



04. SubQuery

● 1. SubQuery

❑ SubQuery

* **Main Query**(SELECT,INSERT,DELETE,UPDATE,CREATE)안에 포함되는 **SELECT**



```
SELECT  ENAME, JOB
FROM    EMP
WHERE   DEPTNO = ( SELECT DEPTNO
                  FROM    EMP
                  WHERE   ENAME = 'SMITH' );
```

❑ 종류

Scalar Subquery	Select List에서 컬럼대신 사용, 단일값(1행,1열값)을 리턴하고 컬럼 대신 사용
Inline View	FROM절에 테이블대신 사용
Nested Subquery	WHERE ,HAVING절에 사용되는 SubQuery

* Query의 2가지 의미 ① SQL ② SELECT

* Scalar : 단일값

● 1. SubQuery

❑ 실행순서

SUBQUERY 실행 → MAIN QUERY 실행

예외) Correlated Subquery (상관서브쿼리)

❑ 분류

1) RETURN 값에 의한 분류

SINGLE ROW SUBQUERY : 리턴되는 데이터가 1건 이하, 단일행 비교 연산자와 함께 사용

Ex) = , > , < , <=..

MULTIPLE ROWS SUBQUERY : 리턴되는 데이터가 1건 이상 , 리스트 연산자와 함께 사용

Ex) IN, ANY, ALL, SOME, EXISTS

SINGLE COLUMN SUBQUERY

MULTIPLE COLUMNS SUBQUERY

2) 동작하는 방식에 따른 분류

Normal Subquery

서브쿼리가 메인쿼리의 컬럼을 참조하지 않는다.

메인쿼리에 값(서브쿼리의 실행결과)을 제공하는 목적으로 사용

Correlated Subquery

서브쿼리가 메인쿼리의 컬럼을 참조 한다.

메인 쿼리가 먼저 실행되고 서브쿼리에서 필터링 하는 목적으로 사용

● 1. SubQuery

❏ SINGLE COLUMN,SINGLE ROW

- ① SELECT ENAME,JOB FROM EMP
WHERE DEPTNO = (SELECT DEPTNO FROM EMP WHERE ENAME = 'SMITH');
- ② SELECT ENAME,SAL FROM EMP WHERE SAL < (SELECT AVG(SAL) FROM EMP);

❏ SINGLE COLUMN, MULTIPLE ROW RETURN SUBQUERY

- ① SELECT ENAME,JOB FROM EMP WHERE DEPTNO = **10,30**; // ??
- ② SELECT ENAME,JOB FROM EMP WHERE DEPTNO **IN (10,30)**; // Multiple Rows // **IN,ANY,ALL**
- ③ SELECT DNAME,LOC FROM DEPT // 3인 이상 근무 부서 정보조회 ??
WHERE DEPTNO = (SELECT DEPTNO FROM EMP GROUP BY DEPTNO HAVING COUNT(*) > 3);

❏ MULTIPLE COLUMN, MULTIPLE ROW RETURN

- ① SELECT DEPTNO,JOB,ENAME,SAL FROM EMP
WHERE (DEPTNO,JOB) IN (SELECT DEPTNO,JOB FROM EMP
GROUP BY DEPTNO,JOB HAVING AVG(SAL) > 2000);

● 1. SubQuery

❑ Scalar Subquery

[장점] 편리성

[질문] 반복되는 실행을 하는가? 실행횟수 * **입/출력값 , Query Execution Cache , hashing**

```
① SELECT DEPTNO,ENAME,JOB,SAL,( SELECT ROUND(AVG(SAL),0)
                                FROM EMP S WHERE S.JOB=M.JOB) AS JOB_AVG_SAL
FROM EMP M
ORDER BY JOB;                                // 실행계획(?) outer-join → 결과가 없으면 NULL 리턴
```

❑ CORRELATED SUBQUERY(상관서브쿼리)

[주의] Subquery는 Mainquery의 컬럼을 참조할수 있지만 Mainquery는 Subquery의 컬럼을 참조할수 없다

[질문] Mainquery에서 Subquery의 컬럼을 참조 하려면 → ① Join 으로 변환 ② Scalar Subquery

```
① SELECT DEPTNO,ENAME,JOB,SAL FROM EMP M
WHERE SAL > ( SELECT AVG(SAL) AS AVG_SAL FROM EMP WHERE JOB = M.JOB );
```

❑ In-Line View (FROM 절에 사용된 SUBQUERY)

[설명] SQL이 실행되는 시점에 동적으로 생성되는 View의 역할을 한다고 해서 Dynamic View 라고도 한다.

일반적으로 Subquery의 컬럼을 Mainquery에서 사용할수 없지만 Inline View에서 Subquery의 컬럼을 Mainquery 에서 사용이 가능하다.

```
① SELECT DEPTNO, ENAME,EMPJOB,SAL,IV.AVG_SAL
FROM EMP, (SELECT JOB,ROUND(AVG(SAL)) AS AVG_SAL FROM EMP GROUP BY JOB) IV
WHERE EMPJOB = IV.JOB AND SAL > IV.AVG_SAL
ORDER BY DEPTNO ,SAL DESC;
```

// 장점?

