Hybrid Columnar Compression
HCC en Oracle Exadata

Documento generado por

Hector Ulloa Ligarius

Para el sitio

Índice

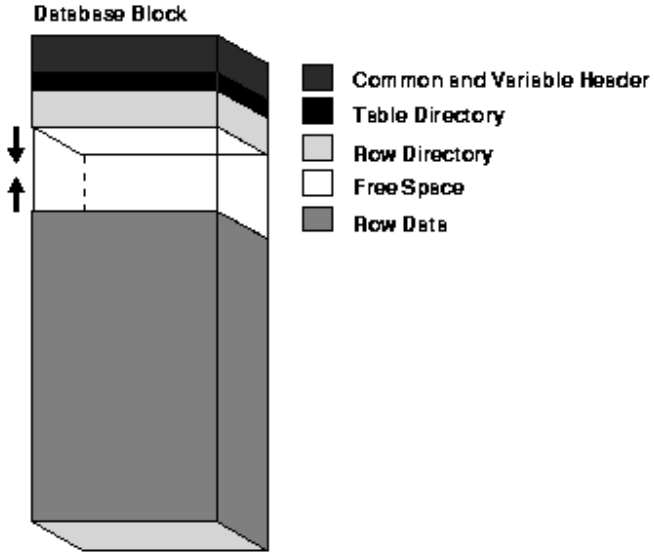
1. Introducción

El siguiente documento explica de manera superficial como es y como trabaja Hybrid Columnar Compression , la nueva forma de comprimir y potente característica de Oracle Exadata.

Se mencionan los tipos de compresión, como se almacena la información dentro de los bloques Oracle, lo que sucede cuando se trabaja con DMLs y por supuesto, una tabla de resultados ,para saber cuando y donde trabajar con HCC

2. Organización de los bloques en Oracle

Antes de comenzar a hablar de HCC , nos tenemos que referir a distintos aspectos para después fundir los conceptos y entender a cabalidad el concepto de HCC, el primero concepto al que nos tenemos que referir es a como Oracle almacena las filas en sus bloques.



En el dibujo se aprecian los siguientes sectores :

- Common and Variable Header :
  Cabecera con información general como la dirección del bloque y el tipo de segmento al cual pertenece

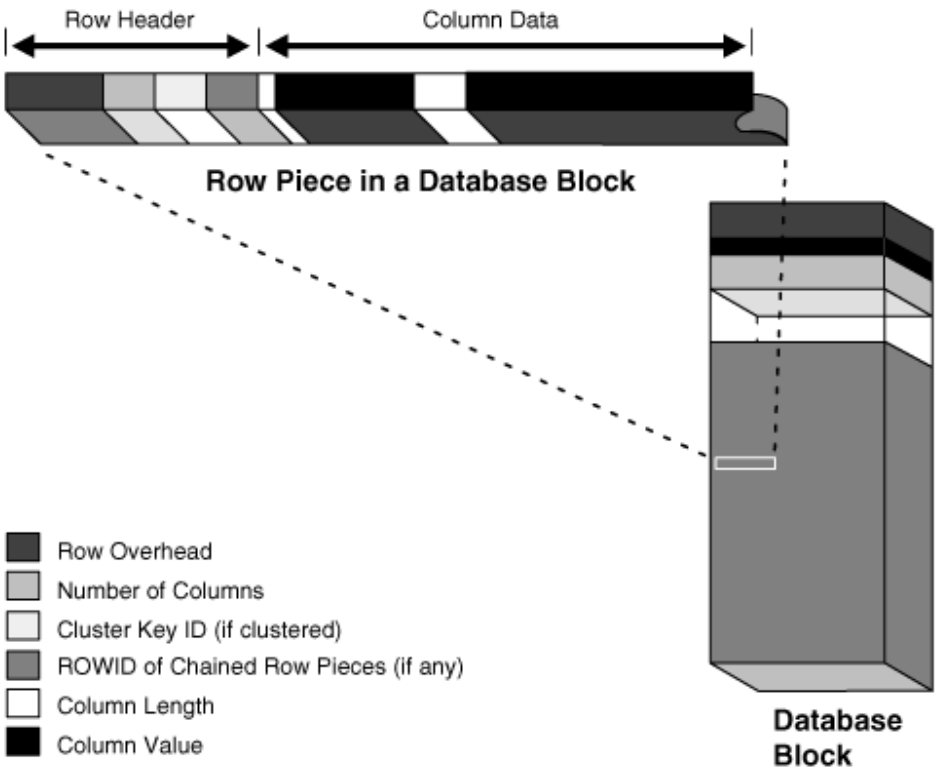
- Table Directory ;
  Este directorio contiene información relacionada de la tabla que posee filas dentro de este bloque

- Row Directory :
  Esta porción del bloque contiene información de las filas que se ubican dentro del bloque, información como el direccionamiento de cada fila (addresses)

- Free Space :
  Acá es donde queda el espacio para que las filas almacenadas en el bloque puedan crecer, o sea, cada vez que una fila aumenta de tamaño y esta almacenada en este bloque ocupa el Free Space para poder crecer

