**SQL의 조건 분기**

* UNION을 사용한 조건 분기는 초보자가 좋아하는 기술 중 하나. 쉽다.
* WHERE구만 조금씩 다른 여러 개의 SELECT구문을 합쳐서, 복수의 조건에 일치하는 하나의 결과 집합을 얻는다. (하단 코드 참고)
* 따라서 내부적으로 여러 개의 SELECT구문을 실행하기 때문에 성능적인 측면에서 큰 단점.
* CASE를 사용해 성능을 개선한 조건분기를 배운다.
* **"조건 분기를 WHERE 구로 하는 사람들은 초보자다. 잘 하는 사람은 SELECT 구만으로 조건분기를 한다.”**

1. UNION & CASE 기본 개념
2. **UNION**

**: 서로 다른 SELECT 구문을 합쳐서 하나의 테이블에 표시.**

**[형식]**  
**SELECT 구문**

**UNION**

**SELECT 구문**

......

* UNION ALL : 두 테이블의 결과를 중복허용 하여 출력
* UNION : 두 테이블의 결과를 중복제거 하여 출력

1. **CASE**

: Java의 if~else와 유사한 기능

**[형식]**

**CASE ~ WHEN ~ THEN ~ ELSE ~ END**  
**CASE** 컬럼명 | 표현식 **WHEN** 조건식1 **THEN** 결과1  
                                **WHEN** 조건식2 **THEN** 결과2  
                                                 .......  
                                **WHEN** 조건식n **THEN** 결과n   
                                **ELSE** 결과  
             **END**

* 조건문과 조건문 사이에는 콤마(,) 를 사용하지 않는다.  
  CASE 문은 반드시 END 로 끝내야 한다.  
  결과 부분은 NULL 을 사용해서는 안 된다.

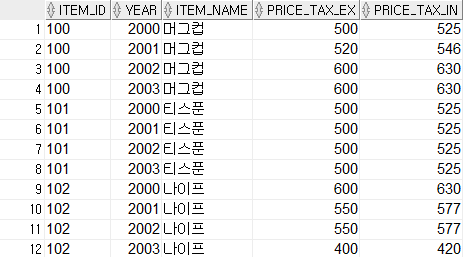
1. **UNION 쓸데없이 길고, 성능도 나쁘다.**

-- 문제1

-- 2001년 까지는 세금이 포함되지 않은 가격을, 2002년 부터는 세금이 포함된 가격을 'price'필드로 표시해라.

-- 표시할 데이터: item\_name, year, price

select \* from items;



-- union을 사용한 조건분기

select \* from (

select item\_name, year, price\_tax\_ex as price

from items

where year<=2001

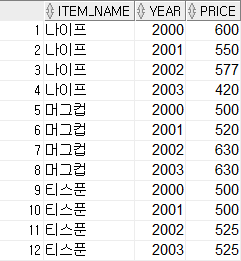
union all

select item\_name, year, price\_tax\_in as price

from items

where year>2001

) order by item\_name, year;



* 정렬이 필요해 부가적으로 SELECT구문을 1번 더 사용.
* 거의 같은 두 개의 쿼리를 두 번이나 실행.
* SQL이 쓸데없이 길다.
* items 테이블에 2번회 접근한다. 테이블의 크기에 따라 선형으로 비용 증가.

-- case를 사용한 조건분기

select item\_name, year,

case when year<=2001 then price\_tax\_ex

when year>2001 then price\_tax\_in

end as price

from items;

UNION을 사용한 결과와 동일하게 출력된다.

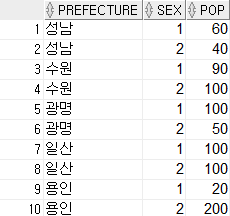
* Items 테이블에 대한 접근이 1회로 줄어든다.

