Workshop Multitenant, Multimodel, In-Memory para la base de Datos Oracle

Parte 3 de 3

Contenidos

In-Memory (30 min)

Configuración del área de memoria (5 min)

Todas las prácticas de hoy se realizarán con la base de datos CDBA, por lo que debemos cargar sus variables de entorno al comienzo de las prácticas. Antes de comenzar, debemos asegurar que la instancia CDBB está parada. Además, dejaremos cargadas las variables de entorno de la instancia CDBA, dado que es la instancia donde realizaremos las prácticas hoy:

```
[oracle@single19c]$  . CDBB.env
[oracle@single19c]$ srvctl stop database -d CDBB -o immediate
[oracle@single19c]$ . CDBA.env
```

En primer lugar comprobamos la configuración de In-Memory que hay en la base de datos.

```
**************************************************************
A. In-Memory Column Store (IM column store) dynamic resizing:
**************************************************************

sqlplus / as sysdba

SQL> show parameter inmemo

NAME                                 TYPE      VALUE
------------------------------------ ----------- ------------------------------
inmemory_adg_enabled                       boolean   TRUE
inmemory_automatic_level             string    OFF
inmemory_clause_default              string
inmemory_expressions_usage           string    ENABLE
inmemory_force                             string    DEFAULT
inmemory_max_populate_servers              integer   0
inmemory_optimized_arithmetic              string    DISABLE
inmemory_prefer_xmem_memcompress           string
inmemory_prefer_xmem_priority              string
inmemory_query                             string    ENABLE
inmemory_size                        big integer 0
inmemory_trickle_repopulate_servers_ integer   1
percent
inmemory_virtual_columns             string    MANUAL
inmemory_xmem_size                   big integer 0
optimizer_inmemory_aware             boolean   TRUE
```

```
SQL> select version from v$instance;

VERSION
-----------------
19.0.0.0.0
```

Para activar IMC, hay que poner inmemory_size > 0 y reiniciar la instancia.
Aprovecharemos también para incrementar la sga de la misma:

```
SQL> show parameter sga

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
allow_group_access_to_sga            boolean     FALSE
lock_sga                             boolean     FALSE
pre_page_sga                         boolean     TRUE
sga_max_size                         big integer 4G
sga_min_size                         big integer 0
sga_target                           big integer 4G
unified_audit_sga_queue_size         integer     1048576

SQL> alter system set sga_max_size=6G scope=spfile;

System altered.

SQL> alter system set sga_target=6G  scope=spfile;

System altered.

SQL> alter system set inmemory_size = 2G scope=spfile;

System altered.

SQL> exit

[oracle@single19c ~]$ srvctl stop database -d $ORACLE_UNQNAME -o immediate
[oracle@single19c ~]$ srvctl start database -d $ORACLE_UNQNAME

$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 12 11:13:37 2022
Version 19.13.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
```

```
Version 19.13.0.0.0

SQL> show parameter inmemo

NAME                                 TYPE      VALUE
------------------------------------ ----------- ------------------------------
inmemory_adg_enabled                         boolean   TRUE
inmemory_automatic_level             string    OFF
inmemory_clause_default              string
inmemory_expressions_usage           string    ENABLE
inmemory_force                               string    DEFAULT
inmemory_max_populate_servers                integer   2
inmemory_optimized_arithmetic                string    DISABLE
inmemory_prefer_xmem_memcompress            string
inmemory_prefer_xmem_priority                string
inmemory_query                               string    ENABLE
inmemory_size                        big integer 2G
inmemory_trickle_repopulate_servers_ integer   1
percent
inmemory_virtual_columns             string    MANUAL
inmemory_xmem_size                   big integer 0
optimizer_inmemory_aware             boolean   TRUE
```

El resize puede ser dinámico :

```
SQL> alter system set inmemory_size = 3G scope=both;

System altered.

SQL> show parameter inmemory

NAME                                 TYPE      VALUE
------------------------------------ ----------- ------------------------------
inmemory_adg_enabled                         boolean   TRUE
inmemory_automatic_level             string    OFF
inmemory_clause_default              string
inmemory_expressions_usage           string    ENABLE
inmemory_force                               string    DEFAULT
inmemory_max_populate_servers                integer   2
inmemory_optimized_arithmetic                string    DISABLE
inmemory_prefer_xmem_memcompress            string
inmemory_prefer_xmem_priority                string
inmemory_query                               string    ENABLE
inmemory_size                        big integer 3G
inmemory_trickle_repopulate_servers_ integer   1
percent
inmemory_virtual_columns             string    MANUAL
inmemory_xmem_size                   big integer 0
optimizer_inmemory_aware             boolean   TRUE
```

El proceso de resize dinámico se puede hacer solo al alza:

```
SQL> alter system set inmemory_size = 1G scope=both;
alter system set inmemory_size = 1G scope=both
*
ERROR at line 1:
ORA-02097: parameter cannot be modified because specified value is invalid
ORA-02095: specified initialization parameter cannot be modified
```

Configurar FastStart (5 min)

El área FastStart es un espacio de tablas designado donde IM FastStart almacena y gestiona los datos de los objetos INMEMORY. Oracle Database gestiona los Espacios de tablas FastStart automáticamente.
En una base de datos Oracle RAC, todos los nodos comparten los datos de FastStart.

```
[oracle@single19]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Thu Jan 13 09:04:06 2022
Version 19.13.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.13.0.0.0

SQL>
Connected.

col tablespace_name format a30

select con_id, TABLESPACE_NAME, STATUS FROM V$INMEMORY_FASTSTART_AREA;

    CON_ID TABLESPACE_NAME         STATUS
---------- ------------------------------ ----------
         1 INVALID_TABLESPACE             DISABLE
         2 INVALID_TABLESPACE             DISABLE
         3 INVALID_TABLESPACE             DISABLE
         4 INVALID_TABLESPACE             DISABLE
         5 INVALID_TABLESPACE             DISABLE

SQL> conn system/We1c0m3_We1c0m3_@SOE

Connected.

SQL> create tablespace TBS_IMC_FASTSTART datafile size 8G;

Tablespace created.

SQL> EXEC DBMS_INMEMORY_ADMIN.FASTSTART_ENABLE('TBS_IMC_FASTSTART')
```

```
PL/SQL procedure successfully completed.


SQL> conn / as sysdba

Connected.

col tablespace_name format a30

SQL> select con_id, TABLESPACE_NAME, STATUS FROM V$INMEMORY_FASTSTART_AREA;

    CON_ID TABLESPACE_NAME         STATUS
---------- ------------------------------ ----------
         1 INVALID_TABLESPACE             DISABLE
         2 INVALID_TABLESPACE             DISABLE
         3 INVALID_TABLESPACE             DISABLE
         4 INVALID_TABLESPACE             DISABLE
         5 TBS_IMC_FASTSTART              ENABLE

SQL> conn system/We1c0m3_We1c0m3_@SOE

Connected.

SQL> COL TABLESPACE_NAME FORMAT a20

SQL> SELECT TABLESPACE_NAME, STATUS,
( (ALLOCATED_SIZE/1024) / 1024 ) AS ALLOC_MB,
( (USED_SIZE/1024) / 1024 ) AS USED_MB
FROM V$INMEMORY_FASTSTART_AREA;

TABLESPACE_NAME      STATUS        ALLOC_MB    USED_MB
-------------------- ---------- ---------- ----------
TBS_IMC_FASTSTART    ENABLE            8192       1
```

Algunas notas sobre Fast Start:

- No se puede forzar de forma manual una escritura al FS !!!
- Se puede migrar el contenido del FS a otro TBS:

```
EXEC DBMS_INMEMORY_ADMIN.FASTSTART_MIGRATE_STORAGE('new_fs_tbs')
```

- Se puede deshabilitar el FS fastStart:

```
EXEC DBMS_INMEMORY_ADMIN.FASTSTART_DISABLE
```

Para desactivar el FAST START

```
SQL> conn system/We1c0m3_We1c0m3_@SOE

Connected.
```

```
SQL> EXEC DBMS_INMEMORY_ADMIN.FASTSTART_DISABLE

PL/SQL procedure successfully completed.

SQL> drop tablespace TBS_IMC_FASTSTART including contents and datafiles;

Tablespace dropped.
```

Publicar las tablas SSB en in-memory (5 min)

```
SQL> conn ssb/ssb@SOE

Connected.

col table_name format a30
set lines 120

--display current status

SQL> select table_name,
       inmemory,
       inmemory_priority,
       inmemory_compression
from user_tables;

TABLE_NAME                     INMEMORY INMEMORY INMEMORY_COMPRESS
------------------------------ -------- -------- -----------------
LINEORDER                      DISABLED
LINEORDER_ACO                  DISABLED
ETL_LO                         DISABLED
RESULTS                        DISABLED
DATE_DIM                       DISABLED
ETL_DD                         DISABLED
SUPPLIER                       DISABLED
PART                           DISABLED
CUSTOMER                       DISABLED
TMP                            DISABLED
YEARLY_PROFIT_REP_MV                  DISABLED
LINEORDER_NO_ACO               DISABLED

12 rows selected.


--alter tables in memory

SQL> alter table lineorder inmemory;
alter table part inmemory;
alter table customer inmemory;
alter table supplier inmemory;
alter table date_dim inmemory;

Table altered.
```

```
Table altered.

Table altered.

Table altered.

Table altered.

SQL> select table_name,
       inmemory,
       inmemory_priority,
       inmemory_compression
from user_tables;

TABLE_NAME                     INMEMORY INMEMORY INMEMORY_COMPRESS
------------------------------ -------- -------- -----------------
LINEORDER                      ENABLED      NONE    FOR QUERY LOW
LINEORDER_ACO                  DISABLED
ETL_LO                         DISABLED
RESULTS                        DISABLED
DATE_DIM                       ENABLED      NONE    FOR QUERY LOW
ETL_DD                         DISABLED
SUPPLIER                       ENABLED      NONE    FOR QUERY LOW
PART                           ENABLED      NONE    FOR QUERY LOW
CUSTOMER                       ENABLED      NONE    FOR QUERY LOW
TMP                            DISABLED
YEARLY_PROFIT_REP_MV                DISABLED
LINEORDER_NO_ACO               DISABLED

11 rows selected.

--fetch all rows to start population

SQL> select count(*) from lineorder;
select count(*) from part;
select count(*) from customer;
select count(*) from supplier;
select count(*) from date_dim;

  COUNT(*)
----------
  29999870

  COUNT(*)
----------
    600000

  COUNT(*)
----------
    150000

  COUNT(*)
----------
     10000
```

```
  COUNT(*)
----------
      2560
```

Monitorizar la publicación de SSB en In Memory

```
SQL> conn system/We1c0m3_We1c0m3_@SOE

Connected.

--new view v$im_segments

SQL> desc v$im_segments

 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
-
 OWNER                                              VARCHAR2(128)
 SEGMENT_NAME                                       VARCHAR2(128)
 PARTITION_NAME                                     VARCHAR2(128)
 SEGMENT_TYPE                                       VARCHAR2(18)
 TABLESPACE_NAME                                    VARCHAR2(128)
 INMEMORY_SIZE                                             NUMBER
 BYTES                                              NUMBER
 BYTES_NOT_POPULATED                                       NUMBER
 POPULATE_STATUS                                    VARCHAR2(13)
 INMEMORY_PRIORITY                                  VARCHAR2(8)
 INMEMORY_DISTRIBUTE                                       VARCHAR2(15)
 INMEMORY_DUPLICATE                                 VARCHAR2(13)
 INMEMORY_COMPRESSION                                      VARCHAR2(17)
 INMEMORY_SERVICE                                   VARCHAR2(12)
 INMEMORY_SERVICE_NAME                                     VARCHAR2(129)
 IS_EXTERNAL                                        VARCHAR2(5)
 CON_ID                                             NUMBER


col owner format a12
col name format a30
col partition_name format a30
set lines 120

--population status

SQL> select v.owner,v.segment_name name,v.partition_name,
v.populate_status status, v.bytes_not_populated
from v$im_segments v
Order by 1;

OWNER       NAME                      PARTITION_NAME           STATUS
BYTES_NOT_POPULATED
```

```
----------- ------------------------------ ------------------------------ ----
--------- -------------------
SSB         LINEORDER                                                 STARTED
          401473536
SSB         PART                                                    COMPLETED
          0
SSB         CUSTOMER                                                COMPLETED
          0
SSB         SUPPLIER                                                COMPLETED
          0
SSB         DATE_DIM                                                COMPLETED
          0

6 rows selected.
```

(La anterior query se puede ejecutar varias veces para ver como van quedando menos bytes por subir a memoria)

```
--size

SQL> select v.owner,v.segment_name name,
round(v.bytes/1024/1024,3) orig_size,
round(v.inmemory_size/1024/1024,3) in_mem_size,
ROUND(v.bytes/v.inmemory_size,2) comp_ratio
from v$im_segments v
order by 1;

OWNER       NAME                         ORIG_SIZE IN_MEM_SIZE COMP_RATIO
----------- ---------------------------- ---------- ----------- ----------
SSB         SUPPLIER                           .836        1.25        .67
SSB         DATE_DIM                           .117        1.25        .09
SSB         LINEORDER                        1745.32         1494.375    1.17
SSB         CUSTOMER                       11.852       11.25     1.05
SSB         PART                           40.563      13.438     3.02

6 rows selected.
```

Queries Sencillas

Comprobar la diferencia de acceso entre las distintas queries.

