제08장

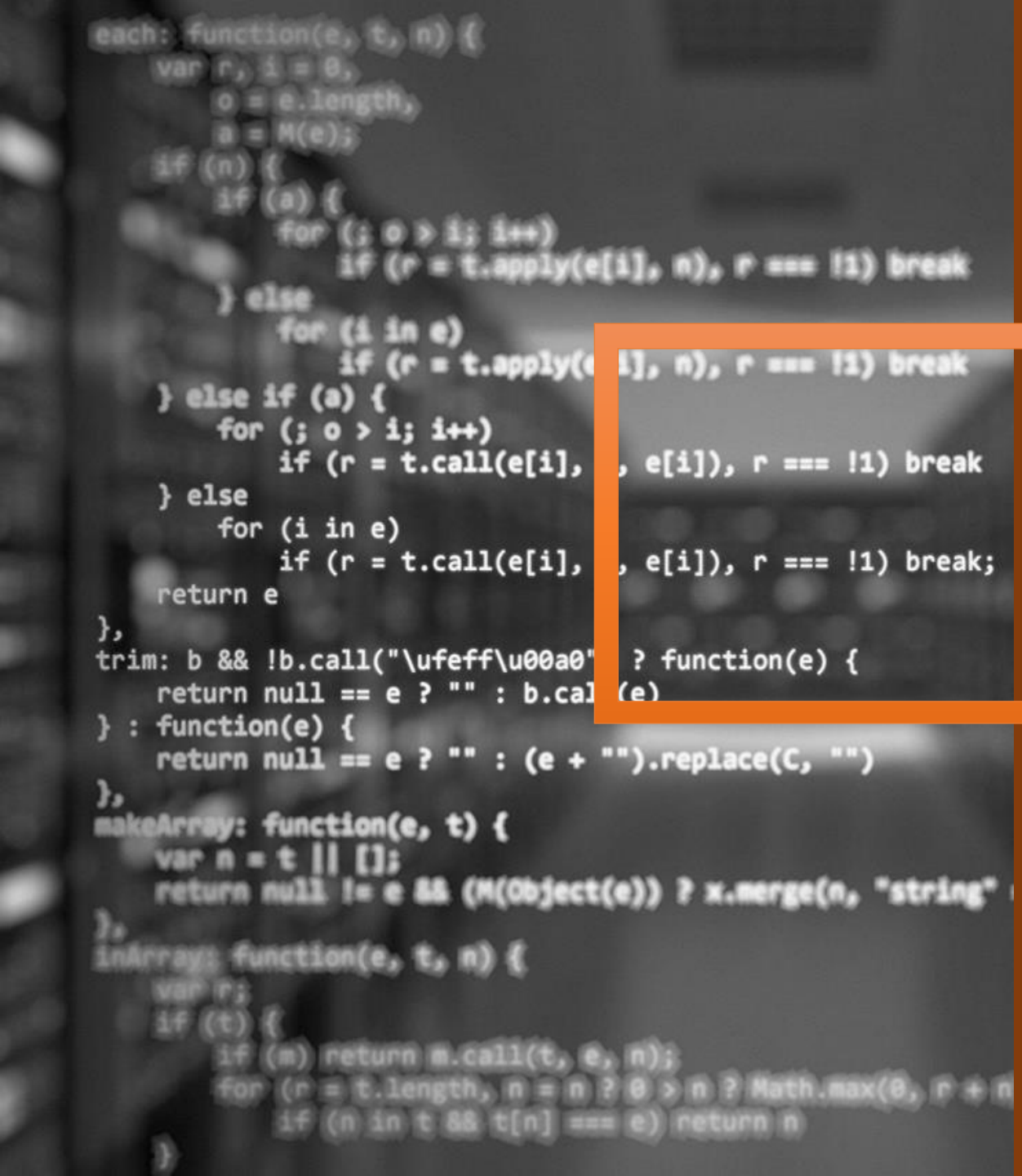
조인과 서브쿼리

MySQL

학습목표

1. 조인에 대해서 알 수 있다.
2. 서브쿼리에 대해서 알 수 있다.

