과목 II

[SQL 활용]

제1장 서브 쿼리

[서브쿼리]
- 하나의 SQL 문 안에 포함 되어 있는 또 다른 SQL문을 말함
- 반드시 괄호로 묶어야 함

서브쿼리 사용 가능한 곳
1. SELECT 절
2. FROM 절
3. WHERE 절
4. HAVING 절
5. ORDER BY 절
6. 기타 DML (INSERT, DELETE, UPDATE)절

● GROUP BY 절 사용 불가

서브 쿼리 종류
1. 동작하는 방식에 따라
 1) UN-CORRELATED(비연관) 서브 쿼리
  - 서브쿼리가 메인쿼리 컬럼을 가지고 있지 않은 형태의 서브 쿼리

 2) CORRELATED(연관) 서브 쿼리
  - 서브쿼리가 메인쿼리 컬럼을 가지고 있는 형태의 서브 쿼리

2. 위치에 따라
 1) 스칼라 서브쿼리
  - SELECT에 사용하는 서브 쿼리
  - 서브 쿼리 결과를 마치 하나의 컬럼처럼 사용하기 위해 주로 사용

```
SELECT * | 컬럼명 | 표현식 ,
       (SELECT * | 컬럼명 | 표현식 ,
        FROM 테이블명 또는 뷰명
        WHERE 조건)
FROM 테이블명 또는 뷰명
```

 2) 인라인뷰
  - FROM절에 사용하는 서브쿼리
  - 서브쿼리 결과를 테이블처럼 사용하기 위해 주로 사용

3) WHERE 절 서브쿼리
- 가장 일반적인 서브쿼리
- 비교 상수에 값을 전달하기 위한 목적으로 주로 사용(상수항의 대체)
- 리턴 데이터 형태에 따라 단일행 서브쿼리, 다중행 서브쿼리, 다중컬럼 서브쿼리, 상호연관 서브쿼리로 구분

WHERE절 서브쿼리 종류

1) 단일행 서브쿼리
- 서브쿼리 결과가 1개의 행이 리턴되는 형태

```
SELECT EMPNO, ENAME, SAL
FROM EMP
WHERE SAL > (SELECT AVG(SAL)
             FROM EMP);
```

2) 다중행 서브쿼리
- 서브쿼리 결과가 여러 행이 리턴되는 경우

연산자	의미
IN	같은 값을 찾음
> ANY	최소값을 반환함
< ANY	최대값을 반환함
< ALL	최소값을 반환함
> ALL	최대값을 반환함

```
SELECT EMPNO, ENAME, SAL
FROM EMP
WHERE SAL > (SELECT MIN(SAL)
             FROM EMP
             WHERE DEPTNO = 10);
```

```
SELECT EMPNO, ENAME, SAL
FROM EMP
WHERE SAL > ANY(SELECT SAL
             FROM EMP
             WHERE DEPTNO = 10);
```

3) 다중컬럼 서브쿼리

- 서브쿼리 결과가 여러 컬럼이 리턴되는 형태
- 대소 비교 전달 불가(두 값을 동시에 묶어 대소 비교 할수 없음)

```
SELECT EMPNO, ENAME, SAL, DEPTNO
FROM EMP
WHERE (DEPTNO, SAL) IN (SELECT DEPTNO, MAX(SAL)
                        FROM EMP
                        GROUP BY DEPTNO);
```

- 비교 시에는 다중행 연산자인 IN을 사용

4) 상호연관 서브쿼리

- 메인쿼리와 서브쿼리의 비교를 수행하는 형태

```
SELECT EMPNO, ENAME, SAL, DEPTNO
FROM EMP
WHERE (DEPTNO, SAL) > (SELECT DEPTNO, MAX(SAL)
                       FROM EMP
                       GROUP BY DEPTNO);
```