● 1. SubQuery

❏ TOP-N, BOTTOM-M

- ① SELECT *
FROM (SELECT EMPNO,ENAME,SAL FROM EMP ORDER BY ORDER BY SAL ASC) BM
WHERE ROWNUM <= 5;

- ② SELECT TN.EMPNO,TN.ENAME,TN.SAL
FROM (SELECT EMPNO,ENAME,SAL FROM EMP ORDER BY SAL DESC) TN
WHERE ROWNUM < 5;

● 2. DML 과 SubQuery

❑ DML 연산과 Subquery

// SUBQUERY로 한번에 **N개 Rows INSERT**

① INSERT INTO BONUS(ENAME,JOB,SAL,COMM) SELECT ENAME,JOB,SAL,COMM FROM EMP;

```
SELECT * FROM BONUS;           // 결과 확인
ROLLBACK;                      // 다음번 실습을 위해서
SELECT * FROM BONUS;           // 결과 확인
```

// 부서별 성과별 보너스 계산후(데이터 연산) **N개 Rows INSERT + 데이터 연산(가공)**

② INSERT INTO BONUS(ENAME,JOB,SAL,COMM)
SELECT ENAME,JOB,SAL,DECODE(DEPTNO,10,SAL*0.3,20,SAL*0.2)+NVL(COMM,0)
FROM EMP WHERE DEPTNO IN (10,20);

```
SELECT * FROM BONUS;
COMMIT;
```

// 평상시 COMM을 받지 못하는 직원들에게 평균 COMM 금액의 50%를 보너스로 지급

③ UPDATE EMP SET COMM = (SELECT AVG(COMM)/2 FROM EMP) WHERE COMM IS NULL OR COMM = 0;
COMMIT;

// 평균 이상의 급여를 받는 직원들은 보너스 지급 대상자에서 제외

④ DELETE FROM BONUS WHERE SAL > (SELECT AVG(SAL) FROM EMP);
COMMIT;

● 3. EXISTS 와 Subquery

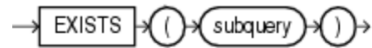
❏ Subquery 연산과 EXISTS

* 데이터의 존재 유무(Exists)를 Boolean(True/False)로 리턴

* Syntax Diagram

EXISTS Condition

An **EXISTS** condition tests for existence of rows in a subquery.



① select deptno from dept where deptno = 30 or deptno = 30 or deptno = 30;

- 3 Rows or 1 Row , 이유는 ?

② select deptno from dept where deptno in (30,30,30,20,20,20,20,10,10,10);

- 9 Rows or 3 Rows , 이유는 ?

③ select deptno as **사원이있는부서** from dept
where deptno in (select deptno from emp where dept.deptno = emp.deptno);

- hash join

④ select deptno as **"사원이 있는 부서"** from dept
where exists (select deptno from emp where dept.deptno = emp.deptno);

- IN 과 동일한 결과 , hash join , 차이점은?

⑤ select deptno as **"사원이 없는 부서"** from dept
where not exists (select deptno from emp where dept.deptno = emp.deptno)

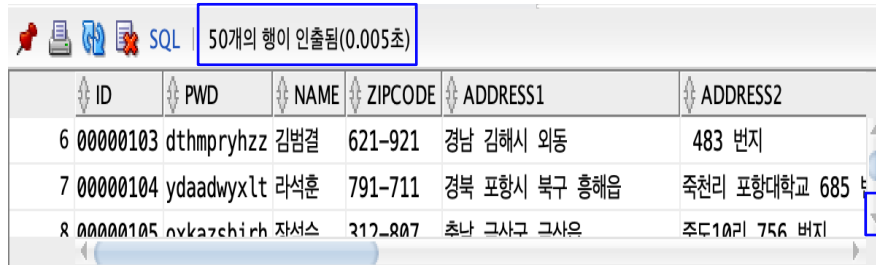
● 3. EXISTS 와 Subquery

- ⑥ select count(*) from customer;
select * from customer;

// 570만 Rows

// 50개의 행만 인출, 이유는? DBMS vs SQL Developer

// 질의결과(Data grid)창 오른쪽 하단부 스크롤을 반복 클릭



ID	PWD	NAME	ZIPCODE	ADDRESS1	ADDRESS2
6 00000103	dthmryhzz	김범결	621-921	경남 김해시 외동	483 번지
7 00000104	ydaadwyxlt	라석훈	791-711	경북 포항시 북구 흥해읍	죽천리 포항대학교 685 번지
8 00000105	nykatzchirh	자서스	312-807	충남 그사구 그사읍	주도10리 756 번지

- ⑦ select * from customer where name like '전지%'; // 최초 실행 2초 , 2번째 실행 0.573초, 이유는?
select * from customer where name like '전지%' and address1 like '%강남%';

- ⑧ select count(*) from customer where account_mgr = 7844;

// empno 7844 사원은 41만명의 고객을 관리하는자(account manager)

- ⑨ select **deptno,job**,empno,ename,sal,comm from emp // 관리하는 고객이 있는 사원/없는 사원?
where **exists** (select **account_mgr** from customer where account_mgr = emp.empno);
select **deptno,job**,empno,ename,sal,comm from emp
where **not exists** (select **1** from customer where account_mgr = emp.empno);

- ⑩ select deptno,job,empno,ename,sal,comm from emp // 동일한 결과? , 동일한 처리 알고리즘(실행계획)?
where empno **in** (select account_mgr from customer where account_mgr = emp.empno);
select deptno,job,empno,ename,sal,comm from emp // 동일한 결과? , 동일한 처리 알고리즘(실행계획)?
where empno **not in** (select account_mgr from customer where account_mgr = emp.empno);