ORACLE DATABASE 12c

Automatic Data Optimization with Oracle Database 12*c*

ORACLE WHITE PAPER | APRIL 2017

ORACLE

Table of Contents

Disclaimer

Introduction

The amount of data that enterprises are storing and managing is growing rapidly - various industry estimates indicate that data volume is doubling every 2-3 years. The rapid growth of data presents daunting challenges for IT, both in cost and performance.

Although the cost of storage keeps declining, fast-growing data volumes make storage one of the costliest elements of most IT budgets. In addition, the accelerating growth of data makes it difficult to meet performance requirements while staying within budget.

Information Lifecycle Management (ILM) is intended to address these challenges by storing data in different storage and compression tiers, according to the enterprise's current business and performance needs. This approach offers the possibility of optimizing storage for both cost savings and maximum performance.

In Oracle Database 12*c*, Heat Map and Automatic Data Optimization were introduced as features of Oracle Advanced Compression. *Heat Map* automatically tracks modification and query timestamps at the row and segment levels, providing detailed insights into how data is being accessed. *Automatic Data Optimization* (ADO) automatically moves and compresses data according to user-defined policies based on the information collected by Heat Map.

Heat Map and ADO make it easy to use existing innovations in Oracle Database compression and partitioning technologies, which help reduce the cost of managing large amounts of data, while also improving application and database performance. Together these capabilities help to implement first-class Information Lifecycle Management in Oracle Database.

Oracle Database 12*c* Release 2 (12.2) is available in the cloud, with Oracle Cloud at Customer, and on-premises.

Storage Tiering and Compression Tiering

An enterprise (or even a single application) does not access all its data equally: the most critical or frequently accessed data will need the best available performance and availability. To provide this best access quality to all the data would be costly, inefficient and is often architecturally impossible. Instead, IT organizations implement *storage tiering*.

With storage tiering, organizations can deploy their data on different tiers of storage so that less-accessed ("colder") data is migrated away from the costliest and fastest storage. The colder data is still available, but at slower speeds – the effect on the overall application performance is minimal, due to the rarity of accessing colder data. Colder data may also be compressed to a greater level (compared to active data) in storage. We use the term Information Lifecycle Management (ILM)[1] to name the managing of data from creation/acquisition to archival or deletion.

Figure 1 indicates that the most active data located on a high performance tier and the less active data/historical data on lower-cost tiers. In this scenario, the business is meeting all of its performance, reliability, and security requirements, but at a significantly lower cost than in a configuration where all data is located on high performance (tier 1) storage. The illustration shows that a higher level of compression can be applied to the less active and historical storage tiers, further improving the cost savings while also improving performance for queries that scan the less active data.

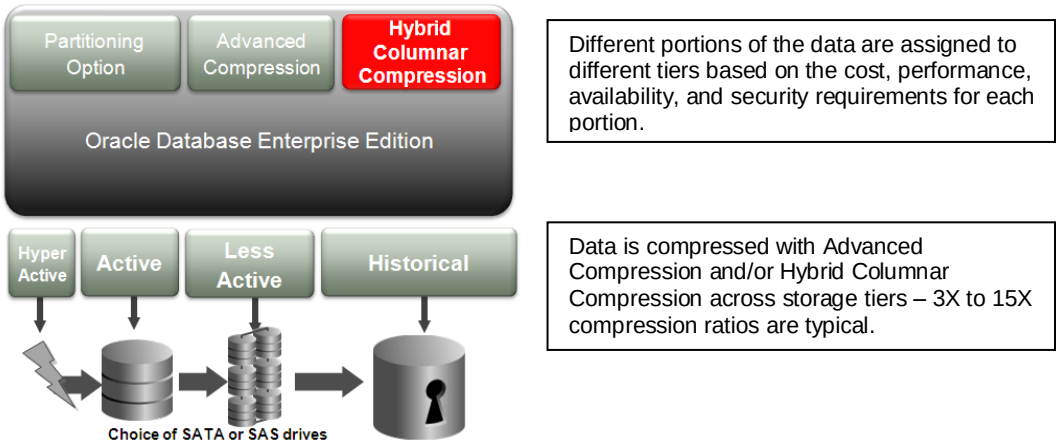

Figure 1 - Partitioning, Advanced Compression and Hybrid Columnar Compression.

