인라인 뷰는 수학의 괄호

인라인 뷰는 실행 방식에 의해 주 쿼리의 조건이 인라인 뷰에서 사용되는 경우와 인라인 뷰에서 사용되지 않는 경우의 두 가지 종류가 존재한다.

결국, 다음과 같이 인라인 뷰의 종류가 존재하게 된다.

- Mergeable 인라인 뷰 : view를 통해 데이터를 가져오는 작업을 최소화 하고자, main query의 조건들과 병합하여 최종적으로 테이블에 접근할 수 있도록 내부적으로 SQL 문장을 변형시키는 것
- Non-Mergeable 인라인 뷰

Non-Mergeable 인라인 뷰는 뷰 Merging이 발생하지 않는 인라인 뷰이다. 이와 같은 인라인 뷰는 어떻게 수행되는 것일까?

```
1  SELECT a.department_name, c.employee_name, c.address
2    FROM department a,
3       ( SELECT department_id, grade, employee_name, address
4           FROM employees b
5          WHERE sal > 200
6          UNION ALL
7         SELECT department_id, grade, employee_name, address
8           FROM employees b
9          WHERE sal < 100
10      ) C
11   WHERE a.department_id = b.department_id
12     AND c.grade = 'S';
```

Main query의

이 SQL에서 WHERE 문의 GRADE 조건이 인라인 뷰 안으로 삽입된다면 뷰 Merging이 발생하여 Mergeable 인라인 뷰로 수행되게 된다.

이와 같은 경우 EMPLOYEES 테이블에 GRADE+SAL 인덱스가 존재한다면 해당 인덱스를 이용할 수 있게 된다.

그렇다면 뷰 Merging이 발생하지 않는 Non-Mergeable 인라인 뷰로 수행된다면 어떻게 수행될까? Non-Mergeable 인라인 뷰로 수행된다면 이는 수학의 괄호와 같다.

수학의 괄호와 비교해 보자. 수학의 괄호는 우선순위를 정하게된다. 그렇기 때문에 괄호가 사용된 부분은 별도로 수행하게 된다.

Non-Mergeable 인라인 뷰는 수학의 괄호와 같다. 그렇기 때문에 EMPLOYEES 테이블에 GRADE+SAL 인덱스가 존재하더라도 해당 인덱스를 이용할 수 없게 된다.

이는 주 쿼리의 조건인 GRADE 조건이 Non-Mergeable 인라인 뷰에서는 인라인 뷰로 삽입되지 못하기 때문에 해당 조건이 처리 범위를 감소시키는 조건으로 사용될 수 없기 때문이다.

결국, 인라인 뷰가 별도로 수행되는 현상이 발생하는 것이 Non-Mergeable 인라인 뷰가 되며 이는 수학의 괄호와 같이 인라인 뷰의 괄호가 연산자의 순서를 결정하는 역할을 수행하게 된다.

Non-Mergeable 인라인 뷰 확인하기

그렇다면 우리가 인라인 뷰만을 보고 어떻게 뷰 Merging이 발생하는지 아닌지를 확인할 수 있겠는가?

물론, SQL의 형태만을 보고 판단할 수는 없다. 하지만, 다음과 같은 문법이 SQL에 사용된다면 Non-Mergeable 인라인 뷰가 되거나 Mergeable 인라인 뷰가 되더라도 주 쿼리의 조건이 인라인 뷰로 삽입되는 현상이 발생된다.

- UNION ALL
- UNION
- DISTINCT
- GROUP BY
- ROWNUM
- 집합 함수

이와 같은 문법을 사용한다면 해당 SQL은 주 쿼리의 조건이 인라인 뷰 안으로 삽입되거나 Non-Mergeable 인라인 뷰로 수행될 가능성이 높다.

물론, 해당 문법을 사용한 인라인 뷰는 인라인 뷰가 주 쿼리와 합쳐지는 형태의 뷰 Merging이 발생하지 않는다. 하지만, 이와 같은 것만으로 우리가 Mergeable 인라인 뷰인지 Non-Mergeable 인라인 뷰인지를 판단할 수는 없다.

