



SQL 활용

중첩 질의문



한국기술교육대학교
온라인평생교육원

학습내용

- 중첩 질의문의 개요
- 다양한 중첩 질의문

학습목표

- 중첩 질의문의 개념에 대해 이해하고, 단일행 서브쿼리와 다중행 서브 쿼리에 대해 설명할 수 있다.
- 다양한 중첩 질의문을 적용하여 두 개 이상의 테이블로부터 데이터를 조회할 수 있다.

● 중첩 질의문의 개요

1. 중첩 질의문의 개념

하나의 SQL문의 결과를 다른 SQL문에 전달함

두 개의 SQL문을 하나의 SQL로 처리함

◆ 조인 질의문과 중첩 질의문

- 이론적으로 중첩 질의문은 조인 구문과 표현능력이 동일함
- 중첩 질의문의 필요성 ⇨ 조인의 필요성과 동일함

◆ 중첩 질의문의 필요성

하나의 SQL 질의문이 하나의 테이블만 검색할 수 있다고 하는 경우

Q 사번이 103인 사원의 부서명을 알고 싶을 때

- ① 사번이 103번인 사원의 부서 번호를 파악함
 - ② 해당 부서 번호와 같은 부서 번호를 가지고 부서명을 부서테이블에서 검색함
- ⇒ 매우 불편함 ⇨ 조인 구문 사용(조인 구문 어려움) ⇨ **중첩 질의문**

Left Screenshot:

```
USE MagicCorp
GO

SELECT DNO
FROM EMPLOYEE
WHERE ENO = 103
```

DNO
1 30

Right Screenshot:

```
USE MagicCorp
GO

SELECT DNAME
FROM DEPARTMENT
WHERE DNO = 30
```

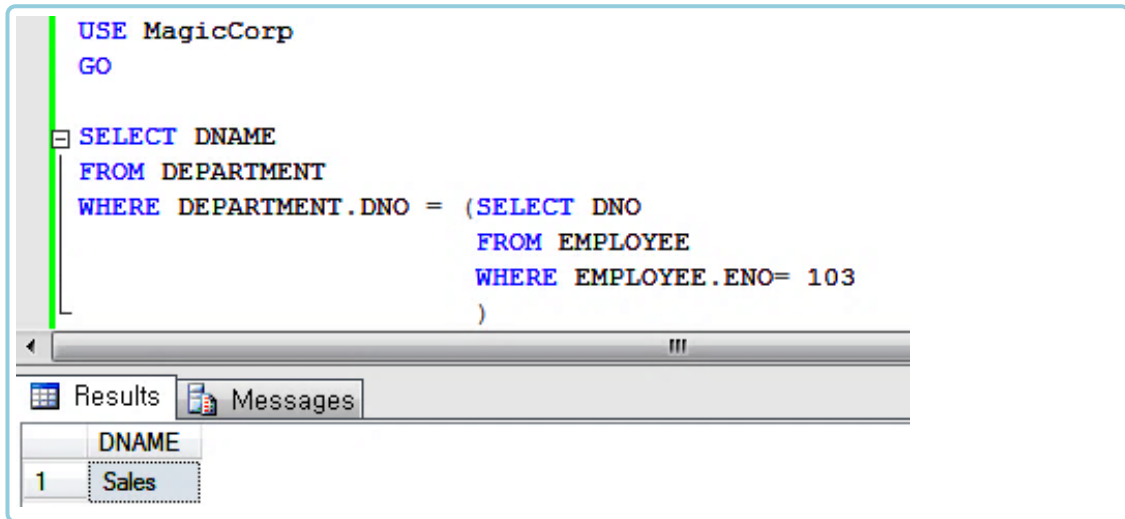
DNAME
1 Sales

● 중첩 질의문의 개요

1. 중첩 질의문의 개념

◆ 중첩 질의문의 표현

- SQL문 안에 SQL문이 포함되어 있음



```
USE MagicCorp
GO

SELECT DNAME
FROM DEPARTMENT
WHERE DEPARTMENT.DNO = (SELECT DNO
                        FROM EMPLOYEE
                        WHERE EMPLOYEE.ENO= 103
                        )
```

The screenshot shows a SQL query execution window. The query is a nested query that selects the department name (DNAME) from the DEPARTMENT table where the department number (DNO) matches the department number of the employee with employee number (ENO) 103. The results pane shows a single row with the department name 'Sales'.

