


① FILTER 연산은 데이터 추출 시 필터링이 일어나고 있음을 알려주는 SQL ROW 연산인데 WHERE 조건 절에서 인덱스를 사용하지 못할 때 발생한다. 
② NESTED LOOP 방식으로 해석할 수 있는데 서브쿼리라면 메인쿼리 로우를 하나씩 읽을때 마다 서브쿼리를 한번씩 실행하는 형태이다.

아래 예문은 MYEMP1 TABLE에서 부서의 최소 급여를 받는 사람들을 추출하는 예문이다.

-- 바깥쪽 메인쿼리에서 한건씩 읽어서 읽은 레코드의 급여가 자신이 속한 부서의 최소급여와
-- 같은지를 반복적으로 비교한다.(중첩루프 방식으로 해석)

```
SQL> SELECT ENAME, SAL, JOB
      FROM MYEMP1 A
      WHERE SAL = (SELECT MIN(SAL)
                   FROM MYEMP1 B
                   WHERE B.DEPTNO = A.DEPTNO);
```

Execution Plan

```
-----
0 SELECT STATEMENT Optimizer=CHOOSE
1 0  FILTER
2 1   TABLE ACCESS (FULL) OF 'MYEMP1'
3 2   SORT (AGGREGATE)
4 3   TABLE ACCESS (BY INDEX ROWID) OF 'MYEMP1'
5 4     INDEX (RANGE SCAN) OF 'idx_myemp1_deptno' (NON-UNIQUE)
```

```
select *
from emp
where deptno in (select /*+ no_unnest */ deptno from dept)
```

WHERE절에 in 또는 exist가 subquery와 함께 사용되면,
FILTER operation 이 발생 (correlated subquery
처럼 동작)

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		5	185	3 (0)	00:00:01
* 1	FILTER					
2	TABLE ACCESS FULL	EMP	14	518	3 (0)	00:00:01
* 3	INDEX UNIQUE SCAN	PK_DEPT	1	3	0 (0)	00:00:01

Main query의 테이블에 대해 scan함

Predicate Information (identified by operation id):

```
1 - filter( EXISTS (SELECT /*+ NO_UNNEST */ 0 FROM "DEPT"
"DEPT" WHERE "DEPTNO"=:B1))
3 - access("DEPTNO"=:B1)
```



옵티마이저가 서브쿼리를 별도의 서브플랜으로 최적화
이처럼, Unnesting하지 않은 서브쿼리를 수행할 때는 메인 쿼리에서 읽히는 레코드마다 값을 넘기면서 서브쿼리를 반복 수행