In addition to storage tiering, it is also possible to use different types of compression to suit different access patterns. For example, colder data can be compressed at a higher level at the cost of slower access. Oracle Database provides several types of compression that can be utilized as data moves through its lifecycle - from hot/active to warm/less active to cold/historical – while meeting the performance and availability requirements for the application. We call this *compression tiering*.

[1] See for example Gartner's definition of ILM. Storage tiering, the practice of allocating different data to different levels of storage service, is one of the tools used for ILM.

Even with the right storage and compression capabilities, deciding which data should reside where and when to migrate data from one tier to another remains a serious challenge. Oracle Database 12*c* addresses this challenge with functionality that automatically discovers data access patterns – Heat Map – and uses Heat Map information to automatically optimize data organization – Automatic Data Optimization. The rest of this document explains the Oracle Database technologies that enable storage and compression tiering, and how to use them to support Information Lifecycle Management.

Heat Map -- Fine Grained Data Usage Tracking

Heat Map, a feature in Oracle Advanced Compression, automatically tracks table/partition usage information at the row and segment levels.[2] Data modification times are tracked at the row level and aggregated to the block level, and modification times, full table scan times, and index lookup times are tracked at the segment level. 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.

Figure 2 shows one way to depict Heat Map data. Each box represents one partition of a table. The size of the box is the relative size of the partition, and the color represents how "hot" (i.e., frequently accessed) the partition is based on the most recent access to any row in the partition.

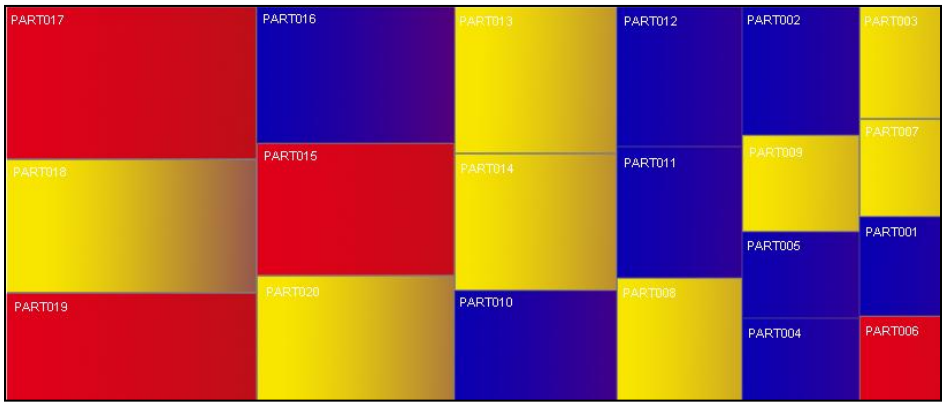

Figure 2 - Heat Map data for access patterns to a partitioned table

Automatic Data Optimization

Automatic Data Optimization (ADO) allows you to create policies for data compression (Smart Compression) and data movement, to implement storage and compression tiering. Smart Compression refers to the ability to utilize Heat Map information to associate compression policies, and compression levels, with actual data usage.

Oracle Database periodically evaluates ADO policies, and uses the information collected by Heat Map to determine when to move and / or compress data. All ADO operations are executed automatically and in the background, without user intervention.

[2] Database rows are stored in database blocks, which are grouped in extents. A segment is a set of extents that contains all the data for a logical storage structure within a tablespace, i.e. a table or partition.

ADO policies can be specified at the segment or row level for tables and/or partitions. Policies will be evaluated and executed automatically in the background during the maintenance window. ADO policies can also be evaluated and executed anytime by a DBA, manually or via a script.