	DNAME
1	Sales

● 중첩 질의문의 개요

2. 단일행 서브 쿼리와 다중행 서브 쿼리

◆ 단일행 서브 쿼리

- 서브 쿼리의 결과로 **하나의 튜플만이 반환됨**
- 서브 쿼리의 검색 조건이 후보키에 연관되어 있을 경우가 많음

◆ 다중행 서브 쿼리

- 서브 쿼리의 결과로 **여러 개의 튜플들이 반환됨**



단일행 서브 쿼리와 다중행 서브 쿼리를 구분해야 할까?

일반적인 비교 연산자인 =, <, <=, >, >=, != 등은 속성값 간의 비교 연산임

- 집합에 대한 비교 연산이 안됨

◆ 단일행 서브 쿼리의 예

Q 사원 번호 110번과 같은 부서에 근무하는 사원들의 사원 번호와 부서번호 검색

- 사원번호가 기본키 임으로 사원번호 110번은 1명 밖에 없음
⇒ 단일행 서브 쿼리
- =, <, <=, >, >=, != 등을 사용할 수 있음

```
USE MagicCorp
GO

SELECT ENO, DNO
FROM EMPLOYEE
WHERE EMPLOYEE.DNO = (SELECT DNO
                       FROM EMPLOYEE
                       WHERE EMPLOYEE.ENO= 110)
```

	ENO	DNO
1	107	10
2	110	10
3	114	10

● 중첩 질의문의 개요

2. 단일행 서브 쿼리와 다중행 서브 쿼리

◆ 다중행 서브 쿼리 - 단일행 비교 연산자 사용 시 오류

Q 봉급이 500이상인 사원과 같은 부서에 근무하는 직원들의 이름, 봉급, 부서번호 구하기

```
USE MagicCorp
GO

SELECT ENAME, SALARY, DNO
FROM EMPLOYEE
WHERE DNO = (SELECT DNO
              FROM EMPLOYEE
              WHERE SALARY >= 500)
```

Results Messages

Msg 512, Level 16, State 1, Line 2
Subquery returned more than 1 value. This is not permitted when the subquery follows =, !

◆ 다중행 비교 연산자

- IN
 - 속성값이 여러 값들 중 하나이기만 하면 참
 - “= OR”의 의미
- ANY 또는 SOME
 - 메인 쿼리 비교 조건에서 서브 쿼리의 결과와 하나라도 일치하면 참
 - IN과의 차이점은 >, >=, <=, <과 같은 범위 비교와도 같이 사용이 가능함
 - = ANY와 = SOME은 IN과 같은 의미임
- ALL
 - 메인 쿼리 비교 조건에서 서브 쿼리의 결과와 모두 일치하면 참

중첩 질의문의 개요

2. 단일행 서브 쿼리와 다중행 서브 쿼리

◆ 다중행 서브 쿼리 : IN

Q 봉급이 500이상인 사원과 같은 부서에 근무하는 직원들의 이름, 봉급, 부서번호 구하기

USE MagicCorp
GO

SELECT ENAME, SALARY, DNO
FROM EMPLOYEE
WHERE DNO IN (SELECT DNO
FROM EMPLOYEE
WHERE SALARY >= 500)

Results

Messages

	ENAME	SALARY	DNO		ENAME	SALARY	DNO
1	e1	300	20	8	e8	500	30
2	e2	250	30	9	e9	1000	20
3	e3	500	30	10	e10	500	10
4	e4	600	20	11	e11	280	30
5	e5	450	30	12	e12	300	20
6	e6	480	30	13	e13	560	20
7	e7	520	10	14	e14	250	10

◆ 다중행 서브 쿼리 : ANY

Q 부서 번호 20에 근무하는 한 직원의 봉급 보다 많은 봉급을 받는 직원들의 이름, 봉급, 부서번호 출력

USE MagicCorp
GO

SELECT ENAME, SALARY, DNO
FROM EMPLOYEE
WHERE SALARY > ANY (SELECT SALARY
FROM EMPLOYEE
WHERE DNO =20)

