

# 데이터조화 (고급)

## 목차

---

I. 서브 쿼리

II. 특수 쿼리

III. 테이블의 결합

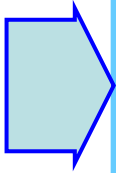
IV. SQL 문 만들기 연습

# 1. 서브 쿼리

- 중첩 질의(nested query)
  - 외부 질의의 WHERE절에 다시 SELECT ... FROM ... WHERE 형태로 포함된 SELECT문
  - 부질의(subquery)라고 함
  - INSERT, DELETE, UPDATE문에도 사용될 수 있음
  - 중첩 질의의 결과로 한 개의 스칼라값(단일 값), 한 개의 애트리뷰트로 이루어진 릴레이션, 여러 애트리뷰트로 이루어진 릴레이션이 반환될 수 있음

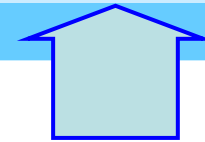
# 1. 서브 쿼리

외부질의



```
SELECT ...  
FROM ...  
WHERE ...
```

```
(SELECT ...  
FROM ...  
WHERE ... );
```



중첩질의

[그림] 중첩 질의의 구조

# 1. 서브 쿼리

예: 한 개의 스칼라 값이 반환되는 경우

질의 : 박영권과 같은 직급을 갖는 모든 사원들의 이름과 직급을 검색하라.  
아래와 같이 각 사원의 이름과 직급을 검색하는 질의(외부질의)가 필요하다

```
SELECT      FIRST_NAME, JOB_ID
FROM        EMPLOYEES
WHERE       JOB_ID =
```

아래와 같이 박영권의 직급을 검색하는 질의(중첩질의)가 필요하다.

```
SELECT      JOB_ID
FROM        EMPLOYEES
WHERE       FIRST_NAME = '박영권'
```

두 질의를 하나로 결합하면 박영권과 같은 직급을 갖는 모든 사원들의 이름과 직급을 검색할 수 있다.

```
SELCECT     FIRST_NAME, JOB_ID
FROM        EMPLOYEES
WHERE       JOB_ID =
```

```
(SELECT      JOB_ID
FROM        EMPLOYEES
WHERE       FIRST_NAME = '박영권' ) ;
```

과장

# 1. 서브 쿼리

- 한 개의 애트리뷰트로 이루어진 릴레이션이 반환되는 경우
  - 중첩 질의의 결과로 한 개의 애트리뷰트로 이루어진 다수의 튜플들이 반환될 수 있음
  - 외부 질의의 WHERE절에서 IN, ANY(SOME), ALL, EXISTS와 같은 연산자를 사용해야 함
    - IN : 한 애트리뷰트가 값들의 집합에 속하는가를 테스트할 때 사용됨
    - ANY : 한 애트리뷰트가 값들의 집합에 속하는 하나 이상의 값들과 어떤 관계를 갖는가를 테스트하는 경우에 사용
    - ALL : 한 애트리뷰트가 값들의 집합에 속하는 모든 값들과 어떤 관계를 갖는가를 테스트하는 경우에 사용
    - EXISTS : 여러 애트리뷰트로 이루어진 릴레이션이 반환되는 경우에 사용

# 1. 서브 쿼리

예: IN

( 3426 IN 

2106
3426
3011

 )은 참이다

( 1365 IN 

2106
3426
3011

 )은 거짓이다

( 1365 NOT IN 

2106
3426
3011

 )은 참이다

# 1. 서브 쿼리

예: ANY

( 3000000 < ANY 

2500000
3000000
4000000

 )은 참이다

( 4000000 < ANY 

2500000
3000000
4000000

 )은 거짓이다

- 다수의 비교값 중 한개라도 만족하면 true 이다.
- IN 과 다른점은 비교 연산자를 사용한다는 점이다.



