼의 역할을 수행하기 위해서 반드시 한 개의 행(Row)만을 반환해야 함
- 성능이 좋지 않기 때문에 자주 사용할 필요가 없음

스칼라 서브쿼리

- 사원명과 부서명 조회하기

DEPT_NO	DEPT_NAME	LOCATION
1	영업부	대구
2	인사부	서울
3	총무부	대구
4	기획부	서울

DEPARTMENT

EMP_NO	NAME	DEPART	POSITION	GENDER	HIRE_DATE	SALARY
1001	구창민	1	과장	M	95-05-01	5000000
1002	김민서	1	사원	M	17-09-01	2500000
1003	이은영	2	부장	F	90-09-01	5500000
1004	한성일	2	과장	M	93-04-01	5000000

EMPLOYEE

- 메인쿼리 + 서브쿼리

```
SELECT E.NAME AS 사원명
     , (SELECT D.DEPT_NAME FROM DEPARTMENT_TBL D WHERE D.DEPT_NO = E.DEPART) AS 부서명
  FROM EMPLOYEE_TBL E;
```

- 결과

사원명	부서명
구창민	영업부
김민서	영업부
이은영	총무부
한성일	총무부

메인쿼리의 값(E.DEPART)을
서브쿼리에서 사용하는
'상호연관 서브쿼리' 형식의
'스칼라 서브쿼리'

인라인 뷰

- Inline View
- 하나의 테이블처럼 사용되는 쿼리
  - ✓ FROM절에서 사용되며 어떤 테이블 형식의 결과를 반환하는 서브쿼리
- 인라인 뷰에서 조회한 칼럼만 메인쿼리에서 조회할 수 있음
- 인라인 뷰에 별명을 지정한 뒤 메인쿼리에서 해당 별명을 사용함
- SELECT절의 수행 순서를 변경하기 위해서 사용 가능함
  - ✓ SELECT절의 수행 순서는 FROM절이 가장 우선임
  - ✓ 먼저 처리하고자 하는 작업을 인라인 뷰로 작업하면 가장 먼저 처리하게 됨

인라인 뷰

- 가장 높은 연봉을 가진 사원 조회하기 (높은 연봉순으로 정렬한 뒤 첫 번째 데이터를 가져오기)

EMP_NO	NAME	DEPART	POSITION	GENDER	HIRE_DATE	SALARY
1001	구창민	1	과장	M	95-05-01	5000000
1002	김민서	1	사원	M	17-09-01	2500000
1003	이은영	2	부장	F	90-09-01	5500000
1004	한성일	2	과장	M	93-04-01	5000000

- 메인쿼리 + 서브쿼리

```
SELECT A.EMP_NO, A.NAME, A.SALARY
  FROM (SELECT EMP_NO, NAME, SALARY
          FROM EMPLOYEE_TBL
         ORDER BY SALARY DESC) A
 WHERE ROWNUM = 1;
```

```
SELECT EMP_NO, NAME, SALARY
  FROM
```

EMP_NO	NAME	SALARY	
1003	이은영	5500000	ROWNUM = 1
1001	구창민	5000000	ROWNUM = 2
1004	한성일	5000000	ROWNUM = 3
1002	김민서	2500000	ROWNUM = 4

```
 WHERE ROWNUM = 1;
```

인라인 뷰를 통해서 실행 우선 순위가 가장 낮은 정렬 (ORDER BY)을 가장 먼저 처리할 수 있음

- 결과

EMP_NO	NAME	SALARY
1003	이은영	5500000

ORACLE

CREATE와 서브쿼리

- CREATE TABLE문에서 서브쿼리를 이용하여 다른 테이블의 구조와 데이터를 복사하여 새로운 테이블을 생성할 수 있음
- 서브쿼리의 출력 결과가 새로운 테이블의 초기 데이터로 사용됨
- 서브쿼리의 출력 결과가 없어도 새로운 테이블은 생성됨. 초기 데이터가 없는 상태로 생성되기 때문에 특정 테이블의 구조만 복사하고자 하는 경우에 유용함
- 제약조건은 NOT NULL만 복사가 되므로 PK, FK 등과 같은 제약조건은 다시 정의해야 함
- CREATE TABLE문에서 지정한 칼럼(개수, 타입)과 서브쿼리의 칼럼(개수, 타입)이 반드시 일치해야 함
- 형식

```
CREATE TABLE 새_테이블(칼럼1, 칼럼2, ...) AS (SELECT 칼럼1, 칼럼2, ... FROM 테이블);
```

INSERT와 서브쿼리

- INSERT문의 VALUES절 대신 서브쿼리를 작성하여 서브쿼리의 결과 집합을 한 번에 삽입할 수 있음
- 서브쿼리의 결과 칼럼(개수, 타입)과 INSERT문의 칼럼(개수, 타입)이 일치해야 함
- 서브쿼리의 결과가 테이블의 제약조건을 위반되면 오류가 발생하므로 주의해야 함
  (예: PK로 설정된 칼럼에 중복된 데이터를 입력하려고 하는 경우)
- 형식

```
INSERT INTO 테이블(칼럼1, 칼럼2, ...) (SELECT 칼럼1, 칼럼2, ... FROM 테이블 WHERE 조건);
```

UPDATE와 서브쿼리

- UPDATE문의 SET절이나 WHERE절에서 서브쿼리를 작성할 수 있음
- 서브쿼리의 결과 칼럼(개수, 타입)과 UPDATE문의 칼럼(개수, 타입)이 일치해야 함
- SET절 형식1

```
UPDATE 테이블
   SET 칼럼 = (SELECT 칼럼 FROM 테이블 WHERE 조건식)
 WHERE 조건식
```

- SET절 형식2

```
UPDATE 테이블
   SET (칼럼1, 칼럼2) = (SELECT 칼럼1, 칼럼2 FROM 테이블 WHERE 조건식)
 WHERE 조건식
```

DELETE와 서브쿼리

- DELETE문의 WHERE절에서 서브쿼리를 작성할 수 있음
- 서브쿼리의 결과 칼럼(개수, 타입)과 DELETE문의 칼럼(개수, 타입)이 일치해야 함
- WHERE절 형식1

```
DELETE
  FROM 테이블
 WHERE 칼럼 = (SELECT 칼럼 FROM 테이블 WHERE 조건식)
```

- WHERE절 형식2

```
DELETE
  FROM 테이블
 WHERE (칼럼1, 칼럼2) = (SELECT 칼럼1, 칼럼2 FROM 테이블 WHERE 조건식)
```

서브쿼리 주의사항

❖ **다음과 같은 경우 오류가 발생하니 주의할 것!**

1. 다중 행 서브쿼리와 단일 행 비교연산자(=, !=, >, >=, <, <=)를 함께 사용하는 경우

2. 서브쿼리가 반환하는 칼럼의 수와 메인쿼리에서 비교되는 칼럼의 수가 다른 경우

3. 서브쿼리 내에서 ORDER BY 절이 사용되는 경우

4. 서브쿼리의 결과가 NULL 인 경우