목차


each: function(e, t, n) {
var r, i = 0,
o = e.length,
a = M(e);
if (n) {
if (a) {
for (; o > i; i++)
if (r = t.apply(e[i], n), r === !1) break
} else
for (i in e)
if (r = t.apply(e[i], n), r === !1) break
} else if (a) {
for (; o > i; i++)
if (r = t.call(e[i], e[i]), r === !1) break
} else
for (i in e)
if (r = t.call(e[i], e[i]), r === !1) break;
return e
},
trim: b && !b.call("\ufeff\u00a0") ? function(e) {
return null == e ? "" : b.call(e)
} : function(e) {
return null == e ? "" : (e + "").replace(C, "")
},
makeArray: function(e, t) {
var n = t || [];
return null != e && (M(Object(e)) ? x.merge(n, "string"
},
inArray: function(e, t, n) {
var r;
if (t) {
if (m) return m.call(t, e, n);
for (r = t.length, n = n ? 0 > n ? Math.max(0, r + n
if (n in t && t[n] === e) return n
}

1. 조인

조인

조인

- Join
- 2개 이상의 테이블을 동시에 조회하는 방식
- 조회할 때 필요한 정보가 여러 테이블에 존재하는 경우에 사용함

조인 종류

- 크로스 조인 (Cross Join)
- 내부 조인 (Inner Join)
- 외부 조인 (Outer Join)

조인 주의사항

- 두 테이블에 동일한 이름의 칼럼이 존재하는 경우 반드시 어느 테이블의 칼럼인지 명시해야 함
  - 이 때 테이블명은 별명을 부여하고 사용하는 것이 일반적임
  - 예시) dept_id 칼럼이 두 테이블에 모두 존재하는 경우 발생하는 오류 메시지
    - Error Code : 1052. Column 'dept_id ' in field list is ambiguous

- 크로스 조인을 제외하면 일대다 관계로 연결된 테이블을 대상으로 조인 처리를 할 수 있음

조인 형식

- 조인 기본 문법

```
SELECT 조회_칼럼
FROM 첫_번째_테이블
  JOIN 두_번째 테이블
  ON 조인_조건
WHERE 조회_조건
```

크로스 조인

크로스 조인

- 상호 조인 또는 Cartesion Product (카테젼 곱)이라고도 함
- 첫 번째 테이블의 모든 행을 두 번째 테이블의 모든 행과 조인하는 방식
- 크로스 조인을 수행할 두 테이블은 일대다 관계를 가지지 않아도 상관 없음
- A 테이블에 10개 행, B 테이블에 5개 행이 있다고 가정하면 두 테이블의 크로스 조인 결과 행은 50개임(10 X 5)
- 시뮬레이션 진행을 위해서 대용량의 테스트 데이터를 생성하는 경우 활용 가능
- 조인 조건을 생략하거나 잘못 작성하면 CROSS JOIN으로 처리됨

형식

- SELECT 조회_칼럼 FROM 첫_번째_테이블 CROSS JOIN 두_번째_테이블;

크로스 조인

- tbl_department

dept_id	dept_name	location
1	영업부	대구
2	인사부	서울

- tbl_employee

emp_id	emp_name	dept_id	position	gender	hire_date	salary
1001	구창민	1	과장	M	1995-05-01	5000000
1002	김민서	1	사원	M	2017-09-01	2500000
1003	이은영	2	부장	F	1990-09-01	5500000
1004	한성일	2	과장	M	1993-04-01	5000000

- Query 문

```
SELECT e.emp_id, e.emp_name, d.dept_name
  FROM tbl_department d
 CROSS JOIN tbl_employee e;
```

- Query 결과

emp_id	emp_name	dept_name
1001	구창민	영업부
1002	김민서	영업부
1003	이은영	영업부
1004	한성일	영업부
1001	구창민	인사부
1002	김민서	인사부
1003	이은영	인사부
1004	한성일	인사부

내부 조인

내부 조인

- Inner Join
- 조인 조건으로 사용된 두 테이블에 공통적으로 존재하는 값만 결합됨
- 두 테이블 중 어느 한 쪽에만 존재하는 값은 조회할 수 없음