-> 에러 발생 / 다중 컬럼 서브쿼리는 동시 두 컬럼에 대한 대소비교 불가

```
SELECT EMPNO, ENAME, SAL, DEPTNO
FROM EMP E1
WHERE SAL > (SELECT MAX(SAL)
             FROM EMP E2
             WHERE E1.DEPTNO = E2.DEPTNO
             GROUP BY DEPTNO);
```

-> 대소 비교할 컬럼을 메인 쿼리에 일치 조건을 서브쿼리에 전달

** 상호연관 서브쿼리 연산 순서 **

1) 메인쿼리 테이블 READ
2) 메인쿼리 WHERE절 확인
3) 서브쿼리 테이블 READ
4) 서브쿼리 WHERE절 확인(다시 E1.DEPNO 요구)
5) E1.DEPTNO 값을 서브쿼리의 DEPTNO 컬럼과 비교하여 조건절 완성
6) 위 조건에 성립하는 행의 그룹연산 결과 확인
7) 위 결과를 메인쿼리에 전달하여 해당 조건을 만족하는 행만 추출

인라인 뷰
- 쿼리 안의 뷰의 형태로 테이블처럼 조회할 데이터를 정의하기 위해 사용
- 테이블명이 존재하지 않기 때문에 다른 테이블과 조인 시 반드시 테이블 별칭 명시
- 서브쿼리 결과를 메인 쿼리의 어느절에서도 사용할 수 있음.
- 모든 연산자 사용 가능

스칼라 서브쿼리
- SELECT 절에 사용하는 쿼리로, 마치 하나의 컬럼처럼 표현하기 위해 사용
- 각 행마다 서브쿼리 결과가 하나여야 함
- 조인의 대체 연산

서브 쿼리 주의 사항
- 특별한 경우(TOP-N 분석 등)을 제외하고는 서브 쿼리 절에 ORDER BY 절을 사용 불가
- 단일 행 서브 쿼리와 다중 행 서브쿼리에 따라 연산자의 선택이 중요

제2장 집합 연산자

[집합 연산자]

- SELECT 문 결과를 하나의 집합으로 간주, 그 집합에 대한 합집합, 교집합, 차집합 연산
- 두 집합 컬럼이 동일하게 구성되어야 함(각 컬럼의 데이터 타입과 순서 일치 필요)

합집합

1) UNION
 - 중복된 데이터를 제거하기 위해 내부적으로 정렬 수행
 - 중복된 데이터가 없을 경우에는 UNION 대신 UNION ALL 사용

2) UNION ALL
 - 중복된 데이터도 전체 출력

교집합
 - 두 집합 사이에 INTERSECT

차집합
 - 두 집합 사이에 MINUS 전달
 - A-B와 B-A는 다르므로 집합의 순서 주의!

집합 연산자 사용 시 주의 사항

1. 두 집합의 컬럼 수 일치
2. 두 집합의 컬럼 순서 일치
3. 두 집합의 각 컬럼의 데이터 타입 일치
4. 각 컬럼의 사이즈는 달라도 됨

제3장 그룹 함수

[그룹 함수]
- 수학/통계 함수들
- 반드시 한 컬럼만 전달
- NULL은 무시하고 연산

<COUNT>
- 행의 수를 세는 함수
- * 또는 단 하나의 컬럼만 전달 가능
- 문자, 숫자, 날짜 컬럼 모두 전달 가능

<SUM>
- 총합 출력
- 숫자 컬럼만 전달 가능

<AVG>
- 평균 출력
- 숫자 컬럼만 가능
- NULL을 제외한 대상의 평균을 리턴하므로 전체 대상 평균 연산 시 주의

<MIN/MAX>
- 최대, 최소 출력
- 날짜, 숫자, 문자 모두 가능(오름차순 순서대로 최소, 최대 출력)