- Row Data :
  Esta parte del bloque es donde se almacena la información (filas) de la tabla o los índices

Y dentro de la Row Data, que es donde se almacenan las filas, pues la información se distribuye de la siguiente forma :



En el dibujo se aprecian los siguientes sectores :

- Filas
- Cantidad de columnas
- La fila propiamente tal
- Cantidad de columnas en la fila
- Largo de la columna
- Valor de la columna

Si buscas un detalle y entendimiento mayor , puedes consultar el siguiente link
http://docs.oracle.com/cd/B19306_01/server.102/b14220/schema.htm

Como se puede apreciar, toda la organización esta orientada a la fila, pues cada vez que se lee , por ejemplo una sola columna, se esta leyendo la fila completa, es más, se lee el bloque Oracle completo.

3. Tipos de compresión en Oracle

En Oracle existen diferentes mecanismos para comprimir la data, estos viene de versiones anteriores de Oracle y son importantes pues algunos tienen directa relación con HCC

3.1. Compresión BASICA (BASIC)

Este método de compresión es una característica básica de Oracle 11g EE. Esta forma comprime sólo DATA , mediante DIRECT PATH LOADS, la compresión se lleva a cabo a nivel de Bloque, este método es el default , acá la documentación de este tipo de compresión

http://docs.oracle.com/cd/E11882_01/server.112/e25494/tables002.htm#CJAGFBFG

Un ejemplo de la creación de una tabla comprimida con BASIC

Creamos una tabla con una gran cantidad de registros

```
13:11:57 SQL> select count(*) from tabla_normal;

  COUNT(*)
----------
   1894412

Elapsed: 00:00:00.16
```

Ejecutamos este scripts (`tamano_tablas.sql`) para saber algunas características de la tabla

```
set linesize 180

col "Dueno" format a20
col "Nombre segmento" format a30
col "Total Mb" for 999G999D9


select s.owner          "Dueno"          ,
       segment_name     "Nombre segmento" ,
       bytes/1024/1024  "Total Mb",
       compress_for     "Compresion"
  from dba_segments s,
       dba_tables t
 where s.owner = t.owner
   and t.table_name = s.segment_name
   and s.owner like '&owner'
and t.table_name like '&table_name'
order by 3;
```

Y su resultado sería

```
13:22:10  11  Enter value for owner: TEST
old   9:    and s.owner like '&owner'
new   9:    and s.owner like 'TEST'
Enter value for table_name: TABLA%
old  10: and t.table_name like '&table_name'
new  10: and t.table_name like 'TABLA%'

Dueno                Nombre segmento                 Total Mb Compresion
-------------------- ------------------------------ ---------- ------------
TEST                 TABLA_NORMAL                       212.8
```

Pues ahora generamos una tabla, pero con el grado de compresión BASIC

```
13:40:32 SQL> create table table_com_basic compress for direct_load operations as
select * from tabla_normal;

Table created.

Elapsed: 00:00:05.21
13:41:01 SQL>
```

Y chequeamos el tamaño de ambas tablas

```
13:42:54 SQL> start tamano_tablas.sql

Enter value for owner: TEST
old   9:    and s.owner like '&owner'
new   9:    and s.owner like 'TEST'
Enter value for table_name: TABL%
old  10: and t.table_name like '&table_name'
new  10: and t.table_name like 'TABL%'

Dueno                Nombre segmento                 Total Mb Compresion
-------------------- ------------------------------ ---------- ------------
TEST                 TABLE_COM_BASIC                    62.0 BASIC
TEST                 TABLA_NORMAL                      212.8

Elapsed: 00:00:00.11
```

Como se puede ver , la compresión de una tabla , pasa de casi 213Mb a 62Mb

Esta compresión sólo se puede usar para Direct Load , o sea, esto se aplica para los siguientes tipos de carga :

- Sentencias INSERT con el hint APPEND
- Parallel DML
- Direct Path de SQL*Loader
- CTAS (Create Table AS SELECT)

3.2. Compresión OLTP (OLTP)

Este método de compresión , permite comprimir bajo todas las operaciones, no sólo DIRECT PATH LOADS, esto forma parte de una opción pagada llamda Advanced Commpression y fue introducido en Oracle 11gr1. Esto es lo mismo que BASIC y se usa una tabla de símbolos para reemplazar los valores repetidos. La compresión por OLTP , deja un PCTFREE de 10 para permitir las actualizaciones de las filas, el método de compresión BASIC usa un PCTFREE de 0 , ejemplo de como habilitar la compresión OLTP

Pues ahora generamos una tabla, pero con el grado de compresión OLTP

```
14:02:05 SQL> create table table_com_oltp compress for all operations as select *
from tabla_normal;

Table created.

Elapsed: 00:00:05.45
14:02:12 SQL>
```

