

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

## Parte 3 de 3



# Contenidos

## WORKSHOP MULTITENANT, MULTIMODEL, IN-MEMORY PARA LA BASE DE DATOS

<b>ORACLE .....</b>	<b>1</b>
<b>PARTE 3 DE 3 .....</b>	<b>1</b>
<b>IN-MEMORY (30 MIN) .....</b>	<b>3</b>
CONFIGURACIÓN DEL ÁREA DE MEMORIA (5 MIN) .....	3
<i>Configurar FastStart (5 min).....</i>	<i>5</i>
<i>Para desactivar el FAST START .....</i>	<i>6</i>
<i>Publicar las tablas SSB en in-memory (5 min).....</i>	<i>7</i>
<i>Monitorizar la publicación de SSB en In Memory.....</i>	<i>8</i>
<i>Queries Sencillas .....</i>	<i>9</i>
<i>Queries de grado medio .....</i>	<i>11</i>
<i>Queries complejas.....</i>	<i>13</i>
<b>ACO - ORACLE ADVANCED COMPRESSION (45 MIN).....</b>	<b>15</b>
ADVANCED ROW COMPRESSION IMPLEMENTATION .....	15
ADVANCED ROW COMPRESSION TABLA/PARTICIÓN .....	19
<i>Utilización de API PL/SQL DBMS_COMRESSION.....</i>	<i>23</i>
<i>Proceso de compresión de índice .....</i>	<i>27</i>
ADVANCED COMPRESSION: ORACLE SECUREFILES .....	29
<i>Consultas sobre tablas Comprimidas/No Comprimidas .....</i>	<i>37</i>
<i>Proceso de compresión de expdp.....</i>	<i>38</i>



# In-Memory (30 min)

## Configuración del área de memoria (5 min)

Lo primero es comprobar 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_percent	integer	1
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



```
sqlplus / as sysdba
```

```
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	6G
sga_min_size	big integer	0
sga_target	big integer	6G
unified_audit_sga_queue_size	integer	1048576

```
SQL> alter system set inmemory_size = 1G scope=spfile;
```

```
[oracle@myoracledb ~]$ srvctl stop database -d $ORACLE_UNQNAME
```

```
[oracle@myoracledb ~]$ srvctl start database -d $ORACLE_UNQNAME
```

```
sqlplus / as sysdba
```

```
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	6G
inmemory_trickle_repopulate_servers_percent	integer	1
inmemory_virtual_columns	string	MANUAL
inmemory_xmem_size	big integer	0
optimizer_inmemory_aware	boolean	TRUE

Luego el resize es dinámico

```
SQL> alter system set inmemory_size = 4G scope=both;
```

```
System altered.
```

```
SQL>
```



```
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	4G
inmemory_trickle_repopulate_servers_1	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 porque es un proceso online.

```
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

SQL>
```

## 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.

```
conn / as sysdba

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
4	INVALID_TABLESPACE	DISABLE



```

conn system/WddFsdF_12_we2@SOE

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.

conn / as sysdba

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
4 TBS_IMC_FASTSTART                ENABLE

conn system/WddFsdF_12_we2@SOE

COL TABLESPACE_NAME FORMAT a20

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

```
conn system/WddFsdF_12_we2@SOE
```



```
SQL> EXEC DBMS_INMEMORY_ADMIN.FASTSTART_DISABLE
```

```
SQL> drop tablespace TBS_IMC_FASTSTART including contents and datafiles;
```

```
Tablespace dropped.
```

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

```
conn ssb/ssb@SOE
col table_name format a30
set lines 120
--display current status
```

```
select table_name,
       inmemory,
       inmemory_priority,
       inmemory_compression
from user_tables;
```

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

```
11 rows selected.
```

```
--alter tables in memory
alter table lineorder inmemory;
alter table part inmemory;
alter table customer inmemory;
alter table supplier inmemory;
alter table date_dim inmemory;
```

```
select table_name,
       inmemory,
       inmemory_priority,
       inmemory_compression
from user_tables;
```

TABLE_NAME	INMEMORY	INMEMORY	INMEMORY_COMPRESS
LINEORDER	ENABLED	NONE	FOR QUERY LOW



RESULTS	DISABLED			
TMP	DISABLED			
DATE_DIM	ENABLED	NONE	FOR QUERY	LOW
YEARLY_PROFIT_REP_MV	DISABLED			
CUSTOMER	ENABLED	NONE	FOR QUERY	LOW
LINEORDER_ACO	DISABLED			
ETL_DD	DISABLED			
SUPPLIER	ENABLED	NONE	FOR QUERY	LOW
PART	ENABLED	NONE	FOR QUERY	LOW
ETL_LO	DISABLED			

11 rows selected.

```
--fetch all rows to start population
select count(*) from lineorder;
select count(*) from part;
select count(*) from customer;
select count(*) from supplier;
select count(*) from date_dim;
```

## Monitorizar la publicación de SSB en In Memory

```
conn system/WddFsdF_12_we2@SOE
```

```
--new view v$im_segments
desc v$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
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
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:

```
conn ssb/ssb@SOE

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

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

clear scr
--flush the buffer_cache
alter system flush buffer_cache;
alter session force parallel query parallel 4;

set lines 120
set autotrace traceonly explain statistics
set timing on

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

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

clear scr
--column store enabled via INMEMORY_QUERY parameter--
alter session disable parallel query;

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

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

clear scr
alter session force parallel query parallel 4;

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

### Query 2



```

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

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

clear scr
--flush the buffer_cache
alter system flush buffer_cache;
alter session force parallel query parallel 4;