<VARIIANCE/STDDEV>
- 평균과 표준편차
- 표준편차는 평균값의 루트값

<GROUP BY FUNCTION>
- GROUP BY 절에 사용하는 함수
- 여러 GROUP BY 결과를 동시에 출력하는 기능

1. GROUPING SETS(A, B, ...)
- A별, B별 그룹 연산 결과 출력
- 나열 순서 중요 X
- 전체 총계는 출력되지 않는다
- NULL 혹은 () 사용하여 전체 총합 출력 가능
** UNION ALL로 대체 가능
** ()는 전체 총합

2. ROLLUP(A, B)
- A별, (A, B) 별, 전체 그룹 연산 결과 출력
- 나열 대상의 순서가 중요함
- 전체 총계가 출력됨
** UNION ALL로 대체 가능

3. CUBE(A, B)
- A별, B별, (A, B)별, 전체 그룹 연산 결과 출력됨
- 그룹으로 묶을 대상의 나열 순서 중요하지 않음
- 기본적으로 전체 총계가 출력됨
** UNION ALL과 GROUPING SET로 대체 가능

```
SELECT DEPTNO, JOB, SUM(SAL)
FROM EMP
GROUP BY CUBE(DEPTNO, JOB);
```

```
SELECT NULL AS DEPTNO, NULL AS JOB, SUM(SAL)
FROM EMP
UNION ALL
SELECT NULL, JOB, SUM(SAL)
FROM EMP
GROUP BY JOB
UNION ALL
SELECT DEPTNO, NULL, SUM(SAL)
FROM EMP
GROUP BY DEPTNO
UNION ALL
SELECT DEPTNO, JOB, SUM(SAL)
FROM EMP
GROUP BY DEPTNO, JOB
```

```
SELECT DEPTNO, JOB, SUM(SAL)
FROM EMP
GROUP BY GROUPING SETS(DEPTNO, JOB, (DEPTNO, JOB), ());
```

제4장 윈도우 함수

[WINDOW FUNCTION]

- 서로 다른 행의 비교나 연산을 위해 만든 함수
- GROUP BY를 쓰지 않고 그룹 연산 가능
- LAG, LEAD, SUM, AVG, MIN, MAX, COUNT, RANK

```
SELECT 윈도우함수([대상]) OVER([PARTITION BY 컬럼]
                              [ORDER BY 컬럼 ASC|DESC]
                              [ROWS|RANGE BETWEEN A AND B]);
```

** PARTITION BY 절 : 출력할 총 데이터 수 변화 없이 그룹 연산 수행할 GROUP BY 칼럼

** ORDER BY 절
 - RANK의 경우 필수
 - SUM, AVG, MIN, MAX, COUNT 등은 누적값 출력 시 사용

** ROWS | RANGE BETWEEN A AND B
 - 연산 범위 설정
 - ORDER BY 절 필수

-> 절 전달 순서 중요(ORDER BY를 PARTITION BY 전에 사용 불가)

그룹 함수의 형태

- SUM, COUNT, AVG, MIN, MAX 등

```
SELECT SUM([대상]) OVER(PARTITION BY 컬럼)
                       [ORDER BY 컬럼 ASC|DESC]
                       [ROWS|RANGE BETWEEN A AND B]);
```

1) SUM OVER()

- 전체 총합, 그룹별 총합 출력 가능

방법 1 : 서브쿼리 사용(스칼라 서브쿼리)

```
SELECT EMPNO, ENAME, SAL, DEPTNO,
       (SELECT SUM(SAL) FROM EMP) AS TOTAL
FROM EMP;
```

방법 2 : 윈도우 함수 사용

```
SELECT EMPNO, ENAME, SAL, DEPTNO,
       SUM(SAL) OVER() AS TOTAL
FROM EMP;
```

** 윈도우 함수의 연산 범위 : 집계 연산 시 행의 범위 설정 가능

1. ROWS, RANGE 차이

 1) ROWS : 값이 같더라도 각 행씩 연산

   -> ROWS는 반드시 범위를 설정해야 함

 2) RANGE : 같은 값의 경우 하나의 RANGE로 묶어서 동시 연산(DEFAULT)