형식

- SELECT 조회_칼럼
  FROM 테이블1 INNER JOIN 테이블2
  ON 조인조건;
- SELECT 조회_칼럼
  FROM 테이블1 INNER JOIN 테이블2
  ON 조인조건 INNER JOIN 테이블3
  ON 조인조건;

내부 조인

- tbl_department

dept_id	dept_name	location
1	영업부	대구
2	인사부	서울
3	총무부	대구
4	기획부	서울

- tbl_employee

emp_id	emp_name	dept_id	position	gender	hire_date	salary
1001	구창민	1	과장	M	1995-05-01	5000000
1002	김민서	1	사원	M	2017-09-01	2500000
1003	이은영	2	부장	F	1990-09-01	5500000
1004	한성일	2	과장	M	1993-04-01	5000000

- Query 문

```
SELECT e.emp_id, e.emp_name, d.dept_name
  FROM tbl_department d
 INNER JOIN tbl_employee e
    ON d.dept_id = e.dept_id;
```

- Query 결과

emp_id	emp_name	dept_name
1001	구창민	영업부
1002	김민서	영업부
1003	이은영	인사부
1004	한성일	인사부

* 총무부와 기획부는 tbl_department 에만 존재하므로 조회가 되지 않음

외부 조인

외부 조인

- Outer Join
- 한 테이블의 내용은 모두 포함되고, 다른 한 테이블의 내용은 일치하는 정보만 포함됨
- 두 테이블 중 어느 한 쪽에만 존재하는 값도 함께 조회할 수 있음
- 왼쪽 테이블의 내용을 모두 포함하는 경우 '왼쪽 외부 조인(Left Outer Join)'을 수행함
- 오른쪽 테이블의 내용을 모두 포함하는 경우 '오른쪽 외부 조인(Right Outer Join)'을 수행함

형식

- SELECT 조회_칼럼
  FROM 테이블1 [LEFT | RIGHT] OUTER JOIN 테이블2
  ON 조인조건;
- SELECT 조회_칼럼
  FROM 테이블1 [LEFT | RIGHT] OUTER JOIN 테이블2
  ON 조인조건 [LEFT | RIGHT] OUTER JOIN 테이블3
  ON 조인조건;

외부 조인

- tbl_department

dept_id	dept_name	location
1	영업부	대구
2	인사부	서울
3	총무부	대구
4	기획부	서울

- tbl_employee

emp_id	emp_name	dept_id	position	gender	hire_date	salary
1001	구창민	1	과장	M	1995-05-01	5000000
1002	김민서	1	사원	M	2017-09-01	2500000
1003	이은영	2	부장	F	1990-09-01	5500000
1004	한성일	2	과장	M	1993-04-01	5000000