그렇다면 어떻게 우리는 정확히 Non-Mergeable 인라인 뷰인지 아닌지를 판단할 수 있을까? 이는 해당 SQL의 실행 계획을 통해 확인 해야 한다. 해당 SQL의 실행 계획을 확인해 보자.

```
SELECT STATEMENT
  NESTED LOOPS
    TABLE ACCESS (FULL) OF 'DEPARTMENT'
    VIEW
      UNION-ALL
        TABLE ACCESS (FULL) OF 'EMPLOYEES'
        TABLE ACCESS (FULL) OF 'EMPLOYEES'
```

이렇게 하면 VIEW 실행 계획이 생성된다.

VIEW 실행 계획이 생성된다면 이는 인라인 뷰가 별도로 수행되어 메모리에 해당 데이터를 생성하는 것이다. 이는 마치 인라인 뷰가 수학의 괄호와 같이 수행되는 것이다.

결국, 실행 계획에 VIEW 실행 계획이 생성된다면 이는 인라인 뷰가 별도로 엑세스되는 것이므로 Non-Mergeable 인라인 뷰로 수행되는 것이다. 그렇기 때문에 VIEW 실행 계획이 생성되지 않는다면 이는 Mergeable 인라인 뷰로 수행되는 형태가 된다.

여기서의 Mergeable 인라인 뷰는 인라인 뷰가 주 쿼리와 합쳐지는 형태의 Mergeable 인라인 뷰로 수행되는 것이다.

만약, Mergeable 인라인 뷰 중 주 쿼리의 조건이 인라인 뷰 안으로 삽입된다면 실행 계획은 어떻게 수행되겠는가? 이와 같다면 실행 계획에는 VIEW PREDICATE라는 실행 계획이 VIEW 실행 계획 위치에 생성되게 된다.

앞서 실행 계획을 통해서 우리는 뷰 Merging이 발생했는지 발생하지 않았는지를 판단할 수 있게 된다. 이런 이유 때문에 인라인 뷰를 작성하면서 해당 SQL의 실행 계획 확인은 매우 중요하다.

하지만 아직도 많은 곳에서는 이와 같이 인라인 뷰에 대한 실행 계획을 확인하지 않는 것이 대부분이다. 이제라도 인라인 뷰에 대한 실행 계획을 확인하는 습관을 가지는 것은 매우 중요하다.

```sql
WITH FIRST_WITH AS(
SELECT /*+ INLINE NO_MERGE FULL(COURSE) PARALLEL(COURSE 4)*/
       CNO,
       CNAME,
       PNO,
       ROW_NUMBER() OVER(ORDER BY CNO DESC) AS RNK
FROM COURSE
WHERE CNO LIKE '%2%'
),

SECOND_WITH AS (
SELECT /*+ INLINE NO_MERGE */
       CNO,
       CNAME,
       PNO,
       RNK
FROM FIRST_WITH
WHERE TO_CHAR(RNK) LIKE '%1%'
),

THIRD_WITH AS (
SELECT /*+ INLINE NO_MERGE */
       A.CNO,
       A.CNAME,
       B.PNAME,
       B.ORDERS,
       A.RNK
FROM SECOND_WITH A, PROFESSOR B
WHERE A.PNO = B.PNO
)

SELECT *
FROM THIRD_WITH;
```

첫번째 view 생성

두번째 view 생성

세번째 view 생성

Id	Operation	Name	E-Rows	OMem	1Mem	Used-Mem
0	SELECT STATEMENT					
1	VIEW		2			
* 2	HASH JOIN		2	1250K	1250K	1192K (0)
3	VIEW		2			
* 4	VIEW		2			
5	WINDOW SORT		2	4096	4096	4096 (0)
* 6	TABLE ACCESS FULL	COURSE	2			
7	TABLE ACCESS FULL	PROFESSOR	24			

해당 operation이 실행될 때 access연산 발생.

해당 operation이 실행될 때, filter연산 발생.

해당 operation이 실행될 때, filter연산 발생.

Predicate Information (identified by operation id):

```
   2 - access("A"."PNO"="B"."PNO")
   4 - filter(TO_CHAR("RNK") LIKE '%1%')
   6 - filter(("CNO" LIKE '%2%' AND "CNO" IS NOT NULL))
```