Single Table Scan

Query 1:

```
SQL> conn ssb/ssb@SOE

Connected.

/********************Query 1************************/
/************Single Column Aggregation***************/
/****************************************************/
```

```
/***************Parallel Disk Access*****************/
/****************************************************/

clear scr

--flush the buffer_cache

SQL> alter system flush buffer_cache;
SQL> alter session force parallel query parallel 4;

System altered.

Session altered.


set lines 120
set autotrace traceonly explain statistics
set timing on

select /*+ NO_INMEMORY */ /* DISK ACCESS */
max(lo_ordtotalprice) most_expensive_order From LINEORDER;

Elapsed: 00:00:32.22

Execution Plan
----------------------------------------------------------
Plan hash value: 396151021

-------------------------------------------------------------------------------
---------------------------------
| Id  | Operation              | Name      | Rows  | Bytes | Cost (%CPU)| Time
|    TQ  |IN-OUT| PQ Distrib |
-------------------------------------------------------------------------------
---------------------------------
|   0 | SELECT STATEMENT       |           |     1 |     6 | 17006   (1)|
00:00:01 |        |      |            |
|   1 |  SORT AGGREGATE        |           |     1 |     6 |            |
|        |      |            |
|   2 |   PX COORDINATOR       |           |       |       |            |
|        |      |            |
|   3 |    PX SEND QC (RANDOM) | :TQ10000  |     1 |     6 |            |
|  Q1,00 | P->S | QC (RAND)  |
|   4 |     SORT AGGREGATE     |           |     1 |     6 |            |
|  Q1,00 | PCWP |            |
|   5 |      PX BLOCK ITERATOR |           |    29M|   171M| 17006   (1)|
00:00:01 |  Q1,00 | PCWC |            |
|   6 |       TABLE ACCESS FULL| LINEORDER |    29M|   171M| 17006   (1)|
00:00:01 |  Q1,00 | PCWP |            |
-------------------------------------------------------------------------------
---------------------------------

Note
-----
   - Degree of Parallelism is 4 because of session
```

```
Statistics
----------------------------------------------------------
       1338  recursive calls
          0  db block gets
     225488  consistent gets
     223484  physical reads
       5548  redo size
        565  bytes sent via SQL*Net to client
        463  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
        110  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

```
/**************In-Memory Serial Access***************/
/*****************************************************/

SQL> clear scr

--column store enabled via INMEMORY_QUERY parameter—

SQL> alter session disable parallel query;

Session altered


SQL> Select /*+ INMEMORY */ /*IN-MEMORY Serial*/
max(lo_ordtotalprice) most_expensive_order From LINEORDER;

Elapsed: 00:00:00.15

Execution Plan
----------------------------------------------------------
Plan hash value: 2267213921

-------------------------------------------------------------------------------
----------
| Id  | Operation                   | Name      | Rows  | Bytes | Cost (%CPU)|
Time     |
-------------------------------------------------------------------------------
----------
|   0 | SELECT STATEMENT            |           |     1 |     6 |  2561  (12)|
00:00:01 |
|   1 |  SORT AGGREGATE             |           |     1 |     6 |            |
|
|   2 |   TABLE ACCESS INMEMORY FULL| LINEORDER |    29M|   171M|  2561  (12)|
00:00:01 |
-------------------------------------------------------------------------------
----------
```

```
Statistics
----------------------------------------------------------
       1252  recursive calls
          8  db block gets
       2156  consistent gets
          4  physical reads
       1408  redo size
        565  bytes sent via SQL*Net to client
        463  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
        125  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

```
/*************In-Memory Parallel Access*************/
/***************************************************/

SQL> clear scr

SQL> alter session force parallel query parallel 4;

Session altered.

SQL> select /*+ INMEMORY */ /*IN-MEMORY Parallel*/
max(lo_ordtotalprice) most_expensive_order From LINEORDER;

Elapsed: 00:00:00.34

Execution Plan
----------------------------------------------------------
Plan hash value: 396151021

-------------------------------------------------------------------------------
----------------------------------------
--

| Id  | Operation                       | Name      | Rows  | Bytes | Cost
(%CPU)| Time     |    TQ  |IN-OUT| PQ Distrib
 |

-------------------------------------------------------------------------------
----------------------------------------
--

|   0 | SELECT STATEMENT                |           |     1 |     6 |   711
(12)| 00:00:01 |        |      |
 |

|   1 |  SORT AGGREGATE                 |           |     1 |     6 |
|          |        |      |
```

```
 |

|   2 |   PX COORDINATOR               |           |       |       |
|          |        |      |
 |

|   3 |    PX SEND QC (RANDOM)         | :TQ10000  |     1 |     6 |
|          |  Q1,00 | P->S | QC (RAND)
 |

|   4 |     SORT AGGREGATE             |           |     1 |     6 |
|          |  Q1,00 | PCWP |
 |

|   5 |      PX BLOCK ITERATOR         |           |    29M|   171M|   711
(12)| 00:00:01 |  Q1,00 | PCWC |
 |

|   6 |       TABLE ACCESS INMEMORY FULL| LINEORDER |    29M|   171M|   711
(12)| 00:00:01 |  Q1,00 | PCWP |
 |

-------------------------------------------------------------------------------
----------------------------------------
--


Note
-----
   - Degree of Parallelism is 4 because of session


Statistics
----------------------------------------------------------
       1323  recursive calls
          2  db block gets
       2386  consistent gets
          0  physical reads
          0  redo size
        565  bytes sent via SQL*Net to client
        465  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
        137  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

Query 2

```
/******Single Column Aggregation with Filter*********/
/************************************************/

/***************Parallel Disk Access*****************/
```

```
/************************************************/

SQL> clear scr

--flush the buffer_cache

SQL> alter system flush buffer_cache;
SQL> alter session force parallel query parallel 4;

SQL> select /*+ NO_INMEMORY */ /* DISK ACCESS */
max(lo_ordtotalprice) most_expensive_order From LINEORDER
where LO_PARTKEY=300023;

Elapsed: 00:00:30.27

Execution Plan
----------------------------------------------------------
Plan hash value: 396151021

-------------------------------------------------------------------------------
---------------------------------
| Id  | Operation              | Name      | Rows  | Bytes | Cost (%CPU)| Time
|    TQ  |IN-OUT| PQ Distrib |
-------------------------------------------------------------------------------
---------------------------------
|   0 | SELECT STATEMENT       |           |     1 |    11 | 16966   (1)|
00:00:01 |        |      |            |
|   1 |  SORT AGGREGATE        |           |     1 |    11 |            |
|        |      |            |
|   2 |   PX COORDINATOR       |           |       |       |            |
|        |      |            |
|   3 |    PX SEND QC (RANDOM) | :TQ10000  |     1 |    11 |            |
|  Q1,00 | P->S | QC (RAND)  |
|   4 |     SORT AGGREGATE     |           |     1 |    11 |            |
|  Q1,00 | PCWP |            |
|   5 |      PX BLOCK ITERATOR |           |    76 |   836 | 16966   (1)|
00:00:01 |  Q1,00 | PCWC |            |
|*  6 |       TABLE ACCESS FULL| LINEORDER |    76 |   836 | 16966   (1)|
00:00:01 |  Q1,00 | PCWP |            |
-------------------------------------------------------------------------------
---------------------------------

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

   6 - filter("LO_PARTKEY"=300023)

Note
-----
   - Degree of Parallelism is 4 because of session


Statistics
----------------------------------------------------------
```

```
      1275  recursive calls
         7  db block gets
    225541  consistent gets
    223510  physical reads
      8540  redo size
       565  bytes sent via SQL*Net to client
       487  bytes received via SQL*Net from client
         2  SQL*Net roundtrips to/from client
       121  sorts (memory)
         0  sorts (disk)
         1  rows processed
```

```
/**************In-Memory Serial Access***************/
/*****************************************************/

SQL> clear scr

SQL> alter session disable parallel query;

SQL> Select /*+ INMEMORY */ /*IN-MEMORY Serial*/
max(lo_ordtotalprice) most_expensive_order From LINEORDER
where LO_PARTKEY=300023;

Elapsed: 00:00:00.09

Execution Plan
----------------------------------------------------------
Plan hash value: 2267213921

-------------------------------------------------------------------------------
----------
| Id  | Operation                   | Name      | Rows  | Bytes | Cost (%CPU)|
Time     |
-------------------------------------------------------------------------------
----------
|   0 | SELECT STATEMENT            |           |     1 |    11 |  2814  (20)|
00:00:01 |
|   1 |  SORT AGGREGATE             |           |     1 |    11 |            |
|
|*  2 |   TABLE ACCESS INMEMORY FULL| LINEORDER |    76 |   836 |  2814  (20)|
00:00:01 |
-------------------------------------------------------------------------------
----------

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

```
   2 - inmemory("LO_PARTKEY"=300023)
       filter("LO_PARTKEY"=300023)


Statistics
----------------------------------------------------------
        736  recursive calls
          0  db block gets
        827  consistent gets
          1  physical reads
        120  redo size
        565  bytes sent via SQL*Net to client
        487  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
         82  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

```
/*************In-Memory Parallel Access*************/
/***************************************************/

SQL> clear scr

SQL> alter session force parallel query parallel 4;

SQL> select /*+ INMEMORY */ /*IN-MEMORY Parallel*/
max(lo_ordtotalprice) most_expensive_order From LINEORDER
where LO_PARTKEY=300023;

Elapsed: 00:00:00.05

Execution Plan
----------------------------------------------------------
Plan hash value: 396151021

-------------------------------------------------------------------------------
----------------------------------------
--

| Id  | Operation                         | Name      | Rows  | Bytes | Cost
(%CPU)| Time     |    TQ  |IN-OUT| PQ Distrib
 |

-------------------------------------------------------------------------------
----------------------------------------
--

|   0 | SELECT STATEMENT                  |           |     1 |    11 |   782
(20)| 00:00:01 |        |      |
 |
```

```
|   1 |  SORT AGGREGATE                |           |     1 |    11 |
|          |        |      |
 |

|   2 |   PX COORDINATOR               |           |       |       |
|          |        |      |
 |

|   3 |    PX SEND QC (RANDOM)         | :TQ10000  |     1 |    11 |
|          |  Q1,00 | P->S | QC (RAND)
 |

|   4 |     SORT AGGREGATE             |           |     1 |    11 |
|          |  Q1,00 | PCWP |
 |

|   5 |      PX BLOCK ITERATOR         |           |    76 |   836 |   782
(20)| 00:00:01 |  Q1,00 | PCWC |
 |

|*  6 |       TABLE ACCESS INMEMORY FULL| LINEORDER |    76 |   836 |   782
(20)| 00:00:01 |  Q1,00 | PCWP |
 |

-------------------------------------------------------------------------------
----------------------------------------
--


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

   6 - inmemory("LO_PARTKEY"=300023)
       filter("LO_PARTKEY"=300023)

Note
-----
   - Degree of Parallelism is 4 because of session


Statistics
----------------------------------------------------------
         13  recursive calls
          0  db block gets
        115  consistent gets
          0  physical reads
          0  redo size
        565  bytes sent via SQL*Net to client
        489  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

