Saved: 01-Dec-2017 18:24

EPAM Systems, RD Dep.

Basic Parallel Execution

REVISION HISTORY					
Ver.	**Description of Change**	**Author**	**Date**	**Approved**	
				Name	**Effective Date**
1.0	Initial status	Valeryia Lupanava	30-NOV-2017		

Saved: 01-Dec-2017 18:24

Содержание

1. Explain plan. Parallel hints

1.1. PARALLEL SELECT

- Один из простых способов ускорения запросов – это их распараллеливание. Параллельное выполнение SQL основывается на вызове координатора **QC** и процессов параллельного выполнения **PX**. Координатор параллельного выполнения проверяет каждую операцию в плане выполнения SQL-запроса, а затем определяет способ разделения или перераспределения строк, используемых операцией.
- Первый вариант.

```
EXPLAIN PLAN FOR
SELECT --+ PARALLEL(a, 10)
  FIRST_NAME,
  LAST_NAME,
  AGE_CATEGORY,
  COUNTRY,
  SUM(TOT_SALE_SUM) AS TOT_SALE_SUM
FROM BL_DM.DIM_CUSTOMERS_SCD a, BL_DM.FCT_RETAIL_SALES_DD b
WHERE a.CUSTOMER_SURR_ID = b.CUSTOMER_SURR_ID
GROUP BY FIRST_NAME,
         LAST_NAME,
         AGE_CATEGORY,
         COUNTRY;

SELECT * FROM TABLE (dbms_xplan.display);
```

- В хинте было явно указано применение распараллеливания в запросе на 10 процессов.
- План выполнения запроса показывает, что сначала каждый набор **a** и **b** имеет четыре процесса выполнения, не смотря на явное указание **10** в запросе. Проверяется **b** таблица, на базе которого строится хэш-таблица и параллельно строки отправляются **a** набору. Другими словами, **b** и **a** работают одновременно: один параллельно сканирует и хэширует строки, второй создает временную хеш-таблицу, чтобы реализовать хэш-соединение со второй таблицей параллельно. Это пример межоперационного параллелизма.
- При работе набор **a** параллельно сканирует таблицу **b**. Он отправляет свои строки на сервер для хеш-соединения со строками **b**. После этого включается параллельное выполнение операции **GROUP BY**. Таким образом, два набора одновременно запускают параллелизм между различными операторами в запросе.

Saved: 01-Dec-2017 18:24

```
PLAN_TABLE_OUTPUT
Plan hash value: 4037101356

-----------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name                | Rows  | Bytes | Cost (%CPU)| Time     |    TQ  |IN-OUT| PQ Distrib |
-----------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                     |  2509K|   591M|  8606   (1)| 00:00:01 |        |      |            |
|   1 |  PX COORDINATOR               |                     |       |       |            |          |        |      |            |
|   2 |   PX SEND QC (RANDOM)         | :TQ10003            |  2509K|   591M|  8606   (1)| 00:00:01 |  Q1,03 | P->S | QC (RAND)  |
|   3 |    HASH GROUP BY              |                     |  2509K|   591M|  8606   (1)| 00:00:01 |  Q1,03 | PCWP |            |
|   4 |     PX RECEIVE                |                     |  2509K|   591M|  8606   (1)| 00:00:01 |  Q1,03 | PCWP |            |
|   5 |      PX SEND HASH             | :TQ10002            |  2509K|   591M|  8606   (1)| 00:00:01 |  Q1,02 | P->P | HASH       |
|   6 |       HASH GROUP BY           |                     |  2509K|   591M|  8606   (1)| 00:00:01 |  Q1,02 | PCWP |            |
|*  7 |        HASH JOIN              |                     |  2509K|   591M|  8600   (1)| 00:00:01 |  Q1,02 | PCWP |            |
|   8 |         PX RECEIVE            |                     | 45666 |  9855K|    34   (0)| 00:00:01 |  Q1,02 | PCWP |            |
|   9 |          PX SEND HYBRID HASH  | :TQ10000            | 45666 |  9855K|    34   (0)| 00:00:01 |  Q1,00 | P->P | HYBRID HASH|
|  10 |           STATISTICS COLLECTOR|                     |       |       |            |          |  Q1,00 | PCWC |            |
|  11 |            PX BLOCK ITERATOR  |                     | 45666 |  9855K|    34   (0)| 00:00:01 |  Q1,00 | PCWC |            |
|  12 |             TABLE ACCESS FULL | DIM_CUSTOMERS_SCD   | 45666 |  9855K|    34   (0)| 00:00:01 |  Q1,00 | PCWP |            |
|  13 |         PX RECEIVE            |                     |  2699K|    66M|  8565   (1)| 00:00:01 |  Q1,02 | PCWP |            |
|  14 |          PX SEND HYBRID HASH  | :TQ10001            |  2699K|    66M|  8565   (1)| 00:00:01 |  Q1,01 | S->P | HYBRID HASH|
|  15 |           PX SELECTOR         |                     |       |       |            |          |  Q1,01 | SCWC |            |
|  16 |            TABLE ACCESS FULL  | FCT_RETAIL_SALES_DD |  2699K|    66M|  8565   (1)| 00:00:01 |  Q1,01 | SCWP |            |
-----------------------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   7 - access("A"."CUSTOMER_SURR_ID"="B"."CUSTOMER_SURR_ID")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - Degree of Parallelism is 4 because of table property
```