Results

Messages

	ENAME	SALARY	DNO
1	e3	500	30
2	e4	600	20
3	e5	450	30
4	e6	480	30
5	e7	520	10
6	e8	500	30
7	e9	1000	20
8	e10	500	10
9	e13	560	20

◆ 다중행 서브 쿼리 : ALL

Q 부서 번호 10에 근무하는 모든 직원들의 봉급 보다 많은 봉급을 받는 직원들의 이름, 봉급, 부서번호 출력

USE MagicCorp
GO

Results

Messages

	ENAME	SALARY	DNO
1	e4	600	20
2	e9	1000	20
3	e13	560	20

```
SELECT ENAME, SALARY, DNO
FROM EMPLOYEE
WHERE SALARY > ALL (SELECT SALARY
                    FROM EMPLOYEE
                    WHERE DNO =10)
```

● 중첩 질의문의 개요

2. 단일행 서브 쿼리와 다중행 서브 쿼리

◆ 다중행 비교 연산자

- EXISTS
 - 서브 쿼리의 결과가 하나라도 존재하면 참이 되는 연산자
- NOT EXISTS
 - EXISTS와 상반되는 연산자

Q 봉급과 커미션의 합이 500이 넘는 사원이 존재하면 모든 사원의 이름 출력

```
USE MagicCorp
GO

SELECT ENAME
FROM EMPLOYEE
WHERE EXISTS (SELECT *
              FROM EMPLOYEE
              WHERE SALARY+COMMISSION > 500)
```

Results		Messages	
	ENAME		ENAME
1	e1	8	e8
2	e2	9	e9
3	e3	10	e10
4	e4	11	e11
5	e5	12	e12
6	e6	13	e13
7	e7	14	e14

● 다양한 중첩 질의문

1. 다중 컬럼 서브 쿼리

◆ 다중 컬럼 서브 쿼리란?

- 다중 컬럼 서브 쿼리 : 서브 쿼리의 결과가 여러 개의 속성들로 구성되어 주 쿼리의 조건과 비교하는 서브 쿼리임
 - 복수 개의 서브 쿼리들로 구성됨
 - 메인 쿼리와 서브 쿼리의 비교 대상 칼럼을 분리하여 개별적으로 비교한 후 AND 연산에 의해 최종 결과를 출력함

◆ 다중 컬럼 서브 쿼리의 예

Q 사원번호 101인 사원과 동일 부서에 동일한 급여를 지급받는 직원 구하기

```
SELECT ENO, ENAME, SALARY, DNO
FROM EMPLOYEE
WHERE DNO IN (SELECT DNO
              FROM EMPLOYEE
              WHERE ENO = 101)
AND SALARY IN (SELECT SALARY
               FROM EMPLOYEE
               WHERE ENO = 101)
```

	ENO	ENAME	SALARY	DNO
1	101	e1	300	20
2	112	e12	300	20

● 다양한 중첩 질의문

2. 상호 연관 서브 쿼리

◆ 비상호 연관 서브 쿼리

- 앞서 보았던 서브 쿼리들은 서브 쿼리의 결과가 메인 쿼리에서 검사하는 튜플에는 영향 받지 않고 그 결과가 일정함
 - ⇒ 부서 테이블의 어떤 튜플들을 검색하던지 간에 서브 쿼리

```
SELECT DNO FROM  
EMPLOYEE WHERE  
EMPLOYEE.ENO = 103
```



항상 일관되게 부서번호 30을 반환함

The screenshot shows a SQL query window with the following text:

```
USE MagicCorp  
GO  
  
SELECT DNAME  
FROM DEPARTMENT  
WHERE DEPARTMENT.DNO = (SELECT DNO  
                        FROM EMPLOYEE  
                        WHERE EMPLOYEE.ENO= 103  
                        )
```

Below the query window, the 'Results' tab is active, displaying a single row of data:

DNAME
Sales

● 다양한 중첩 질의문

2. 상호 연관 서브 쿼리

◆ 상호 연관 서브 쿼리

메인 쿼리결과와 서브 쿼리 간에 검색 결과를 교환하는 서브 쿼리

- 메인 쿼리와 서브 쿼리 간의 결과를 교환하기 위하여 서브 쿼리의 WHERE 조건절에서 메인 쿼리의 테이블과 연결함
- 서브 쿼리의 조건절에 메인 쿼리에서 사용하는 테이블의 속성이 나타남



