학습목표

1. 서브쿼리
2. 집합연산

학습내용

- 서브쿼리에 대해 배울 수 있습니다.
- 집합연산(UNION, INTERSECT, MINUS)에 대해 배울 수 있습니다.

사전퀴즈

1. 서브쿼리는 SQL 쿼리 안에 또다른 쿼리문이 들어있는 형태를 말한다.
정답 : O
해설 : 서브쿼리는 쿼리문 안에 또다른 쿼리문이 들어있는 형태를 말한다. 서브쿼리문은 보통 ()로 감싼다.

2. MariaDB는 합집합(UNION), 교집합(INTERSECT), 차집합(MINUS) 연산을 지원한다.
정답 : X
해설 : MariaDB는 합집합(UNION)만 지원한다.

수업

1. 서브쿼리(Subquery)

서브쿼리

- 쿼리문 내에 또 다른 쿼리문이 있는 형태
- 서브쿼리는 메인쿼리에 포함되는 관계
  · ()를 사용해 감싸는 형태
  · ORDER BY를 사용하지 못한다.
- 사용 가능한 위치
  · SELECT
  · FROM
  · WHERE
  · HAVING
  · ORDER BY
  · VALUES (INSERT)
  · SET (UPDATE)
- 종류

서브쿼리의 종류	설명
Single Row 서브쿼리 (단일 행 서브쿼리)	서브쿼리의 실행결과가 항상 1건 이하인 서브쿼리를 의미한다. 단일 행 서브쿼리는 단일 행 비교 연산자와 함께 사용된다. 단일 행 비교 연산자에는 =, <, <=, >, >=, <>이 있다.
Multi Row 서	서브쿼리의 실행 결과가 여러 건인 서브쿼리를 의미한다. 다중 행 서브쿼리는 다중 행 비교 연산자와 함께 사용된다. 다

브쿼리 (다중 행 서브쿼리)	중 행 비교 연산자에는 IN, ALL, ANY, SOME, EXISTS가 있다.
Multi Column 서브쿼리 (다중 컬럼 서브쿼리)	서브쿼리의 실행 결과로 여러 컬럼을 반환한다. 메인쿼리의 조건절에 여러 컬럼을 동시에 비교할 수 있다. 서브쿼리와 메인쿼리에서 비교하고자 하는 컬럼 개수와 컬럼의 위치가 동일해야 한다. 다중 컬럼 비교 연산자에는 IN이 있다.

예제 1-1

국가명이 'South Korea'의 국가코드를 찾아 이에 해당되는 도시의 수를 표기하시오.

```
 7 select count(*)
 8 from city
 9 where CountryCode = (select code
10                      from country
11                      where Name = 'South Korea')
12 ;
13
```

결과 #1 (1×1)

count(*)
70

예제 1-2

city 테이블에서 국가코드가 'KOR'인 도시의 평균 인구 수보다 인구수가 많은 도시들의 이름을 표시하시오.

```sql
select name, Population
from city
where Population > (select avg(Population)
                    from city
                    where CountryCode = 'KOR')
order by Population asc
;
```

결과 #1 (1×1)

avg(Population)
557,141.3286

```sql
select name, Population
from city
where Population > (select avg(Population)
                    from city
                    where CountryCode = 'KOR')
order by Population asc
;
```

city (2×473)

name	Population
Asunción	557,776
Jaboatão dos Guararapes	558,680
Quilmes	559,249
Rajkot	559,407
Quetta	560,307
Novokuznetsk	561,600
Chonju	563,153
Lisboa	563,210
Seattle	563,374
El Paso	563,662
Cochin (Kochi)	564,589
Antalya	564,914
Durban	566,120
Nampo	566,200
Hamamatsu	568,796
Düsseldorf	568,855

