Oracle Database – Compression Best Practices

Cris Pedregal, CMTS
Product Management
Oracle Database Development
June, 2015
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Program Agenda

1. Types of Compression
2. DOs/DON’Ts of Compression
3. Enabling Compression
4. Heat Map/Automatic Data Compression
5. Using Compression Advisor
Types of Compression
Basic, Index, Advanced Row and Hybrid Columnar
## Data/Index Compression Techniques

<table>
<thead>
<tr>
<th>COMPRESSION TYPE:</th>
<th>SUITABLE FOR:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Compression</td>
<td>Read only tables and partitions in Data Warehouse environments or “inactive” data partitions in OLTP environments</td>
</tr>
<tr>
<td>Advanced Row Compression</td>
<td>Active tables and partitions in OLTP and Data Warehouse environments</td>
</tr>
<tr>
<td>Advanced LOB Compression and Deduplication</td>
<td>Non-relational data in OLTP and Data Warehouse environments</td>
</tr>
<tr>
<td>Advanced Network Compression and Data Guard Redo Transport Compression</td>
<td>All environments</td>
</tr>
<tr>
<td>RMAN/Data Pump Backup Compression</td>
<td>All environments</td>
</tr>
<tr>
<td>Index Compression</td>
<td>Indexes on tables for OLTP and Data Warehouse</td>
</tr>
<tr>
<td>Hybrid Columnar Compression – Query Level</td>
<td>Query mostly tables and partitions in OLTP and Data Warehouse environments</td>
</tr>
<tr>
<td>Hybrid Columnar Compression – Archive Level</td>
<td>“Inactive” data tables/partitions in OLTP and Data Warehouse environments</td>
</tr>
</tbody>
</table>
Basic Compression

- **Introduced in Oracle Database 9i Release 2**
  - Compression during bulk load operations only (Direct Load, CTAS...)
  - Data modified using conventional DML not compressed
    - Modified data will degrade compression ratio over time unless table is recompressed

- **Ideal for “DW” data – not intended for use with active data**
  - Use for Data Warehouse applications
  - *Customers report 2x to 4x compression ratios*
  - Included with Oracle Database Enterprise Edition (no additional license required)
Advanced Row Compression

- Ideal for “ACTIVE” data -- can be used for cold/inactive also
  - Use for both OLTP and Data Warehouse applications
  - Customers report 2x to 4x compression ratios

- Balances Good Compression Ratios and Performance
  - Maintains compression across all DML operations and bulk load operations
  - Does not require data to be uncompressed – keeps data compressed in memory
  - Reads often see improved performance due to fewer I/Os and enhanced memory efficiency
  - Not as aggressive as HCC in terms of compression ratios
Advanced LOB Compression/Deduplication

• **LOBS typically experience a reduction of 2x to 3x times in size**
  - Automatically avoids compressing data that would not benefit from compression
    - LOBS already compressed (by 3rd party application) may not compress any further
    - Typically provides compression when inline LOB compression does not
  - Useful for **content management, email applications and data archival** applications
  - Included with Advanced Compression option
    - Compression Advisor with Oracle Database 12c works with LOBS

• **No adverse impact on read operations**
  - Often improves read performance for cache data

• **Enables storage of a single physical image for duplicate data**
  - Significantly reduces space consumption
  - Dramatically improves writes and copy operations
Prefix (Index) Compression

- **Included with Oracle Database Enterprise Edition**
  - No additional license required
  - Can compress just indexes and not data, or can compress both

- **Customers report 2x compression is typical**
  - Compression ratio depends on how many columns are selected/compressibility of those columns

- **ANALYZE INDEX will give advice on whether / how many columns to choose**
  - Index data is **NOT decompressed when read from disk into memory**
Advanced Index Compression

• **Advanced Index Compression automatically chooses the right compression per block**
  – No user decision/analysis required

• **Average compression ratio for indexes is 3x**
  – Little or no discernible overhead
  – May see compression where Prefix Compression did not – avoids compressing indexes that don’t provide optimal compression
  – Requires Advanced Compression option license

• **Compression Advisor extended to provide estimated compression ratio**
Hybrid Columnar Compression (HCC)

• **Ideal for inactive/cold data**
  – Use with OLTP/Data Warehouse applications (data with no/few modifications)
  – DML INSERTS/UPDATES could reduce compression ratio
  – *Customers report 6x to 50x compression ratios*
  – HCC Compressed data stays compressed in memory