Queries de grado medio

Two table scan

Query 1

```
SQL> conn ssb/ssb@SOE

/**********Group Aggregate with two table join*******/
/*************************************************/

/**************Parallel Disk Access*****************/
/*************************************************/

SQL> clear scr

--flush the buffer_cache

SQL> alter system flush buffer_cache;
SQL> alter session force parallel query parallel 4;
SQL> alter session set inmemory_query='DISABLE';
SQL> set autotrace traceonly explain statistics
SQL> Set timing on

SQL> select /*+ NO_INMEMORY */ /* DISK ACCESS */
d_date,sum(l.lo_revenue) "Total Revenue"
From   LINEORDER l, DATE_DIM d
Where  l.lo_orderdate = d.d_datekey
and      D_DAYNUMINMONTH = 25
and    d.d_month = 'December'
group by d_date
order by d_date;

6 rows selected.

Elapsed: 00:00:30.12

Execution Plan
----------------------------------------------------------
Plan hash value: 803921888

-------------------------------------------------------------------------------
----------------------------------------
--------------------------

| Id  | Operation                                 | Name                      |
Rows  | Bytes | Cost (%CPU)| Time     |
  TQ  |IN-OUT| PQ Distrib |

-------------------------------------------------------------------------------
----------------------------------------
```

```
---------------------------

|   0 | SELECT STATEMENT                          |                                |
7 |   343 | 17032   (2)| 00:00:01 |
      |      |            |

|   1 |  TEMP TABLE TRANSFORMATION                |                                |
|       |            |          |
      |      |            |

|   2 |   LOAD AS SELECT (CURSOR DURATION MEMORY)| SYS_TEMP_0FD9D6651_290D25 |
|       |            |          |
      |      |            |

|   3 |    PX COORDINATOR                         |                                |
|       |            |          |
      |      |            |

|   4 |     PX SEND QC (RANDOM)                   | :TQ10001                       |
7 |   294 |     5  (20)| 00:00:01 |
Q1,01 | P->S | QC (RAND)  |

|   5 |      HASH GROUP BY                        |                                |
7 |   294 |     5  (20)| 00:00:01 |
Q1,01 | PCWP |            |

|   6 |       PX RECEIVE                          |                                |
7 |   294 |     4   (0)| 00:00:01 |
Q1,01 | PCWP |            |

|   7 |        PX SEND HASH                       | :TQ10000                       |
7 |   294 |     4   (0)| 00:00:01 |
Q1,00 | P->P | HASH       |

|   8 |         KEY VECTOR CREATE BUFFERED        | :KV0000                        |
7 |   294 |     4   (0)| 00:00:01 |
Q1,00 | PCWC |            |

|   9 |          PX BLOCK ITERATOR                |                                |
7 |   266 |     4   (0)| 00:00:01 |
Q1,00 | PCWC |            |

|* 10 |           TABLE ACCESS FULL               | DATE_DIM                       |
7 |   266 |     4   (0)| 00:00:01 |
Q1,00 | PCWP |            |

|  11 |   PX COORDINATOR                          |                                |
|       |            |          |
      |      |            |

|  12 |    PX SEND QC (ORDER)                     | :TQ20004                       |
7 |   343 | 17026   (2)| 00:00:01 |
Q2,04 | P->S | QC (ORDER) |
```

```
|  13 |     SORT ORDER BY                      |                          |
7 |   343 | 17026   (2)| 00:00:01 |
Q2,04 | PCWP |            |

|  14 |      PX RECEIVE                        |                          |
7 |   343 | 17025   (2)| 00:00:01 |
Q2,04 | PCWP |            |

|  15 |       PX SEND RANGE                    | :TQ20003                 |
7 |   343 | 17025   (2)| 00:00:01 |
Q2,03 | P->P | RANGE      |

|* 16 |        HASH JOIN BUFFERED              |                          |
7 |   343 | 17025   (2)| 00:00:01 |
Q2,03 | PCWP |            |

|  17 |         PX RECEIVE                     |                          |
7 |   119 | 17023   (2)| 00:00:01 |
Q2,03 | PCWP |            |

|  18 |          PX SEND HYBRID HASH           | :TQ20001                 |
7 |   119 | 17023   (2)| 00:00:01 |
Q2,01 | P->P | HYBRID HASH|

|  19 |           STATISTICS COLLECTOR         |                          |
|       |            |          |
Q2,01 | PCWC |            |

|  20 |            VIEW                        | VW_VT_80F21617           |
7 |   119 | 17023   (2)| 00:00:01 |
Q2,01 | PCWP |            |

|  21 |             HASH GROUP BY              |                          |
7 |   112 | 17023   (2)| 00:00:01 |
Q2,01 | PCWP |            |

|  22 |              PX RECEIVE                |                          |
7 |   112 | 17023   (2)| 00:00:01 |
Q2,01 | PCWP |            |

|  23 |               PX SEND HASH             | :TQ20000                 |
7 |   112 | 17023   (2)| 00:00:01 |
Q2,00 | P->P | HASH       |

|  24 |                VECTOR GROUP BY         |                          |
7 |   112 | 17023   (2)| 00:00:01 |
Q2,00 | PCWP |            |

|  25 |                 HASH GROUP BY          |                          |
7 |   112 | 17023   (2)| 00:00:01 |
Q2,00 | PCWP |            |

|  26 |                  KEY VECTOR USE        | :KV0000                  |
87245 |  1363K| 17023   (2)| 00:00:01 |
```

```
Q2,00 | PCWC |            |

|  27 |                   PX BLOCK ITERATOR      |                              |
29M|   343M| 17023   (2)| 00:00:01 |
Q2,00 | PCWC |            |

|* 28 |                    TABLE ACCESS FULL     | LINEORDER                    |
29M|   343M| 17023   (2)| 00:00:01 |
Q2,00 | PCWP |            |

|  29 |         PX RECEIVE                       |                              |
7 |   224 |     2   (0)| 00:00:01 |
Q2,03 | PCWP |            |

|  30 |          PX SEND HYBRID HASH             | :TQ20002                     |
7 |   224 |     2   (0)| 00:00:01 |
Q2,02 | P->P | HYBRID HASH|

|  31 |           PX BLOCK ITERATOR              |                              |
7 |   224 |     2   (0)| 00:00:01 |
Q2,02 | PCWC |            |

|  32 |            TABLE ACCESS FULL             | SYS_TEMP_0FD9D6651_290D25 |
7 |   224 |     2   (0)| 00:00:01 |
Q2,02 | PCWP |            |

--------------------------------------------------------------------------------
---------------------------------------
---------------------------


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

  10 - filter("D_DAYNUMINMONTH"=25 AND "D"."D_MONTH"='December')
  16 - access("ITEM_5"=INTERNAL_FUNCTION("C0"))
  28 - filter(SYS_OP_KEY_VECTOR_FILTER("L"."LO_ORDERDATE",:KV0000))

Note
-----
   - dynamic statistics used: dynamic sampling (level=5)
   - Degree of Parallelism is 4 because of session
   - vector transformation used for this statement


Statistics
----------------------------------------------------------
       1182  recursive calls
         43  db block gets
     225566  consistent gets
     223559  physical reads
      18976  redo size
        860  bytes sent via SQL*Net to client
        604  bytes received via SQL*Net from client
```

```
          2  SQL*Net roundtrips to/from client
        271  sorts (memory)
          0  sorts (disk)
          6  rows processed
```

```
/**************In-Memory Serial Access***************/

SQL> clear scr

SQL> alter session disable parallel query;
SQL> alter session set inmemory_query='ENABLE';

SQL> Select /*+ INMEMORY */ /*IN-MEMORY Serial*/
d_date,sum(l.lo_revenue) "Total Revenue"
From   LINEORDER l, DATE_DIM d
Where  l.lo_orderdate = d.d_datekey
and      D_DAYNUMINMONTH = 25
and    d.d_month = 'December'
group by d_date
order by d_date;

6 rows selected.

Elapsed: 00:00:00.08

Execution Plan
----------------------------------------------------------
Plan hash value: 2937007760

-------------------------------------------------------------------------------
---------------------------------------
| Id  | Operation                                 | Name                      |
Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
---------------------------------------
|   0 | SELECT STATEMENT                          |                           |
7 |   343 |  2724  (17)| 00:00:01 |
|   1 |  TEMP TABLE TRANSFORMATION                |                           |
|       |            |          |
|   2 |   LOAD AS SELECT (CURSOR DURATION MEMORY)| SYS_TEMP_0FD9D6655_290D25 |
|       |            |          |
|   3 |    HASH GROUP BY                          |                           |
7 |   294 |     2  (50)| 00:00:01 |
|   4 |     KEY VECTOR CREATE BUFFERED            | :KV0000                   |
7 |   294 |     1   (0)| 00:00:01 |
|*  5 |      TABLE ACCESS INMEMORY FULL           | DATE_DIM                  |
7 |   266 |     1   (0)| 00:00:01 |
```

```
|   6 |   SORT GROUP BY                          |                            |
7 |   343 |  2722  (17)| 00:00:01 |
|*  7 |    HASH JOIN                             |                            |
7 |   343 |  2721  (17)| 00:00:01 |
|   8 |     VIEW                                 | VW_VT_80F21617             |
7 |   119 |  2719  (17)| 00:00:01 |
|   9 |      VECTOR GROUP BY                     |                            |
7 |   112 |  2719  (17)| 00:00:01 |
|  10 |       HASH GROUP BY                      |                            |
7 |   112 |  2719  (17)| 00:00:01 |
|  11 |        KEY VECTOR USE                    | :KV0000                    |
87245 |  1363K|  2719  (17)| 00:00:01 |
|* 12 |         TABLE ACCESS INMEMORY FULL       | LINEORDER                  |
29M|   343M|  2719  (17)| 00:00:01 |
|  13 |     TABLE ACCESS FULL                    | SYS_TEMP_0FD9D6655_290D25 |
7 |   224 |     2   (0)| 00:00:01 |
---------------------------------------------------------------------------------
---------------------------------------

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

   5 - inmemory("D_DAYNUMINMONTH"=25 AND "D"."D_MONTH"='December')
       filter("D_DAYNUMINMONTH"=25 AND "D"."D_MONTH"='December')
   7 - access("ITEM_5"=INTERNAL_FUNCTION("C0"))
  12 - inmemory(SYS_OP_KEY_VECTOR_FILTER("L"."LO_ORDERDATE",:KV0000))
       filter(SYS_OP_KEY_VECTOR_FILTER("L"."LO_ORDERDATE",:KV0000))

Note
-----
   - vector transformation used for this statement


Statistics
-----------------------------------------------------------
         12  recursive calls
          5  db block gets
         20  consistent gets
          0  physical reads
       1096  redo size
        860  bytes sent via SQL*Net to client
        604  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
          6  rows processed
```

```
/*************In-Memory Parallel Access*************/
/***************************************************/
```

```
SQL> clear scr

SQL> alter session force parallel query parallel 4;

SQL> select /*+ INMEMORY */ /*IN-MEMORY Parallel*/
d_date,sum(l.lo_revenue) "Total Revenue"
From   LINEORDER l, DATE_DIM d
Where  l.lo_orderdate = d.d_datekey
and      D_DAYNUMINMONTH = 25
and    d.d_month = 'December'
group by d_date
order by d_date;

6 rows selected.

Elapsed: 00:00:02.75

Execution Plan
----------------------------------------------------------
Plan hash value: 746328703

-------------------------------------------------------------------------------
----------------------------------------
----------

| Id  | Operation                                | Name      | Rows  | Bytes |
Cost (%CPU)| Time     |    TQ  |IN-OUT| PQ
 Distrib |

-------------------------------------------------------------------------------
----------------------------------------
----------

|   0 | SELECT STATEMENT                         |           |     7 |   399 |
755  (17)| 00:00:01 |        |      |
         |

|   1 |  PX COORDINATOR                          |           |       |       |
|          |        |      |
         |

|   2 |   PX SEND QC (ORDER)                     | :TQ10002  |     7 |   399 |
755  (17)| 00:00:01 |  Q1,02 | P->S | QC
 (ORDER) |

|   3 |    SORT GROUP BY                         |           |     7 |   399 |
755  (17)| 00:00:01 |  Q1,02 | PCWP |
         |

|   4 |     PX RECEIVE                           |           |     7 |   399 |
755  (17)| 00:00:01 |  Q1,02 | PCWP |
         |
```

```
|   5 |      PX SEND RANGE                      | :TQ10001  |     7 |   399 |
755  (17)| 00:00:01 |  Q1,01 | P->P | RA
NGE       |

|   6 |       HASH GROUP BY                     |           |     7 |   399 |
755  (17)| 00:00:01 |  Q1,01 | PCWP |
          |

|*  7 |        HASH JOIN                        |           |     7 |   399 |
754  (17)| 00:00:01 |  Q1,01 | PCWP |
          |

|*  8 |         TABLE ACCESS INMEMORY FULL      | DATE_DIM  |     7 |   266 |
2   (0)| 00:00:01 |  Q1,01 | PCWP |
          |

|   9 |         VIEW                            | VW_GBC_5  |  2407 | 45733 |
752  (17)| 00:00:01 |  Q1,01 | PCWP |
          |

|  10 |          HASH GROUP BY                  |           |  2407 | 28884 |
752  (17)| 00:00:01 |  Q1,01 | PCWP |
          |

|  11 |           PX RECEIVE                    |           |  2407 | 28884 |
752  (17)| 00:00:01 |  Q1,01 | PCWP |
          |

|  12 |            PX SEND HASH                 | :TQ10000  |  2407 | 28884 |
752  (17)| 00:00:01 |  Q1,00 | P->P | HA
SH        |

|  13 |             HASH GROUP BY               |           |  2407 | 28884 |
752  (17)| 00:00:01 |  Q1,00 | PCWP |
          |

|  14 |              PX BLOCK ITERATOR          |           |   29M|   343M|
755  (17)| 00:00:01 |  Q1,00 | PCWC |
          |

|  15 |               TABLE ACCESS INMEMORY FULL| LINEORDER |   29M|   343M|
755  (17)| 00:00:01 |  Q1,00 | PCWP |
          |

---------------------------------------------------------------------------
-----------------------------------------
----------


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

   7 - access("ITEM_1"="D"."D_DATEKEY")
   8 - inmemory("D_DAYNUMINMONTH"=25 AND "D"."D_MONTH"='December')
```

```
       filter("D_DAYNUMINMONTH"=25 AND "D"."D_MONTH"='December')

Note
-----
   - dynamic statistics used: dynamic sampling (level=5)
   - Degree of Parallelism is 4 because of session


Statistics
----------------------------------------------------------
        887  recursive calls
          2  db block gets
       1253  consistent gets
         10  physical reads
        804  redo size
        860  bytes sent via SQL*Net to client
        606  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
         96  sorts (memory)
          0  sorts (disk)
          6  rows processed
```