다중행 연산자

다중 행 연산자	설명
IN (서브쿼리)	- 서브쿼리의 결과에 존재하는 임의의 값과 동일한 조건을 의미한다. (Multiple OR 조건) - 전체레코드를 스캔하여 실제 존재하는 데이터들의 모든 값까지 확인한다. - 서브쿼리의 결과값을 메인쿼리에 대입하여 조건 비교 후 결과를 출력한다. (IN쿼리 → 메인쿼리)
EXISTS (서브쿼리)	- 서브쿼리의 결과를 만족하는 값이 존재하는지 여부를 확인하는 조건을 의미한다. 조건을 만족하는 건이 여러 건이더라도 1건만 찾으면 더 이상 검색하지 않는다. - 존재여부만 확인하고 스캔하지 않음(IN 연산자보다 상대적으로 빠름) - 메인쿼리의 결과값을 서브쿼리에 대입하여 조건 비교 후 결과를 출력한다. (메인쿼리 → EXISTS 쿼리) - EXISTS 연산자는 연관서브쿼리로만 사용된다. · 연관(Correlated)서브쿼리

	서브쿼리가 메인쿼리 컬럼을 가지고 있는 형태의 서브쿼리. 일반적으로 메인쿼리가 먼저 수행되어 읽혀진 데이터를 서브쿼리에서 조건이 맞는지 확인하고자 할 때 주로 사용된다. · 비연관(Un-Correlated)서브쿼리 서브쿼리가 메인쿼리 컬럼을 가지고 있지 않은 형태의 서브쿼리이다. 메인쿼리에 값(서브쿼리가 실행된 결과)을 제공하기 위한 목적으로 주고 사용한다.
비교연산자 ANY (서브쿼리)	서브쿼리의 결과에 존재하는 어느 하나의 값이라도 만족하는 조건을 의미한다. 비교 연산자로 “>”를 사용했다면 메인쿼리는 서브쿼리의 값들 중 어떤 값이라도 만족하면 되므로, 서브쿼리의 결과의 **최소값**보다 큰 모든 건이 조건을 만족한다. (SOME은 ANY와 동일함)
비교연산자 ALL (서브쿼리)	서브쿼리의 결과에 존재하는 모든 값을 만족하는 조건을 의미한다. 비교 연산자로 “>”를 사용했다면 메인쿼리는 서브쿼리의 모든 결과 값을 만족해야 하므로, 서브쿼리 결과의 **최대값**보다 큰 모든 건이 조건을 만족한다.

*** 참고 ***

IN 연산자와 EXISTS 연산자의 차이점?

구글 검색하다가 설명을 잘해주신 분이 있어서 퍼왔다.

출처: http://database.sarang.net/?inc=read&aid=28979&criteria=oracle&subcrit=&id=&limit=20&keyword=&page=

문제) 2006년 자격수당을 받은 직원 명단을 구하시오.

두가지 방법으로 접근하겠습니다.

1) IN

```
SELECT 사번, 성명
  FROM 직원
 WHERE 사번 IN (SELECT 사번
                  FROM 급여
                 WHERE 급여일자 LIKE '2006%'
                   AND 자격수당 > 0
                )
```

여기서 급여 테이블에서 구한 사번은 제공자로써 역할을 하게 됩니다.
직원 테이블에서 결과를 얻기 위한 조건으로 제공되어지는것이죠.
실행순서 : 급여테이블검색 => 사번 제공 => 직원테이블 검색

2) EXISTS

```
SELECT 사번, 성명
  FROM 직원 a
 WHERE EXISTS (SELECT ''
                 FROM 급여
                WHERE 급여일자 LIKE '2006%'
                  AND 자격수당 > 0
                  AND 사번 = a.사번
                )
```

여기서 급여 테이블은 제공자 역할이 아닌 확인자 역할을 합니다.
직원테이블의 사번과 급여의 사번이 서브쿼리 안에서 조인되는것이 보일 것입니다.
즉, 직원테이블을 검색하면서 해당 사번의 직원이 조건에 만족하는지를
  급여테이블과 조인해가면서 확인하는 것입니다.