2. BETWEEN A AND B

 A) 시작점 정의

  - CURRENT ROW : 현재 행부터
  - UNBOUNDED PRECEDING : 처음부터(DEFAULT)
  - N PRECEDING : N 이전부터

 B) 마지막 시점 정의

  - CURRENT ROW : 현재 행까지
  - UNBOUNDED FOLLOWING : 마지막까지
  - N FOLLOWING : N 이후까지

RANGE 범위 전달(DEFAULT)

```
SELECT R.*,
       SUM(SAL) OVER(ORDER BY SAL)
FROM RANGE_TEST R;
```

ROWS 범위 설정 시

```
SELECT R.*,
       SUM(SAL) OVER(ORDER BY SAL
                     ROWS BETWEEN UNBOUNDED PRECEDING
                     AND CURRENY ROW) AS RESULT1
FROM RANGE_TEST R;
```

순위 관련 함수

1) RANK(순위)

 1-1) RANK WITHIN GROUP

  - 특정 값에 대한 순위 확인
  - 윈도우 함수가 아닌 일반 함수

```
SELECT RANK(값) WITHIN GROUP(ORDER BY 컬럼);
```

2) RANK() OVER()
- 전체 중/특정 그룹 중 값의 순위 확인
- ORDER BY 절 필수
- 순위를 구할 대상을 ORDER BY 절에 명시(여러개 나열 가능)
- 그룹 내 순위 구할 시 PARTITION BY 절 사용

```
SELECT RANK() OVER(PARTITION BY 컬럼)
                    ORDER BY 컬럼 ASC|DESC);
```

3) DENSE_RANK
- 누적 순위
- 값이 같을 때 동일한 순위 부여 후 다음 순위가 바로 이어지는 순위 부여 방식
- 1등이 5명이더라도 그 다음 순위가 2등

4) ROW_NUMBER
- 연속된 행 번호
- 동일한 순위를 인정하지 않고 단순히 순서대로 나열한대로의 순서 값 리턴

LAG, LEAD
- 행 순서대로 각각 이전 값(LAG), 이후 값(LEAD) 가져오기
- ORDER BY 절 필수

```
SELECT LAG(컬럼, N)  // 가져올 값을 갖는 컬럼, 몇 번째 값을 가져올지(DEFAULT:1)
            OVER(PARTITION BY 컬럼)
            ORDER BY 컬럼 ASC|DESC);
```

FIRST_VALUE, LAST_VALUE
- 정렬 순서대로 정해진 범위에서의 처음 값, 마지막 값 출력
- 순서와 범위 정의에 따라 최솟값 최댓값 리턴 가능

NTILE
- 행을 특정 컬럼 순서에 따라 정해진 수의 그룹으로 나누기 위한 함수
- 그룹 번호가 리턴됨
- ORDER BY 필수
- PARTITION BY를 사용하여 특정 그룹을 또 원하는 수 만큼 그룹 분리 가능

```
SELECT NTILE(N) OVER([PARTITION BY 컬럼] // N은 그룹의 개수
                    ORDER BY 컬럼 ASC|DESC)
```

비율 관련 함수

1) RATIO_TO_REPORT

- 각 값의 비율 리턴
- ORDER BY 사용 불가

```
RATIO_TO_REPORT(대상) OVER([PARTITION BY ...])
```

2) CUME_DIST : 누적 비율

- 각 값의 누적 비율 리턴
- ORDER BY 필수(누적 비율을 구하는 순서 정할 수 있음)

```
CUME_DIST() OVER([PARTITION BY 컬럼]
                    ORDER BY 컬럼 ASC|DESC)
```

3) PERCENT_RANK

- PERCENMTILE(분위수 출력)
- 전체 COUNT 중 상대적 위치 출력(0~1 범위 내)
- ORDER BY 필수

```
PERCENT_RANK() OVER([PARTITION BY 컬럼]
                      ORDER BY 컬럼 ASC|DESC)
```