Queries complejas

Three table scan

Query 1

```
SQL> conn ssb/ssb@SOE


/*********Group Aggregate with three table join******/
/************************************************/
/***************Parallel Disk Access*****************/

SQL> clear scr

--flush the buffer_cache

SQL> alter system flush buffer_cache;
SQL> set autotrace traceonly explain statistics
SQL> alter session force parallel query parallel 4;
SQL> alter session set inmemory_query='DISABLE';
SQL> set timing on

SQL> select /*+ NO_INMEMORY */ /* DISK ACCESS */
p.p_name, sum(l.lo_revenue)
From     LINEORDER l, DATE_DIM d, PART p
Where    l.lo_orderdate = d.d_datekey
And      l.lo_partkey   = p.p_partkey
And      p.p_name       = 'misty gainsboro'
```

```
And      d.d_year = 1992
And      d.d_month = 'December'
Group by p.p_name;

Elapsed: 00:00:37.16

Execution Plan
----------------------------------------------------------
Plan hash value: 2989091434

-------------------------------------------------------------------------------------
----------------------------------------
--------------------------

| Id  | Operation                                | Name                      |
Rows  | Bytes | Cost (%CPU)| Time     |
  TQ  |IN-OUT| PQ Distrib |

-------------------------------------------------------------------------------------
----------------------------------------
--------------------------

|   0 | SELECT STATEMENT                         |                           |
50 |  3400 | 17430   (2)| 00:00:01 |
      |      |            |

|   1 |  TEMP TABLE TRANSFORMATION               |                           |
|       |            |          |
      |      |            |

|   2 |   LOAD AS SELECT (CURSOR DURATION MEMORY)| SYS_TEMP_0FD9D6666_290D25 |
|       |            |          |
      |      |            |

|   3 |    PX COORDINATOR                        |                           |
|       |            |          |
      |      |            |

|   4 |     PX SEND QC (RANDOM)                  | :TQ10001                  |
1 |    24 |     5  (20)| 00:00:01 |
Q1,01 | P->S | QC (RAND)  |

|   5 |      HASH GROUP BY                       |                           |
1 |    24 |     5  (20)| 00:00:01 |
Q1,01 | PCWP |            |

|   6 |       PX RECEIVE                         |                           |
1 |    24 |     4   (0)| 00:00:01 |
Q1,01 | PCWP |            |

|   7 |        PX SEND HASH                      | :TQ10000                  |
1 |    24 |     4   (0)| 00:00:01 |
Q1,00 | P->P | HASH       |
```

```
|   8 |         KEY VECTOR CREATE BUFFERED       | :KV0000                  |
1 |    24 |     4   (0)| 00:00:01 |
Q1,00 | PCWC |            |

|   9 |          PX BLOCK ITERATOR               |                          |
31 |   620 |     4   (0)| 00:00:01 |
Q1,00 | PCWC |            |

|* 10 |           TABLE ACCESS FULL              | DATE_DIM                 |
31 |   620 |     4   (0)| 00:00:01 |
Q1,00 | PCWP |            |

|  11 |   LOAD AS SELECT (CURSOR DURATION MEMORY)| SYS_TEMP_0FD9D6667_290D25 |
|       |            |          |
      |      |            |

|  12 |    PX COORDINATOR                        |                          |
|       |            |          |
      |      |            |

|  13 |     PX SEND QC (RANDOM)                  | :TQ20001                 |
70 |  1610 |   397   (1)| 00:00:01 |
Q2,01 | P->S | QC (RAND)  |

|  14 |      HASH GROUP BY                       |                          |
70 |  1610 |   397   (1)| 00:00:01 |
Q2,01 | PCWP |            |

|  15 |       PX RECEIVE                         |                          |
70 |  1610 |   396   (1)| 00:00:01 |
Q2,01 | PCWP |            |

|  16 |        PX SEND HASH                      | :TQ20000                 |
70 |  1610 |   396   (1)| 00:00:01 |
Q2,00 | P->P | HASH       |

|  17 |         KEY VECTOR CREATE BUFFERED       | :KV0001                  |
70 |  1610 |   396   (1)| 00:00:01 |
Q2,00 | PCWC |            |

|  18 |          PX BLOCK ITERATOR               |                          |
71 |  1349 |   396   (1)| 00:00:01 |
Q2,00 | PCWC |            |

|* 19 |           TABLE ACCESS FULL              | PART                     |
71 |  1349 |   396   (1)| 00:00:01 |
Q2,00 | PCWP |            |

|  20 |   PX COORDINATOR                         |                          |
|       |            |          |
      |      |            |

|  21 |    PX SEND QC (RANDOM)                   | :TQ30003                 |
50 |  3400 | 17027   (2)| 00:00:01 |
```

```
Q3,03 | P->S | QC (RAND)  |

|* 22 |     HASH JOIN BUFFERED                  |                                |
50 |  3400 | 17027   (2)| 00:00:01 |
Q3,03 | PCWP |            |

|  23 |      PX RECEIVE                         |                                |
50 |  2250 | 17025   (2)| 00:00:01 |
Q3,03 | PCWP |            |

|  24 |       PX SEND HYBRID HASH               | :TQ30001                       |
50 |  2250 | 17025   (2)| 00:00:01 |
Q3,01 | P->P | HYBRID HASH|

|  25 |        STATISTICS COLLECTOR             |                                |
|       |            |          |
Q3,01 | PCWC |            |

|* 26 |         HASH JOIN                       |                                |
50 |  2250 | 17025   (2)| 00:00:01 |
Q3,01 | PCWP |            |

|  27 |          TABLE ACCESS FULL              | SYS_TEMP_0FD9D6666_290D25 |
1 |    24 |     2   (0)| 00:00:01 |
Q3,01 | PCWP |            |

|  28 |          VIEW                           | VW_VT_80F21617                 |
50 |  1050 | 17023   (2)| 00:00:01 |
Q3,01 | PCWP |            |

|  29 |           HASH GROUP BY                 |                                |
50 |  1250 | 17023   (2)| 00:00:01 |
Q3,01 | PCWP |            |

|  30 |            PX RECEIVE                   |                                |
50 |  1250 | 17023   (2)| 00:00:01 |
Q3,01 | PCWP |            |

|  31 |             PX SEND HASH                | :TQ30000                       |
50 |  1250 | 17023   (2)| 00:00:01 |
Q3,00 | P->P | HASH       |

|  32 |              VECTOR GROUP BY            |                                |
50 |  1250 | 17023   (2)| 00:00:01 |
Q3,00 | PCWP |            |

|  33 |               HASH GROUP BY             |                                |
50 |  1250 | 17023   (2)| 00:00:01 |
Q3,00 | PCWP |            |

|  34 |                KEY VECTOR USE           | :KV0000                        |
70 |  1750 | 17023   (2)| 00:00:01 |
Q3,00 | PCWC |            |
```

```
|  35 |                 KEY VECTOR USE           | :KV0001                    |
5417 |   111K| 17023   (2)| 00:00:01 |
Q3,00 | PCWC |            |

|  36 |                  PX BLOCK ITERATOR       |                            |
29M|   486M| 17023   (2)| 00:00:01 |
Q3,00 | PCWC |            |

|* 37 |                  TABLE ACCESS FULL       | LINEORDER                  |
29M|   486M| 17023   (2)| 00:00:01 |
Q3,00 | PCWP |            |

|  38 |      PX RECEIVE                          |                            |
70 |  1610 |     2   (0)| 00:00:01 |
Q3,03 | PCWP |            |

|  39 |       PX SEND HYBRID HASH                | :TQ30002                   |
70 |  1610 |     2   (0)| 00:00:01 |
Q3,02 | P->P | HYBRID HASH|

|  40 |        PX BLOCK ITERATOR                 |                            |
70 |  1610 |     2   (0)| 00:00:01 |
Q3,02 | PCWC |            |

|  41 |         TABLE ACCESS FULL                | SYS_TEMP_0FD9D6667_290D25 |
70 |  1610 |     2   (0)| 00:00:01 |
Q3,02 | PCWP |            |

--------------------------------------------------------------------------------
----------------------------------------
---------------------------


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

  10 - filter("D"."D_YEAR"=1992 AND "D"."D_MONTH"='December')
  19 - filter("P"."P_NAME"='misty gainsboro')
  22 - access("ITEM_8"=INTERNAL_FUNCTION("C0"))
  26 - access("ITEM_7"=INTERNAL_FUNCTION("C0"))
  37 - filter(SYS_OP_KEY_VECTOR_FILTER("L"."LO_PARTKEY",:KV0001) AND
SYS_OP_KEY_VECTOR_FILTER("L"."LO_ORDERDATE",:KV0000
))


Note
-----
   - dynamic statistics used: dynamic sampling (level=5)
   - Degree of Parallelism is 4 because of session
   - vector transformation used for this statement


Statistics
-----------------------------------------------------------
```

```
        300  recursive calls
         37  db block gets
     229041  consistent gets
     228675  physical reads
     284504  redo size
        655  bytes sent via SQL*Net to client
        677  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
         27  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

```
/**************In-Memory Serial Access***************/
/****************************************************/

SQL> clear scr

SQL> alter session disable parallel query;
SQL> alter session set inmemory_query='DISABLE';


SQL> Select /*+ INMEMORY */ /*IN-MEMORY Serial*/
p.p_name, sum(l.lo_revenue)
From     LINEORDER l, DATE_DIM d, PART p
Where    l.lo_orderdate = d.d_datekey
And      l.lo_partkey   = p.p_partkey
And      p.p_name       = 'misty gainsboro'
And      d.d_year = 1992
And      d.d_month = 'December'
Group by p.p_name;


Elapsed: 00:00:00.52

Execution Plan
----------------------------------------------------------
Plan hash value: 3307065880

-------------------------------------------------------------------------------------
---------------------------------------
| Id  | Operation                                 | Name                        |
Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
---------------------------------------
|   0 | SELECT STATEMENT                          |                             |
50 |  3400 |  2952  (22)| 00:00:01 |
|   1 |  TEMP TABLE TRANSFORMATION                |                             |
|       |            |          |
|   2 |   LOAD AS SELECT (CURSOR DURATION MEMORY)| SYS_TEMP_0FD9D666C_290D25 |
|       |            |          |
```

```
|   3 |    HASH GROUP BY                          |                                |
1 |    24 |     2  (50)| 00:00:01 |
|   4 |     KEY VECTOR CREATE BUFFERED            | :KV0000                        |
1 |    24 |     1   (0)| 00:00:01 |
|*  5 |      TABLE ACCESS INMEMORY FULL           | DATE_DIM                       |
31 |   620 |     1   (0)| 00:00:01 |
|   6 |   LOAD AS SELECT (CURSOR DURATION MEMORY)| SYS_TEMP_0FD9D666D_290D25 |
|       |             |          |
|   7 |    HASH GROUP BY                          |                                |
70 |  1610 |    67  (21)| 00:00:01 |
|   8 |     KEY VECTOR CREATE BUFFERED            | :KV0001                        |
70 |  1610 |    66  (20)| 00:00:01 |
|*  9 |      TABLE ACCESS INMEMORY FULL           | PART                           |
71 |  1349 |    66  (20)| 00:00:01 |
|  10 |   HASH GROUP BY                           |                                |
50 |  3400 |  2883  (22)| 00:00:01 |
|* 11 |    HASH JOIN                              |                                |
50 |  3400 |  2882  (22)| 00:00:01 |
|  12 |     MERGE JOIN CARTESIAN                  |                                |
70 |  3290 |     4   (0)| 00:00:01 |
|  13 |      TABLE ACCESS FULL                    | SYS_TEMP_0FD9D666C_290D25 |
1 |    24 |     2   (0)| 00:00:01 |
|  14 |      BUFFER SORT                          |                                |
70 |  1610 |     2   (0)| 00:00:01 |
|  15 |       TABLE ACCESS FULL                   | SYS_TEMP_0FD9D666D_290D25 |
70 |  1610 |     2   (0)| 00:00:01 |
|  16 |     VIEW                                  | VW_VT_80F21617                 |
50 |  1050 |  2878  (22)| 00:00:01 |
|  17 |      VECTOR GROUP BY                      |                                |
50 |  1250 |  2878  (22)| 00:00:01 |
|  18 |       HASH GROUP BY                       |                                |
50 |  1250 |  2878  (22)| 00:00:01 |
|  19 |        KEY VECTOR USE                     | :KV0000                        |
70 |  1750 |  2878  (22)| 00:00:01 |
|  20 |          KEY VECTOR USE                   | :KV0001                        |
5417 |   111K|  2878  (22)| 00:00:01 |
|* 21 |           TABLE ACCESS INMEMORY FULL      | LINEORDER                      |
29M|   486M|  2878  (22)| 00:00:01 |
-------------------------------------------------------------------------------
---------------------------------------

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

   5 - inmemory("D"."D_YEAR"=1992 AND "D"."D_MONTH"='December')
       filter("D"."D_YEAR"=1992 AND "D"."D_MONTH"='December')
   9 - inmemory("P"."P_NAME"='misty gainsboro')
       filter("P"."P_NAME"='misty gainsboro')
  11 - access("ITEM_7"=INTERNAL_FUNCTION("C0") AND
"ITEM_8"=INTERNAL_FUNCTION("C0"))
  21 - inmemory(SYS_OP_KEY_VECTOR_FILTER("L"."LO_PARTKEY",:KV0001) AND
              SYS_OP_KEY_VECTOR_FILTER("L"."LO_ORDERDATE",:KV0000))
       filter(SYS_OP_KEY_VECTOR_FILTER("L"."LO_PARTKEY",:KV0001) AND
              SYS_OP_KEY_VECTOR_FILTER("L"."LO_ORDERDATE",:KV0000))
```

```
Note
-----
   - vector transformation used for this statement


Statistics
----------------------------------------------------------
         21  recursive calls
          0  db block gets
         38  consistent gets
          0  physical reads
          0  redo size
        655  bytes sent via SQL*Net to client
        677  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

```
/**************In-Memory Parallel Access*************/
/*****************************************************/

SQL> clear scr

SQL> alter session force parallel query parallel 4;

SQL> select /*+ INMEMORY */ /*IN-MEMORY Parallel*/
p.p_name, sum(l.lo_revenue)
From     LINEORDER l, DATE_DIM d, PART p
Where    l.lo_orderdate = d.d_datekey
And      l.lo_partkey   = p.p_partkey
And      p.p_name       = 'misty gainsboro'
And      d.d_year = 1992
And      d.d_month = 'December'
Group by p.p_name;

Elapsed: 00:00:00.44

Execution Plan
----------------------------------------------------------
Plan hash value: 2989091434

-------------------------------------------------------------------------------------
----------------------------------------
-----------------------------

| Id  | Operation                                    | Name
| Rows  | Bytes | Cost (%CPU)| Time
```

```
|    TQ  |IN-OUT| PQ Distrib |

-------------------------------------------------------------------------------
----------------------------------------
------------------------------

|   0 | SELECT STATEMENT                              |
|    50 |  3400 |   827  (22)| 00:00:01
|       |       |            |

|   1 |  TEMP TABLE TRANSFORMATION                    |
|       |       |            |
|       |       |            |

|   2 |   LOAD AS SELECT (CURSOR DURATION MEMORY)     | SYS_TEMP_0FD9D6672_290D25
|       |       |            |
|       |       |            |

|   3 |    PX COORDINATOR                             |
|       |       |            |
|       |       |            |

|   4 |     PX SEND QC (RANDOM)                       | :TQ10001
|     1 |    24 |     3  (34)| 00:00:01
|  Q1,01 | P->S | QC (RAND)  |

|   5 |      HASH GROUP BY                            |
|     1 |    24 |     3  (34)| 00:00:01
|  Q1,01 | PCWP |            |

|   6 |       PX RECEIVE                              |
|     1 |    24 |     2   (0)| 00:00:01
|  Q1,01 | PCWP |            |

|   7 |        PX SEND HASH                           | :TQ10000
|     1 |    24 |     2   (0)| 00:00:01
|  Q1,00 | P->P | HASH       |

|   8 |         KEY VECTOR CREATE BUFFERED            | :KV0000
|     1 |    24 |     2   (0)| 00:00:01
|  Q1,00 | PCWC |            |

|   9 |          PX BLOCK ITERATOR                    |
|    31 |   620 |     2   (0)| 00:00:01
|  Q1,00 | PCWC |            |

|* 10 |           TABLE ACCESS INMEMORY FULL          | DATE_DIM
|    31 |   620 |     2   (0)| 00:00:01
|  Q1,00 | PCWP |            |

|  11 |   LOAD AS SELECT (CURSOR DURATION MEMORY)     | SYS_TEMP_0FD9D6673_290D25
|       |       |            |
|       |       |            |
```

```
|  12 |    PX COORDINATOR                          |
|       |       |            |
|        |      |            |

|  13 |     PX SEND QC (RANDOM)                    | :TQ20001
|    70 |  1610 |    19  (22)| 00:00:01
|  Q2,01 | P->S | QC (RAND)  |

|  14 |      HASH GROUP BY                         |
|    70 |  1610 |    19  (22)| 00:00:01
|  Q2,01 | PCWP |            |

|  15 |       PX RECEIVE                           |
|    70 |  1610 |    18  (17)| 00:00:01
|  Q2,01 | PCWP |            |

|  16 |        PX SEND HASH                        | :TQ20000
|    70 |  1610 |    18  (17)| 00:00:01
|  Q2,00 | P->P | HASH       |

|  17 |         KEY VECTOR CREATE BUFFERED         | :KV0001
|    70 |  1610 |    18  (17)| 00:00:01
|  Q2,00 | PCWC |            |

|  18 |          PX BLOCK ITERATOR                 |
|    71 |  1349 |    18  (17)| 00:00:01
|  Q2,00 | PCWC |            |

|* 19 |           TABLE ACCESS INMEMORY FULL       | PART
|    71 |  1349 |    18  (17)| 00:00:01
|  Q2,00 | PCWP |            |

|  20 |   PX COORDINATOR                           |
|       |       |            |
|        |      |            |

|  21 |    PX SEND QC (RANDOM)                     | :TQ30003
|    50 |  3400 |   803  (22)| 00:00:01
|  Q3,03 | P->S | QC (RAND)  |

|* 22 |     HASH JOIN BUFFERED                     |
|    50 |  3400 |   803  (22)| 00:00:01
|  Q3,03 | PCWP |            |

|  23 |      PX RECEIVE                            |
|    50 |  2250 |   801  (22)| 00:00:01
|  Q3,03 | PCWP |            |

|  24 |       PX SEND HYBRID HASH                  | :TQ30001
|    50 |  2250 |   801  (22)| 00:00:01
|  Q3,01 | P->P | HYBRID HASH|

|  25 |        STATISTICS COLLECTOR                |
|       |       |            |
```

```
|  Q3,01 | PCWC |            |                                    |
|* 26 |         HASH JOIN                          |
|    50 |  2250 |   801  (22)| 00:00:01
|  Q3,01 | PCWP |            |                                    |
|  27 |          TABLE ACCESS FULL                 | SYS_TEMP_0FD9D6672_290D25
|     1 |    24 |     2   (0)| 00:00:01
|  Q3,01 | PCWP |            |                                    |
|  28 |          VIEW                              | VW_VT_80F21617
|    50 |  1050 |   799  (22)| 00:00:01
|  Q3,01 | PCWP |            |                                    |
|  29 |           HASH GROUP BY                    |
|    50 |  1250 |   799  (22)| 00:00:01
|  Q3,01 | PCWP |            |                                    |
|  30 |            PX RECEIVE                      |
|    50 |  1250 |   799  (22)| 00:00:01
|  Q3,01 | PCWP |            |                                    |
|  31 |             PX SEND HASH                   | :TQ30000
|    50 |  1250 |   799  (22)| 00:00:01
|  Q3,00 | P->P | HASH       |                                    |
|  32 |              VECTOR GROUP BY               |
|    50 |  1250 |   799  (22)| 00:00:01
|  Q3,00 | PCWP |            |                                    |
|  33 |               HASH GROUP BY                |
|    50 |  1250 |   799  (22)| 00:00:01
|  Q3,00 | PCWP |            |                                    |
|  34 |                KEY VECTOR USE              | :KV0000
|    70 |  1750 |   799  (22)| 00:00:01
|  Q3,00 | PCWC |            |                                    |
|  35 |                 KEY VECTOR USE             | :KV0001
|  5417 |   111K|   799  (22)| 00:00:01
|  Q3,00 | PCWC |            |                                    |
|  36 |                  PX BLOCK ITERATOR         |
|    29M|   486M|   799  (22)| 00:00:01
|  Q3,00 | PCWC |            |                                    |
|* 37 |                   TABLE ACCESS INMEMORY FULL| LINEORDER
|    29M|   486M|   799  (22)| 00:00:01
|  Q3,00 | PCWP |            |                                    |
|  38 |      PX RECEIVE                            |
|    70 |  1610 |     2   (0)| 00:00:01
|  Q3,03 | PCWP |            |                                    |
```

```
|  39 |       PX SEND HYBRID HASH                  | :TQ30002
|    70 |  1610 |     2   (0)| 00:00:01
|  Q3,02 | P->P | HYBRID HASH|

|  40 |        PX BLOCK ITERATOR                   |
|    70 |  1610 |     2   (0)| 00:00:01
|  Q3,02 | PCWC |            |

|  41 |         TABLE ACCESS FULL                  | SYS_TEMP_0FD9D6673_290D25
|    70 |  1610 |     2   (0)| 00:00:01
|  Q3,02 | PCWP |            |

--------------------------------------------------------------------------------
----------------------------------------
------------------------------


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

  10 - inmemory("D"."D_YEAR"=1992 AND "D"."D_MONTH"='December')
       filter("D"."D_YEAR"=1992 AND "D"."D_MONTH"='December')
  19 - inmemory("P"."P_NAME"='misty gainsboro')
       filter("P"."P_NAME"='misty gainsboro')
  22 - access("ITEM_8"=INTERNAL_FUNCTION("C0"))
  26 - access("ITEM_7"=INTERNAL_FUNCTION("C0"))
  37 - inmemory(SYS_OP_KEY_VECTOR_FILTER("L"."LO_PARTKEY",:KV0001) AND
SYS_OP_KEY_VECTOR_FILTER("L"."LO_ORDERDATE",:KV00
00))

       filter(SYS_OP_KEY_VECTOR_FILTER("L"."LO_PARTKEY",:KV0001) AND
SYS_OP_KEY_VECTOR_FILTER("L"."LO_ORDERDATE",:KV0000
))


Note
-----
   - dynamic statistics used: dynamic sampling (level=5)
   - Degree of Parallelism is 4 because of session
   - vector transformation used for this statement


Statistics
----------------------------------------------------------
        108  recursive calls
          0  db block gets
        162  consistent gets
          0  physical reads
          0  redo size
        655  bytes sent via SQL*Net to client
        679  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          8  sorts (memory)
          0  sorts (disk)
```

```
1  rows processed
```