실행순서 : 직원테이블 검색 ==> 급여테이블 확인

두쿼리의 차이점을 볼까요.
1)의 서브쿼리에선 자격수당이 0보다 큰 레코드를 모두 검색합니다.
2)의 쿼리에선 해당 사원의 자격수당이 0보다 큰건 단 1건만 검색하고 종료됩니다.
12개월 모두 자격수당을 받앋어도 1건만 검색하면 참이 되는 것이죠.
12건을 모두 검색 안해도 1건만 있다면 조건을 만족하니까요.
바로 이것이 EXISTS의 개념입니다.
SELECT절에 * 가 오든 1이 오든 컬럼명이 오든 그건 중요치 않습니다.
오직 데이터가 존재하느냐 안하느냐가 중요하죠.

이상입니다.

예제 1-3

한국의 모든(ALL) 도시의 인구 수보다 많은 도시를 찾아 표시하시오
→ 한국 도시 인구수의 최대값인 9,981,619보다 큰 도시들을 표시

```
23 select Population
24 from city
25 where CountryCode = 'KOR'
26 order by Population desc
27 ;
28
```

city (1×70)

Population
9,981,619
3,804,522
2,559,424
2,548,568
1,425,835
1,368,341
1,084,891

```
29 select name, Population
30 from city
31 where Population > ALL (select Population
32                         from city
33                         where CountryCode = 'KOR')
34 ;
35
```

city (2×1)

name	Population
Mumbai (Bombay)	10,500,000

예제 1-4

한국에서 어떤(ANY) 도시의 인구 수보다 많은 도시를 찾아 표시하시오

→ 한국 도시 인구수의 최소값인 92,239보다 큰 도시들을 표시

```
select Population
from city
where CountryCode = 'KOR'
order by Population asc
;
```

city (1×70)

Population
92,239
95,472
103,544
107,831
108,788

```
select name, Population
from city
where Population > ANY (select Population
                        from city
                        where CountryCode = 'KOR')
order by Population asc
;
```

city (2×3,854)

name	Population
Fairfield	92,256
Torrejón de Ardoz	92,262
Cairns	92,273
Gainesville	92,291

★ 예제 1-5 ★

다시 풀어보기 !!!

국가명이 'South Korea', 'China', 'Japan'의 국가코드를 찾아 이에 해당되는 도시의 수를 각각 표시하시오.

```
select count(*), CountryCode
from city
where CountryCode IN (select code
                      from country
                      where name IN ('South Korea', 'China', 'Japan')
                      )
group by CountryCode
;
```

city (2×3)

count(*)	CountryCode
363	CHN
248	JPN
70	KOR

★ 예제 1-6 ★

다시 풀어보기 !!!

인구가 5,000,000명이 넘어가는 도시의 도시이름, 국가코드, 인구 수를 표시하시오(다중컬럼과 IN을 써서)

*** 주의 ***

다중컬럼 서브쿼리 사용시 **소괄호()로 비교하려는 컬럼들을 감싸야함**

```sql
select name, CountryCode, Population
from city
where (name, CountryCode) IN (select name, CountryCode
                              from city
                              where Population > 5000000)
order by Population asc
;
```

city (3×24)

name	CountryCode	Population
Lahore	PAK	5,063,499
Kinshasa	COD	5,064,000
Tianjin	CHN	5,286,800
Rio de Janeiro	BRA	5,598,953
Santafé de Bogotá	COL	6,260,862
Bangkok	THA	6,320,174
Chongqing	CHN	6,351,600

예제 1-7

국가코드가 'KOR', 'CHN', 'JPN'인 도시명과 국가코드, 인구수를 출력하시오. (EXISTS)

1)

아래 쿼리는 실수했던 쿼리이다.

city 테이블의 레코드를 하나씩 검색하면서 서브쿼리의 조건을 만족하는지 확인하는데

서브쿼리의 조건이 메인쿼리와 상관없이 항상 true이기 때문에 city테이블의 모든 레코드들을 보여준다.

```sql
select name, CountryCode, Population
from city
where EXISTS (select code
              from country
              where code IN ('KOR', 'CHN', 'JPN')
              )
;
```

city (3×4,079)

name	CountryCode	Population
Kabul	AFG	1,780,000
Qandahar	AFG	237,500
Herat	AFG	186,800
Mazar-e-Sharif	AFG	127,800
Amsterdam	NLD	731,200
Rotterdam	NLD	593,321
Haag	NLD	440,900
Utrecht	NLD	234,323
Eindhoven	NLD	201,843
Tilburg	NLD	193,238
Groningen	NLD	172,701
Breda	NLD	160,398
Apeldoorn	NLD	153,491
Nijmegen	NLD	152,463
Enschede	NLD	149,544
Haarlem	NLD	148,772