1. **집계와 조건분기**
2. **집계 대상으로 조건분기** : 집계함수에 CASE식(조건분기) 삽입

-- 문제2

-- 지역별로 남녀 인구를 표시 (지역, 남자인구, 여자인구)

select \* from population;



-- UNION을 사용한 방법

select prefecture, sum(pop\_men) as pop\_men, sum(pop\_wom) as pop\_wom

from(

select prefecture, pop as pop\_men, null as pop\_wom

from population

where sex=1

union

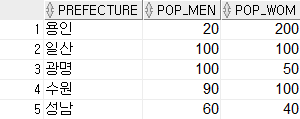
select prefecture, null as pop\_men, pop as pop\_wom

from population

where sex=2

) TMP

group by prefecture;



-- 위 쿼리의 서브쿼리

select prefecture, pop as pop\_men, null as pop\_wom

from population

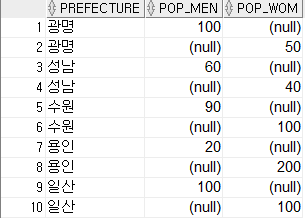
where sex=1

union

select prefecture, null as pop\_men, pop as pop\_wom

from population

where sex=2



* 서브쿼리 TMP는 위와 같이 남성과 여성의 인구가 별도의 레코드에 출력.
* 따라서 외측에 있는 GROUP BY 구를 사용해 하나의 레코드로 집약.
* Population 테이블에 풀 스캔이 2회 수행되므로 성능저하.

-- CASE를 사용한 방법

select prefecture,

sum(case when sex=1 then pop else null end) as pop\_men,

sum(case when sex=2 then pop else null end) as pop\_wom

from population

group by prefecture;

UNION 사용시와 동일 결과물 출력.

* 테이블 스캔 1회만으로 동일 결과 출력 – 성능개선.

1. **집약 결과로 조건분기** : CASE WHEN 집계함수 THEN ~

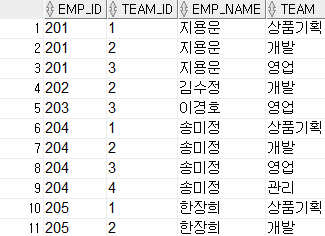
-- 문제3

-- 소속팀이 1개라면 해당 직원은 팀의 이름을 그대로 출력,

-- 2개라면 '2개를 겸무', 3개 이상이라면 '3개 이상을 겸무'라는 문자열을 'team'필드로 표시.

-- 출력대상: emp\_name, team

select \* from employees;



-- UNION을 사용

select emp\_name, max(team) as team

from employees

group by emp\_name

having count(\*)=1

union all

select emp\_name, '2개의 업무를 겸비' as team

from employees

group by emp\_name

having count(\*)=2

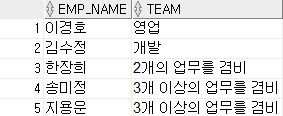
union all

select emp\_name, '3개 이상의 업무를 겸비' as team

from employees

group by emp\_name

having count(\*)>=3;



* GROUP 구를 사용하는 경우, 특정 필드의 단일 값을 출력하고 싶을 때 MAX(), MIN() 등 단일 결과를 출력해주는 집계함수 라면 무엇이든 사용 가능.
* 1번 라인의 max(team)은 단지 스칼라(더 이상 분할 불가능한 값)을 출력하기 위한 목적.

-- CASE를 사용

select emp\_name,

case when count(\*)=1 then max(team)

when count(\*)=2 then '2개의 업무를 겸비'

when count(\*)>=3 then '3개 이상의 업무를 겸비'

end as team

from employees

group by emp\_name;

결과는 UNION을 사용한 경우와 동일

* UNION 사용시 3회의 테이블 스캔이 2회로 감소.

1. **그래도 UNION이 필요한 경우**

* SELECT 구문들에서 사용하는 테이블이 다른 경우.

select col\_1

from TABLE\_A

union

select col\_2

from TABLE\_B

* 예외적인 몇 가지 상황