# 1. 서브 쿼리

예: ALL

( 3000000 < ALL	2500000	)은 거짓이다	( 3000000 = ALL	2500000	)은 거짓이다
	3000000			3000000	
	4000000			4000000	

( 1500000 < ALL	2500000	)은 참이다	( 1500000 <> ALL	2500000	)은 참이다
	3000000			3000000	
	4000000			4000000	

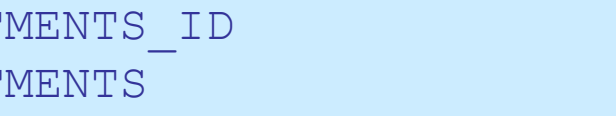
- 전체 값을 비교하여 모두 만족해야만 true 이다.

# 1. 서브 쿼리

## 예: IN을 사용한 질의

질의 : 영업부나 개발부에 근무하는 직원들의 이름을 검색하라.

```
SELECT      FIRST_NAME
FROM        EMPLOYEES
WHERE       DEPARTMENTS_ID IN (1, 3)
```



```
(SELECT      DEPARTMENTS_ID
FROM        DEPARTMENTS
WHERE       DEPARTMENT_NAME = '영업' OR
            DEPARTMENT_NAME = '개발' );
```

(쇼핑몰) 문제연습

카테고리가 스포츠 또는 가전 인 상품의 상품명, 판매가, 카테고리 코드를 검색하시오

# 1. 서브 쿼리

## 예: 중첩질의가 없는 질의로 표현

이 질의를 중첩질의를 사용하지 않은 다음과 같은 질의로 나타낼 수 있다. 실제로 중첩질의를 사용하여 표현된 대부분의 질의가 없는 질의로 표현할 수 있다.

```
SELECT      FIRST_NAME
FROM        EMPLOYEES E, DEPARTMENTS D
WHERE       E.DEPARTMENTS_ID = D.DEPARTMENTS_ID
AND
(D.DEPARTMENT_NAME = '영업'
OR D.DEPARTMENT_NAME = '개발' );
```

(쇼핑몰) 문제 연습  
앞에 문제를 조인을 이용하여 출력

FIRST_NAME
박영권
이수민
최종철
김상원

## 1. 서브 쿼리

- 여러 애트리뷰트들로 이루어진 릴레이션이 반환되는 경우
  - 중첩 질의의 결과로 여러 애트리뷰트들로 이루어진 릴레이션이 반환되는 경우에는 EXISTS 연산자를 사용하여 중첩 질의의 결과가 빈 릴레이션인지 여부를 검사함
  - 중첩 질의의 결과가 빈 릴레이션이 아니면 참이 되고, 그렇지 않으면 거짓

# 1. 서브 쿼리

- 여러 애트리뷰트들로 이루어진 릴레이션이 반환되는 경우

- EXISTS 구문

SELECT	애트리뷰트(들)
FROM	릴레이션(들)
[WHERE	EXISTS (또는 NOT EXISTS)
	[서브 쿼리] ]

- 사용하는 용도 (마스터 : 디테일 = 1:N ) 경우 사용

회사코드 마스터  
COMPANY\_MST

COMPANY_CODE	회사코드	PK
COMPANY_NAME	회사명	

	COMPANY_CODE	COMPANY_NAME
1	1	삼성전자
2	2	SK하이닉스
3	3	현대자동차
4	4	NAVER

회사코드 디테일  
COMPANY\_DTL

COMPANY_CODE	회사코드	PK
DEPT_CODE	부서코드	PK
DEPT_EMP	부서인원	

	COMPANY_CODE	DEPT_CODE	DEPT_EMP
1	1	1	100
2	1	2	300
3	2	1	50

# 1. 서브 쿼리

- 여러 애트리뷰트들로 이루어진 릴레이션이 반환되는 경우
  - COMPANY\_CODE를 공유하고있는 1,2 삼성전자와 현대차만 출력됩니다
  - 해당되는 경우만 삭제하는 경우도 사용된다

```
SELECT * FROM COMPANY_MST MST
WHERE EXISTS
(
    SELECT * FROM COMPANY_DTL DTL
    WHERE MST.COMPANY_CODE = DTL.COMPANY_CODE
)
```