메인 쿼리에서 어떤 튜플에 대한 조건을 비교하는가에
따라서 서브 쿼리의 결과가 다르게 나타남

● 상호 연관 서브 쿼리 사용법

```
SELECT 속성리스트  
FROM table1  
WHERE table1.속성 비교연산자 (  
    SELECT 속성리스트  
    FROM table2  
    WHERE table2.속성 비교연산자 table1 속성)
```

● 주의 사항

- 메인 쿼리에서 table1에 속한 튜플을 하나씩 접근하여 WHERE 절 수행 시 서브 쿼리가 반복적으로 수행됨으로 성능이 매우 떨어질 수 있음
⇒ 조인 구문을 이용하는 것이 더 효율적임

● 다양한 중첩 질의문

2. 상호 연관 서브 쿼리

◆ 상호 연관 서브 쿼리

Q 각 사원에 대하여 관리자와 동일 부서에서 근무하는 직원들의 이름, 급여, 부서 번호 출력

- 사원마다 관리자가 달라짐

<pre>SELECT ENO, ENAME, SALARY, DNO FROM EMPLOYEE E WHERE DNO IN (SELECT DNO FROM EMPLOYEE M WHERE E.MANAGER = M.ENO)</pre>																																												
<table><tr><th colspan="2">Results</th><th colspan="3">Messages</th></tr><tr><th></th><th>ENO</th><th>ENAME</th><th>SALARY</th><th>DNO</th></tr><tr><td>1</td><td>101</td><td>e1</td><td>300</td><td>20</td></tr><tr><td>2</td><td>102</td><td>e2</td><td>250</td><td>30</td></tr><tr><td>3</td><td>103</td><td>e3</td><td>500</td><td>30</td></tr><tr><td>4</td><td>105</td><td>e5</td><td>450</td><td>30</td></tr><tr><td>5</td><td>106</td><td>e6</td><td>480</td><td>30</td></tr><tr><td>6</td><td>108</td><td>e8</td><td>500</td><td>30</td></tr></table>					Results		Messages				ENO	ENAME	SALARY	DNO	1	101	e1	300	20	2	102	e2	250	30	3	103	e3	500	30	4	105	e5	450	30	5	106	e6	480	30	6	108	e8	500	30
Results		Messages																																										
	ENO	ENAME	SALARY	DNO																																								
1	101	e1	300	20																																								
2	102	e2	250	30																																								
3	103	e3	500	30																																								
4	105	e5	450	30																																								
5	106	e6	480	30																																								
6	108	e8	500	30																																								

● 다양한 중첩 질의문

3. 중첩 질의문 작성 시 주의점

◆ 다중행 서브 쿼리 시 단일행 비교 연산자와 사용하는 경우

- 중첩 질의문 사용 시 오류가 없도록 IN, ANY, ALL을 기본적으로 사용함

```
USE MagicCorp
GO

SELECT ENAME, SALARY, DNO
FROM EMPLOYEE
WHERE DNO = (SELECT DNO
             FROM EMPLOYEE
             WHERE SALARY >= 500)
```

Results Messages

Msg 512, Level 16, State 1, Line 2
Subquery returned more than 1 value. This is not permitted when the subquery follows =, !=

- 서브 쿼리 내에서는 ORDER BY 절을 사용하면 안됨

```
SELECT ENO, ENAME, SALARY, DNO
FROM EMPLOYEE
WHERE DNO IN (SELECT DNO
             FROM EMPLOYEE
             WHERE ENO = 101
             ORDER BY DNO)
```

Messages

Msg 1033, Level 15, State 1, Line 7
The ORDER BY clause is invalid in views, inline functions, derived tables, subqueries, and common table express

- 서브 쿼리의 결과가 NULL일 경우, 메인 쿼리의 결과 또한 NULL임

```
SELECT ENO, ENAME, SALARY, DNO
FROM EMPLOYEE
WHERE DNO IN (SELECT DNO
             FROM EMPLOYEE
             WHERE ENO = 500)
```