Y chequeamos el tamaño de todas tablas

```
14:03:13 SQL> start tamano_tablas.sql
Enter value for owner: TEST
old   9:    and s.owner like '&owner'
new   9:    and s.owner like 'TEST'
Enter value for table_name: TABL%
old  10: and t.table_name like '&table_name'
new  10: and t.table_name like 'TABL%'

Dueno                Nombre segmento                 Total Mb Compresion
-------------------- ------------------------------ ---------- ------------
TEST                 TABLE_COM_BASIC                      62.0 BASIC
TEST                 TABLE_COM_OLTP                       72.0 OLTP
TEST                 TABLA_NORMAL                        212.8

Elapsed: 00:00:00.11
```

Acá verificamos que la compresión OLTP , comprime menos que la BASIC, aunque la BASIC es más reducida , dado que es solo para DIRECT LOADS , en cambio la OLTP es para toda operación

Por que reviste importancia la compresión OLTP, pues cuando no se pueda usar HCC (Hybrid Columnar Compression) se debe usar OLTP y cuando se hace actualizaciones a filas en tablas que están comprimidas con HCC, de inmediato esa fila pasa a ser OLTP, más adelante explicaremos este raro evento.

3.3. Compresión HCC (Hybrid Columnar Compression)

Solamente está disponible para tablas que sean almacenadas en un Exadata Storage y solamente las tablas que sean cargadas con DIRECT PATH serán comprimidas

Si hacemos Insert y update comunes sobre las tablas con HCC , estas DMLs causarán registros que son almacenados en formato OLTP, si se actualiza un registro, ese registro es migrado a un nuevo bloque y ese bloque es marcado para compresión OLTP (por eso se menciona la importancia de la compresión por OLTP en el punto anterior)

HCC , provee 4 métodos de compresión , los cuales son :

3.3.1. QUERY LOW

Este nivel de compresión usal el algoritmo LZO, provee el nivel más bajo de compresión , pero también es el que involucra menos CPU para comprimir y descomprimir, este método es el más rápido y em mucha documentación se conoce como WAREHOUSE LOW, el grado de compresión es 4x

En el siguiente ejemplo , efectuaremos una compresión HCC con Query Low

```
16:31:27 SQL> create table table_hcc_querylow COMPRESS FOR QUERY LOW as select *
from tabla_normal;

Table created.

Elapsed: 00:00:03.17
```

Y chequeamos el tamaño de todas tablas

```
16:32:17 SQL> start tamano_tablas.sql
Enter value for owner: TEST
old   9:    and s.owner like '&owner'
new   9:    and s.owner like 'TEST'
Enter value for table_name: TABL%
old  10: and t.table_name like '&table_name'
new  10: and t.table_name like 'TABL%'

Dueno                Nombre segmento                  Total Mb Compresion
-------------------- ------------------------------ ---------- ------------
TEST                 TABLE_HCC_QUERYLOW                   24.0 QUERY LOW
TEST                 TABLE_COM_BASIC                      62.0 BASIC
TEST                 TABLE_COM_OLTP                       72.0 OLTP
TEST                 TABLA_NORMAL                        212.8
```

Podemos apreciar claramente la potencia en la compresión de datos ,pues pasamos de casi 213Mb a sólo 24Mb

3.3.2. QUERY HIGH

Este método de compresión usa ZLIB (gzip) , se denomina a veces como WAREHOUSE HIGH y el grado de compresión es como 6x

En el siguiente ejemplo, efectuaremos una compresión HCC con Query High

```
16:34:10 SQL> create table table_hcc_queryhigh COMPRESS FOR QUERY HIGH as select
* from tabla_normal;

Table created.

Elapsed: 00:00:05.58
16:34:17 SQL>
```

Y chequeamos el tamaño del resto de las tablas

```
16:34:57 SQL> start tamano_tablas.sql
Enter value for owner: TEST
old   9:    and s.owner like '&owner'
new   9:    and s.owner like 'TEST'
Enter value for table_name: TABL%
old  10: and t.table_name like '&table_name'
new  10: and t.table_name like 'TABL%'

Dueno                Nombre segmento                  Total Mb Compresion
-------------------- ------------------------------ ---------- ------------
TEST                 TABLE_HCC_QUERYHIGH                  12.0 QUERY HIGH
TEST                 TABLE_HCC_QUERYLOW                   24.0 QUERY LOW
TEST                 TABLE_COM_BASIC                      62.0 BASIC
TEST                 TABLE_COM_OLTP                       72.0 OLTP
TEST                 TABLA_NORMAL                        212.8
```

A medida que vamos aplicando un más alto grado de compresión para HCC, la tabla va disminuyendo en tamaño, pasando de 212.8 Mb a 12Mb

3.3.3. ARCHIVE LOW

Este usa el método de compresión ZLIB (gzip) , pero con un grado de compresión mayor que el QUERY HIGH, puede alcanzar un grado de compresión de 7x

Ejecutamos un comando para crear la tabla

```
16:36:41 SQL> create table table_hcc_archivelow COMPRESS FOR ARCHIVE LOW as
select * from tabla_normal;

Table created.

Elapsed: 00:00:06.46
```

```
16:36:50 SQL>
```