- Второй вариант.
- Можно по умолчанию задать для DDL-объекта распараллеливание при работе с ним, что и было сделано ниже. Такая команда применяется к уже созданным объектам. Для новых объектов можно указать параметр PARALLEL при создании, что будет описано ниже.

```
ALTER TABLE SALES_BY_CUSTOMERS PARALLEL 8;

EXPLAIN PLAN FOR
SELECT AGE_CATEGORY,
       COUNTRY,
       SUM(RECEIPTS_SUM)
FROM   SALES_BY_CUSTOMERS
GROUP BY AGE_CATEGORY,
         COUNTRY;

SELECT * FROM TABLE (dbms_xplan.display);
```

- Можно увидеть, что в плане выполнения запрос был распараллелен и без явного указания хинта. Операции прошли в порядке, аналогичном для предыдущего примера. Сначала таблица была захэширована и параллельно выполнялась к захэшированным наборам данных операция группировки.

Saved: 01-Dec-2017 18:24

```
PLAN_TABLE_OUTPUT
Plan hash value: 380891532

-----------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name               | Rows  | Bytes | Cost (%CPU)| Time     |    TQ  |IN-OUT| PQ Distrib |
-----------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                    |   824 | 20600 |    11  (19)| 00:00:01 |        |      |            |
|   1 |  PX COORDINATOR          |                    |       |       |            |          |        |      |            |
|   2 |   PX SEND QC (RANDOM)    | :TQ10001           |   824 | 20600 |    11  (19)| 00:00:01 |  Q1,01 | P->S | QC (RAND)  |
|   3 |    HASH GROUP BY         |                    |   824 | 20600 |    11  (19)| 00:00:01 |  Q1,01 | PCWP |            |
|   4 |     PX RECEIVE           |                    |   824 | 20600 |    11  (19)| 00:00:01 |  Q1,01 | PCWP |            |
|   5 |      PX SEND HASH        | :TQ10000           |   824 | 20600 |    11  (19)| 00:00:01 |  Q1,00 | P->P | HASH       |
|   6 |       HASH GROUP BY      |                    |   824 | 20600 |    11  (19)| 00:00:01 |  Q1,00 | PCWP |            |
|   7 |        PX BLOCK ITERATOR |                    | 42054 |  1026K|     9   (0)| 00:00:01 |  Q1,00 | PCWC |            |
|   8 |         TABLE ACCESS FULL| SALES_BY_CUSTOMERS | 42054 |  1026K|     9   (0)| 00:00:01 |  Q1,00 | PCWP |            |
-----------------------------------------------------------------------------------------------------------------------------

Note
-----
   - Degree of Parallelism is 4 because of table property
```