ACO - Oracle Advanced Compression (45 min)

Oracle Advanced Compression (ACO) permite aumentar el rendimiento al mismo tiempo que se reduce el coste del almacenamiento. Permite reducir significativamente el espacio total de almacenamiento de la base de datos al permitir la compresión para todo tipo de datos relacionales (tabla), no estructurados (archivo), índice, red, Data Guard, backup con RMAN.

Mediante *Advanced Row Compression* se puede reducir el consumo de almacenamiento en un factor de 2x a 4x. De igual forma, los accesos por parte del optimizador, se incrementan en 2,5x en procesos de table-scan comparados con los datos no comprimidos.

Los datos se leen comprimidos (datos e índices) directamente, en memoria, sin descomprimir los bloques.

Advanced Row Compression Implementation

Se utilizará el usuario SSB donde se crearán dos tablas

- TEST_TAB_NO (sin compresión)
- TEST_TAB_COMP (con compression)

Ejecutaremos inserciones en ambas tablas para comprobar el factor de compresión obtenido mediante el uso del API (DBMS_COMPRESSION) y comprobaremos los bloques de ambas tablas.

Para ello utilizaremos la PDB (SOE) y nos conectaremos con usuario SSB.

Crear dos tablas: TEST_TAB_NO y TEST_TAB_COMP y se procederá a insertar registros sobre la tabla TEST_TAB_NO que no está comprimida.

```
[oracle@single19c ~]$ sqlplus ssb/ssb@soe

CREATE TABLE test_tab_no (
  id            NUMBER(10)    NOT NULL,
  description   VARCHAR2(100) NOT NULL,
  created_date  DATE          NOT NULL,
  created_by    VARCHAR2(50)  NOT NULL,
  updated_date  DATE,
  updated_by    VARCHAR2(50)
);

Table created.

SQL> SET SERVEROUTPUT ON

SQL> DECLARE
  v_date        test_tab_no.created_date%TYPE := SYSDATE;
  v_user        test_tab_no.created_by%TYPE   := USER;
  l_start_time  NUMBER;
  l_start_cpu   NUMBER;
BEGIN

  l_start_time := DBMS_UTILITY.get_time;
  l_start_cpu  := DBMS_UTILITY.get_cpu_time;

  INSERT /*+ APPEND */ INTO test_tab_no (id, description, created_date,
created_by)
  SELECT level,
         'A very repetitive, and therefore very compressible column value',
         v_date,
         v_user
  FROM   dual
  CONNECT BY level <= 1000000;
  COMMIT;

  DBMS_OUTPUT.put_line('CPU Time (hsecs)    : ' || (DBMS_UTILITY.get_cpu_time -
l_start_cpu));
  DBMS_OUTPUT.put_line('Elapsed Time (hsecs): ' || (DBMS_UTILITY.get_time -
l_start_time));
END;
/


CPU Time (hsecs)    : 535
Elapsed Time (hsecs): 702

PL/SQL procedure successfully completed.
```

Utilizaremos el API DBMS_COMPRESSION con el tipo de compresión avanzada (DBMS_COMPRESSION.comp_advanced) y veremos el factor de compresión que se podría lograr en esta tabla no comprimida.

```
SET SERVEROUTPUT ON
DECLARE
  l_blkcnt_cmp    PLS_INTEGER;
  l_blkcnt_uncmp  PLS_INTEGER;
  l_row_cmp       PLS_INTEGER;
  l_row_uncmp     PLS_INTEGER;
  l_cmp_ratio     NUMBER;
  l_comptype_str  VARCHAR2(32767);
BEGIN
  DBMS_COMPRESSION.get_compression_ratio (
    scratchtbsname  => 'SYSAUX',
    ownname         => 'SSB',
    objname         => 'TEST_TAB_NO',
    subobjname      => NULL,
    comptype        => DBMS_COMPRESSION.comp_advanced,
    blkcnt_cmp      => l_blkcnt_cmp,
    blkcnt_uncmp    => l_blkcnt_uncmp,
    row_cmp         => l_row_cmp,
    row_uncmp       => l_row_uncmp,
    cmp_ratio       => l_cmp_ratio,
    comptype_str    => l_comptype_str,
    subset_numrows  => DBMS_COMPRESSION.comp_ratio_allrows,
    objtype         => DBMS_COMPRESSION.objtype_table
  );
  DBMS_OUTPUT.put_line('Number of blocks used (compressed)       : ' ||
l_blkcnt_cmp);
  DBMS_OUTPUT.put_line('Number of blocks used (uncompressed)     : ' ||
l_blkcnt_uncmp);
```

```
  DBMS_OUTPUT.put_line('Number of rows in a block (compressed)   : ' ||
l_row_cmp);
  DBMS_OUTPUT.put_line('Number of rows in a block (uncompressed) : ' ||
l_row_uncmp);
 DBMS_OUTPUT.put_line('Compression ratio                        : ' ||
l_cmp_ratio);
  DBMS_OUTPUT.put_line('Compression type                         : ' ||
l_comptype_str);
END;
/
Number of blocks used (compressed)      : 216
Number of blocks used (uncompressed)     : 1664
Number of rows in a block (compressed)   : 647
Number of rows in a block (uncompressed) : 84
Compression ratio                        : 7.7
Compression type                         : "Compress Advanced"

PL/SQL procedure successfully completed.
```

Crear la tabla TEST_TAB_COMP para insertar registros sobre la tabla comprimida. Utilizaremos el atributo COMPRESS FOR ALL OPERATIONS en la creación de la tabla.

```
SQL> CREATE TABLE test_tab_comp (
  id            NUMBER(10)    NOT NULL,
  description   VARCHAR2(100) NOT NULL,
  created_date  DATE          NOT NULL,
  created_by    VARCHAR2(50)  NOT NULL,
  updated_date  DATE,
  updated_by    VARCHAR2(50)
)
COMPRESS FOR ALL OPERATIONS;
Table created.

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
```

```
  v_date        test_tab_comp.created_date%TYPE := SYSDATE;
  v_user        test_tab_comp.created_by%TYPE   := USER;
  l_start_time  NUMBER;
  l_start_cpu   NUMBER;
BEGIN

  l_start_time := DBMS_UTILITY.get_time;
  l_start_cpu  := DBMS_UTILITY.get_cpu_time;

  INSERT /*+ APPEND */ INTO test_tab_comp (id, description, created_date,
created_by)
  SELECT level,
         'A very repetitive, and therefore very compressible column value',
         v_date,
         v_user
  FROM   dual
  CONNECT BY level <= 1000000;
  COMMIT;

  DBMS_OUTPUT.put_line('CPU Time (hsecs)    : ' || (DBMS_UTILITY.get_cpu_time -
l_start_cpu));
  DBMS_OUTPUT.put_line('Elapsed Time (hsecs): ' || (DBMS_UTILITY.get_time -
l_start_time));
END;
/
CPU Time (hsecs)    : 349
Elapsed Time (hsecs): 370


PL/SQL procedure successfully completed.
```

Comprobar la ocupación de bloques de ambas tablas para ver la mejora con la compresión.

```
col table_name format a20
```

```
SQL> SELECT table_name,compression, compress_for
FROM   user_tables
WHERE  table_name like 'TEST_TAB%';

TABLE_NAME           COMPRESS COMPRESS_FOR
-------------------- -------- ------------------------------
TEST_TAB_COMP        ENABLED  ADVANCED
TEST_TAB_NO          DISABLED

SQL> SELECT table_name,
       compression,
       num_rows,
       blocks,
       empty_blocks
FROM   user_tables
WHERE  table_name like 'TEST_TAB%'
ORDER BY 1;

TABLE_NAME           COMPRESS   NUM_ROWS     BLOCKS EMPTY_BLOCKS
-------------------- -------- ---------- ---------- ------------
TEST_TAB_COMP        ENABLED     1000000       1578            0
TEST_TAB_NO          DISABLED    1000000      12055            0
```

Advanced Row Compression Tabla/Partición

De igual forma, se podría comprimir particiones de tablas particionadas para implementar si se quisiera un entorno ILM (*Information Lifecycle Management*) donde podríamos definir políticas sobre las particiones y así poder almacenar información que va a ser accedida frecuentemente o cada cierto tiempo.

Veamos un ejemplo de compresión a nivel de partición. Se creará una tabla con dos particiones y procederemos ejecutar varios procesos DML (Data Manipulation Language) para insertar información sobre las particiones creadas.