# 1. 서브 쿼리

## 예: EXISTS를 사용한 질의

질의: 영업부나 개발부에 근무하는 직원들의 이름을 검색하라.

```
SELECT FIRST_NAME
FROM EMPLOYEES E
WHERE EXISTS
    (SELECT *
     FROM DEPARTMENTS D
     WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID
     AND
        (DEPARTMENT_NAME = '영업'
         OR DEPARTMENT_NAME = '개발')
    );
```

FIRST_NAME
박영권
이수민
최종철
김상원

내부쿼리의 결과로 테이블 구성한다

이 테이블 값으로 중첩쿼리의 값이  
빈값인지 검사한다

## 2. 특수쿼리

- DECODE 함수
  - 특정 컬럼의 값을 기준으로 마치 IF문을 사용하는 것과 같은 효과를 내는 함수
  - 해당 컬럼의 값이 'A'이면 지정한 특정한 값을 출력하고, 'B'이면 또 다른 값을 출력
  - 기본값을 정해서 조건을 만족하지 않는 경우의 출력 제어 가능
- 구문
  - DECODE(expr, search, result, default)
  - expr과 각 search 값을 비교하여
  - 같으면 result 값을 반환
  - 다르면 default 값을 반환

[연습문제] DECODE 예제 복습

직원테이블에서 부서번호와  
부서번호가 60이면 개발 90 이면  
영업 그 외는 기타 라고 출력하시오



## 2. 특수쿼리

- 단일행함수
- NVL( 컬럼, 표현식 )
  - 컬럼값이 null 일경우 표현식을 반환
  - 그렇지 않을경우 컬럼값을 반환한다
- NVL2( 컬럼, 표현식1, 표현식2 )
  - 컬럼값이 null이 아니면 표현식 1을 반환하고,
  - 컬럼값이 null이면        표현식 2를 반환한다

[연습예제]  
부서번호가 널인 경우  
표현식으로 출력하시오

## 2. 특수쿼리

- CASE 함수

- CASE 함수 역시 여러 가지 경우에 대해서 하나를 선택하는 함수입니다.
- DECODE 함수와 차이점이 있다면 DECODE 함수는 조건이 일치 (= 비교 연산자)하는 경우에 대해서만 적용되는 반면, CASE 함수는 다양한 비교 연산자를 이용하여 조건을 제시할 수 있으므로 범위를 지정할 수도 있습니다.
- CASE 함수는 프로그램 언어의 if else if else 와 유사한 구조를 갖습니다.

형식	<pre>CASE <i>표현식</i> WHEN <i>조건1</i> THEN <i>결과1</i>       WHEN <i>조건2</i> THEN <i>결과2</i>       WHEN <i>조건3</i> THEN <i>결과3</i>       ELSE <i>결과n</i> END</pre>
----	--

## 2. 특수쿼리

- ROWNUM
  - 오라클에서 ROWNUM 은 쿼리집합이 메모리에 생성된 뒤 결과 집합 내의 레코드에 붙여지는 가상의 순번을 의미한다.  
ROWNUM에 숫자 1,2,3, ... N 의 값이 할당되며 여기서 N의 값은 ROWNUM과 사용하는 로우의 수를 의미한다
- 사용용도 예
  - Top-N 프로세싱 ( 전체에서 N개의 레코드만 조회 )
  - 페이지네이션 (Pagination) : LIMIT 구분과 페이징을 처리에 사용

## 2. 특수쿼리

### 예: TOP-N 프로세싱

질의: DEPARTMENTS 에서 10개 이하만 목록을 출력

```
SELECT      ROWNUM, FIRST_NAME, JOB_ID, SALARY
FROM        EMPLOYEES
WHERE       ROWNUM <= 10;
```

## 2. 특수쿼리

### 예: 페이징 구현 예제

질의: EMPLOYEES 테이블에서 ROWNUM 이 11 에서 20 까지 목록 출력

```
SELECT * FROM (  
    SELECT ROWNUM AS RNUM, E.* FROM EMPLOYEES E  
) WHERE RNUM >= 11 AND RNUM <= 20;
```

rownum은 where절이 먼저 실행되고 그 조건에 맞는 리스트를 먼저 검색후 해당 리스트에 번호를 매긴다.