Y chequeamos el tamaño del resto de las tablas que hemos creado

```
16:37:12 SQL> start tamano_tablas.sql
Enter value for owner: TEST
old   9:    and s.owner like '&owner'
new   9:    and s.owner like 'TEST'
Enter value for table_name: TABL%
old  10: and t.table_name like '&table_name'
new  10: and t.table_name like 'TABL%'

Dueno                Nombre segmento                 Total Mb Compresion
-------------------- ------------------------------ ---------- ------------
TEST                 TABLE_HCC_ARCHIVELOW                 12.0 ARCHIVE LOW
TEST                 TABLE_HCC_QUERYHIGH                  12.0 QUERY HIGH
TEST                 TABLE_HCC_QUERYLOW                   24.0 QUERY LOW
TEST                 TABLE_COM_BASIC                      62.0 BASIC
TEST                 TABLE_COM_OLTP                       72.0 OLTP
TEST                 TABLA_NORMAL                        212.8

6 rows selected.

Elapsed: 00:00:00.02
```

Se puede apreciar que la compresión con ARCHIVE LOW es muy similar a la de QUERY HIGH, en cuanto a valores finales de compresión

3.3.4. ARCHIVE HIGH

Este método usa Bzip2 para comprimir, es el más alto grado de compresión existente, pero ocupa mucha CPU, puede llegar hasta 12x

Creamos una tabla con este tipo de compresión y veremos los resultados

```
17:27:43 SQL> create table table_hcc_archivehigh COMPRESS FOR ARCHIVE HIGH as
select * from tabla_normal;


Table created.

Elapsed: 00:01:00.39
```

Y chequemos el resto de las tablas

```
17:32:22 SQL> start tamano_tablas.sql
Enter value for owner: TEST
old   9:    and s.owner like '&owner'
new   9:    and s.owner like 'TEST'
Enter value for table_name: TABL%
old  10: and t.table_name like '&table_name'
```

```
new  10: and t.table_name like 'TABL%'

Dueno                Nombre segmento                 Total Mb Compresion
-------------------- ------------------------------ ---------- ------------
TEST                 TABLE_HCC_ARCHIVEHIGH                10.0 ARCHIVE HIGH
TEST                 TABLE_HCC_ARCHIVELOW                 12.0 ARCHIVE LOW
TEST                 TABLE_HCC_QUERYHIGH                  12.0 QUERY HIGH
TEST                 TABLE_HCC_QUERYLOW                   24.0 QUERY LOW
TEST                 TABLE_COM_BASIC                      62.0 BASIC
TEST                 TABLE_COM_OLTP                       72.0 OLTP
TEST                 TABLA_NORMAL                        212.8
```

En efecto , la compresión HCC con ARCHIVE HIGH , es la que produce mejores resultados en cuanto a compresión, pero … es la que tiene peores resultados en cuanto a tiempo, dado que se demoro muchísimo mas que las otras opciones.

4. Performance de las tablas comprimidas

Un factor a tener en cuenta , a parte de lo que es el tiempo para comprimir una tabla es la performance de estas tablas , a continuación se harán consultas a estas tablas y se verificará cuando es lo que le toma a Oracle llevar a cabo una descompresión de las tablas y entregar un resultado.

Para ello , lo que haremos es sacar estadísticas full sobre la tabla, le aplicamos este comando pues es uno de los que más trabajo intensivo sobre los bloques hace .

A continuación los resultados.

```
-
11:48:14 SQL> exec dbms_stats.gather_table_stats(ownname => 'TEST', tabname =>
'TABLE_HCC_ARCHIVEHIGH',estimate_percent => 100, method_opt => 'FOR TABLE FOR ALL
COLUMNS FOR ALL INDEXES', cascade => true );

PL/SQL procedure successfully completed.

Elapsed: 00:00:44.79


-
11:52:26 SQL> exec dbms_stats.gather_table_stats(ownname => 'TEST', tabname =>
'TABLE_HCC_ARCHIVELOW',estimate_percent => 100, method_opt => 'FOR TABLE FOR ALL
COLUMNS FOR ALL INDEXES', cascade => true );

PL/SQL procedure successfully completed.

Elapsed: 00:00:40.11


-
11:53:07 SQL> exec dbms_stats.gather_table_stats(ownname => 'TEST', tabname =>
'TABLE_HCC_QUERYHIGH',estimate_percent => 100, method_opt => 'FOR TABLE FOR ALL
COLUMNS FOR ALL INDEXES', cascade => true );

PL/SQL procedure successfully completed.

Elapsed: 00:00:39.77


-
11:58:54 SQL> exec dbms_stats.gather_table_stats(ownname => 'TEST', tabname =>
'TABLE_HCC_QUERYLOW',estimate_percent => 100, method_opt => 'FOR TABLE FOR ALL
COLUMNS FOR ALL INDEXES', cascade => true );


PL/SQL procedure successfully completed.

Elapsed: 00:00:40.37
```

```
-
12:00:16 SQL> exec dbms_stats.gather_table_stats(ownname => 'TEST', tabname =>
'TABLE_COM_BASIC',estimate_percent => 100, method_opt => 'FOR TABLE FOR ALL
COLUMNS FOR ALL INDEXES', cascade => true );


PL/SQL procedure successfully completed.

Elapsed: 00:00:43.48

-
12:01:52 SQL> exec dbms_stats.gather_table_stats(ownname => 'TEST', tabname =>
'TABLE_COM_OLTP',estimate_percent => 100, method_opt => 'FOR TABLE FOR ALL
COLUMNS FOR ALL INDEXES', cascade => true );

PL/SQL procedure successfully completed.

Elapsed: 00:00:42.25
```