2)

동영상 강의에서 선생님이 작성한 쿼리를 두 번째로 작성해 보았다.

원했던 결과는 나왔는데...

서브쿼리 안에 city 테이블과 country 테이블을 조인하는 조건을 명시하지 않았는데도 어떻게 결과가 제대로 나왔을까?

한참 고민해봤는데, 서브쿼리 안에 있는 테이블명이나 select 절에 있는 컬럼들은 아무 상관이 없을것 같다!!! 는 생각이 들었다.

그래서 서브쿼리 안에 있는 테이블명과 select 절에 있는 컬럼들을 변경해봄 (세번째 쿼리 참고)

```
select name, CountryCode, Population
from city
where EXISTS (select code
                from country
                where CountryCode IN ('KOR', 'CHN', 'JPN')
                )
;
```

city (3×681)

name	CountryCode	Population
Tokyo	JPN	7,980,230
Jokohama [Yokohama]	JPN	3,339,594
Osaka	JPN	2,595,674
Nagoya	JPN	2,154,376
Sapporo	JPN	1,790,886
Kioto	JPN	1,461,974
Kobe	JPN	1,425,139
Fukuoka	JPN	1,308,379
Kawasaki	JPN	1,217,359
Hiroshima	JPN	1,119,117
Kitakyushu	JPN	1,016,264
Sendai	JPN	989,975
Chiba	JPN	863,930

3)

서브쿼리 안에 있는 테이블명을 country에서 dual로, select 절에 있는 컬럼을 code에서 *로 변경해보았다

두 번째 쿼리랑 동일한 결과가 나옴!!!

즉, 서브쿼리의 테이블명, select 절에 있는 컬럼에 상관없이

city 테이블 레코드의 CountryCode가 KOR, CHN, JPN을 만족하기만 하면 city 테이블의 레코드를 결과값으로 반환하는 것!!!

```
select name, CountryCode, Population
from city
where EXISTS (select *
                from dual
                where CountryCode IN ('KOR', 'CHN', 'JPN')
)
;
```

city (3×681)

name	CountryCode	Population
Tokyo	JPN	7,980,230
Jokohama [Yokohama]	JPN	3,339,594
Osaka	JPN	2,595,674
Nagoya	JPN	2,154,376
Sapporo	JPN	1,790,886
Kioto	JPN	1,461,974
Kobe	JPN	1,425,139
Fukuoka	JPN	1,308,379
Kawasaki	JPN	1,217,359
Hiroshima	JPN	1,119,117
Kitakyushu	JPN	1,016,264
Sendai	JPN	989,975
Chiba	JPN	863,930

4)

SQLD 자격증 책에 나와있는 내용을 참고해서 쿼리를 작성해보았다.

city 테이블의 레코드를 하나씩 검색하면서 각 레코드가 서브쿼리의 조건을 만족하는지를 country테이블과 조인하면서 확인한다.

```sql
select A.name, A.CountryCode, A.Population
from city A
where EXISTS (select B.code
                from country B
                where A.CountryCode = B.code
                and code IN ('KOR', 'CHN', 'JPN')
                )
;
```

city (3×681)

name	CountryCode	Population
Shanghai	CHN	9,696,300
Peking	CHN	7,472,000
Chongqing	CHN	6,351,600
Tianjin	CHN	5,286,800
Wuhan	CHN	4,344,600
Harbin	CHN	4,289,800
Shenyang	CHN	4,265,200
Kanton [Guangzhou]	CHN	4,256,300
Chengdu	CHN	3,361,500
Nanking [Nanjing]	CHN	2,870,300
Changchun	CHN	2,812,000
Xi´an	CHN	2,761,400
Dalian	CHN	2,697,000
Qingdao	CHN	2,596,000
Jinan	CHN	2,278,100
Hangzhou	CHN	2,190,500
Zhengzhou	CHN	2,107,200
Shijiazhuang	CHN	2,041,500
Taiyuan	CHN	1,968,400