ADO policies specify *what* conditions (of data access) will initiate an ADO operation – such as ***no access***, or ***no modification***, or ***creation time*** – and *when* the policy will take effect – for example, after "*n"* days or months or years. It is important to note that ADO "conditions" cannot be mixed. For example, if the first ADO condition, on a table/partition, uses the condition "no modification", then the other conditions used for the same table/partition must use the same condition type.

Conditions in ADO policies are not limited to Heat Map data: you can also create custom conditions using PL/SQL functions, extending the flexibility of ADO to use your own data and logic to determine when to move or compress data.

The next examples use the "best practice" approach of compressing using both Advanced Row Compression and Hybrid Columnar Compression. While compression tiering best practice does include the use of HCC, if an organization doesn't have access to HCC, then they would use only Advanced Row Compression in their ADO policies.

Automatic Data Optimization Examples

The following examples assume there is an orders table containing sales orders, and the table is range partitioned by order date.

In the first example, a segment-level ADO policy is created to automatically compress partitions using Advanced Row Compression after there have been no modifications for 30 days. This will automatically reduce storage used by older sales data, as well as improve performance of queries that scan through large numbers of rows in the older partitions of the table.

```
ALTER TABLE orders ILM ADD POLICY
ROW STORE COMPRESS ADVANCED SEGMENT
AFTER 30 DAYS OF NO MODIFICATION;
```

In the following example, a segment-level ADO policy is created to automatically compress the partition, to a higher compression level, using Hybrid Columnar Compression after there have been no modifications for 90 days. This makes sense when HCC is available, and when the data will no longer be updated, but will continue to be queried; moving to HCC will save a lot of storage AND give a big boost to query performance.

```
ALTER TABLE orders ILM ADD POLICY
COLUMN STORE COMPRESS FOR QUERY HIGH SEGMENT
AFTER 90 DAYS OF NO MODIFICATION;
```

Sometimes it is necessary to load data at the highest possible speed, which requires creating the partition without any compression enabled. This allows the post-load activity to subside on the table before compression is implemented. For organizations with SLA's around the load times, this allows the table to be created and populated as quickly as possible, before implementing compression.

In the next example, an organization wants the benefits of compression, but also needs to ensure SLA requirements are met. It would be beneficial to compress the data, in the partition, on a more granular basis (block by block instead of the entire segment). You can create row-level ADO policies to automatically

compress blocks in the partition after no row, in a given block, has been modified for at least 3 days. With row-level policies, all rows in the block must meet the policy conditions before the block is compressed.

Below is an example ADO policy where rows are inserted uncompressed, and then later moved to Advanced Row Compression (row-level policies can also be specified for Hybrid Columnar Compression) on a per-block basis. Note that this policy uses the ROW keyword instead of the SEGMENT keyword.

```
ALTER TABLE orders ILM ADD POLICY
ROW STORE COMPRESS ADVANCED ROW
AFTER 3 DAYS OF NO MODIFICATION;
```

With the above policy in place, Oracle Database will evaluate blocks during the maintenance window, and any blocks that qualify will be compressed, freeing up space for new rows as they are inserted. This allows you to achieve the highest possible performance for data loads, but also get the storage savings and performance benefits of compression without having to wait for an entire partition to be ready for compression.

In addition to Smart Compression, ADO policy actions include data movement to other storage tiers, including lower cost storage tiers or storage tiers with other compression capabilities such as Oracle' s Hybrid Columnar Compression (HCC).[3]

ADO-based storage tiering is not based upon the ADO condition clause (i.e. after "x" days of NO MODIFICATION) as is compression tiering and instead, is based upon tablespace space pressure. The justification for making storage tiering dependent on "space pressure" is exactly as you might imagine, the belief that organizations will want to keep as much data as possible on their high performance (and most expensive) storage tier, and not move data to a lower performance storage tier until it is absolutely required.