• **Compression Levels Balance Query Performance and Storage Reduction**
  • **QUERY** compression level optimized to increase scan *query performance* –
    6x to 10x compression ratios typical
  • **ARCHIVE** compression best approach for long term data management and data *archival* –
    10x to 15x compression ratios typical

*HCC Requires: Exadata, SuperCluster, Pillar Axiom or ZFSSA storage*
DOs/DON’Ts of Compression

Best Practice Suggestions
Compression DOs

• **Compress all tables/partitions, except...**
  – The general recommendation is to compress all the tables in the database
    – **Exception:** If the table is used as a queue, i.e. rows are inserted into the table, then later most or all of the rows are deleted, then more rows are inserted then deleted... (don’t compress this table)

• **Test environment = production environment!**
  – The best test environment for each Advanced Compression capability is where you can most closely duplicate the production environment – this will provide the most realistic (pre- and post- compression) performance and functionality comparisons

• **The greater duplication the greater the compression**
  – Space usage reduction gives the best results where the most duplicate data is stored (low cardinality)
    – **Sorting data** (on the columns with the most duplicates) may increase the compression ratio
    – **Larger block sizes** may provide higher compression ratios – always test first
Compression DOs

• **Tablespace level compression (vs. table/partition)**
  - For custom applications, we recommend compressing at the Tablespace level, but users should consider turning off compression on very high traffic or very small tables, such as tables used as queues.
  - For commercial packaged applications, where typically the number of objects can be very large, the recommended approach is **object selection instead of exclusion**
    - Often the largest tables and indexes consume the majority of the database space -- compress those objects while excluding high traffic objects like tables used as queues.

• **Regarding overhead.... *Don’t be CPU bound before compression***
  - CPU overhead is typically minimal (<5% CPU), implementing Advanced Row Compression will have additional, although minor overhead for some DML operations.
  - Compression best suited for systems where there are “spare” CPU cycles.

• **Use SecureFiles for unstructured data**
  - Storing LOBS in SecureFiles allows you to use SecureFiles compression and deduplication.
Compression DOs

• Use Direct-path vs. Conventional path for bulk loads
  – Direct path is optimized for compression (provides better performance):
    – When performing Inserts, specify `insert /*+ append */` for better performance
    – Use direct-path loads (CTAS, Insert as Select...) when possible

• Works best with tablespace encryption
  – With tablespace encryption, table and index compression is done before encryption
    – With column encryption, compression is done after encryption -- compression will have minimal effectiveness on encrypted columns

• See these MOS notes....
  – Advanced Compression Master Note (Doc ID 1223705.1)
  – How to compress a table that is online (Doc ID 1353967.1)
  – Advanced Compression critical patches (Doc ID 1061366.1)
  – Redo Transport compression with Data Guard (Doc ID 729551.1)
  – How to see if rows are compressed in a table (Doc ID 1477918.1)
Compression DON’Ts

• Unsupported Datatypes
  – Basic, Advanced Row Compression and HCC do NOT support LONG data types

• Don’t use HCC with heavily modified tables/partitions
  – Hybrid Columnar Compression best suited for tables that are not modified, or are lightly modified. DML inserts/updates will be at a lower compression ratio than data that is bulk loaded
    – Can recompress HCC table to regain degraded compression ratio

• Advanced Compression/HCC is for data only
  – IOT's are essentially indexes, so they can't be compressed with HCC, Advanced Row or Basic Compression
    – Prefix compression can be used with IOT's (but not with Advanced Index Compression)
    – Indexes are compressed separately from the data
Enabling Compression

Best Practice Migration Suggestions
The following example of a partial `CREATE TABLE` statement specifies advanced row compression for one partition and basic table compression for the other partition:

```sql
CREATE TABLE sales (  
  prod_id NUMBER NOT NULL,  
  cust_id NUMBER NOT NULL, ...  
)  
PARTITION BY RANGE (time_id)  
  (  
    partition sales_2013 VALUES LESS THAN (TO_DATE('...')) ROW STORE COMPRESS BASIC,  
    partition sales_2014 VALUES LESS THAN (MAXVALUE) ROW STORE COMPRESS ADVANCED  
  );
```