1.2. PARALLEL DDL

- Операции DDL могут быть параллельны, если в синтаксисе это явно объявлено. CTAS – широко используемая и полезная операция при для создания таблицы с параметром PARALLEL. Оператор CTAS содержит две части: часть CREATE (DDL) и часть SELECT (запрос). Oracle может парализовать обе части инструкции. Часть CREATE CTAS должна соответствовать тем же правилам, которые применяются к другим операциям DDL.

```
CREATE TABLE employees
PARALLEL 5
AS
SELECT position_name, count(position_name) AS amount
FROM bl_dm.dim_employees_scd
GROUP BY position_name;
```

```
EXPLAIN PLAN FOR
SELECT position_name, count(position_name) AS amount
FROM employees
WHERE position_name = 'financial advisor'
GROUP BY position_name;

SELECT * FROM TABLE (dbms_xplan.display);
```

- В плане выполнения можно наблюдать, что теперь DML-операции будут выполняться с распараллеливанием на данной таблице.

```
PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name       | Rows  | Bytes | Cost (%CPU)| Time     |    TQ  |IN-OUT| PQ Distrib |
-------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |            |     1 |    21 |     2   (0)| 00:00:01 |        |      |            |
|   1 |  PX COORDINATOR          |            |       |       |            |          |        |      |            |
|   2 |   PX SEND QC (RANDOM)    | :TQ10001   |     1 |    21 |     2   (0)| 00:00:01 |  Q1,01 | P->S | QC (RAND)  |
|   3 |    HASH GROUP BY         |            |     1 |    21 |     2   (0)| 00:00:01 |  Q1,01 | PCWP |            |
|   4 |     PX RECEIVE           |            |     1 |    21 |     2   (0)| 00:00:01 |  Q1,01 | PCWP |            |
|   5 |      PX SEND HASH        | :TQ10000   |     1 |    21 |     2   (0)| 00:00:01 |  Q1,00 | P->P | HASH       |
|   6 |       HASH GROUP BY      |            |     1 |    21 |     2   (0)| 00:00:01 |  Q1,00 | PCWP |            |
|   7 |        PX BLOCK ITERATOR |            |     1 |    21 |     2   (0)| 00:00:01 |  Q1,00 | PCWC |            |
|*  8 |         TABLE ACCESS FULL| EMPLOYEES  |     1 |    21 |     2   (0)| 00:00:01 |  Q1,00 | PCWP |            |
-------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   8 - filter("POSITION_NAME"='financial advisor')

Note
-----
   - Degree of Parallelism is 4 because of table property
```

1.3. PARALLEL DML

- При выполнении DML-операции с распараллеливанием можно увидеть, что такие же шаги предпринимают QC и PX, как и при выполнении обычного запроса. Выполняется параллельное хэширование двух таблиц. Первая таблица хэшируется и помещается в темповую таблицу, вторая хэшируется и онлайн сравниваются значения хэш-функции по строкам. Для каждой такой обработанной строки осуществляется обновление. Поскольку реализация операций выполняется параллельно, то перфоманс будет лучше, в чем и состоит главное преимущество PARALLEL.

```
UPDATE /*+ PARALLEL(a) */ sales_by_customers a
SET receipts_sum = receipts_sum * 1.1
WHERE country = 'India' AND age_category IN
  (SELECT age_category
   FROM BL_DM.DIM_CUSTOMERS_SCD
   WHERE region = 'Caribbean');
```

```
EXPLAIN PLAN FOR
UPDATE /*+ PARALLEL(a) */ sales_by_customers a
SET receipts_sum = receipts_sum * 1.1
WHERE country = 'India' AND age_category IN
  (SELECT age_category FROM BL_DM.DIM_CUSTOMERS_SCD WHERE region = 'Caribbean');

SELECT * FROM TABLE (dbms_xplan.display);
```

Script Output | Query Result

Task completed in 0.019 seconds

```
216 rows updated.

Plan FOR succeeded.

Plan FOR succeeded.
```