La tabla será creada para comprimir sólo datos sobre una partición definida para compresión.

```
SQL> CREATE TABLE test_tab_particion (
  id            NUMBER(10)    NOT NULL,
  description   VARCHAR2(100) NOT NULL,
  created_date  DATE          NOT NULL,
  created_by    VARCHAR2(50)  NOT NULL,
  updated_date  DATE,
  updated_by    VARCHAR2(50)
)
NOCOMPRESS
PARTITION BY RANGE (created_date) (
  PARTITION test_tab_q1 VALUES LESS THAN (TO_DATE('01/04/2003', 'DD/MM/YYYY'))
COMPRESS FOR ALL OPERATIONS,
  PARTITION test_tab_q2 VALUES LESS THAN (MAXVALUE)
);

Table created

SQL> set linesize 1000;
SQL> COLUMN partition_name FORMAT A30

SQL> SELECT partition_name, compression, compress_for
FROM   user_tab_partitions
WHERE  table_name = 'TEST_TAB_PARTICION'
ORDER BY 1;

PARTITION_NAME                 COMPRESS COMPRESS_FOR
------------------------------ -------- ------------------------------
TEST_TAB_Q1                    ENABLED  ADVANCED
TEST_TAB_Q2                    DISABLED
```

Ejecutaremos dos procesos DML para insertar datos de varias fechas repartidos en las dos particiones creadas, una con compresión y otra sin ella, y simular una implementación de tablas tipo ILM. En primer lugar sólo se inserta en la partición sin compresión.

```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  v_date        test_tab_particion.created_date%TYPE := SYSDATE;
  v_user        test_tab_particion.created_by%TYPE   := USER;
  l_start_time  NUMBER;
  l_start_cpu   NUMBER;
BEGIN

  l_start_time := DBMS_UTILITY.get_time;
  l_start_cpu  := DBMS_UTILITY.get_cpu_time;

  INSERT /*+ APPEND */ INTO test_tab_particion (id, description, created_date,
created_by)
  SELECT level,
         'A very repetitive, and therefore very compressible column value',
         v_date,
         v_user
  FROM   dual
  CONNECT BY level <= 1000000;
  COMMIT;

  DBMS_OUTPUT.put_line('CPU Time (hsecs)    : ' || (DBMS_UTILITY.get_cpu_time -
l_start_cpu));
  DBMS_OUTPUT.put_line('Elapsed Time (hsecs): ' || (DBMS_UTILITY.get_time -
l_start_time));
END;
/
CPU Time (hsecs)    : 359
Elapsed Time (hsecs): 382
```

```
PL/SQL procedure successfully completed.
```

Analizar la tabla (Partición) mediante DBMS_STATS.gather_table_stats. Posteriormente veremos la ocupación de los bloques y datos insertados sobre la tabla particionada.

```
SQL> EXEC DBMS_STATS.gather_table_stats(USER, 'TEST_TAB_PARTICION');
PL/SQL procedure successfully completed.


SQL> SELECT table_name,
       partition_name,
       compression,
       num_rows,
       blocks,
       empty_blocks
FROM   user_tab_partitions
WHERE  table_name = 'TEST_TAB_PARTICION'
ORDER BY 1;


TABLE_NAME           PARTITION_NAME                 COMPRESS   NUM_ROWS
BLOCKS EMPTY_BLOCKS
-------------------- ------------------------------ -------- ---------- -------
--- ------------
TEST_TAB_PARTICION   TEST_TAB_Q1                    ENABLED           0
0            0
TEST_TAB_PARTICION   TEST_TAB_Q2                    DISABLED    1000000
12038            0
```

Repetimos el proceso de inserción en la tabla particionada por diferentes fechas para crear registros en la partición definida con el atributo COMPRESS FOR ALL OPERATIONS.

```
SQL> SET SERVEROUTPUT ON

SQL> DECLARE
  v_date        test_tab_particion.created_date%TYPE := TO_DATE('31/03/2003',
'DD/MM/YYYY');
  v_user        test_tab_particion.created_by%TYPE   := USER;
  l_start_time  NUMBER;
```

```
  l_start_cpu   NUMBER;
BEGIN

  l_start_time := DBMS_UTILITY.get_time;
  l_start_cpu  := DBMS_UTILITY.get_cpu_time;

  INSERT /*+ APPEND */ INTO test_tab_particion (id, description, created_date,
created_by)
  SELECT level,
         'A very repetitive, and therefore very compressible column value',
         v_date,
         v_user
  FROM   dual
  CONNECT BY level <= 1000000;
  COMMIT;

  DBMS_OUTPUT.put_line('CPU Time (hsecs)    : ' || (DBMS_UTILITY.get_cpu_time -
l_start_cpu));
  DBMS_OUTPUT.put_line('Elapsed Time (hsecs): ' || (DBMS_UTILITY.get_time -
l_start_time));
END;
/

CPU Time (hsecs)    : 329
Elapsed Time (hsecs): 391

PL/SQL procedure successfully completed.
```

Analizar la tabla (Partición) mediante DBMS_STATS.gather_table_stats. Posteriormente veremos la ocupación de los bloques y datos insertados sobre la tabla particionada. Como vemos, en la partición comprimida hay menos bloques con respecto a la partición sin comprimir

```
SQL> EXEC DBMS_STATS.gather_table_stats(USER, 'TEST_TAB_PARTICION');

PL/SQL procedure successfully completed.

SQL> COLUMN table_name FORMAT A20
SQL> COLUMN partition_name FORMAT A20

SQL> SELECT table_name,
       partition_name,
       compression,
       num_rows,
       blocks,
       empty_blocks
FROM   user_tab_partitions
WHERE  table_name = 'TEST_TAB_PARTICION'
ORDER BY 1;
```

```
TABLE_NAME           PARTITION_NAME       COMPRESS   NUM_ROWS     BLOCKS
EMPTY_BLOCKS
-------------------- -------------------- -------- ---------- ---------- ------
------
TEST_TAB_PARTICION   TEST_TAB_Q1          ENABLED     1000000       1578
0
TEST_TAB_PARTICION   TEST_TAB_Q2          DISABLED    1000000      12038
0
```

Utilización de API PL/SQL DBMS_COMRESSION

A continuación, veremos los distintos usos que podemos lograr con el API PL/SQL DBMS_COMPRESSION utilizando distintos atributos de tipo de compresión. Para una lista completa de todos los valores soportados consultar el manual de esta API.

Para ello, crearemos una tabla particionada e insertaremos y chequearemos los distintos factores de compresión.

Posteriormente, se creará un índice local sobre la tabla TAB_DBMS_COMPRESS.

El procedimiento GET_COMPRESSION_RATIO estima el impacto de diferentes niveles de compresión en una tabla o partición especificada.

```
SQL> CREATE TABLE tab_dbms_compress (
  id                NUMBER,
  code              VARCHAR2(20),
  description       VARCHAR2(50),
  clob_description  CLOB,
  created_date      DATE,
  CONSTRAINT tab_dbms_compress_pk PRIMARY KEY (id)
)
PARTITION BY RANGE (created_Date)
(PARTITION tab_dbms_compress_part_2015 VALUES LESS THAN (TO_DATE('01/01/2016',
'DD/MM/YYYY')) TABLESPACE sysaux,
 PARTITION tab_dbms_compress_part_2016 VALUES LESS THAN (TO_DATE('01/01/2017',
'DD/MM/YYYY')) TABLESPACE sysaux);

Table created.

--- Creamos un indice sobre la tabla tab_dbms_compress ---

SQL> CREATE INDEX tab_dbms_compress_code_idx ON tab_dbms_compress(code) LOCAL;

Index created


SQL> INSERT INTO tab_dbms_compress
```

```
SELECT level,
       CASE
         WHEN MOD(level,2)=0 THEN 'CODE1'
         ELSE 'CODE2'
       END,
       CASE
         WHEN MOD(level,2)=0 THEN 'Description for CODE1'
         ELSE 'Description for CODE2'
       END,
       CASE
         WHEN MOD(level,2)=0 THEN 'CLOB description for CODE1'
         ELSE 'CLOB description for CODE2'
       END,
       CASE
         WHEN MOD(level,2)=0 THEN TO_DATE('01/07/2015','DD/MM/YYYY')
         ELSE TO_DATE('01/07/2016','DD/MM/YYYY')
       END
FROM   dual
CONNECT BY level <= 100000;
COMMIT;

100000 rows created.

Commit complete.

SQL> EXEC DBMS_STATS.gather_table_stats(USER, 'tab_dbms_compress');

PL/SQL procedure successfully completed.
```

El primer ejemplo muestra el efecto de la compresión OLTP en una tabla específica, utilizando todas las filas de la tabla como tamaño de muestra.

```
SQL> SET SERVEROUTPUT ON

SQL> DECLARE
  l_blkcnt_cmp    PLS_INTEGER;
  l_blkcnt_uncmp  PLS_INTEGER;
  l_row_cmp       PLS_INTEGER;
  l_row_uncmp     PLS_INTEGER;
  l_cmp_ratio     NUMBER;
  l_comptype_str  VARCHAR2(32767);
BEGIN
  DBMS_COMPRESSION.get_compression_ratio (
    scratchtbsname  => 'SYSAUX',
    ownname         => 'SSB',
    objname         => 'TAB_DBMS_COMPRESS',
    subobjname      => NULL,
    comptype        => DBMS_COMPRESSION.comp_advanced,
    blkcnt_cmp      => l_blkcnt_cmp,
    blkcnt_uncmp    => l_blkcnt_uncmp,
    row_cmp         => l_row_cmp,
    row_uncmp       => l_row_uncmp,
    cmp_ratio       => l_cmp_ratio,
```

```
    comptype_str    => l_comptype_str,
    subset_numrows  => DBMS_COMPRESSION.comp_ratio_allrows,
    objtype         => DBMS_COMPRESSION.objtype_table
  );

  DBMS_OUTPUT.put_line('Number of blocks used (compressed)       : ' ||
l_blkcnt_cmp);
  DBMS_OUTPUT.put_line('Number of blocks used (uncompressed)     : ' ||
l_blkcnt_uncmp);
  DBMS_OUTPUT.put_line('Number of rows in a block (compressed)   : ' ||
l_row_cmp);
  DBMS_OUTPUT.put_line('Number of rows in a block (uncompressed) : ' ||
l_row_uncmp);
  DBMS_OUTPUT.put_line('Compression ratio                        : ' ||
l_cmp_ratio);
  DBMS_OUTPUT.put_line('Compression type                         : ' ||
l_comptype_str);
END;
/

Number of blocks used (compressed)       : 1313
Number of blocks used (uncompressed)     : 1737
Number of rows in a block (compressed)   : 74
Number of rows in a block (uncompressed) : 55
Compression ratio                        : 1.3
Compression type                         : "Compress Advanced"

PL/SQL procedure successfully completed.
```

El segundo ejemplo muestra el efecto de la compresión OLTP en un índice particionado, utilizando todas las filas de la tabla como tamaño de muestra.

```
SQL> SET SERVEROUTPUT ON

SQL> DECLARE
  l_blkcnt_cmp    PLS_INTEGER;
  l_blkcnt_uncmp  PLS_INTEGER;
  l_row_cmp       PLS_INTEGER;
  l_row_uncmp     PLS_INTEGER;
  l_cmp_ratio     NUMBER;
  l_comptype_str  VARCHAR2(32767);
BEGIN
  DBMS_COMPRESSION.get_compression_ratio (
    scratchtbsname  => 'SYSAUX',
    ownname         => 'SSB',
    objname         => 'TAB_DBMS_COMPRESS_CODE_IDX',
    subobjname      => 'TAB_DBMS_COMPRESS_PART_2015',
    comptype        => DBMS_COMPRESSION.comp_index_advanced_low,
    blkcnt_cmp      => l_blkcnt_cmp,
    blkcnt_uncmp    => l_blkcnt_uncmp,
    row_cmp         => l_row_cmp,
    row_uncmp       => l_row_uncmp,
```

```
    cmp_ratio       => l_cmp_ratio,
    comptype_str    => l_comptype_str,
    subset_numrows  => DBMS_COMPRESSION.comp_ratio_minrows,
    objtype         => DBMS_COMPRESSION.objtype_index
  );

  DBMS_OUTPUT.put_line('Number of blocks used (compressed)       : ' ||
l_blkcnt_cmp);
  DBMS_OUTPUT.put_line('Number of blocks used (uncompressed)     : ' ||
l_blkcnt_uncmp);
  DBMS_OUTPUT.put_line('Number of rows in a block (compressed)   : ' ||
l_row_cmp);
  DBMS_OUTPUT.put_line('Number of rows in a block (uncompressed) : ' ||
l_row_uncmp);
  DBMS_OUTPUT.put_line('Compression ratio                        : ' ||
l_cmp_ratio);
  DBMS_OUTPUT.put_line('Compression type                         : ' ||
l_comptype_str);
END;
/

Number of blocks used (compressed)       : 78
Number of blocks used (uncompressed)     : 120
Number of rows in a block (compressed)   : 641
Number of rows in a block (uncompressed) : 417
Compression ratio                        : 1.5
Compression type                         : "Compress Advanced Low"

PL/SQL procedure successfully completed.
```

El tercer ejemplo muestra el efecto de la compresión OLTP en un campo LOB, utilizando todas las filas de la tabla como tamaño de muestra.

```
SQL> SET SERVEROUTPUT ON

SQL> DECLARE
  l_blkcnt_cmp    PLS_INTEGER;
  l_blkcnt_uncmp  PLS_INTEGER;
  l_lobcnt        PLS_INTEGER;
  l_cmp_ratio     NUMBER;
  l_comptype_str  VARCHAR2(32767);
BEGIN
  DBMS_COMPRESSION.get_compression_ratio (
    scratchtbsname  => 'SYSAUX',
    tabowner        => 'SSB',
    tabname         => 'TAB_DBMS_COMPRESS',
    lobname         => 'CLOB_DESCRIPTION',
    partname        => NULL,
    comptype        => DBMS_COMPRESSION.comp_lob_high,
    blkcnt_cmp      => l_blkcnt_cmp,
    blkcnt_uncmp    => l_blkcnt_uncmp,
    lobcnt          => l_lobcnt,
```

```
    cmp_ratio       => l_cmp_ratio,
    comptype_str    => l_comptype_str,
    subset_numrows  => DBMS_COMPRESSION.comp_ratio_lob_maxrows
  );

  DBMS_OUTPUT.put_line('Number of blocks used (compressed)       : ' ||
l_blkcnt_cmp);
  DBMS_OUTPUT.put_line('Number of blocks used (uncompressed)     : ' ||
l_blkcnt_uncmp);
  DBMS_OUTPUT.put_line('Number of rows in a block (compressed)   : ' ||
l_lobcnt);
  DBMS_OUTPUT.put_line('Number of lobs sampled                   : ' ||
l_cmp_ratio);
  DBMS_OUTPUT.put_line('Compression type                         : ' ||
l_comptype_str);
END;
/

Number of blocks used (compressed)        : 68
Number of blocks used (uncompressed)     : 61
Number of rows in a block (compressed)   : 4999
Number of lobs sampled                   : .8
Compression type                         : "Compress High"

PL/SQL procedure successfully completed.
```

La función GET_COMPRESSION_TYPE muestra el nivel de compresión para una fila especificada en una tabla. En este caso concreto, al tratarse de una tabla que no está comprimida la salida será `'COMP_NOCOMPRESS'`.

```
SQL> SELECT rowid,
       CASE DBMS_COMPRESSION.get_compression_type ('SSB', 'TAB_DBMS_COMPRESS',
rowid, 'TAB_DBMS_COMPRESS_PART_2015')
         WHEN 1     THEN 'COMP_NOCOMPRESS'
         WHEN 2     THEN 'COMP_ADVANCED'
         WHEN 4     THEN 'COMP_QUERY_HIGH'
         WHEN 8     THEN 'COMP_QUERY_LOW'
         WHEN 16    THEN 'COMP_ARCHIVE_HIGH'
         WHEN 32    THEN 'COMP_ARCHIVE_LOW'
         WHEN 64    THEN 'COMP_BLOCK'
         WHEN 128   THEN 'COMP_LOB_HIGH'
         WHEN 256   THEN 'COMP_LOB_MEDIUM'
         WHEN 512   THEN 'COMP_LOB_LOW'
         WHEN 1024  THEN 'COMP_INDEX_ADVANCED_HIGH'
         WHEN 2048  THEN 'COMP_INDEX_ADVANCED_LOW'
         WHEN 1000  THEN 'COMP_RATIO_LOB_MINROWS'
         WHEN 4096  THEN 'COMP_BASIC'
         WHEN 5000  THEN 'COMP_RATIO_LOB_MAXROWS'
         WHEN 8192  THEN 'COMP_INMEMORY_NOCOMPRESS'
         WHEN 16384 THEN 'COMP_INMEMORY_DML'
         WHEN 32768 THEN 'COMP_INMEMORY_QUERY_LOW'
         WHEN 65536 THEN 'COMP_INMEMORY_QUERY_HIGH'
```

```
         WHEN 32768 THEN 'COMP_INMEMORY_CAPACITY_LOW'
         WHEN 65536 THEN 'COMP_INMEMORY_CAPACITY_HIGH'
       END AS compression_type
FROM   ssb.TAB_DBMS_COMPRESS PARTITION (TAB_DBMS_COMPRESS_part_2015)
WHERE  rownum <= 5;
/

ROWID              COMPRESSION_TYPE
------------------ --------------------------
AAASYdAAEAAAMwDAAA COMP_NOCOMPRESS
AAASYdAAEAAAMwDAAB COMP_NOCOMPRESS
AAASYdAAEAAAMwDAAC COMP_NOCOMPRESS
AAASYdAAEAAAMwDAAD COMP_NOCOMPRESS
AAASYdAAEAAAMwDAAE COMP_NOCOMPRESS
```

Proceso de compresión de índice

De igual forma que se puede comprimir una tabla/partición, también es importante poder comprimir los índices aplicados sobre los bloques de las tablas.