2. 집합연산

- 각종 집합연산 지원
- 합집합(UNION), 교집합(INTERSECT), 차집합(MINUS) ...
- MariaDB은 INTERSECT/MINUS는 지원하지 않음

UNION - 두 쿼리의 결과값을 합쳐서 리턴함

- SELECT 쿼리1 UNION SELECT 쿼리2 UNION ...
- 두 쿼리의 결과 형식이 동일해야 함 (기본적으로 DISTINCT 적용)
- 다른 테이블이라도 결과값의 형식만 일치하면 됨

UNION ALL - 중복을 허용하는 UNION

- SELECT 쿼리1 UNION ALL SELECT 쿼리2 UNION ...

예제 2-1

city 테이블에서 국가코드가 'KOR', 국가코드가 'CHN'인 도시를 구하시오 (UNION 사용)

```
select *
from city
where CountryCode = 'KOR'
UNION
select *
from city
where CountryCode = 'CHN'
;
```

결과 #1 (5×433)

ID	Name	CountryCode	District
2,331	Seoul	KOR	Seoul
2,332	Pusan	KOR	Pusan
2,333	Inchon	KOR	Inchon
2,334	Taegu	KOR	Taegu
2,335	Taejon	KOR	Taejon
2,336	Kwangju	KOR	Kwangju
2,337	Ulsan	KOR	Kyongsangnam
2,338	Songnam	KOR	Kyonggi
2,339	Puchon	KOR	Kyonggi

예제 2-2

UNION / UNION ALL을 사용해 차이를 확인하시오 (국가코드를 사용)

1) UNION 사용

```
select CountryCode
from city
where CountryCode = 'KOR'
UNION
select CountryCode
from city
where CountryCode = 'CHN'
;
```

결과 #1 (1×2)

CountryCode
KOR
CHN

2) UNION ALL 사용

```
20 select CountryCode
21 from city
22 where CountryCode = 'KOR'
23 UNION ALL
24 select CountryCode
25 from city
26 where CountryCode = 'CHN'
27 ;
```

결과 #1 (1×433)

CountryCode
KOR
KOR
KOR
KOR
KOR
KOR
KOR

INTERSECT - 두 쿼리의 결과값 중 공통값을 찾아서 리턴함

- SELECT 쿼리1 INTERSECT SELECT 쿼리2
- 두 쿼리의 결과 형식이 동일해야 함(기본적으로 DISTINCT 적용)
- 다른 테이블이라도 결과값의 형식만 일치하면 됨

MINUS/EXCEPT - 쿼리1 결과값에서 쿼리2 결과값을 빼서 리턴함

- SELECT 쿼리1 MINUS SELECT 쿼리2

* MariaDB은 INTERSECT/MINUS는 지원하지 않음
* 다른 쿼리로 대체해서 사용해야 함

서브쿼리(SubQuery)를 사용하는 의미?

전문가의견

서브쿼리를 잘 사용하면 여러 쿼리를 별도로 사용해서 처리하는 것보다 한 줄로 처리할 수 있는 경우가 많습니다.
게다가 SQL 언어가 비절차적(Non-procedual) 언어이기 때문에 절차적으로 프로그래밍하는 일반 언어(C/JAVA)와 같은 특성과 조금 달라 서브쿼리가 발달했다고 볼 수 있습니다.
서브쿼리는 장점도 있지만 지나치게 쿼리가 길어지면 읽기가 어려운 문제(코드가독성)가 있습니다.
결과적으로 잘 사용하면 약이고 남용하면 독이 되는 경우가 있으므로 주의해서 사용하는 것이 좋습니다.