Saved: 01-Dec-2017 18:24

```
PLAN_TABLE_OUTPUT
Plan hash value: 4138917518

-----------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                  | Name                | Rows  | Bytes | Cost (%CPU)| Time     |    TQ  |IN-OUT| PQ Distrib |
-----------------------------------------------------------------------------------------------------------------------------------
|   0 | UPDATE STATEMENT           |                     |   180 | 24120 |   376   (1)| 00:00:01 |        |      |            |
|   1 |  UPDATE                    | SALES_BY_CUSTOMERS  |       |       |            |          |        |      |            |
|   2 |   PX COORDINATOR           |                     |       |       |            |          |        |      |            |
|   3 |    PX SEND QC (RANDOM)     | :TQ10002            |   180 | 24120 |   376   (1)| 00:00:01 | Q1,02  | P->S | QC (RAND)  |
|*  4 |     HASH JOIN SEMI BUFFERED|                     |   180 | 24120 |   376   (1)| 00:00:01 | Q1,02  | PCWP |            |
|   5 |      PX RECEIVE            |                     |   180 |  5400 |    68   (0)| 00:00:01 | Q1,02  | PCWP |            |
|   6 |       PX SEND HASH         | :TQ10000            |   180 |  5400 |    68   (0)| 00:00:01 | Q1,00  | P->P | HASH       |
|   7 |        PX BLOCK ITERATOR   |                     |   180 |  5400 |    68   (0)| 00:00:01 | Q1,00  | PCWC |            |
|*  8 |         TABLE ACCESS FULL  | SALES_BY_CUSTOMERS  |   180 |  5400 |    68   (0)| 00:00:01 | Q1,00  | PCWP |            |
|   9 |      PX RECEIVE            |                     |  5719 |   580K|   308   (1)| 00:00:01 | Q1,02  | PCWP |            |
|  10 |       PX SEND HASH         | :TQ10001            |  5719 |   580K|   308   (1)| 00:00:01 | Q1,01  | S->P | HASH       |
|  11 |        PX SELECTOR         |                     |       |       |            |          | Q1,01  | SCWC |            |
|* 12 |         TABLE ACCESS FULL  | DIM_CUSTOMERS_SCD   |  5719 |   580K|   308   (1)| 00:00:01 | Q1,01  | SCWP |            |
-----------------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
```

2. PARALLEL LOAD STRATEGY

Параллельная загрузка данных может повысить производительность выполняемых DML-операторов коренным образом. При последовательном выполнении операций каждая последующая должна ожидать окончания предшествующей. Параллельное выполнение помогает избежать простоя и позволяет следующей по очереди операции начинать обработку данных, которые уже были обработаны первой операцией, не смотря на то что полностью первая операция не завершилась.

При загрузке сгенерированных данных такую возможность можно применить к загрузке данных на BL_DM-слой. Поскольку при данной загрузке для каждой таблицы применяются операции SELECT и INSERT, то PARALLEL LOAD позволит делать выборку данных для вставки и загружать уже отобранные данные одновременно, что в теории должно повысить производительность выполнения наших запросов.

Проверим время план выполнения запроса до применения хинта на фактовой таблице.

```
Plan hash value: 4253285374

-----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                 | Name                    | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT          |                         |     1 |   185 | 13269   (1)| 00:00:01 |       |       |
|   1 |  LOAD TABLE CONVENTIONAL  | FCT_RETAIL_SALES_DD     |       |       |            |          |       |       |
|   2 |   SEQUENCE                | FCT_RETAIL_SALES_DD_SEQ |       |       |            |          |       |       |
|*  3 |    HASH JOIN              |                         |     1 |   185 | 13269   (1)| 00:00:01 |       |       |
|   4 |     NESTED LOOPS          |                         |     1 |   159 | 12177   (1)| 00:00:01 |       |       |
|*  5 |      HASH JOIN            |                         |     1 |   146 | 12177   (1)| 00:00:01 |       |       |
|*  6 |       HASH JOIN           |                         |     1 |   120 | 10539   (1)| 00:00:01 |       |       |
|*  7 |        HASH JOIN          |                         |     1 |    94 |  8901   (1)| 00:00:01 |       |       |
|   8 |         PARTITION RANGE ALL|                        |     1 |    26 |  3375   (1)| 00:00:01 |     1 |    11 |
|   9 |          PARTITION LIST ALL|                        |     1 |    26 |  3375   (1)| 00:00:01 |     1 |     6 |
|  10 |           TABLE ACCESS FULL| DIM_CUSTOMERS_SCD      |     1 |    26 |  3375   (1)| 00:00:01 |     1 |    66 |
|  11 |         TABLE ACCESS FULL  | CLS_FCT_RETAIL_SALES_DD|  2001K|  129M|  5520   (1)| 00:00:01 |       |       |
|  12 |        PARTITION LIST ALL  |                         |     1 |    26 |  1638   (1)| 00:00:01 |     1 |     6 |
```