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

```

```

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

clear scr
alter session disable parallel query;

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

```

```

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

clear scr
alter session force parallel query parallel 4;

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

```

## Queries de grado medio

### Two table scan



## Query 1

```
conn ssb/ssb@SOE

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

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

clear scr
--flush the buffer_cache
alter system flush buffer_cache;
alter session force parallel query parallel 4;
alter session set inmemory_query='DISABLE';
set autotrace traceonly explain statistics
set timing on

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;
```

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

clear scr
alter session disable parallel query;
alter session set inmemory_query='ENABLE';

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;
```



```

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

clear scr
alter session force parallel query parallel 4;

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;

```

## Queries complejas

### Three table scan

#### Query 1

```

conn ssb/ssb@SOE

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

clear scr
--flush the buffer_cache
alter system flush buffer_cache;
set autotrace traceonly explain statistics
alter session force parallel query parallel 4;
alter session set inmemory_query='DISABLE';
set timing on

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;

```



```

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

clear scr
alter session disable parallel query;
alter session set inmemory_query='DISABLE';

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;

```

```

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

clear scr
alter session force parallel query parallel 4;

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;

```



# 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@myoracledb ~]$ 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)
```



```

);

SET SERVEROUTPUT ON
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;
/

```

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;
/

```

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.

```

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)
)
COMPRESSION FOR ALL OPERATIONS;

SET SERVEROUTPUT ON

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;
/

```



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

```
col table_name format a20

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

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

## 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.

```
CREATE TABLE test_tab_partition (
  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)
);

set linesize 1000;
COLUMN partition_name FORMAT A30

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

```

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.

```

SET SERVEROUTPUT ON
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;
/

```

Analizar la tabla (Partición) mediante DBMS\_STATS.gather\_table\_stats. Posteriormente veremos la ocupación de los bloques y datos insertados sobre la tabla particionada.

```

EXEC DBMS_STATS.gather_table_stats(USER, 'TEST_TAB_PARTICION');

SELECT table_name,
       partition_name,
       compression,
       num_rows,
       blocks,
       empty_blocks
  FROM   user_tab_partitions
 WHERE  table_name = 'TEST_TAB_PARTICION'

```



```
ORDER BY 1;
```

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.

```
SET SERVEROUTPUT ON
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;
/
```

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

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

COLUMN table_name FORMAT A20
COLUMN partition_name FORMAT A20

SELECT table_name,
       partition_name,
       compression,
```



```

        num_rows,
        blocks,
        empty_blocks
FROM   user_tab_partitions
WHERE  table_name = 'TEST_TAB_PARTICION'
ORDER BY 1;

```

## Utilización de API PL/SQL DBMS\_COMPRESSION

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.

```

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);

--- Creamos un índice sobre la tabla tab_dbms_compress ---

CREATE INDEX tab_dbms_compress_code_idx ON tab_dbms_compress(code) LOCAL;

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;

EXEC DBMS_STATS.gather_table_stats(USER, 'tab_dbms_compress');
```

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.

```

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         => '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;
/

```

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.

```

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         => '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;
/

```



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.

```
SET SERVEROUTPUT ON
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;
/
```

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'.

```
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;
/

```

## 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.

```

conn ssb/ssb@soe

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

```

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

```

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;

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

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

```

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.

```

conn sys/WddFsdF_12_we2@SOE as sysdba

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

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

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.

```

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

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

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.

```

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

```



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

drop index ssb.test_idx;
```

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

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

drop index ssb.test_idx;
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@myoracledb ~]$ 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
```

```
alter session set container=SOE;
```

```
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;
```

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

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

```
CREATE USER sf
  IDENTIFIED BY WddFsdf_12_we2
  DEFAULT TABLESPACE sysaux
  TEMPORARY TABLESPACE temp
  QUOTA UNLIMITED ON sysaux
  QUOTA UNLIMITED ON basicfiles
  QUOTA UNLIMITED ON securefiles
/
```

```
GRANT dba TO sf;
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

conn sf/WddFsdf_12_we2@soe;

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



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).

```
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)  
;  
  
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  
    )  
;  
  
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  
    )  
;
```

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.

```
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;
/

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 (tkr_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;
/

```

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

```

SET SERVEROUTPUT ON

BEGIN

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

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

  sf.pkg_securefiles.AddTroubleTicket (
    tkr_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;
/

```

Analizamos el schema SF mediante DBMS\_STATS.GATHER\_SCHEMA\_STATS.

```

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

```

Insertamos en las tablas creadas como SECUREFILES a partir de los datos en la tabla creada como BASICFILES mediante insert ... select ...

```

INSERT INTO sf.secure_tickets
SELECT * FROM sf.tickets;

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

COMMIT;

```

Comprobar los metadatos 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

```

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_LOB0000074019C00005\$\$	LOBSEGMENT	SECUREFILE
SECUREFILES		
SYS_LOB0000074022C00005\$\$	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.

```

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_LOB00000XXXX0C00005$$', '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;
/

```

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;

set linesize 1000
col table_name format a30

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

SELECT table_name,
       compression,
       num_rows,
       blocks,
       empty_blocks
FROM   user_tables
WHERE  table_name LIKE ('LINEORDER%')
ORDER BY 1;
```

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

```
set timing on

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

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

Otras dos consultas un poco más complejas

```
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;

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;

```

## 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 transféralo 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		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/WddFsdF_12_we2@SOE parfile=exp_pdbsoe_comp.par
$ expdp system/WddFsdF_12_we2@SOE parfile=exp_pdbsoe_nocomp.par
```

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