Como podemos ver no hubo mayores variaciones para las tablas en sus distintos grados de compresión, a pesar de eso, se debe tener muy en cuenta la cantidad de filas de las tablas, para el caso de nuestros ejemplos tiene un poco menos de 1.900.000 filas, lo cual es muy reducido en comparación con tablas transaccionales que dentro de Exadata pueden alcanzar sin problemas los miles de millones

5. Comparación de resultados entre la creación y consulta

La siguiente tabla nos muestra de una manera tabular , la información que se desprende de la generación de las tablas , así como las consultas y actualizaciones masivas

Nota : Toda la información generada se realizó sobre una tabla de 1.9 millones de registros y todos los datos están expresados en segundos

Tipo compresión	Tiempo de creación	Tiempo de consulta (toma de estadísticas)
BASIC	00:05.21	00:43.48
OLTP	00:05.45	00:42.25
HCC QUERY LOW	00:03.17	00:40.37
HCC QUERY HIGH	00:05.58	00:39.77
HCC ARCHIVE LOW	00:06.46	00:40.11
HCC ARCHIVE HIGH	01:00.39	00:44.79

Se pueden apreciar variaciones en todas las compresiones, tanto al consultar, como lo más importante al crear la tabla

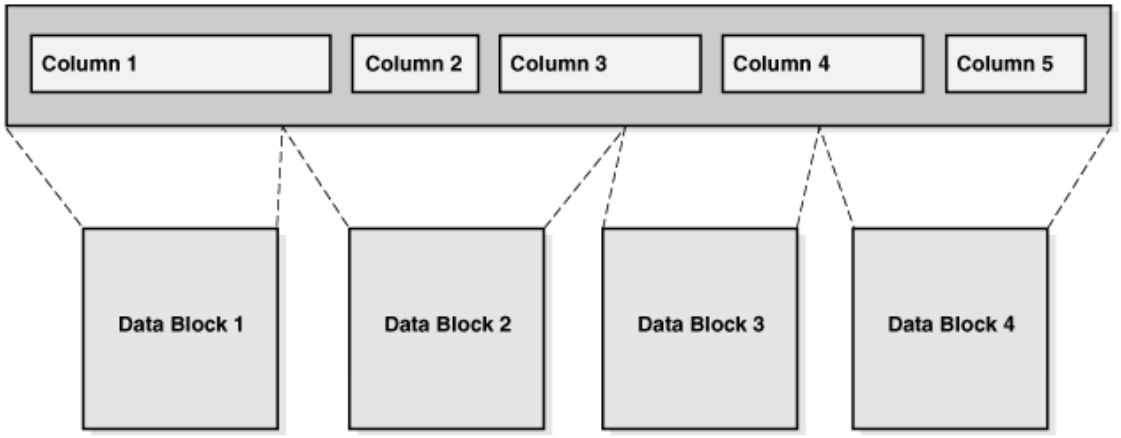
6. ¿Cómo almacena HCC la información?

HCC almacena la información de forma distinta a lo que lo hace un bloque común y corriente, como lo explicábamos en un principio , un bloque Oracle contiene las filas de una tabla y cada vez que se consulta por una fila que se encuentre dentro de un bloque, el bloque completo es llevado a memoria.

En cambio HCC almacena la información de la siguiente forma :

Los bloques Oracle son reorganizados en estructuras lógicas llamadas Compression Unit o CU , estas estructuras lógicas consisten de múltiples bloques Oracle , tantos como para juntar 32Kb o 64Kb.

Un ejemplo gráfico de CU, donde Oracle toma las columans existentes en los bloques de datos (4 de 8Kb) y las agrupa y comprime, se puede apreciar como dentro de la CU existen valores comprimidos para Column1, el cual es más grande que los valores comprimidos



Dentro de cada CU Oracle almacena la misma columna para un grupo determinado de filas, al almacenar la misma columna, lo que Oracle está haciendo es mejorar claramente las medidas de compresión sobre esos datos.