SQL | 0.123 seconds

OPERATION	OBJECT_NAME	CARDINALITY	COST	PARTITION_START	PARTITION_STOP
INSERT STATEMENT		1	13269		
LOAD TABLE CONVENTIONAL	FCT_RETAIL_SALES_DD				
SEQUENCE	FCT_RETAIL_SALES_DD_SEQ				
HASH JOIN		1	13269		
Access Predicates					
A.EMPLOYEE_SURR_ID=C.EMPLOYEE_ID					
HASH JOIN		1	12177		
Access Predicates					
A.PRODUCT_SURR_ID=F.PRODUCT_ID					

```
2,001,998 rows inserted.

Elapsed: 00:06:53.634
```

План выполнения запроса с применением **-- + PARALLEL (4).**

```
Plan hash value: 2766409812

-----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                        | Name                    | Rows  | Bytes | Cost (%CPU)| Time
-----------------------------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT                 |                         |     1 |   185 |  3683   (1)| 00:00:01
|   1 |  LOAD TABLE CONVENTIONAL         | FCT_RETAIL_SALES_DD     |       |       |            |
|   2 |   SEQUENCE                       | FCT_RETAIL_SALES_DD_SEQ |       |       |            |
|   3 |    PX COORDINATOR                |                         |       |       |            |
|   4 |     PX SEND QC (RANDOM)          | :TQ10006                |     1 |   185 |  3683   (1)| 00:00:01
|*  5 |      HASH JOIN                   |                         |     1 |   185 |  3683   (1)| 00:00:01
|   6 |       PX RECEIVE                 |                         |     1 |   159 |  3380   (1)| 00:00:01
|   7 |        PX SEND BROADCAST         | :TQ10005                |     1 |   159 |  3380   (1)| 00:00:01
|   8 |         BUFFER SORT              |                         |   902 |  126K|            |
|   9 |          NESTED LOOPS            |                         |     1 |   159 |  3380   (1)| 00:00:01
|* 10 |           HASH JOIN              |                         |     1 |   146 |  3380   (1)| 00:00:01
|  11 |            PX RECEIVE            |                         |     1 |   120 |  2925   (1)| 00:00:01
|  12 |             PX SEND HYBRID HASH  | :TQ10003                |     1 |   120 |  2925   (1)| 00:00:01
```

SQL | 1.474 seconds

OPERATION	OBJECT_NAME	CARDINALITY	COST	PARTITION_START	PARTITION_STOP
INSERT STATEMENT		1	3683		
LOAD TABLE CONVENTIONAL	FCT_RETAIL_SALES_DD				
SEQUENCE	FCT_RETAIL_SALES_DD_SEQ				
PX COORDINATOR					
PX SEND (QC (RANDOM))	:TQ10006	1	3683		
HASH JOIN		1	3683		
Access Predicates					
A.EMPLOYEE_SURR_ID=C.EMPLOYEE_ID					
PX RECEIVE		1	3380		

```
2,001,998 rows inserted.

Elapsed: 00:09:02.015
```

Можно наблюдать, что несмотря на уменьшение параметра **COST**, параллель в данном случае оказала обратный эффект – **время выполнения запроса увеличилось на две минуты**, что весьма существенно. Применение параллели в данном случае нецелесообразно. Попробуем увеличить количество паралелльных процессов до 20.

План выполнения запроса с применением **-- + PARALLEL (10).**

Saved: 01-Dec-2017 18:24

```
Plan hash value: 2766409812

-----------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT         |                     |     1 |   185 |  1473   (1)| 00:00:01 |
|   1 |  LOAD TABLE CONVENTIONAL | FCT_RETAIL_SALES_DD |       |       |            |          |
|   2 |   SEQUENCE               | FCT_RETAIL_SALES_DD_SEQ |   |       |            |          |
|   3 |    PX COORDINATOR        |                     |       |       |            |          |
|   4 |     PX SEND QC (RANDOM)  | :TQ10006            |     1 |   185 |  1473   (1)| 00:00:01 |
|*  5 |      HASH JOIN           |                     |     1 |   185 |  1473   (1)| 00:00:01 |
|   6 |       PX RECEIVE         |                     |     1 |   159 |  1352   (1)| 00:00:01 |
|   7 |        PX SEND BROADCAST | :TQ10005            |     1 |   159 |  1352   (1)| 00:00:01 |
```