### Table Compression Method

<table>
<thead>
<tr>
<th>Table Compression Method</th>
<th><code>CREATE/ALTER TABLE</code> Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic table compression</td>
<td><code>ROW STORE COMPRESS [BASIC]</code></td>
</tr>
<tr>
<td>Advanced row compression</td>
<td><code>ROW STORE COMPRESS ADVANCED</code></td>
</tr>
<tr>
<td>Warehouse compression (Hybrid Columnar Compression)</td>
<td>`COLUMN STORE COMPRESS FOR QUERY [LOW</td>
</tr>
<tr>
<td>Archive compression (Hybrid Columnar Compression)</td>
<td>`COLUMN STORE COMPRESS FOR ARCHIVE [LOW</td>
</tr>
</tbody>
</table>
ALTER TABLE .... MOVE (existing tables)

- Enables Advanced Row Compression for future DML and also compresses existing data
- Table is locked during MOVE operation – queries will run, all DML will be blocked
  - Run in parallel for best performance
  - Indexes on the partition or table will be invalidated; those indexes will need to be rebuilt
- ALTER TABLE... MOVE PARTITION with the UPDATE INDEXES clause
  - Will maintain indexes
  - All DML will be blocked until the move command completes
  - Not available for non-partitioned tables
- ALTER TABLE ... MOVE PARTITION ONLINE
  - Allows DML operations to continue to run uninterrupted on the partition that is being moved
  - Global indexes maintained during the move partition operation
- ALTER TABLE...MOVE ROW STORE COMPRESS ADVANCED creates new extents for the compressed data
  - The positioning of the new segment can be anywhere within the data file
  - When the original segment is released it may or may not be possible to shrink the data file
Online Redefinition (existing tables)

- Enables Advanced Row Compression for future DML and also compresses existing data
- Using DBMS_REDEFINITION keeps the table online for both read/write activity during the migration
  - Run DBMS_REDEFINITION in parallel for best performance
- Online redefinition clones the indexes to the interim table during the operation
  - All cloned indexes are incrementally maintained during the sync (refresh) operation
  - No interruption in the use of the indexes during, or after, the online redefinition
  - Exception when online redefinition is used for redefining a partition: all global indexes are invalidated and need to be rebuilt after the online redefinition

- See MOS Note (Doc ID 1353967.1) for Online Redefinition usage examples...
Heat Map/Automatic Data Optimization

Smart Compression – New in Oracle Database 12c
Heat Map

• Heat Map gives you a detailed view of how your data is being accessed, and how access patterns are changing over time

• Programmatic access to Heat Map data is available through a set of PL/SQL table functions, as well as through data dictionary views
  – Data modification times are tracked at the row level and aggregated to the block level
  – Modification times, full table scan times, and index lookup times are tracked at the segment level
Automatic Data Optimization (ADO)
Usage Based Data Compression

**Hot Data**
- Advanced Row Compression: 3X
  - 01110101010010
  - 10000100010101
  - 01011100001010

**Warm Data**
- Columnar Query Compression: 10X
  - 1010101011010100101101110001010010
  - 111101001101001000110000101011001

**Archive Data**
- Columnar Archive Compression: 15X
  - 10101001011010011001101010001000110
  - 11010101011010011100101010101010110
  - 1110010100100101001010110111011010

---

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.
### Automatic Data Optimization (new/existing tables)

**ALTER TABLE** `sales` ILM add

<table>
<thead>
<tr>
<th>Access Level</th>
<th>Description</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Active</strong></td>
<td>- Advanced Row Compressed (2-4x) &lt;br&gt;- Affects ONLY Candidate Rows &lt;br&gt;- Cached in DRAM &amp; FLASH</td>
<td>row store compress advanced row after 2 days of no modification</td>
</tr>
<tr>
<td><strong>Frequent Access</strong></td>
<td>- HCC Query Compressed (10x) &lt;br&gt;- High Performance Storage</td>
<td>compress for query low after 1 week of no modification</td>
</tr>
<tr>
<td><strong>Occasional Access</strong></td>
<td>- Advanced/HCC Compressed &lt;br&gt;- Low Cost Storage</td>
<td>tier to low_cost_storage Tablespace (Must be Exadata/ZFSSA/Axiom/FS1 for HCC)</td>
</tr>
<tr>
<td><strong>Dormant</strong></td>
<td>- HCC Archive Compressed (15-50X) &lt;br&gt;- Archival Storage</td>
<td>compress for archive high after 6 months no access</td>
</tr>
</tbody>
</table>
As data ages:
- Activity declines
- Volume grows
- Older data primarily for reporting

```
alter table ...
add policy
... compress for query
after 3 months of no modification

... compress for archive
after 1 year of no modification
```

As data cools down, Automatic Data Optimization automatically converts data to columnar compressed
Using Compression Advisor