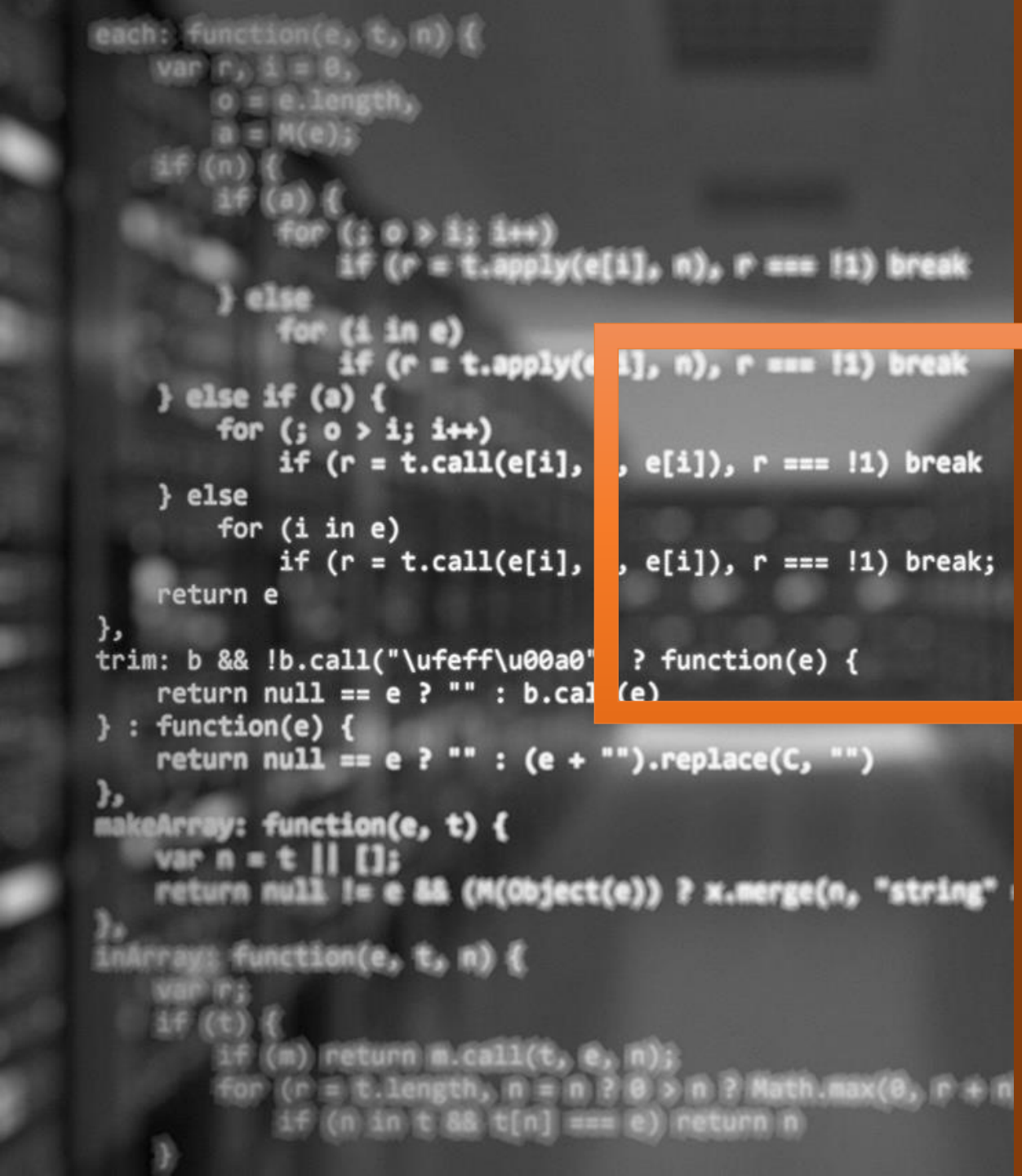
- Query 문

```
SELECT e.emp_id, e.emp_name, d.dept_name
  FROM tbl_department d
  LEFT OUTER JOIN tbl_employee e
    ON d.dept_id = e.dept_id;
```

- Query 결과

emp_id	emp_name	dept_name
1001	구창민	영업부
1002	김민서	영업부
1003	이은영	인사부
1004	한성일	인사부
null	null	총무부
null	null	인사부

* 총무부와 기획부는 tbl_department 에만 존재하지만 함께 조회됨


each: function(e, t, n) {
var r, i = 0,
o = e.length,
a = M(e);
if (n) {
if (a) {
for (; o > i; i++)
if (r = t.apply(e[i], n), r === !1) break
} else
for (i in e)
if (r = t.apply(e[i], n), r === !1) break
} else if (a) {
for (; o > i; i++)
if (r = t.call(e[i], e[i]), r === !1) break
} else
for (i in e)
if (r = t.call(e[i], e[i]), r === !1) break;
return e
},
trim: b && !b.call("\ufeff\u00a0") ? function(e) {
return null == e ? "" : b.call(e)
} : function(e) {
return null == e ? "" : (e + "").replace(C, "")
},
makeArray: function(e, t) {
var n = t || [];
return null != e && (M(Object(e)) ? x.merge(n, "string"
},
inArray: function(e, t, n) {
var r;
if (t) {
if (m) return m.call(t, e, n);
for (r = t.length, n = n ? 0 > n ? Math.max(0, r + n
if (n in t && t[n] === e) return n
}