En el siguiente ejemplo utilizaremos distintos factores de compresión para los índices bien sean únicos/no únicos creados en la tabla TEST.

```
SQL> conn ssb/ssb@soe

SQL> CREATE TABLE test (
ENAME VARCHAR2(75),
EADD1 VARCHAR2(75),
EADD2 VARCHAR2(75),
EADD3 VARCHAR2(75),
EADD4 VARCHAR2(75),
CITY VARCHAR2(75)
);

Table created.
```

Ahora insertaremos datos mediante `insert con hint /*+ APPEND */` sobre la tabla `TEST`

```
SQL> INSERT /*+ APPEND */ INTO test
SELECT RPAD('X',75, 'X'),
RPAD('X',75, 'X'),
RPAD('X',75, 'X'),
RPAD('X',75, 'X'),
RPAD('X',75, 'X'),
RPAD(TO_CHAR(level),75, 'X')
FROM dual
CONNECT BY level <= 10000;
COMMIT;
```

```
10000 rows created.

Commit complete.

col owner format a10
col segment_name format a15
col segment_type format a15

SQL> SELECT table_name, blocks
FROM dba_tables
WHERE table_name IN ('TEST')
/

TABLE_NAME               BLOCKS
-------------------- ----------
TEST                        687
```

Creamos un primer índice no único sin compresión (`'TEST_IDX'`), verificamos su tamaño y lo borramos. Estas operaciones las vamos a realizar como sys para tener acceso a determinadas vistas del diccionario de datos que requieren privilegios de sysdba.

```
SQL> conn sys/We1c0m3_We1c0m3_@SOE as sysdba

Connected.

SQL> CREATE INDEX ssb.test_idx ON ssb.test(ENAME, EADD1, EADD2, EADD3, EADD4,
CITY);

Index created

SQL> select owner,segment_name,segment_type,bytes from dba_segments where
owner='SSB' and segment_name in ('TEST','TEST_IDX');

OWNER      SEGMENT_NAME    SEGMENT_TYPE         BYTES
---------- --------------- --------------- ----------
SSB        TEST            TABLE              6291456
SSB        TEST_IDX        INDEX              6291456

SQL> drop index ssb.TEST_IDX;
```

Creamos un segundo índice no único CON compresión (`'TEST_IDX'`) y verificamos su tamaño. Para ello, se usará el atributo COMPRESS en la creación del índice. Después lo borramos.

```
SQL> CREATE INDEX ssb.test_idx ON ssb.test(ENAME, EADD1, EADD2, EADD3, EADD4,
CITY) COMPRESS 5;

SQL> select owner,segment_name,segment_type,bytes from dba_segments where
owner='SSB' and segment_name in ('TEST','TEST_IDX');
```

```
OWNER      SEGMENT_NAME    SEGMENT_TYPE         BYTES
---------- --------------- --------------- ----------
SSB        TEST            TABLE              6291456
SSB        TEST_IDX        INDEX              2097152

drop index ssb.test_idx;
```

Por último creamos un índice no único con compresión COMPRESS ADVANCED LOW y COMPRESS ADVANCED HIGH verificando el tamaño de este índice `'TEST_IDX'`. Como podemos comprobar la compresión tipo COMPRESS ADVANCED HIGH comprime los índices de una forma más óptima. El tipo de compresión HIGH está disponible a partir de la versión 12.2 de Oracle Database.

```
SQL> CREATE UNIQUE INDEX ssb.test_idx ON ssb.test(ENAME, EADD1, EADD2, EADD3,
EADD4, CITY) COMPRESS ADVANCED LOW;

SQL> select owner,segment_name,segment_type,bytes from dba_segments where
owner='SSB' and segment_name in ('TEST','TEST_IDX');

OWNER      SEGMENT_NAME    SEGMENT_TYPE         BYTES
---------- --------------- --------------- ----------
SSB        TEST            TABLE              6291456
SSB        TEST_IDX        INDEX              2097152

drop index ssb.test_idx;
```

```
SQL> CREATE UNIQUE INDEX ssb.test_idx ON ssb.test(ENAME, EADD1, EADD2, EADD3,
EADD4, CITY) COMPRESS ADVANCED HIGH;

SQL> select owner,segment_name,segment_type,bytes from dba_segments where
owner='SSB' and segment_name in ('TEST','TEST_IDX');

OWNER      SEGMENT_NAME    SEGMENT_TYPE         BYTES
---------- --------------- --------------- ----------
SSB        TEST            TABLE              6291456
SSB        TEST_IDX        INDEX               196608

SQL> drop index ssb.test_idx;

SQL> exit;
```

Advanced Compression: Oracle SecureFiles

La funcionalidad SecureFiles es un rediseño completo de la implementación del almacenamiento de objetos grandes (LOB) en Oracle 12c. El almacenamiento LOB original, conocido como BASICFILE, sigue siendo el método de almacenamiento predeterminado en versión 11g, pero la palabra clave SECUREFILE habilita el nuevo método de almacenamiento, que permite el cifrado y ahorro de espacio mediante compresión y deduplicación. En 12c es posible establecer a nivel de parámetros de base de datos el poder manejar los campos tipo LOB (Securefiles) como predeterminado en la creación de cualquier tipo LOB. En 18c y 19c por defecto se crean como SECUREFILES.

Para utilizar los campos tipo Securefiles los tablespaces tienen que ser definidos como SEGMENT SPACE MANAGEMENT AUTO;

Para el siguiente ejemplo se crean dos tablespaces (uno para cargar ficheros en base de datos BASICFILES y otro como SECUREFILES).

Se cargarán unos cuantos ficheros de texto .TXT existentes en la carpeta `/home/oracle/sf/docs` en LOB (BASICFILES) y posteriormente los insertaremos en tipo LOB SECUREFILES (comprimidos).

```
[oracle@single19c ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Sep 4 12:50:16 2020
Version 19.8.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.8.0.0.0

SQL> alter session set container=SOE;

Session altered.

SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         5 SOE                            READ WRITE NO
```

```
SQL> CREATE TABLESPACE securefiles
    DATAFILE '+DATA/'
    SIZE 300M REUSE
    EXTENT MANAGEMENT LOCAL
    UNIFORM SIZE 4M
    SEGMENT SPACE MANAGEMENT AUTO;

Tablespace created.
```

```
SQL> CREATE TABLESPACE basicfiles
    DATAFILE '+DATA/'
    SIZE 300M REUSE
    EXTENT MANAGEMENT LOCAL
    UNIFORM SIZE 4M
    SEGMENT SPACE MANAGEMENT AUTO;

Tablespace created.

-- Crear un usuario para la carga de los ficheros de texto .txt

SQL> CREATE USER sf
    IDENTIFIED BY We1c0m3_We1c0m3_
    DEFAULT TABLESPACE sysaux
    TEMPORARY TABLESPACE temp
    QUOTA UNLIMITED ON sysaux
    QUOTA UNLIMITED ON basicfiles
    QUOTA UNLIMITED ON securefiles
/

User created

SQL> GRANT dba TO sf;
SQL> GRANT EXECUTE ANY PROCEDURE, CREATE ANY DIRECTORY TO sf;
```

Se crea con el usuario `sf` un directorio para almacenar los ficheros que luego se utilizarán para cargarlos en la base de datos tipo Securefiles.

```
-- Crear el directorio de objetos donde estarán los ficheros de texto .txt

SQL> conn sf/We1c0m3_We1c0m3_@soe;

SQL> CREATE OR REPLACE DIRECTORY sf_docs
    AS '/home/oracle/sf/docs';

Directory created.
```

Creamos la tabla TICKETS que contendrá los campos LOB tipo BASICFILES y dos tablas que contendrán los campos LOB tipo SECUREFILES: SECURE_TICKETS (no comprimido) y SECURE_TICKETS_COMP (comprimido).

```
SQL> CREATE TABLE sf.tickets (
     tkt_id         NUMBER
    ,description    VARCHAR2(30)
    ,submit_dtm     TIMESTAMP
    ,status         VARCHAR2(8)
    ,document       BLOB
)
    LOB(document)   STORE AS BASICFILE (TABLESPACE basicfiles)
```

```
;

Table created

SQL> CREATE TABLE sf.secure_tickets (
     tkt_id         NUMBER
    ,description    VARCHAR2(30)
    ,submit_dtm     TIMESTAMP
    ,status         VARCHAR2(8)
    ,document       BLOB
)
    LOB(document)
        STORE AS SECUREFILE (
            TABLESPACE securefiles
        )
;

Table created.

SQL> CREATE TABLE sf.secure_tickets_comp (
     tkt_id         NUMBER
    ,description    VARCHAR2(30)
    ,submit_dtm     TIMESTAMP
    ,status         VARCHAR2(8)
    ,document       BLOB
)
    LOB(document)
        STORE AS SECUREFILE (
            TABLESPACE securefiles
            COMPRESS HIGH
        )
;

Table created.
```

Utilizaremos un procedimiento para cargar los ficheros de texto .TXT existentes en la carpeta `/home/oracle/sf/docs` en la tabla TICKETS en tipo BASICFILES.

```
SQL> CREATE OR REPLACE PACKAGE sf.pkg_securefiles
AS
    PROCEDURE AddTroubleTicket (
         tkt_id         IN sf.tickets.tkt_id%TYPE
        ,description    IN sf.tickets.description%TYPE
        ,submit_dts     IN VARCHAR2
        ,status         IN sf.tickets.status%TYPE
        ,docFileName    IN VARCHAR2
    );

END pkg_securefiles;
/
```

```
Package created.


SQL> CREATE OR REPLACE PACKAGE BODY sf.pkg_securefiles
AS
    PROCEDURE LoadBFILEIntoLOB (
     src_dir    IN      VARCHAR2
    ,src_file   IN      VARCHAR2
    ,target_lob IN OUT  BLOB
    )

    IS
        src_loc     BFILE   := BFILENAME(src_dir, src_file);
        load_amt    INTEGER := 4000;
    BEGIN
        -- Open the source document file in read-only mode
        DBMS_LOB.OPEN(
             file_loc => src_loc
            ,open_mode => DBMS_LOB.LOB_READONLY
        );

        -- Calculate the size of the external BFILE
        load_amt := DBMS_LOB.GETLENGTH(file_loc => src_loc);

        -- Load the LOB from the source file
        DBMS_LOB.LOADFROMFILE(target_lob, src_loc, load_amt);

             -- Close the opened BFILE external LOB
        DBMS_LOB.FILECLOSE(file_loc => src_loc);

    EXCEPTION
        WHEN OTHERS THEN
             DBMS_OUTPUT.PUT_LINE('LoadLOBFromFILE Error: ' || SQLCODE || ' - '
|| SQLERRM);

    END LoadBFILEIntoLob;

    PROCEDURE AddTroubleTicket (
         tkt_id         IN sf.tickets.tkt_id%TYPE
        ,description    IN sf.tickets.description%TYPE
        ,submit_dts     IN VARCHAR2
        ,status         IN sf.tickets.status%TYPE
        ,docFileName    IN VARCHAR2
    )
    IS
        submit_dtm  TIMESTAMP;
        docBlob     BLOB;
    BEGIN

        -- Calculate timestamp value
        submit_dtm := TO_TIMESTAMP(submit_dts, 'yyyy-mm-dd hh24:mi:ss');

        -- Add new row, returning references to the document and image BLOBs
        INSERT INTO sf.tickets
```

```
        VALUES (tkt_id, description, submit_dtm, status, EMPTY_BLOB())
        RETURNING document INTO docBlob;

        -- Build the document LOB from the supplied file name
        LoadBFILEIntoLOB('SF_DOCS', docFileName, docBlob);

    EXCEPTION
        WHEN OTHERS THEN
             DBMS_OUTPUT.PUT_LINE('Severe error! ' || SQLCODE || ' - ' ||
SQLERRM);

    END AddTroubleTicket;
END pkg_securefiles;
/

Package body created.
```