Por ejemplo, una tabla de empleados, Oracle toma las 50 primeras filas y saca todos los nombre de las columna NOMBRE y los comprime, después hace lo mismo con los rut, direcciones, etc y los va dejando en estos CU de 32Kb y 64Kb

De hecho se puede apreciar que Oracle almacena los valores, con orientación a las columnas ☺ Aunque esto no es tan así… pues al igual que los bloques comunes, también podemos leer una columna y el nos traerá la fila completa, por eso es el nombre que posee… compresión hibrida…

7. El impdp y el expdp

Al llevar a cabo un export Datapump de la tabla que se encuentra con compresión HCC, nos podemos fijar que el archivo resultante es mucho mayor que la tabla, pero si ese mismo respaldo , la generamos sobre la tabla de origen, que no tiene compresión HCC, nos fijaremos en algo curioso, ambos respaldos pesan lo mismo, pues las tablas se descomprimen para llevar la información al archivo dump

Export de una tabla con compresión HCC

```
[oracle@cltexdb01 ~]$ expdp test/test tables=table_hcc_with_dml
directory=data_pump_dir
Export: Release 11.2.0.2.0 - Production on Wed Mar 7 22:49:28 2012

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - 64bit
Production
With the Partitioning, Real Application Clusters, Automatic Storage Management,
OLAP,
Data Mining and Real Application Testing options
Starting "TEST"."SYS_EXPORT_TABLE_01":  test/******** tables=table_hcc_with_dml
directory=data_pump_dir
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 80 MB
Processing object type TABLE_EXPORT/TABLE/TABLE
. . exported "TEST"."TABLE_HCC_WITH_DML"                  181.6 MB 1894412 rows
Master table "TEST"."SYS_EXPORT_TABLE_01" successfully loaded/unloaded
******************************************************************************
Dump file set for TEST.SYS_EXPORT_TABLE_01 is:
  /u01/app/oracle/product/11.2.0/dbhome_CLCRME4/rdbms/log/expdat.dmp
Job "TEST"."SYS_EXPORT_TABLE_01" successfully completed at 22:49:42
```

Podemos chequear que el respaldo pesa poco más de 181Mb y la tabla pesa 80Mb

Export de una tabla sin compresión HCC

```
[oracle@cltexdb01 ~]$ expdp test/test tables=tabla_normal dumpfile=respaldo3.dmp
directory=data_pump_dir

Export: Release 11.2.0.2.0 - Production on Wed Mar 7 23:17:20 2012

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - 64bit
Production
With the Partitioning, Real Application Clusters, Automatic Storage Management,
OLAP,
Data Mining and Real Application Testing options
Starting "TEST"."SYS_EXPORT_TABLE_01":  test/******** tables=tabla_normal
dumpfile=respaldo3.dmp directory=data_pump_dir
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 212.7 MB
```

```
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
. . exported "TEST"."TABLA_NORMAL"                       183.1 MB 1894412 rows
Master table "TEST"."SYS_EXPORT_TABLE_01" successfully loaded/unloaded
******************************************************************************
Dump file set for TEST.SYS_EXPORT_TABLE_01 is:
  /u01/app/oracle/product/11.2.0/dbhome_CLCRME4/rdbms/log/respaldo3.dmp
Job "TEST"."SYS_EXPORT_TABLE_01" successfully completed at 23:17:28
```

Podemos verificar que el respaldo pesa poco más de 183Mb y la tabla pesa 212Mb

Export de una tabla con compresión HCC y con la claúsula COMPRESSION=ALL

```
[oracle@cltexdb01 ~]$ expdp test/test tables=table_hcc_with_dml
dumpfile=respaldo4.dmp directory=data_pump_dir compression=all

Export: Release 11.2.0.2.0 - Production on Wed Mar 7 23:23:06 2012

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - 64bit
Production
With the Partitioning, Real Application Clusters, Automatic Storage Management,
OLAP,
Data Mining and Real Application Testing options
Starting "TEST"."SYS_EXPORT_TABLE_01":  test/******** tables=table_hcc_with_dml
dumpfile=respaldo4.dmp directory=data_pump_dir compression=all
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 80 MB
Processing object type TABLE_EXPORT/TABLE/TABLE
. . exported "TEST"."TABLE_HCC_WITH_DML"                 21.74 MB 1894412 rows
Master table "TEST"."SYS_EXPORT_TABLE_01" successfully loaded/unloaded
******************************************************************************
Dump file set for TEST.SYS_EXPORT_TABLE_01 is:
  /u01/app/oracle/product/11.2.0/dbhome_CLCRME4/rdbms/log/respaldo4.dmp
Job "TEST"."SYS_EXPORT_TABLE_01" successfully completed at 23:23:23
```

