

# 데이터베이스

# 관계형 데이터베이스

## 07. 서브 쿼리

first  
coding

- SCOTT의 부서명을 알아내기 위한 서브 쿼리문 부터 살펴봅시다.

**Main Query ,  
Outer Query**

```
SELECT DNAME  
FROM DEPT  
WHERE DEPTNO = (  
  
)
```

**Sub Query ,  
INNER Query**

```
SELECT DEPTNO  
FROM EMP  
WHERE ENAME='SCOTT'
```

- 서브 쿼리는 하나의 SELECT 문장의 절 안에 포함된 또 하나의 SELECT 문장입니다.
- 그렇기에 서브 쿼리를 포함하고 있는 쿼리문을 메인 쿼리, 포함된 또 하나의 쿼리를 서브 쿼리라 합니다.
- 서브 쿼리는 비교 연산자의 오른쪽에 기술해야 하고 반드시 괄호로 둘러싸아야 합니다.
- 서브 쿼리는 메인 쿼리가 실행되기 이전에 한번만 실행이 됩니다.

- 부속질의의 종류

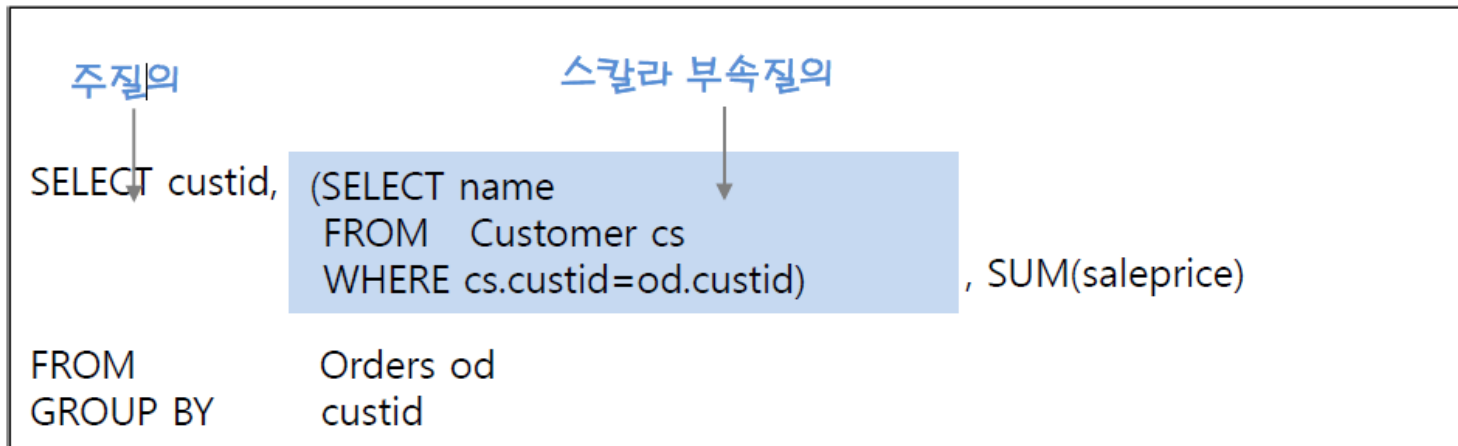
- 스칼라 부속질의 - SELECT 부속질의
- 인라인 뷰 - FROM 부속질의
- 중첩질의 - WHERE 부속질의

명칭	위치	영문 및 동의어	설명
스칼라 부속질의	SELECT 절	scalar subquery	SELECT 절에서 사용되며 단일 값을 반환하기 때문에 스칼라 부속질의라고 함.
인라인 뷰	FROM 절	inline view, table subquery	FROM 절에서 결과를 뷰(view) 형태로 반환하기 때문에 인라인 뷰라고 함.
중첩질의	WHERE 절	nested subquery, predicate subquery	WHERE 절에 술어와 같이 사용되며 결과를 한정시키기 위해 사용됨. 상관 혹은 비상관 형태.

## 03 스칼라 부속질의(scalar subquery)

- 스칼라 부속질의(scalar subquery)

- SELECT 절에서 사용되는 부속질의로, 부속질의의 결과 값을 단일 행, 단일 열의 스칼라 값으로 반환함.
- 스칼라 부속질의는 원칙적으로 스칼라 값이 들어갈 수 있는 모든 곳에 사용 가능하며, 일반적으로 SELECT 문과 UPDATE SET 절에 사용됨.
- 주질의와 부속질의와의 관계는 상관/비상관 모두 가능함.