The value for the ADO parameter **TBS_PERCENT_USED** specifies the percentage of the tablespace quota when a tablespace is considered full. The value for **TBS_PERCENT_FREE** specifies the targeted free percentage for the tablespace. When the percentage of the tablespace quota reaches the value of TBS_PERCENT_USED, ADO begins to move segments so that percent free of the tablespace quota approaches the value of TBS_PERCENT_FREE. This action by ADO is a best effort and not a guarantee.

You can set ILM ADO parameters with the CUSTOMIZE_ILM procedure in the DBMS_ILM_ADMIN PL/SQL package, for example:

```
BEGIN
DBMS_ILM_ADMIN.CUSTOMIZE_ILM(DBMS_ILM_ADMIN.TBS_PERCENT_USED, 85):
DBMS_ILM_ADMIN.CUSTOMIZE_ILM(DBMS_ILM_ADMIN.TBS_PERCENT_FREE, 25):
END;
```

In the example above, when a tablespace reaches the fullness threshold (85%) defined by the user, the database will automatically move the coldest table/partition(s) in the tablespace to the target tablespace until the tablespace quota has at least 25 percent free. This only applies to tables and partitions that have a "TIER TO" ADO policy defined. This frees up space on your tier 1 storage for the segments that would truly benefit from the performance while moving colder segments, that don't need Tier 1 performance, to lower cost Tier 2 storage.

[3] HCC requires the use of Exadata, SuperCluster, Pillar Axiom, Sun ZFS Storage Appliance or FS1

In the following example, a tablespace-level ADO policy automatically moves the partition to a different tablespace when the current tablespace runs low on space. The “tier to” keywords indicate that data will be moved to a new tablespace when the current tablespace becomes too full. The “low_cost_store” tablespace was created on a lower cost storage tier.

```
ALTER TABLE orders ILM ADD POLICY tier to low_cost_store;
```

ADO storage tiering operates at the segment level, so when an ADO policy implements storage tiering the entire segment is moved and this movement is one direction, meaning that ADO storage tiering is meant to move colder segments from high performance storage to slower, lower cost storage. If an ADO compression policy AND a storage tiering policy both qualify, the database will execute both in a single segment reorganization step.

All of this storage tiering behavior can be overridden with custom conditions on "TIER TO" policies: the DBA can write their own conditions using PL/SQL, and implement segment movement based on any conditions they want to encode.

Another option when moving a segment to another tablespace is to set the target tablespace to READ ONLY after the object is moved. This is beneficial for historical data and during backups, since subsequent RMAN full database backups will skip READ ONLY tablespaces.

Automatic Data Optimization for OLTP

The previous examples show individual ADO policies that implement one action –compression tiering (Smart Compression) or storage tiering. The following example shows how to combing multiple ADO policies for an OLTP application.

In OLTP applications, you should use Advanced Row Compression for the most active tables/partitions, to ensure that newly added or updated data will be compressed as DML operations are performed against the active tables/partitions.

For cold or historic data within the OLTP tables, use either Warehouse or Archive Hybrid Columnar Compression. This ensures that data which is infrequently or never updated is compressed to the highest levels – compression ratios of 6x to 15x are typical with Hybrid Columnar Compression, whereas 2x to 4x compression ratios are typical with Advanced Row Compression.

For example, see Figure 3:

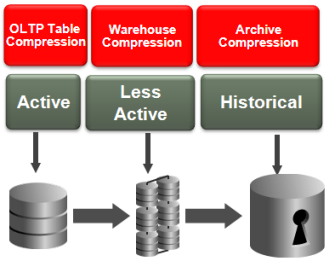

Figure 3 - Advanced Row Compression, Hybrid Columnar Compression and tiering.

To implement this approach with ADO, use the following policies:

```
ALTER TABLE orders ILM ADD POLICY
COLUMN STORE COMPRESS FOR QUERY HIGH SEGMENT
AFTER 30 DAYS OF NO MODIFICATION;

ALTER TABLE orders ILM ADD POLICY
COLUMN STORE COMPRESS FOR ARCHIVE HIGH SEGMENT
AFTER 90 DAYS OF NO MODIFICATION;

ALTER TABLE orders ILM ADD POLICY tier to low_cost_store;
```

In this example of Smart Compression and storage tiering, we assume that the orders partition is defined with Advanced Row Compression enabled, so that rows are compressed at that level when they are first inserted.