Podemos verificar que el respaldo pesa poco más de 21Mb y la tabla pesa 80Mb

Export de una tabla sin compresión HCC y con la claúsula COMPRESSION=ALL

```
[oracle@cltexdb01 ~]$ expdp test/test tables=tabla_normal dumpfile=respaldo5.dmp
directory=data_pump_dir compression=all

Export: Release 11.2.0.2.0 - Production on Wed Mar 7 23:24:35 2012

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - 64bit
Production
```

```
With the Partitioning, Real Application Clusters, Automatic Storage Management,
OLAP,
Data Mining and Real Application Testing options
Starting "TEST"."SYS_EXPORT_TABLE_01":  test/******** tables=tabla_normal
dumpfile=respaldo5.dmp directory=data_pump_dir compression=all
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 212.7 MB
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
. . exported "TEST"."TABLA_NORMAL"                          20.80 MB 1894412 rows
Master table "TEST"."SYS_EXPORT_TABLE_01" successfully loaded/unloaded
******************************************************************************
Dump file set for TEST.SYS_EXPORT_TABLE_01 is:
  /u01/app/oracle/product/11.2.0/dbhome_CLCRME4/rdbms/log/respaldo5.dmp
Job "TEST"."SYS_EXPORT_TABLE_01" successfully completed at 23:24:45

[oracle@cltexdb01 ~]$
```

Podemos verificar que el respaldo pesa poco más de 21Mb y la tabla pesa 80Mb

Como se puede apreciar, el hecho de que las tablas estén con compresión HCC no afecta en nada al respaldo en cuanto a sus tamaños finales.

8. ¿Qué sucede cuando hago Update sobre una tabla HCC?

Cuando se produce una DML de actualización en el formato de filas y bloques antiguo (me refiero antiguo… a un formato normal) sólo se produce un bloqueo a nivel de fila, por eso el evento Row Lock , pero cuando se está trabajando con Exadata, al momento de hacer un update , Oracle se ve en la necesidad de hacer un bloqueo a nivel del CU (Compression Unit) , aunque sea un update a una sola fila.

Cuando se actualiza una fila dentro de una tabla con HCC, el registro se tiene que migrar a un nuevo bloque y este nuevo bloque está comprimido con formato OLTP . Lo gracioso del tema es que permanecen los 2 bloques, el original…donde estaba la fila , el cual va a contener un puntero hacía el nuevo bloque, donde se migra la fila modificada.

Por ende y poniendo atención al hecho de que habrán filas migradas, implicaría que al hacer un update sobre una tabla HCC…¿está crecería en tamaño? Pues la respuesta nefasta es sí..

He aquí un ejemplo :

Creamos una tabla con compresión HCC

```
00:29:54 SQL> create table table_hcc_with_DML COMPRESS FOR ARCHIVE HIGH as select
* from tabla_normal;

Table created.

Elapsed: 00:00:57.20
00:30:53 SQL>
```

Podemos ver el tamaño de la tabla

```
00:48:33 SQL> start tamano_tablas
Enter value for owner: TEST
old   9:    and s.owner like '&owner'
new   9:    and s.owner like 'TEST'
Enter value for table_name: %%
old  10: and t.table_name like '&table_name'
new  10: and t.table_name like '%%'

Dueno                Nombre segmento                  Total Mb Compresion
-------------------- ------------------------------ ---------- ------------
TEST                 TABLE_HCC_WITH_DML                   10.0 ARCHIVE HIGH
```

Incluso podemos ver la cantidad de bloques que posee

```
00:53:13 SQL> r
  1  select owner , segment_name , segment_type , bytes/1024/1024 Mb , blocks from dba_segments
  2* where segment_name like 'TABLE_HCC_WITH_DML'

OWNER                          SEGMENT_NAME             SEGMENT_TYPE               MB     BLOCKS
------------------------------ ------------------------ ------------------ ---------- ----------
TEST                           TABLE_HCC_WITH_DML       TABLE                      10       1280

Elapsed: 00:00:00.02
```

Sólo 1280 bloques utilizados

Se actualiza la tabla completa

```
00:53:59 SQL> update TABLE_HCC_WITH_DML set owner = 'TITO';

1894412 rows updated.

Elapsed: 00:01:43.24
```

Y vemos nuevamente cuanto pesa

```
00:55:59 SQL> 00:55:59 SQL> start tamano_tablas
Enter value for owner: TEST
old   9:    and s.owner like '&owner'
new   9:    and s.owner like 'TEST'
Enter value for table_name: TABLE_HCC_WITH_DML
old  10: and t.table_name like '&table_name'
new  10: and t.table_name like 'TABLE_HCC_WITH_DML'

Dueno                Nombre segmento                  Total Mb Compresion
-------------------- ------------------------------ ---------- ------------
TEST                 TABLE_HCC_WITH_DML                   80.0 ARCHIVE HIGH

Elapsed: 00:00:00.11
```

Subió inmediatamente de 10Mb a 80Mb , aunque es mucho menor a la tabla original

```
00:57:02 SQL> start tamano_tablas
Enter value for owner: TEST
old   9:    and s.owner like '&owner'
new   9:    and s.owner like 'TEST'
Enter value for table_name: TABLA_NORMAL
old  10: and t.table_name like '&table_name'
new  10: and t.table_name like 'TABLA_NORMAL'

Dueno                Nombre segmento                  Total Mb Compresion
-------------------- ------------------------------ ---------- ------------
TEST                 TABLA_NORMAL                        212.8
```

Y procedemos a verificar la cantidad de bloques usada por el segmento

```
00:58:50 SQL> select owner , segment_name , segment_type , bytes/1024/1024 Mb , blocks from dba_segments
00:58:54   2  where segment_name like 'TABLE_HCC_WITH_DML';

OWNER                          SEGMENT_NAME             SEGMENT_TYPE              MB     BLOCKS
------------------------------ ------------------------ ------------------ ---------- ----------
TEST                           TABLE_HCC_WITH_DML       TABLE                      80      10240

Elapsed: 00:00:00.02
```

Se puede verificar, que paso de tener 1280 bloques a poseer 10240, o sea, el salto cuantitativo fue gigante.