## 2. 특수쿼리

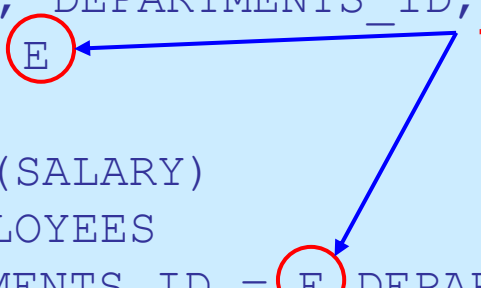
- **상관 중첩 질의(correlated nested query)**
  - 중첩 질의의 WHERE절에 있는 프레디키트에서 외부 질의에 선언된 릴레이션의 일부 애트리뷰트를 참조하는 질의
  - 중첩 질의의 수행 결과가 단일 값이든, 하나 이상의 애트리뷰트로 이루어진 릴레이션이든 외부 질의로 한 번만 결과를 반환하면 상관 중첩 질의가 아님
  - 상관 중첩 질의에서는 외부 질의를 만족하는 각 튜플이 구해진 후에 중첩 질의가 수행되므로 상관 중첩 질의는 외부 질의를 만족하는 튜플 수만큼 여러 번 수행될 수 있음

## 2. 특수쿼리

### 예: 상관 중첩 질의

질의 : 자신이 속한 부서의 직원들의 평균 급여보다 많은 급여를 받는 직원들에 대해 이름, 부서번호, 급여를 검색하라

```
SELECT      FIRST_NAME, DEPARTMENTS_ID, SALARY
FROM        EMPLOYEES E
WHERE       SALARY >
            (SELECT      AVG (SALARY)
             FROM        EMPLOYEES
             WHERE       DEPARTMENTS_ID = E.DEPARTMENTS_ID);
```



중첩질의에서 릴레이션 이름을 함께 표현하지 않은 DEPARTMENTS\_ID 애트리뷰트는 중첩 질의의 EMPLOYEES 릴레이션을 참조하는 것이다.

외부 질의에 명시된 EMPLOYEES 릴레이션의 DEPARTMENTS\_ID 애트리뷰트를 참조하려면 E, DEPARTMENTS\_ID와 같이 별칭을 사용해야한다. 다시 말해, 중첩 질의에서 별칭을 붙이지 않은 모든 애트리뷰트들은 중첩 질의에 명시된 릴레이션에 속하는 것이다.

### 3. 테이블의 결합

- 집합 연산

- 집합 연산을 적용하려면 두 릴레이션이 합집합 호환성을 가져야 함
- 입력릴레이션 결과 릴레이션에서 중복 튜플 배제
  - UNION(합집합)
  - EXCEPT(차집합)
  - INTERSECT(교집합)
- 입력릴레이션 결과 릴레이션에서 중복 튜플 허용
  - UNION ALL(합집합)
  - EXCEPT ALL(차집합)
  - INTERSECT ALL(교집합)



### 3. 테이블의 결합

예: 집합연산

질의 : 김창섭이 속한 부서이거나 개발 부서의 부서번호를 검색하라.

```
(SELECT      DEPARTMENTS_ID
FROM      EMPLOYEES
WHERE      FIRST_NAME = '김창섭' )
UNION
(SELECT      DEPARTMENTS_ID
FROM      DEPARTMENTS
WHERE      DEPARTMENT_NAME = '개발' );
```

DEPARTMENTS_ID
2
3