SQL | 4.581 seconds

OPERATION	OBJECT_NAME	CARDINALITY	COST	PARTITION_START	PARTITION_STOP
INSERT STATEMENT		1	1473		
LOAD TABLE CONVENTIONAL	FCT_RETAIL_SALES_DD				
SEQUENCE	FCT_RETAIL_SALES_DD_SEQ				
PX COORDINATOR					
PX SEND (QC (RANDOM))	:TQ10006	1	1473		
HASH JOIN		1	1473		
Access Predicates					
A.EMPLOYEE_SURR_ID=C.EMPLOYEE_ID					
PX RECEIVE		1	1352		
PX SEND (BROADCAST)	:TQ10005	1	1352		
BUFFER (SORT)		902			
NESTED LOOPS		1	1352		

```
2,001,998 rows inserted.

Elapsed: 00:20:18.325
```

Данный запрос выполнялся 20 минут.

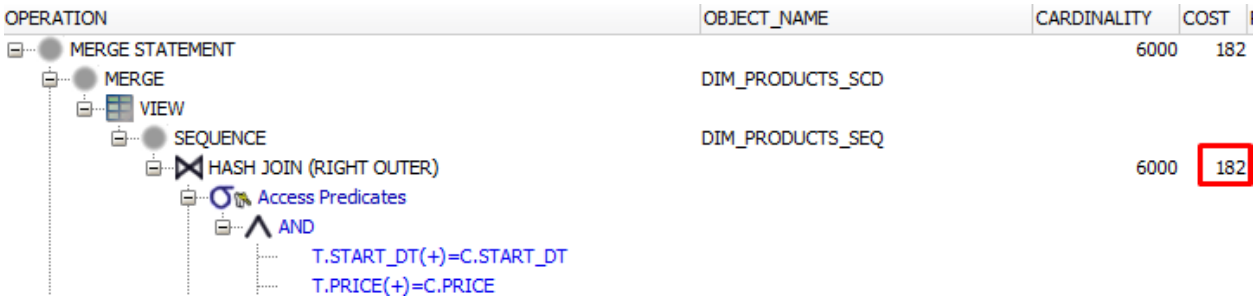
Очевидно, время выполнение загрузки только увеличивается с применением PARALLEL на фактовой таблице.

Протестируем хинт на DIMENSION-таблице DIM_PRODUCTS_SCD, поскольку она самая большая из DIMENSION-таблиц.

План выполнения запроса без применения **-- + PARALLEL.**

```
Plan hash value: 24526241

-----------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name                | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------------
|   0 | MERGE STATEMENT          |                     |  6000 |   14M|       |   182   (2)| 00:00:01 |
|   1 |  MERGE                   | DIM_PRODUCTS_SCD    |       |      |       |            |          |
|   2 |   VIEW                   |                     |       |      |       |            |          |
|   3 |    SEQUENCE              | DIM_PRODUCTS_SEQ    |       |      |       |            |          |
|*  4 |     HASH JOIN RIGHT OUTER|                     |  6000 |   14M|       |   182   (2)| 00:00:01 |
|   5 |      PARTITION LIST ALL  |                     |     1 | 1251 |       |     2   (0)| 00:00:01 |
|   6 |       PARTITION LIST ALL |                     |     1 | 1251 |       |     2   (0)| 00:00:01 |
|   7 |        TABLE ACCESS FULL | DIM_PRODUCTS_SCD    |     1 | 1251 |       |     2   (0)| 00:00:01 |
|   8 |      VIEW                |                     |  6000 | 7183K|       |   180   (2)| 00:00:01 |
|   9 |       MINUS              |                     |       |      |       |            |          |
|  10 |        SORT UNIQUE       |                     |  6000 |  621K|  920K|            |          |
```



OPERATION	OBJECT_NAME	CARDINALITY	COST
MERGE STATEMENT		6000	182
MERGE	DIM_PRODUCTS_SCD		
VIEW			
SEQUENCE	DIM_PRODUCTS_SEQ		
HASH JOIN (RIGHT OUTER)		6000	182
Access Predicates			
AND			
T.START_DT(+)=C.START_DT			
T.PRICE(+)=C.PRICE			