## 03 스칼라 부속질의(scalar subquery)

- 스칼라 부속질의(scalar subquery)
  - 마당서점의 고객별 판매액을 보이시오(결과는 고객이름과 고객별 판매액을 출력).

```
SELECT ( SELECT name
          FROM Customer cs
          WHERE cs.custid=od.custid ) "name",
        SUM(saleprice) "total"
FROM Orders od
GROUP BY od.custid;
```

name	total
박지성	39000
김연아	15000
추신수	33000
장미란	31000

### 03 스칼라 부속질의(scalar subquery)

- 스칼라 부속질의(scalar subquery)

- 마당서점의 고객별 판매액을 보이시오(결과를 고객이름과 고객별 판매액을 출력).

```
SELECT    custid,  
          SUM(saleprice) "total"  
FROM      Orders od  
GROUP BY  custid ;
```



CUSTID	total
1	39000
2	15000
4	33000
3	31000

```
SELECT    name  
FROM      Customer cs  
WHERE     cs.custid = od.custid
```

custid 1의 이름은?



```
SELECT    (SELECT    name  
          FROM      Customer cs  
          WHERE     cs.custid = od.custid "name",  
          SUM(saleprice) "total"  
FROM      Orders od  
GROUP BY  od.custid ;
```



name	total
박지성	39000
김연아	15000
추신수	33000
장미란	31000

- 인라인 뷰(inline view)
  - FROM 절에서 사용되는 부속질의.
  - 테이블 이름 대신 인라인 뷰 부속질의를 사용하면 보통의 테이블과 같은 형태로 사용할 수 있음.
  - 부속질의 결과 반환되는 데이터는 다중 행, 다중 열이어도 상관없음.
  - 다만 가상의 테이블인 뷰 형태로 제공되어 상관 부속질의로 사용될 수는 없음.



- 인라인 뷰(inline view)
  - 고객번호가 2 이하인 고객의 판매액을 보이시오 (결과는 고객이름과 고객별 판매액 출력)

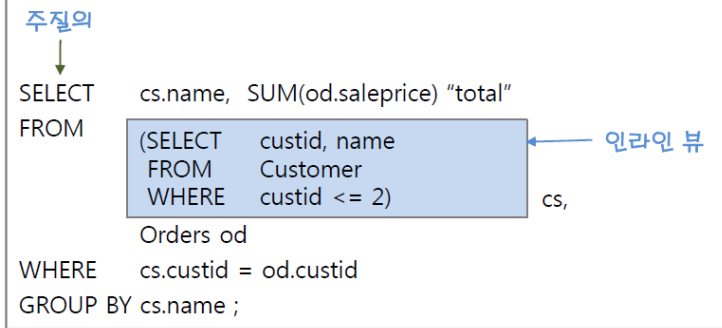
```
SELECT cs.name, SUM(od.saleprice) "total"
FROM (SELECT custid, name
      FROM Customer
      WHERE custid <= 2) cs,
      Orders od
WHERE cs.custid=od.custid
GROUP BY cs.name;
```

NAME	total
박지성	39000
김연아	15000

- 인라인 뷰(inline view)

- 고객번호가 2 이하인 고객의 판매액을 보이시오 (결과를 고객이름과 고객별 판매액 출력)

```
SELECT cs.name, SUM(od.saleprice) "total"
FROM (SELECT custid, name
      FROM Customer
      WHERE custid <= 2) cs,
      Orders od
WHERE cs.custid=od.custid
GROUP BY cs.name;
```



NAME	total
박지성	39000
김연아	15000

## 05 중첩질의(nested subquery)

- 중첩질의(nested subquery)
  - 중첩질의(nested subquery)는 WHERE 절에서 사용되는 부속질의.
  - WHERE 절은 보통 데이터를 선택하는 조건 혹은 술어(predicate)와 같이 사용됨.
  - 중첩질의 연산자의 종류

술어	연산자	반환 행	반환 열	상관
비교	=, >, <, >=, <=, < >	단일	단일	가능
집합	IN, NOT IN	다중	단일	가능
한정(quantified)	ALL, SOME(ANY)	다중	단일	가능
존재	EXISTS, NOT EXISTS	다중	단일	필수

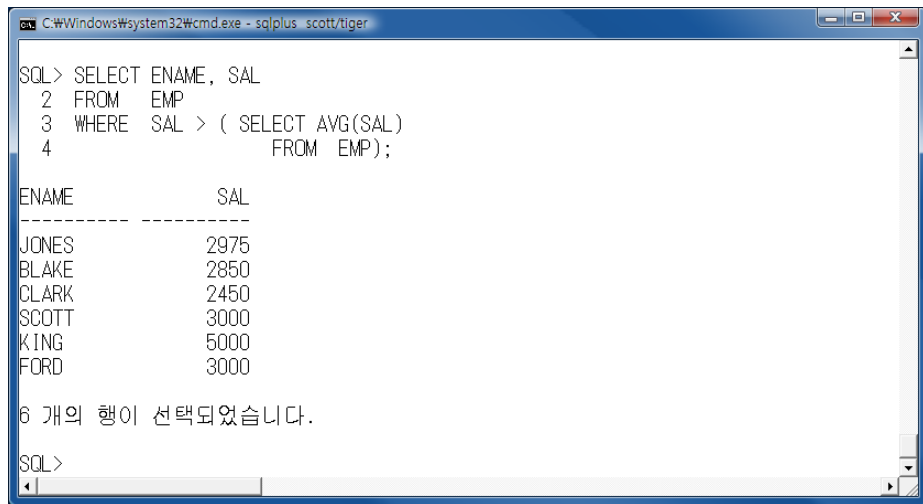
## 05 중첩질의(nested subquery)

