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

1) 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

2) 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.

2. UNION 쓸데없이 길고, 성능도 나쁘다.

```
-- 문제 1
-- 2001 년 까지는 세금이 포함되지 않은 가격을, 2002 년 부터는 세금이 포함된 가격을
'price'필드로 표시해라.
-- 표시할 데이터:  item_name, year, price
select * from items;
```

	ITEM_ID	YEAR	ITEM_NAME	PRICE_TAX_EX	PRICE_TAX_IN
1	100	2000	머그컵	500	525
2	100	2001	머그컵	520	546
3	100	2002	머그컵	600	630
4	100	2003	머그컵	600	630
5	101	2000	티스푼	500	525
6	101	2001	티스푼	500	525
7	101	2002	티스푼	500	525
8	101	2003	티스푼	500	525
9	102	2000	나이프	600	630
10	102	2001	나이프	550	577
11	102	2002	나이프	550	577
12	102	2003	나이프	400	420

```
-- union 을 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;
```

	ITEM_NAME	YEAR	PRICE
1	나이프	2000	600
2	나이프	2001	550
3	나이프	2002	577
4	나이프	2003	420
5	머그컵	2000	500
6	머그컵	2001	520
7	머그컵	2002	630
8	머그컵	2003	630
9	티스푼	2000	500
10	티스푼	2001	500
11	티스푼	2002	525
12	티스푼	2003	525

- 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.

```
-- case 를 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.

3. 집계와 조건분기

1) **집계 대상으로 조건분기** : 집계함수에 CASE식(조건분기) 삽입

```
-- 문제 2
-- 지역별로 남녀 인구를 표시 (지역, 남자인구, 여자인구)
```

```sql
select * from population;
```

	PREFECTURE	SEX	POP
1	성남	1	60
2	성남	2	40
3	수원	1	90
4	수원	2	100
5	광명	1	100
6	광명	2	50
7	일산	1	100
8	일산	2	100
9	용인	1	20
10	용인	2	200

```sql
-- 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;
```

	PREFECTURE	POP_MEN	POP_WOM
1	용인	20	200
2	일산	100	100
3	광명	100	50
4	수원	90	100
5	성남	60	40

```sql
-- 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
```

	PREFECTURE	POP_MEN	POP_WOM
1	광명	100	(null)
2	광명	(null)	50
3	성남	60	(null)
4	성남	(null)	40
5	수원	90	(null)
6	수원	(null)	100
7	용인	20	(null)
8	용인	(null)	200
9	일산	100	(null)
10	일산	(null)	100

- 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.

```
-- CASE 를 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.

2) 집약 결과로 조건분기 : CASE WHEN 집계함수 THEN ~

```
-- 문제 3
-- 소속팀이 1 개라면 해당 직원은 팀의 이름을 그대로 출력,
-- 2 개라면 '2 개를 겸무', 3 개 이상이라면 '3 개 이상을 겸무'라는 문자열을
'team'필드로 표시.
-- 출력대상: emp_name, team
select * from employees;
```

	EMP_ID	TEAM_ID	EMP_NAME	TEAM
1	201	1	지용운	상품기획
2	201	2	지용운	개발
3	201	3	지용운	영업
4	202	2	김수정	개발
5	203	3	이경호	영업
6	204	1	송미정	상품기획
7	204	2	송미정	개발
8	204	3	송미정	영업
9	204	4	송미정	관리
10	205	1	한장희	상품기획
11	205	2	한장희	개발

```sql
-- UNION 을 사용
select emp_name, max(team) as team
from employees
group by emp_name
having count(*)=1
union all
select emp_name, '2 개의 업무를 겸비' as team
from employees
group by emp_name
having count(*)=2
union all
select emp_name, '3 개 이상의 업무를 겸비' as team
from employees
group by emp_name
having count(*)>=3;
```

	EMP_NAME	TEAM
1	이경호	영업
2	김수정	개발
3	한장희	2개의 업무를 겸비
4	송미정	3개 이상의 업무를 겸비
5	지용운	3개 이상의 업무를 겸비

- 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.

```sql
-- CASE 를 사용
select emp_name,
```

```
    case when count(*)=1 then max(team)
           when count(*)=2 then '2 개의 업무를 겸비'
           when count(*)>=3 then '3 개 이상의 업무를 겸비'
           end as team
from employees
group by emp_name;
```

결과는 UNION을 사용한 경우와 동일

- UNION 사용시 3회의 테이블 스캔이 2회로 감소.

4. 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