Oracle Database will automatically evaluate the ADO policies to determine if the partition is eligible to be moved to a higher compression level, and if the partition is eligible to be moved to a lower cost storage tier. As discussed earlier, storage tiering is primarily triggered when the current tablespace becomes too full, but can be customized to occur based on user-defined conditions.

The capabilities of Heat Map and ADO in Oracle Database 12c make it easy for DBAs to implement ILM for OLTP applications, and enable the use of HCC with OLTP data. With HCC, DBAs can significantly reduce the amount of storage space used by colder OLTP data, while increasing the performance of reports and analytics.

Automatic Data Optimization for Data Warehousing

In data warehousing applications on Exadata storage, or on Oracle Storage that supports HCC, HCC Warehouse Compression should be used for heavily queried tables/partitions. For cold or historic data within the data warehousing application, using Archive Compression ensures that data which is infrequently accessed is compressed to the highest level – compression ratios of 15x to 50x are typical with Archive Compression.

For example, see Figure 4:

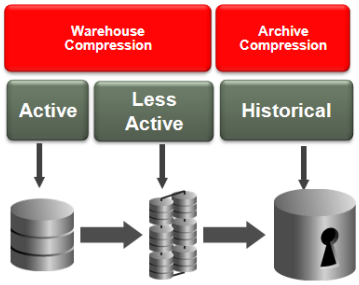

Figure 4 - Partitioning and Hybrid Columnar Compression.

To implement this approach with ADO, use the following statements:

```
ALTER TABLE orders ILM ADD POLICY
COLUMN STORE COMPRESS FOR ARCHIVE HIGH SEGMENT
AFTER 90 DAYS OF NO MODIFICATION;

ALTER TABLE orders ILM ADD POLICY tier to lessactivetbs;
```

In this example, we assume that the orders table is defined with Warehouse Compression enabled, so that rows are compressed at that level when they are first inserted.

Although this example implemented HCC compression at the segment-level, it should be noted that Automatic Data Optimization also supports Hybrid Columnar Compression (HCC) for row-level policies. Rows from cold blocks can be HCC-compressed even if there is DML insert activity on other parts of the table or partition. Hybrid Columnar Compression (HCC) can be used during array inserts into tables. This means that the SQL INSERT statement, without the APPEND hint, can use HCC (without degrading the compression level), and array inserts from programmatic interfaces such as PL/SQL and the Oracle Call Interface (OCI) can use HCC. ADO policies for HCC are now effective even if changes are constantly being made to a small subset of the table or partition (updates will still be performed uncompressed and could impact the HCC compression level).

Oracle Database will automatically evaluate the ADO policies to determine if the partition is eligible to be moved to a higher compression level, and if the partition is eligible to be moved to a different tablespace. As with the previous Smart Compression example for OLTP, the automatic capabilities of ADO in Oracle Database 12c make it simple and easy for DBAs to implement ILM for Data Warehousing, and significantly reduce the amount of time and effort DBAs need to spend optimizing storage usage and storage performance.

Conclusion

Information Lifecycle Management enables organizations to understand how their data is accessed over time, and manage the data accordingly. However, most ILM solutions for databases lack two key capabilities – automatic classification of data, and automatic data compression and movement across storage tiers.

The Heat Map and Automatic Data Optimization features of Oracle Advanced Compression support comprehensive and automated ILM solutions that minimize costs while maximizing performance. In combination with the comprehensive compression features in Oracle Database 12*c*, Oracle Database 12*c* provides an ideal platform for implementing ILM.

For more information about Automatic Data Optimization, please see the Storage Optimization blog here: https://blogs.oracle.com/DBStorage/

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

blogs.oracle.com/oracle

facebook.com/oracle

twitter.com/oracle

oracle.com

Hardware and Software, Engineered to Work Together