2. 서브쿼리

서브쿼리

서브쿼리

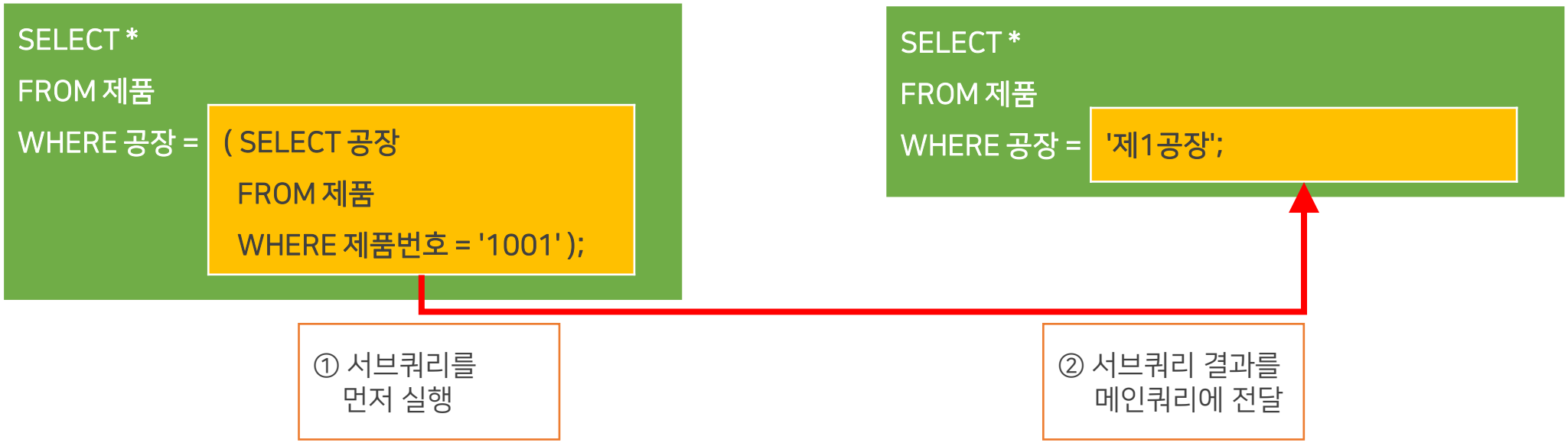
- Subquery
- 어떤 SQL문의 결과를 다른 SQL문에게 전달하기 위해 두 개 이상의 SQL문을 하나의 SQL문으로 연결하는 방법
- 메인쿼리(Main Query)에 포함되는 쿼리를 서브쿼리(Sub Query)라고 함
- 서브쿼리를 먼저 실행한 뒤 그 결과를 메인쿼리로 전달하는 방식으로 동작함

서브쿼리의 위치에 따른 구분

- 중첩 서브쿼리 (Nested Subquery)
  - 단일 행 서브쿼리
  - 다중 행 서브쿼리
- 스칼라 서브쿼리 (Scalar Subquery)
- 인라인 뷰 (Inline View)

서브쿼리가 필요한 상황

- 제품아이디가 '1001'인 제품이 생산되는 공장에서 생산되는 제품들의 목록을 조회하기
  - 제품번호 '1001'인 제품이 생산되는 공장을 조회한다. (서브쿼리)
  - 해당 공장에서 생산되는 제품들의 목록을 조회한다. (메인쿼리)



중첩 서브쿼리

중첩 서브쿼리

- WHERE 절에서 사용되는 서브쿼리
- 어떤 값을 반환하는 서브쿼리
- 서브쿼리가 반환하는 값의 개수에 따라 2가지로 구분함
  - 단일 행 서브쿼리
  - 다중 행 서브쿼리

```
SELECT 조회_칼럼
FROM 테이블
WHERE 조건_칼럼 = (중첩_서브쿼리)
```