- 단일 행(Single Row) 서브 쿼리는 수행 결과가 오직 하나의 로우(행, row)만을 반환하는 서브 쿼리를 갖는 것을 말합니다.
- 단일 행 서브 쿼리문에서는 이렇게 오직 하나의 로우(행, row)로 반환되는 서브 쿼리의 결과는 메인 쿼리에 보내게 되는데 메인 쿼리의 WHERE 절에서는 단일 행 비교 연산자인 =, >, >=, <, <=, <>를 사용해야 합니다.

## 05 중첩질의(nested subquery)

- 평균 급여를 구하는 쿼리문을 서브 쿼리로 사용하여 평균 급여보다 더 많은 급여를 받는 사원을 검색하는 문장은 다음과 같습니다.

```
SELECT ENAME, SAL
FROM EMP
WHERE SAL > ( SELECT AVG(SAL)
               FROM EMP);
```



```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger

SQL> SELECT ENAME, SAL
2 FROM EMP
3 WHERE SAL > ( SELECT AVG(SAL)
4               FROM EMP);

ENAME          SAL
-----
JONES          2975
BLAKE          2850
CLARK          2450
SCOTT          3000
KING           5000
FORD           3000

6 개의 행이 선택되었습니다.

SQL>
```

## 05 중첩질의(nested subquery)

- 비교 연산자

- 부속질의가 반드시 단일 행, 단일 열을 반환해야 하며, 아닐 경우 질의를 처리할 수 없음.
- 평균 주문금액 이하의 주문에 대해서 주문번호와 금액을 보이시오.

```
SELECT orderid, saleprice  
FROM Orders  
WHERE saleprice <= (SELECT AVG(saleprice) FROM Orders);
```

ORDERID	SALEPRICE
1	6000
3	8000
4	6000
9	7000

- 각 고객의 평균 주문금액보다 큰 금액의 주문 내역에 대해서 주문번호, 고객번호, 금액을 보이시오.

```
SELECT orderid, saleprice  
FROM Orders  
WHERE saleprice <= (SELECT AVG(saleprice) FROM Orders);
```

ORDERID	CUSTID	SALEPRICE
2	1	21000
3	2	8000
5	4	20000
8	3	12000
10	3	13000

## 05 중첩질의(nested subquery)

- 다중 행 서브 쿼리는 서브 쿼리에서 반환되는 결과가 하나 이상의 행일 때 사용하는 서브 쿼리입니다. 다중 행 서브 쿼리는 반드시 다중 행 연산자(Multiple Row Operator)와 함께 사용해야 합니다.

종류	의미
IN	메인 쿼리의 비교 조건('=' 연산자로 비교할 경우)이 서브 쿼리의 결과 중에서 하나라도 일치하면 참입니다.
ANY, SOME	메인 쿼리의 비교 조건이 서브 쿼리의 검색 결과와 하나 이상이 일치하면 참입니다.
ALL	메인 쿼리의 비교 조건이 서브 쿼리의 검색 결과와 모든 값이 일치하면 참입니다.
EXIST	메인 쿼리의 비교 조건이 서브 쿼리의 결과 중에서 만족하는 값이 하나라도 존재하면 참입니다.

- 결과가 2개 이상 구해지는 쿼리문을 서브 쿼리로 기술할 경우에는 다중 행 연산자와 함께 사용해야 합니다.
  - 주어진 문제가 3000 이상 받는 사원이 소속된 부서(10번, 20번)와 동일한 부서에서 근무하는 사원이기에 서브 쿼리의 결과 중에서 하나라도 일치하면 참인 결과를 구하는 IN 연산자와 함께 사용되어야 합니다.

```
SELECT ENAME, SAL, DEPTNO
FROM EMP
WHERE DEPTNO IN ( SELECT DISTINCT DEPTNO
FROM EMP
WHERE SAL>=3000);
```