Todo lo anterior nos indica que obviamente hubo una migración de filas, por ende … el crecimiento en cuanto a Mb y bloques fue tan grande.

Para verificar que si hubo migración de filas, procedemos a consultar un registro en particular desde nuestra tabla

La función para construir el RowId antiguo fue sacada desde el blog de Kerry Osborne, he aquí el link
http://kerryosborne.oracle-guy.com/scripts/create_old_rowid.sql

Y he acá el código

```
create or replace function old_rowid (p_rowid rowid)
return varchar as

  rowid_type NUMBER;
  object_id NUMBER;
  fileno NUMBER;
  blockno   NUMBER;
  rowno  NUMBER;

BEGIN

   dbms_rowid.rowid_info(p_rowid, rowid_type, object_id, fileno, blockno, rowno);
/*
   dbms_output.put_line('Row Typ-' || TO_CHAR(rowid_type));
   dbms_output.put_line('Obj No-' || TO_CHAR(object_id));
   dbms_output.put_line('RFNO-' || TO_CHAR(fileno));
   dbms_output.put_line('Block No-' || TO_CHAR(blockno));
   dbms_output.put_line('Row No-' || TO_CHAR(rowno));
*/
return(to_char(fileno)||'.'||to_char(blockno)||'.'||to_char(rowno));

END;
/
```

Imaginemos que borramos la tabla y la volvemos a crear con formato HCC .

Seleccionamos una sola fila , aplicando la función old_rowid

```
SQL> select rowid, old_rowid(rowid) "Rowid antiguo" from table_hcc_with_DML
  where object_id = 8458

ROWID              Rowid antiguo
------------------ --------------------
AAAWKIAAEAAAMupB+n 4.52137.8103
```

Y el formato antiguo nos dice que la fila está en el file 4, bloque 52137 y en la fila 8103

Aplicamos nuestra actualización y realizamos la misma consulta

```
01:35:19 SQL> select rowid, old_rowid(rowid) "Rowid antiguo" from
table_hcc_with_DML
  where object_id = 8458
01:36:10   2  01:36:10   3  ;

ROWID              Rowid antiguo
------------------ --------------------
AAAWKIAAEAAAPkxAA2 4.63793.54
```

Con lo anterior verificamos que si hubo un movimiento de los datos de esa fila a un nuevo bloque, de hecho, podemos consultar por cualquiera de los RowId y obtendremos acceso a la misma fila ☺ ¿cómo es eso? Pues bien , recuerden que cuando se actualiza una fila en HCC, la fila es migrada, pero se genera un vínculo entre el bloque que contenía la fila y el bloque que la contiene, por eso es el crecimiento en bloques desde la tabla, al consultarla en la DBA_SEGMENTS.

RowId original : **AAAWKIAAEAAAMupB+n**
RowId de la fila migrada : **AAAWKIAAEAAAPkxAA2**

Si hacemos la consulta por cualquiera de los 2 tendremos acceso al mismo registro

```
01:37:36 SQL> select object_id from table_hcc_with_DML where rowid =
'AAAWKIAAEAAAMupB+n';

 OBJECT_ID
----------
      8458

Elapsed: 00:00:00.89
01:38:01 SQL> select object_id from table_hcc_with_DML where rowid =
'AAAWKIAAEAAAPkxAA2';

 OBJECT_ID
----------
      8458

Elapsed: 00:00:00.00
01:38:15 SQL>
```

9. Conclusiones

Se puede apreciar con todos los ejemplos mostrados que la compresión mediante HCC es poderosísima.

Todo esto se relaciona al nuevo mecanismo de compresión y a las unidades lógicas que almacenamiento que se generan a raíz de la agrupación de columnas para un grupo de filas, esto hace que la compresión sea más eficiente.

Todo esto aplica para tablas en donde la data no es modificada… o mejor dicho, la tasa de update es muy baja, dado que con las DMLs (update) se pasa de una compresión HCC a una compresión OLTP, además de generar encadenamiento de filas.

Cuando se hace la actualización de una fila, se bloquea toda la CU que almacena la fila, esto puede redundar en bloqueos de hasta 64Kb, que es la medida de un CU.

La compresión HCC puede ser definida a nivel de partición en una tabla y no hay problemas con los respaldos de una tabla HCC , pero el respaldo final pesa casi lo mismo que una tabla normal, dado que Oracle descomprime la tabla antes de respaldarla.