단일 행 서브쿼리

단일 행 서브쿼리

- 실행 결과가 하나의 값인 서브쿼리

단일 행 서브쿼리 조건

- 기본키(PK)나 고유키(UNIQUE)를 가진 칼럼과 동등비교(=)한 서브쿼리
- 그룹 함수를 사용하여 어떤 하나의 값을 얻어낸 서브쿼리

단일 행 서브쿼리 연산자

종류	의미	종류	의미
>	크다	>=	크거나 같다
<	작다	<=	작거나 같다
=	같다	!=	같지 않다

단일 행 서브쿼리

- tbl_employee

emp_id	emp_name	dept_id	position	gender	hire_date	salary
1001	구창민	1	과장	M	1995-05-01	5000000
1002	김민서	1	사원	M	2017-09-01	2500000
1003	이은영	2	부장	F	1990-09-01	5500000
1004	한성일	2	과장	M	1993-04-01	5000000

- 평균 연봉 이상을 받는 사원 조회하기

```
SELECT * FROM tbl_employee WHERE salary > (SELECT AVG(salary) FROM tbl_employee);

                                          ↓

SELECT * FROM tbl_employee WHERE salary > (4500000);
```

다중 행 서브쿼리

다중 행 서브쿼리

- 실행 결과가 여러 값인 서브쿼리
- 단일 행 서브쿼리의 연산자를 사용할 수 없음

다중 행 서브쿼리 연산자

종류	개념	의미	기본 형식
IN	or	서브쿼리의 결과 중에서 하나라도 일치하면 true	WHERE 칼럼 IN (서브쿼리) WHERE 칼럼 NOT IN (서브쿼리)
ANY	or	서브쿼리의 결과 중에서 하나라도 일치하면 true	WHERE 칼럼 = ANY(서브쿼리) WHERE 칼럼 != ANY(서브쿼리) WHERE 칼럼 < ANY(서브쿼리) WHERE 칼럼 <= ANY(서브쿼리) WHERE 칼럼 > ANY(서브쿼리) WHERE 칼럼 >= ANY(서브쿼리)
ALL	and	서브쿼리의 모든 결과와 일치하면 true	WHERE 칼럼 = ALL(서브쿼리) WHERE 칼럼 != ALL(서브쿼리) WHERE 칼럼 < ALL(서브쿼리) WHERE 칼럼 <= ALL(서브쿼리) WHERE 칼럼 > ALL(서브쿼리) WHERE 칼럼 >= ALL(서브쿼리)

다중 행 서브쿼리

- tbl_employee

emp_id	emp_name	dept_id	position	gender	hire_date	salary
1001	구창민	1	과장	M	1995-05-01	5000000
1002	김민서	1	사원	M	2017-09-01	2500000
1003	이은영	2	부장	F	1990-09-01	5500000
1004	한성일	2	과장	M	1993-04-01	5000000

- 부서번호가 1인 부서의 사원들과 같은 직급을 가진 사원 조회하기

```
SELECT * FROM tbl_employee WHERE position IN (SELECT position
                                                FROM tbl_employee
                                               WHERE dept_id = 1);

                                                ↓

SELECT * FROM tbl_employee WHERE position IN ('과장', '사원');
```

스칼라 서브쿼리

스칼라 서브쿼리

- SELECT 절에서 사용되는 서브쿼리
- 하나의 값만 반환하는 서브쿼리
- 서브쿼리의 조건을 일치하는 데이터가 없는 경우 NULL을 반환함
- 다른 테이블에서 어떤 값을 가져올 때 사용함

```
SELECT 조회_칼럼, (스칼라_서브쿼리)
FROM 테이블
WHERE 조건_칼럼 = 값
```

스칼라 서브쿼리

- tbl_employee

emp_id	emp_name	dept_id	position	gender	hire_date	salary
1001	구창민	1	과장	M	1995-05-01	5000000
1002	김민서	1	사원	M	2017-09-01	2500000
1003	이은영	2	부장	F	1990-09-01	5500000
1004	한성일	2	과장	M	1993-04-01	5000000

- 사원번호가 1001인 사원의 이름, 급여, 전체사원평균급여 조회하기

```
SELECT emp_name, salary, (SELECT AVG(salary) FROM tbl_employee) AS '평균급여'
  FROM tbl_employee WHERE emp_id = 1001;


                                                ↓


SELECT emp_name, salary, (4500000) AS '평균급여'
  FROM tbl_employee WHERE emp_id = 1001;
```

