

`/*+ PARALLEL(테이블명 degree수) */`

↑ 실행 구성 개수

• PARALLEL 뿐만 아니라 모든 힌트는 `/*+ 힌트구문 */` 형태로 사용하며 SELECT 바로 다음에 기술한다.

그럼 힌트를 어떤 식으로 사용하는지 알아 보자.

입력


```
DECLARE
  -- 변수 선언 ...
BEGIN

  SELECT /*+ PARALLEL(a 4), PARALLEL(b 4), */
         a.column1, a.column2, ...
  INTO ...
  FROM TABELA a
       TABLEB b
 WHERE a.column_1 = b.column_1
       AND ...

  --나머지 로직
  ...
END;
```

힌트를 사용하면 병렬 쿼리는 힌트가 사용된 SQL문에만 적용된다. 또한 기본적으로 PARALLEL QUERY 옵션은 활성화된 상태이므로 별도로 활성화하거나 비활성화할 필요는 없는데, 만약 비활성 상태에서 힌트를 적용하면 병렬 쿼리가 수행되지 않는다.

- 병렬 처리란, SQL문이 수행해야 할 작업 범위를 여러 개의 작은 단위로 나누어 여러 프로세스(또는 스레드)가 동시에 처리하는 것을 말한다. 여러 프로세스가 동시에 작업하므로 대용량 데이터를 처리할 때 수행 속도를 극적으로 단축시킬 수 있다.

 parallel 힌트를 사용할 때는 반드시 full 힌트도 함께 꼭 사용해야 한다. 옵티마이저에 의해 인덱스 스캔이 선택되면 parallel 힌트가 무시되기 때문이다.

```
SQL> SELECT /*+ full(o) parallel(o, 4) */  
        COUNT(*) 주문건수, SUM(주문수량) 주문수량, SUM(주문금액) 주문금액  
FROM 주문 o  
WHERE 주문일시 BETWEEN '20120101' AND '20120301'  
;
```

- 아래와 같이  parallel_index 힌트를 사용할 때, 반드시 index 또는 index_ffs 힌트를 함께 사용하는 습관도 필요하다.

```
SQL> SELECT /*+ index_ffs(o, 주문_idx) parallel_index(o, 주문_idx, 4) */  
        COUNT(*) 주문건수, SUM(주문수량) 주문수량, SUM(주문금액) 주문금액  
FROM 주문 o  
WHERE 주문일시 BETWEEN '20120101' AND '20120301'  
;
```

(3) 병렬 인덱스 스캔

- Index Fast Full Scan이 아닌 한 인덱스는 기본적으로 병렬로 스캔 할 수 없다.
- 파티션된 인덱스일 때 병렬 스캔이 가능하며 파티션 기반 **Granule**이므로 병렬도는 파티션개수 이하로만 지정가능 함.

set autotrace on 커맨드를 실행해주면, 이후에 실행되는 모든 SQL문에 대해 실행 후, Plan 정보를 함께 보여줍니다. Execution Plan 정보를 보면 PX COORDINATOR 가 나오는 걸로 보아 병렬처리를 했음을 알 수 있습니다. 밑에 Note 섹션에 DOP 4 로 실행되었다고 적혀져 나오네요.

```
SQL>
SQL> set autotrace on
SQL>
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17 00:00:00	800		20
7499	ALLEN	SALESMAN	7698	1981-02-20 00:00:00	1600	300	30
7521	WARD	SALESMAN	7698	1981-02-22 00:00:00	1250	500	30
7566	JONES	MANAGER	7839	1981-04-02 00:00:00	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-28 00:00:00	1250	1400	30
7698	BLAKE	MANAGER	7839	1981-05-01 00:00:00	2850		30
7782	CLARK	MANAGER	7839	1981-06-09 00:00:00	2450		10
7788	SCOTT	ANALYST	7566	1987-04-19 00:00:00	3000		20
7839	KING	PRESIDENT		1981-11-17 00:00:00	5000		10
7844	TURNER	SALESMAN	7698	1981-09-08 00:00:00	1500	0	30
7876	ADAMS	CLERK	7788	1987-05-23 00:00:00	1100		20
7900	JAMES	CLERK	7698	1981-12-03 00:00:00	950		30
7902	FORD	ANALYST	7566	1981-12-03 00:00:00	3000		20
7934	MILLER	CLERK	7782	1982-01-23 00:00:00	1300		10

14 rows selected.

Execution Plan

Plan hash value: 2873591275

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT		14	532	2 (0)	00:00:01			
1	PX COORDINATOR								
2	PX SEND QC (RANDOM)	:TQ10000	14	532	2 (0)	00:00:01	Q1,00	P->S	QC (RAND)
3	PX BLOCK ITERATOR		14	532	2 (0)	00:00:01	Q1,00	PCWC	
4	TABLE ACCESS FULL	EMP	14	532	2 (0)	00:00:01	Q1,00	PCWP	

Note

- Degree of Parallelism is 4 because of table property

Statistics

```
12 recursive calls
0 db block gets
18 consistent gets
0 physical reads
0 redo size
1593 bytes sent via SQL*Net to client
607 bytes received via SQL*Net from client
2 SQL*Net roundtrips to/from client
0 sorts (memory)
0 sorts (disk)
14 rows processed
```

싱글처리할 때는 PX Coordinator 얘기도 없고, Note 섹션이 아예 안나오네요.

싱글처리로 바꾸는 Table 의 degree 를 초기화하는 방법은 위에서처럼 `alter table ... parallel 1;` 이라고 해도 되고, `alter table ... noparallel;` 이라고 해도 같은 겁니다.

밑에 Note 섹션에 DOP 4 로 실행되었다고 적혀져 나오네요.

원래 싱글일 때와 비교해보면 확실히 다른 것을 확인할 수 있습니다.

```
SQL>
SQL> alter table EMP parallel 1;
Table altered.
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17 00:00:00	800		20
7499	ALLEN	SALESMAN	7698	1981-02-20 00:00:00	1600	300	30
7521	WARD	SALESMAN	7698	1981-02-22 00:00:00	1250	500	30
7566	JONES	MANAGER	7839	1981-04-02 00:00:00	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-28 00:00:00	1250	1400	30
7698	BLAKE	MANAGER	7839	1981-05-01 00:00:00	2850		30
7782	CLARK	MANAGER	7839	1981-06-09 00:00:00	2450		10
7788	SCOTT	ANALYST	7566	1987-04-19 00:00:00	3000		20
7839	KING	PRESIDENT		1981-11-17 00:00:00	5000		10
7844	TURNER	SALESMAN	7698	1981-09-08 00:00:00	1500	0	30
7876	ADAMS	CLERK	7788	1987-05-23 00:00:00	1100		20
7900	JAMES	CLERK	7698	1981-12-03 00:00:00	950		30
7902	FORD	ANALYST	7566	1981-12-03 00:00:00	3000		20
7934	MILLER	CLERK	7782	1982-01-23 00:00:00	1300		10

14 rows selected.

Execution Plan

Plan hash value: 3956160932

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	532	3 (0)	00:00:01
1	TABLE ACCESS FULL	EMP	14	532	3 (0)	00:00:01

Statistics

```
11 recursive calls
0 db block gets
26 consistent gets
0 physical reads
0 redo size
1646 bytes sent via SQL*Net to client
607 bytes received via SQL*Net from client
2 SQL*Net roundtrips to/from client
2 sorts (memory)
0 sorts (disk)
14 rows processed
```