DBMS_COMPRESSION
DBMS_COMPRESSION

– See MOS Note (Doc ID 1284972.1) for Compression Advisor usage examples...
– See Oracle Database documentation (search for DBMS_COMPRESSION)
  – 12c Advisor now includes LOBS and Advanced Index Compression

```sql
set serveroutput on
DECLARE
  blkcnt_cmp pls_integer;
  blkcnt_uncomp pls_integer;
  row_cmp pls_integer;
  row_uncomp pls_integer;
  cmp_ratio pls_integer;
  comp_type_str varchar2(100);
BEGIN
  DBMS_COMPRESSION.GET_COMPRESSION_RATIO ('USERS', 'SH', 'SALES', NULL, DBMS_COMPRESSION.COMP_FOR_OLTP, blkcnt_cmp, blkcnt_uncomp, row_cmp, row_uncomp, cmp_ratio, comp_type_str);
  DBMS_OUTPUT.PUT_LINE('Block count compressed - ' || blkcnt_cmp);
  DBMS_OUTPUT.PUT_LINE('Block count uncompressed - ' || blkcnt_uncomp);
  DBMS_OUTPUT.PUT_LINE('Row count per block compressed - ' || row_cmp);
  DBMS_OUTPUT.PUT_LINE('Row count per block uncompressed - ' || row_uncomp);
  DBMS_OUTPUT.PUT_LINE('Compression type - ' || comp_type_str);
  DBMS_OUTPUT.PUT_LINE('Compression ratio = ' || blkcnt_uncomp/blkcnt_cmp || ' to 1');
  DBMS_OUTPUT.PUT_LINE('Compression ratio orig = ' || cmp_ratio);
END;
/```
Advisor Results

SAMPLE OUTPUT FOR COMPRESSION RATIO FOR THE WHOLE TABLE OF SH.SALES

SYS@dw23> set serveroutput on
SYS@dw23> DECLARE
  2  blkcnt_cmp pls_integer;
  3  blkcnt_uncmp pls_integer;
  4  row_cmp pls_integer;
  5  row_uncmp pls_integer;
  6  cmp_ratio pls_integer;
  7  comp_type_str varchar2(100);
  8 BEGIN
  9  DBMS_COMPRESSION.GET_COMPRESSION_RATIO ('USERS', 'SH', 'SALES', '', DBMS_COMPRESSION.COMP_FOR OLTP, blkcnt_cmp, blkcnt_uncmp, row_cmp, row_uncmp, cmp_ratio, comp_type_str);
 10  DBMS_OUTPUT.PUT_LINE('Block count compressed = ' || blkcnt_cmp);
 11  DBMS_OUTPUT.PUT_LINE('Block count uncompressed = ' || blkcnt_uncmp);
 12  DBMS_OUTPUT.PUT_LINE('Row count per block compressed = ' || row_cmp);
 13  DBMS_OUTPUT.PUT_LINE('Row count per block uncompressed = ' || row_uncmp);
 14  DBMS_OUTPUT.PUT_LINE('Compression type = ' || comp_type_str);
 15  DBMS_OUTPUT.PUT_LINE('Compression ratio = ' || blkcnt_uncmp/blkcnt_cmp || ' to 1');
 16  DBMS_OUTPUT.PUT_LINE('Compression ratio orig= ' || cmp_ratio);
 17  END;
 18 /
Block count compressed = 161
Block count uncompressed = 427
Row count per block compressed = 555
Row count per block uncompressed = 208
Compression type = "Compress For OLTP"
Compression ratio = 2.5217391304347826086856521739130434783 to 1
Compression ratio orig= 3

PL/SQL procedure successfully completed.
About DBMS_COMPRESSION

- **Free to use, no additional license required**
- **Use with Oracle Database 11g Release 2 and later**
  - Use Advisor version on OTN Advanced Compression web page for Oracle Database 9i thru Oracle Database 11g Release 1:
- **Use for Hybrid Columnar Compression ratios on any Oracle Database platform**
  - Very accurate, results can be used for planning purposes
Program Agenda

1. Types of Compression
2. DOs/DON’Ts of Compression
3. Enabling Compression
4. Heat Map/Automatic Data Compression
5. Using Compression Advisor
Resources

Advanced Compression on OTN

Heat Map, ADO, and Information Lifecycle Management on OTN

Related Sessions at Oracle OpenWorld 2014 (San Francisco):
https://blogs.oracle.com/DBStorage/entry/open_world_sessions

Product Manager: gregg.christman@oracle.com