스칼라 서브쿼리

- tbl_department

dept_id	dept_name	location
1	영업부	대구
2	인사부	서울
3	총무부	대구
4	기획부	서울

- tbl_employee

emp_id	emp_name	dept_id	position	gender	hire_date	salary
1001	구창민	1	과장	M	1995-05-01	5000000
1002	김민서	1	사원	M	2017-09-01	2500000
1003	이은영	2	부장	F	1990-09-01	5500000
1004	한성일	2	과장	M	1993-04-01	5000000

- 사원번호가 1001인 사원의 사원명과 부서명 조회하기

```
SELECT emp_name, (SELECT dept_name FROM tbl_department WHERE dept_id = e.dept_id)
  FROM tbl_employee e WHERE emp_id = 1001;
```

↓

```
SELECT emp_name, ('영업부')
  FROM tbl_employee e WHERE emp_id = 1001;
```

메인쿼리의 값(e.dept_id)을
서브쿼리에서 사용하는
'상호연관 서브쿼리' 형식의
'스칼라 서브쿼리'

인라인 뷰

인라인 뷰

- FROM 절에서 사용되는 쿼리
- 어떤 테이블 형식의 결과를 반환하는 쿼리
- 인라인 뷰에서 조회한 칼럼만 메인쿼리에서 조회할 수 있음
- 인라인 뷰는 반드시 별명(Alias)을 지정해야만 함
- SELECT 문의 수행 순서 변경을 위해서 사용함
  - SELECT 문에서는 FROM 절이 가장 먼저 처리됨
  - 먼저 처리하고 싶은 작업을 인라인 뷰로 작업하면 가장 먼저 처리됨

```
SELECT  조회_칼럼
FROM  (인라인 뷰)
WHERE 조건_칼럼 = 값
```

인라인 뷰

- tbl_employee

emp_id	emp_name	dept_id	position	gender	hire_date	salary
1001	구창민	1	과장	M	1995-05-01	5000000
1002	김민서	1	사원	M	2017-09-01	2500000
1003	이은영	2	부장	F	1990-09-01	5500000
1004	한성일	2	과장	M	1993-04-01	5000000

- 부서번호가 1인 사원의 사원명 조회하기

```
SELECT e.emp_name FROM (SELECT *
                          FROM tbl_employee
                         WHERE dept_id = 1) e;
```

인라인 뷰 결과를
FROM 절의 테이블로 사용

↓

emp_id	emp_name	dept_id	position	gender	hire_date	salary
1001	구창민	1	과장	M	95-05-01	5000000
1002	김민서	1	사원	M	17-09-01	2500000

INSERT 문과 서브쿼리

- 서브쿼리의 결과를 한 번에 INSERT 할 때 사용함
- INSERT 문의 VALUES 절 대신 서브쿼리를 작성함
- 형식
  - INSERT INTO 테이블(칼럼, 칼럼, ...) (SELECT 칼럼, 칼럼, ... FROM 테이블 WHERE 조건식);

UPDATE 문과 서브쿼리

- UPDATE 문의 SET 절이나 WHERE 절에서 서브쿼리를 작성함

- 형식
  - SET 절
    - UPDATE 테이블
      SET 칼럼 = (SELECT 칼럼 FROM 테이블 WHERE 조건식)
      WHERE 조건식;
    - UPDATE 테이블
      SET (칼럼1, 칼럼2) = (SELECT 칼럼1, 칼럼2 FROM 테이블 WHERE 조건식)
      WHERE 조건식;
  - WHERE 절
    - UPDATE 테이블 SET 칼럼 = 값 WHERE 칼럼 = (단일 행 서브쿼리);
    - UPDATE 테이블 SET 칼럼 = 값 WHERE 칼럼 IN (다중 행 서브쿼리);

DELETE 문과 서브쿼리

- DELETE 문의 WHERE 절에서 서브쿼리를 작성할 수 있음

- 형식
  - DELETE FROM 테이블 WHERE 칼럼 = (단일 행 서브쿼리);
  - DELETE FROM 테이블 WHERE 칼럼 IN (다중 행 서브쿼리);
  - DELETE FROM 테이블
    WHERE (칼럼1, 칼럼2) = (SELECT 칼럼1, 칼럼2 FROM 테이블 WHERE 조건식);