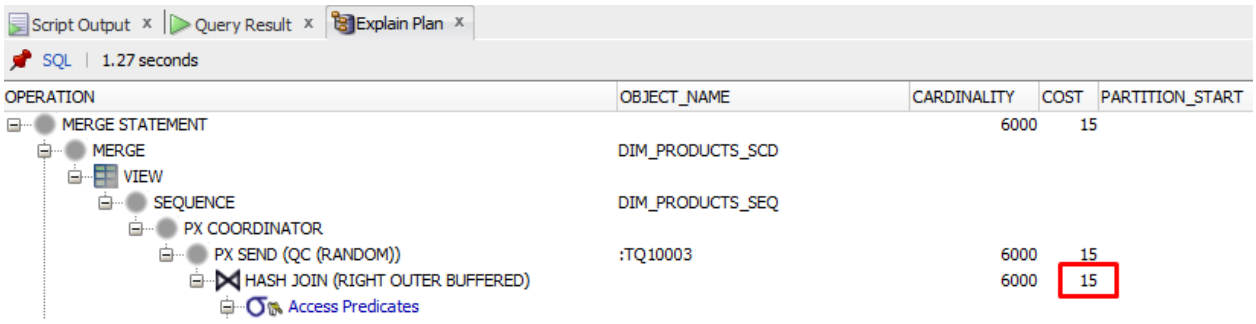
```
6,000 rows merged.

Elapsed: 00:00:01.440
```

План выполнения запроса с применением **-- + PARALLEL (4).**

```
Plan hash value: 3260348670

-----------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                        | Name            | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------------------------
|   0 | MERGE STATEMENT                  |                 |  6000 |    14M|       |    15  (20)| 00:00:01 |       |       |
|   1 |  MERGE                           | DIM_PRODUCTS_SCD |       |       |       |            |          |       |       |
|   2 |   VIEW                           |                 |       |       |       |            |          |       |       |
|   3 |    SEQUENCE                      | DIM_PRODUCTS_SEQ |       |       |       |            |          |       |       |
|   4 |     PX COORDINATOR               |                 |       |       |       |            |          |       |       |
|   5 |      PX SEND QC (RANDOM)         | :TQ10003        |  6000 |    14M|       |    15  (20)| 00:00:01 |       |       |
|*  6 |       HASH JOIN RIGHT OUTER BUFFERED|              |  6000 |    14M|       |    15  (20)| 00:00:01 |       |       |
|   7 |        PX PARTITION LIST ALL     |                 |     1 |  1251 |       |     2   (0)| 00:00:01 |     1 |     3 |
|   8 |         PX PARTITION LIST ALL    |                 |     1 |  1251 |       |     2   (0)| 00:00:01 |     1 |     2 |
|   9 |          TABLE ACCESS FULL       | DIM_PRODUCTS_SCD |     1 |  1251 |       |     2   (0)| 00:00:01 |     1 |     6 |
|  10 |        PX RECEIVE                |                 |  6000 |  7183K|       |    13  (24)| 00:00:01 |       |       |
-----------------------------------------------------------------------------------------------------------------------------
```



Script Output | Query Result | Explain Plan

SQL | 1.27 seconds

OPERATION	OBJECT_NAME	CARDINALITY	COST	PARTITION_START
MERGE STATEMENT		6000	15	
MERGE	DIM_PRODUCTS_SCD			
VIEW				
SEQUENCE	DIM_PRODUCTS_SEQ			
PX COORDINATOR				
PX SEND (QC (RANDOM))	:TQ10003	6000	15	
HASH JOIN (RIGHT OUTER BUFFERED)		6000	15	
Access Predicates				

```
6,000 rows merged.

Elapsed: 00:00:01.204
```

Дименшен-таблицы намного меньше фактовой таблицы и загружаются довольно быстро, поэтому применение параллельной загрузки на них также нецелесообразно. **Ускорение запроса на милисекунды не окажет существенной разницы на перфоманс в спроектированном DWH.**

Таким образом стретегия параллельной загрузки в данной работе применяться не будет.