Results Messages

ENO	ENAME	SALARY	DNO
-----	-------	--------	-----

● 다양한 중첩 질의문

3. 중첩 질의문 작성 시 주의점

◆ 다중행 서브 쿼리 시 단일행 비교 연산자와 사용하는 경우

- 서브 쿼리가 NULL을 반환할 경우, 메인 쿼리에서 결과를 생성하고 싶으면 “NOT EXISTS” 를 사용함

```
SELECT ENO, ENAME, SALARY, DNO
FROM EMPLOYEE
WHERE NOT EXISTS (SELECT DNO
                  FROM EMPLOYEE
                  WHERE ENO = 500)
```

Results

Messages

	ENO	ENAME	SALARY	DNO		ENO	ENAME	SALARY	DNO
1	101	e1	300	20	8	108	e8	500	30
2	102	e2	250	30	9	109	e9	1000	20
3	103	e3	500	30	10	110	e10	500	10
4	104	e4	600	20	11	111	e11	280	30
5	105	e5	450	30	12	112	e12	300	20
6	106	e6	480	30	13	113	e13	560	20
7	107	e7	520	10	14	114	e14	250	10

핵심요약

1. 중첩 질의문의 개요

■ 중첩 질의문의 개념

- 하나의 SQL문의 결과를 다른 SQL문에 전달함
- 두 개의 SQL문을 하나의 SQL로 처리함
- 이론적으로 중첩 질의문은 조인 구문과 표현능력이 동일함
- SQL문 안에 SQL문이 포함되어 있음

■ 단일행 서브 쿼리와 다중행 서브 쿼리

- 단일행 서브 쿼리
 - 서브 쿼리의 결과로 하나의 튜플만이 반환됨
- 다중행 서브 쿼리
 - 서브 쿼리의 결과로 여러 개의 튜플들이 반환됨
- 단일행 서브 쿼리와 다중행 서브 쿼리를 구분해야 하는 이유
 - 일반적인 비교 연산자인 =, <, <=, >, >=, !=등은 속성값 간의 비교 연산임

핵심요약

1. 중첩 질의문의 개요

■ 단일행 서브 쿼리와 다중행 서브 쿼리

■ 다중행 비교 연산자

① IN

- 속성값이 여러 값들 중 하나이기만 하면 참
- “= OR”의 의미

② ANY 또는 SOME

- 메인 쿼리 비교 조건에서 서브 쿼리의 결과와 하나라도 일치하면 참
- IN과의 차이점은 >, >=, <=, <과 같은 범위 비교와도 같이 사용 가능함

③ ALL

- 메인 쿼리 비교 조건에서 서브 쿼리의 결과와 모두 일치하면 참

④ EXISTS 연산자

- 서브 쿼리의 결과가 하나라도 존재하면 참이 되는 연산자

⑤ NOT EXISTS

- EXISTS와 상반되는 연산자

핵심요약

2. 다양한 중첩 질의문

■ 다중 컬럼 서브 쿼리

- 서브 쿼리의 결과가 여러 개의 속성들로 구성되어 주 쿼리의 조건과 비교하는 서브 쿼리임
- 복수 개의 서브 쿼리들로 구성됨
- 메인 쿼리와 서브 쿼리의 비교 대상 칼럼을 분리하여 개별적으로 비교한 후 AND 연산에 의해 최종 결과를 출력함

■ 상호 연관 서브 쿼리

- 비상호 연관 서브 쿼리
 - 서브 쿼리의 결과가 메인 쿼리에서 검사하는 튜플에는 영향 받지 않고 그 결과가 일정함
- 상호 연관 서브 쿼리
 - 메인 쿼리결과 서브 쿼리 간에 검색 결과를 교환하는 서브 쿼리
- 주의 사항
 - 메인 쿼리에서 table1에 속한 튜플을 하나씩 접근하여 WHERE 절 수행 시 서브 쿼리가 반복적으로 수행됨으로 성능이 매우 떨어질 수 있음

■ 중첩 질의문 작성 시 주의점

- 중첩 질의문 사용 시 오류가 없도록 IN, ANY, ALL을 기본적으로 사용함
- 서브 쿼리 내에서는 ORDER BY 절을 사용하면 안됨
- 서브 쿼리의 결과가 NULL일 경우, 메인 쿼리의 결과 또한 NULL임
- 서브 쿼리가 NULL을 반환할 경우, 메인 쿼리에서 결과를 생성하고 싶으면 “NOT EXISTS” 를 사용함