Cargamos los ficheros de texto .TXT existentes en la carpeta `/home/oracle/sf/docs` en la tabla TICKETS, para posteriormente migrarlos a formato Securefiles.

```
SQL> SET SERVEROUTPUT ON

SQL> BEGIN

    sf.pkg_securefiles.AddTroubleTicket (
         tkt_id => 101
        ,description => 'Trouble Ticket 101'
        ,submit_dts => '2008-12-31 23:45:00'
        ,status => 'OPEN'
        ,docFileName => 'prueba.txt'
    );

    sf.pkg_securefiles.AddTroubleTicket (
         tkt_id => 102
        ,description => 'Trouble Ticket 102'
        ,submit_dts => '2009-01-04 00:00:00'
        ,status => 'OPEN'
        ,docFileName => 'prueba1.txt'
    );

    sf.pkg_securefiles.AddTroubleTicket (
         tkt_id => 103
        ,description => 'Trouble Ticket 103'
        ,submit_dts => '2009-01-02 00:00:00'
        ,status => 'OPEN'
        ,docFileName => 'prueba2.txt'
     );

    sf.pkg_securefiles.AddTroubleTicket (
         tkt_id => 104
        ,description => 'Trouble Ticket 104'
        ,submit_dts => '2009-01-14 12:30:00'
```

```
        ,status => 'OPEN'
        ,docFileName => 'prueba3.txt'
  );

   sf.pkg_securefiles.AddTroubleTicket (
        tkt_id => 105
       ,description => 'Trouble Ticket 105'
       ,submit_dts => '2009-01-09 00:00:00'
       ,status => 'OPEN'
       ,docFileName => 'prueba4.txt'
   );

   sf.pkg_securefiles.AddTroubleTicket (
        tkt_id => 106
       ,description => 'Trouble Ticket 106'
       ,submit_dts => '2009-01-11 00:00:00'
       ,status => 'OPEN'
       ,docFileName => 'prueba5.txt'
   );

   sf.pkg_securefiles.AddTroubleTicket (
        tkt_id => 107
       ,description => 'Trouble Ticket 107'
       ,submit_dts => '2009-01-16 00:00:00'
       ,status => 'OPEN'
       ,docFileName => 'prueba6.txt'
   );
   sf.pkg_securefiles.AddTroubleTicket (
        tkt_id => 108
       ,description => 'Trouble Ticket 108'
       ,submit_dts => '2009-01-12 00:00:00'
       ,status => 'OPEN'
       ,docFileName => 'prueba7.txt'
   );

   sf.pkg_securefiles.AddTroubleTicket (
        tkt_id => 109
       ,description => 'Trouble Ticket 109'
       ,submit_dts => '2009-01-02 00:00:00'
       ,status => 'OPEN'
       ,docFileName => 'prueba8.txt'
    );

   sf.pkg_securefiles.AddTroubleTicket (
        tkt_id => 110
       ,description => 'Trouble Ticket 110'
       ,submit_dts => '2009-01-14 12:45:00'
       ,status => 'OPEN'
       ,docFileName => 'prueba9.txt'
  );
sf.pkg_securefiles.AddTroubleTicket (
        tkt_id => 111
       ,description => 'Trouble Ticket 111'
       ,submit_dts => '2009-01-14 12:45:00'
```

```
        ,status => 'OPEN'
        ,docFileName => 'prueba10.txt'
   );
    COMMIT;
END;
/

PL/SQL procedure successfully completed.
```

Analizamos el schema SF mediante DBMS_STATS.GATHER_SCHEMA_STATS.

```
SQL> BEGIN
    DBMS_STATS.GATHER_SCHEMA_STATS(ownname => 'sf', CASCADE => TRUE);
END;
/

PL/SQL procedure successfully completed.
```

Insertamos en las tablas creadas como SECUREFILES a partir de los datos en la tabla creada como BASICFILES mediate `insert … select …`

```
SQL> INSERT INTO sf.secure_tickets
SELECT * FROM sf.tickets;

11 rows created.

SQL> INSERT INTO sf.secure_tickets_comp
SELECT * FROM sf.tickets;

11 rows created.

SQL> COMMIT;
```

Comprobar los metadados creados en ambas tablas accediendo a la vista DBA_SEGMENTS:

```
SET PAGESIZE 1000
SET LINESIZE 140
set serveroutput on
-----
-- View: DBA_SEGMENTS
-- Shows metadata about individual BasicFile and SecureFile segments
-----

TTITLE 'LOB Segment Information|(from DBA_SEGMENTS)'
COL segment_name        FORMAT A30      HEADING 'Segment Name'
COL segment_type        FORMAT A20      HEADING 'Segment|Type'
COL segment_subtype     FORMAT A20      HEADING 'Segment|SubType'
COL partition_name      FORMAT A12      HEADING 'Partition|Name'
COL tablespace_name     FORMAT A12      HEADING 'Tablespace'
SELECT
```

```
    segment_name
   ,segment_type
   ,segment_subtype
   ,partition_name
   ,tablespace_name
  FROM dba_segments
 WHERE owner = 'SF'
 ORDER BY segment_name
;
TTITLE OFF


TTITLE OFF
  3    4    5    6    7    8    9   10
Wed Jan 12
page    1
                                                        LOB Segment
Information
                                                          (from DBA_SEGMENTS)

                               Segment              Segment
Partition
Segment Name                   Type                 SubType              Name
Tablespace
------------------------------ -------------------- -------------------- ------
------ ------------
SECURE_TICKETS                 TABLE                ASSM
SYSAUX
SECURE_TICKETS_COMP            TABLE                ASSM
SYSAUX
SYS_IL0000075331C00005$$       LOBINDEX             ASSM
BASICFILES
SYS_IL0000075334C00005$$       LOBINDEX             ASSM
SECUREFILES
SYS_IL0000075337C00005$$       LOBINDEX             ASSM
SECUREFILES
SYS_LOB0000075331C00005$$      LOBSEGMENT           ASSM
BASICFILES
SYS_LOB0000075334C00005$$      LOBSEGMENT           SECUREFILE
SECUREFILES
SYS_LOB0000075337C00005$$      LOBSEGMENT           SECUREFILE
SECUREFILES
TICKETS                        TABLE                ASSM
SYSAUX

9 rows selected.
```

Basado en la información obtenida de la vista DBA_SEGMENTS modificar el nombre de los segmentos (LOB) correspondientes al tablespace SECUREFILES en el bloque PL/SQL más abajo, concretamente la parte resaltada en color rojo. A continuación, se muestra una salida de ejemplo:

```
SYS_LOB0000075334C00005$$       LOBSEGMENT              SECUREFILE
SECUREFILES
SYS_LOB0000075337C00005$$       LOBSEGMENT              SECUREFILE
SECUREFILES
```

Antes de ejecutar el siguiente bloque, modificar el código PL/SQL y añadir los valores recuperados en la consulta anterior reemplazando el texto en rojo.

```
SQL> declare
  segment_size_block NUMBER;
  segment_size_byte NUMBER;
  used_block NUMBER;
  used_byte NUMBER;
  expired_block NUMBER;
  expired_byte NUMBER;
  unexpired_block NUMBER;
  unexpired_byte NUMBER;
begin
dbms_space.space_usage ('SF', 'SYS_LOB0000075334C00005$$', 'LOB',
dbms_space.spaceusage_exact, segment_size_block,
segment_size_byte, used_block, used_byte, expired_block, expired_byte,
unexpired_block, unexpired_byte, null);
dbms_output.put_line('segment_size_blocks = '||segment_size_block);
dbms_output.put_line('segment_size_bytes = '||segment_size_byte);
dbms_output.put_line('used_blocks = '||used_block);
dbms_output.put_line('used_bytes = '||used_byte);
dbms_output.put_line('expired_blocks = '||expired_block);
dbms_output.put_line('expired_bytes = '||expired_byte);
dbms_output.put_line('unexpired_blocks = '||unexpired_block);
dbms_output.put_line('unexpired_bytes = '||unexpired_byte);
end;
/

segment_size_blocks = 3072
segment_size_bytes = 25165824
used_blocks = 2468
used_bytes = 20217856
expired_blocks = 604
expired_bytes = 4947968
unexpired_blocks = 0
unexpired_bytes = 0

PL/SQL procedure successfully completed.

segment_size_blocks = 512
segment_size_bytes = 4194304
used_blocks = 87
used_bytes = 712704
expired_blocks = 425
expired_bytes = 3481600
unexpired_blocks = 0
```

```
unexpired_bytes = 0

PL/SQL procedure successfully completed.
```

Observar la ocupación de los segmentos de la columna ‘segment_size_bytes’ de ambos objetos por el nombre de los segmentos LOBS.

Consultas sobre tablas Comprimidas/No Comprimidas

En el siguiente ejemplo se ejecutarán varias sentencias sobre tablas Comprimidas y No comprimidas para ver el rendimiento de ambas. ACO nos proporciona beneficios tanto de reducción del almacenamiento, como rendimiento en consultas al tener que leer menos bloques en memoria de los Buffers de la SGA.

Con las siguientes sentencias, vemos que hay dos tablas (LINEORDER y LINEORDER_NO_ACO) una comprimida con compresión BASIC y otra sin compresión. De igual forma vemos la ocupación de bloques de cada tabla.

```
$ sqlplus ssb/ssb@soe;

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 12 16:07:29 2022
Version 19.13.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.

Last Successful login time: Wed Jan 12 2022 13:13:03 +00:00

Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.13.0.0.0

SQL> set linesize 1000
SQL> col table_name format a30

SQL> SELECT table_name,compression, compress_for
FROM   user_tables
WHERE  table_name like 'LINEORDE%';

TABLE_NAME                     COMPRESS COMPRESS_FOR
------------------------------ -------- ------------------------------
LINEORDER                      ENABLED  BASIC
LINEORDER_ACO                  ENABLED  ADVANCED
LINEORDER_NO_ACO               DISABLED


SQL> SELECT table_name,
       compression,
       num_rows,
       blocks,
```

```
       empty_blocks
FROM   user_tables
WHERE  table_name LIKE ('LINEORDER%')
ORDER BY 1;

TABLE_NAME                     COMPRESS   NUM_ROWS     BLOCKS EMPTY_BLOCKS
------------------------------ -------- ---------- ---------- ------------
LINEORDER                      ENABLED    29999800     223987            0
LINEORDER_ACO                  ENABLED     1000000       8416            0
LINEORDER_NO_ACO               DISABLED   29999870     417832            0
```

Ejecutar las siguientes sentencias para ver como responden sobre tablas comprimidas/no comprimidas.

```
SQL> set timing on

SQL> select /* no compresion */ max(LO_ORDTOTALPRICE) most_expensive_order From
LINEORDER_NO_ACO where LO_PARTKEY=300023;

MOST_EXPENSIVE_ORDER
--------------------
            40614642

Elapsed: 00:00:29.68


SQL> select /* compresion */ max(LO_ORDTOTALPRICE) most_expensive_order From
LINEORDER where LO_PARTKEY=300023;

MOST_EXPENSIVE_ORDER
--------------------
            40614642

Elapsed: 00:00:00.17
```

Otras dos consultas un poco más complejas

```
SQL> select  /* no compresion */ d_date,sum(l.lo_revenue) "Total Revenue"
From   LINEORDER_NO_ACO l, DATE_DIM d
Where  l.lo_orderdate = d.d_datekey
and      D_DAYNUMINMONTH = 25
and     d.d_month = 'December'
group by d_date
order by d_date;

D_DATE             Total Revenue
------------------ -------------
December 25, 1992     4.6038E+10
December 25, 1993     4.3746E+10
December 25, 1994     4.5026E+10
```

```
December 25, 1995     4.5589E+10
December 25, 1996     4.6722E+10
December 25, 1997     4.4994E+10

6 rows selected.

Elapsed: 00:00:29.77

SQL> select  /* compresion */ d_date,sum(l.lo_revenue) "Total Revenue"
From   LINEORDER l, DATE_DIM d
Where  l.lo_orderdate = d.d_datekey
and      D_DAYNUMINMONTH = 25
and     d.d_month = 'December'
group by d_date
order by d_date;

D_DATE             Total Revenue
------------------ -------------
December 25, 1992     4.6038E+10
December 25, 1993     4.3746E+10
December 25, 1994     4.5026E+10
December 25, 1995     4.5589E+10
December 25, 1996     4.6722E+10
December 25, 1997     4.4994E+10

6 rows selected.

Elapsed: 00:00:02.84
```

Proceso de compresión de expdp

Desde la versión 11g, Data Pump permite comprimir el backup antes de escribir a fichero dump con el parámetro ‘compression’.

El parámetro ‘compression’ puede contener 4 valores:

- ALL
- DATA_ONLY
- METAGATA_ONLY
- NONE

Utilizando el valor ‘ALL’, el tamaño del fichero de backup puede ser reducido hasta 10 veces. El tiempo de expdp se incrementará con respecto al export sin compresión.

Vamos a crear dos ficheros de parámetros para EXPDP, uno con compresión y otro si ella. Para crear estos ficheros use el editor vi o cree el fichero con un editor de texto local y transfiéralo a la máquina con SFTP.

El contenido de `exp_pdbsoe_comp.par` (con compresión) debe ser el siguiente.

```
directory=DATA_PUMP_DIR
```

```
dumpfile=EXP_PDBSOE_COMP%U.dmp
logfile=EXP_PDBSOE_COMP.log
COMPRESSION=ALL
COMPRESSION_ALGORITHM=HIGH
SCHEMAS=SOE
Exclude=materialized_view
```

El contenido de `exp_pdbsoe_nocomp.par` (sin compresión) debe ser el siguiente.

```
directory=DATA_PUMP_DIR
dumpfile=EXP_PDBSOE_NOCOMP%U.dmp
logfile=EXP_PDBSOE_NOCOMP.log
SCHEMAS=SOE
Exclude=materialized_view
```

Por defecto existe un directorio con nombre 'DATA_PUMP_DIR'. Ejecutar la siguiente sentencia para ver el PATH completo de disco.

```
col owner format a20;
col directory_name format a30;
col directory_path format a65;

SQL> select * from dba_directories where DIRECTORY_NAME= 'DATA_PUMP_DIR';

OWNER                DIRECTORY_NAME
-------------------- ------------------------------
DIRECTORY_PATH                                                   ORIGIN_CON_ID
---------------------------------------------------------------- -------------
SYS                  DATA_PUMP_DIR
/u01/app/oracle/product/19.0.0.0/dbhome_1/rdbms/log/                         1
```

Lanzar los comandos expdp con los ficheros .PAR creado en el paso anterior, para la pluggable database SOE.

```
$ expdp system/We1c0m3_We1c0m3_@SOE parfile=exp_pdbsoe_comp.par
$ expdp system/We1c0m3_We1c0m3_@SOE parfile=exp_pdbsoe_nocomp.par
```

Comprobar el tiempo que tarda cada uno y el espacio ocupado por el fichero .dmp generado.

31060 -rw-r----- 1 oracle asmadmin 31801344 Jan 12 16:21
EXP_PDBSOE_COMP01.dmp
87156 -rw-r----- 1 oracle asmadmin 89243648 Jan 12 16:23
EXP_PDBSOE_NOCOMP01.dmp