- IN, NOT IN

- IN 연산자는 주질의 속성 값이 부속질의에서 제공한 결과 집합에 있는지 확인(check)하는 역할을 함.
- IN 연산자는 부속질의의 결과 다중 행을 가질 수 있음.
- 주 질의는 WHERE 절에 사용되는 속성 값을 부속질의의 결과 집합과 비교해 하나라도 있으면 참이 됨.
- NOT IN은 이와 반대로 값이 존재하지 않으면 참이 됨.
- 대한민국에 거주하는 고객에게 판매한 도서의 총 판매액을 구하시오.

```
SELECT SUM(saleprice) "total"  
FROM Orders  
WHERE custid IN (    SELECT custid  
                     FROM Customer  
                     WHERE address LIKE '%대한민국%')  
;
```

total
46000

## 05 중첩질의(nested subquery)

- ALL, SOME(ANY)

- ALL은 모두, SOME(ANY)은 어떠한(최소한 하나라도)이라는 의미를 가짐.
- ALL 조건은 메인 쿼리의 비교 조건이 서브 쿼리의 검색 결과와 모든 값이 일치하면 참.
- ANY 조건은 메인 쿼리의 비교 조건이 서브 쿼리의 검색 결과와 하나 이상만 일치하면 참.
- 구문 구조

scalar\_expression {비교연산자(= | < > | != | > | >= | != | < | <= | !<)}

{ALL | SOME | ANY} (부속질의)

- 3번 고객이 주문한 도서의 최고 금액보다 더 비싼 도서를 구입한 주문의 주문번호와 금액을 보이시오.

```
SELECT orderid, saleprice
FROM Orders
WHERE saleprice > ALL (SELECT saleprice
                       FROM Orders
                       WHERE custid='3') ;
```

ORDERID	SALEPRICE
5	20000
2	21000

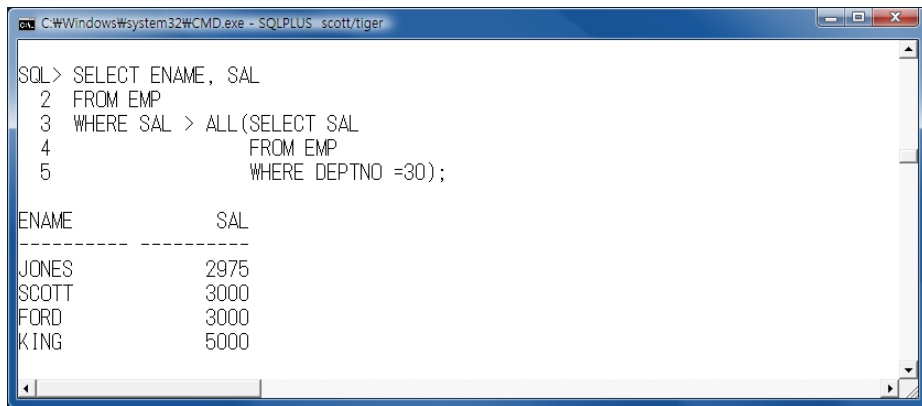
- 다음은 부서번호가 30번인 직원들의 급여 중 가장 작은 값(950)보다 많은 급여를 받는 직원의 이름, 급여를 출력하는 예제를 작성해 봅시다.

```
SELECT ENAME, SAL
FROM EMP
WHERE SAL > ANY ( SELECT SAL
                   FROM EMP
                   WHERE DEPTNO = 30 );
```

## 05 중첩질의(nested subquery)

- 30번 소속 직원들 중에서 급여를 가장 많이 받는 직원보다 더 많은 급여를 받는 사람의 이름, 급여를 출력하는 쿼리문을 작성해 봅시다.

```
SELECT ENAME, SAL
FROM EMP
WHERE SAL > ALL(SELECT SAL
                 FROM EMP
                 WHERE DEPTNO =30);
```



```
SQL> SELECT ENAME, SAL
2 FROM EMP
3 WHERE SAL > ALL(SELECT SAL
4                 FROM EMP
5                 WHERE DEPTNO =30);
```

ENAME	SAL
JONES	2975
SCOTT	3000
FORD	3000
KING	5000

## 05 중첩질의(nested subquery)

- EXISTS, NOT EXISTS

- 데이터의 존재 유무를 확인하는 연산자
- 주 질의에서 부속질의로 제공된 속성의 값을 가지고 부속질의에 조건을 만족하여 값이 존재하면 참이 되고, 주 질의는 해당 행의 데이터를 출력함.
- NOT EXISTS의 경우 이와 반대로 동작함.

- 구문 구조

WHERE [NOT] EXISTS (부속질의)

- EXISTS 연산자로 대한민국에 거주하는 고객에게 판매한 도서의 총 판매액을 구하시오.

```
SELECT SUM(saleprice) "total"
FROM Orders od
WHERE EXISTS (SELECT *
              FROM Customer cs
              WHERE address LIKE '%대한민국%' AND cs.custid=od.custid)
;
```

total
46000