ORACLE

Implementing Data Lifecycle Management with Oracle Automatic Data Optimization

July, 2024, Version 23ai

Purpose statement

This document provides an overview of features and enhancements included in release 23ai. It is intended solely to help you assess the business benefits of upgrading to 23ai and planning for the implementation and upgrade of the product features described.

Disclaimer

This document in any form, software, or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. 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, timing, and pricing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

ORACLE

Table of contents

Exponential increases in data volumes are putting enterprise IT infrastructures under severe pressure – from a cost, performance, scalability, and manageability perspective. It has become imperative to employ more efficient ways of storing, and managing, data to meet growing demands on IT systems. Dramatic increases in storage volumes are evident in all types of applications, and enterprise applications are no exception.

Although most organizations have long regarded their data as one of their most valuable corporate assets, only recently has the amount of data under management become a major issue. Originally, data helped achieve operational goals to run the business, but as technology capabilities have grown, ever-larger databases have become feasible for both operational (OLTP) and analytical (Data Warehouse) applications.

Regulatory requirements are also changing data retention, as many organizations are now required to retain, and control, much more information for much longer periods. These requirements often extend beyond structured data - typically stored in relational databases such as Oracle Database – to semi-structured and unstructured data such as medical images, videos, photos, contracts, documents, etc. The result is an explosion in the amount of data (and indexes) that organizations are required to obtain, organize, manage, and store securely (and safely), while still providing easy, scalable, and high-performance access.

Consequently, organizations are trying to store fast growing quantities of data for the lowest possible cost while meeting increasingly stringent regulatory requirements for data retention and protection.

Data Lifecycle Management

Data lifecycle management is the practice of applying policies for the effective management (including storage) of information throughout its useful life.

Data lifecycle management for Oracle Database includes every phase of information from its beginning to its end. This includes the policies, processes practices and tools used to align the business value of information with the most appropriate and cost-effective IT infrastructure -- from the time information is created, or acquired, through its final disposition.

There are **FIVE STEPS** to implement a data lifecycle management strategy:

1. **Define the Data Classes**:

For the primary databases that drive your business, identify the types of data in each database and where it is stored, and then determine:

- Which data is important, where it is, and what must be retained
- How this data flows within the organization
- What happens to this data over time and when is it no longer actively needed
- The degree of data availability, and protection, that is needed
- Data retention for legal and business requirements

2. **Create Logical Storage Tiers**:

For the data classes that represent the different types of storage tiers available in your environment.

3. **Define a Lifecycle:**

A lifecycle definition describes how data migrates across logical storage tiers during its lifetime. A lifecycle definition comprises one or more lifecycle stages that select a logical storage tier, data attributes such as compression and read-only, and a duration for data residing on that lifecycle stage. To summarize, a lifecycle defines WHERE to store data, HOW to store data and HOW LONG data should be retained.

4. **Assign a Lifecycle to Database Tables/Partitions**
5. **Define and Enforce Compliance Policies**

Automating Data Lifecycle Management with Oracle Database

Previously, when implementing a data lifecycle management strategy with Oracle Database, organizations typically used Advanced Row Compression and Data Partitioning to manually create and deploy a compression and storage tiering solution – a solution that required organizations to have sharp insight into data access and usage patterns across applications and tables/partitions.

Based upon this insight, DBAs, along with their storage counterparts, would manually compress and/or move data based upon their best estimations regarding actual data usage, ideally trying to ensure that the most frequently accessed data remained on the highest performance storage.

The data lifecycle management features of Oracle Database can now automate this previously manual operation using the Heat Map and Automatic Data Optimization (Automatic Data Optimization is a feature of Advanced Compression).

Heat Map

Heat Map automatically tracks usage information at the row and segment levels. 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 enables a detailed view of how data is accessed, and how access patterns change 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. In addition, Oracle Enterprise Manager provides graphical representations of Heat map data.

Heat map is included with Oracle Database Enterprise Edition.

Automatic Data Optimization

Automatic Data Optimization (ADO) allows organizations to create compression tiering and/or storage tiering ADO policies and Oracle Database evaluates these ADO policies during the database maintenance window, and uses the information collected by Heat map to determine which operations to execute.

ADO policies specify *what* conditions (of data/index 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. Custom conditions can be created by the DBA, allowing other factors to be used to determine when to move or compress data.

ADO compression policies can be set at the following levels:

- **SEGMENT:** Specify SEGMENT to create a segment-level compression policy. This type of policy instructs the database to compress table segments when the condition specified in

the AFTER clause is met or when the PL/SQL function specified in the ON clause returns TRUE.

- **GROUP:** Specify GROUP to create a group-level compression policy. This type of policy instructs the database to compress the table and its dependent objects, such as indexes and SecureFiles LOBs, when the condition specified in the AFTER clause is met or when the PL/SQL function specified in the ON clause returns TRUE.
- **ROW**: Specify ROW to create a row-level compression policy. This type of policy instructs the database to compress database blocks in which all the rows have not been modified for a specified period of time.

All ADO operations execute automatically and in the background, with no user intervention required. In addition to being evaluated and executed automatically in the background during the maintenance window, policies can also be evaluated and executed anytime by a DBA, manually or via a script.

Implementing Automatic Data Lifecycle Management

At the core of an Oracle Database data lifecycle management solution is the ability to define multiple data classes and tiers of storage and assign different portions of data/indexes to different tiers based on the desired cost, performance, and security for each portion.

This is enabled using Data Partitioning, Advanced Row Compression, Advanced Index Compression and Hybrid Columnar Compression, which are briefly described below.

Data Partitioning

At the most basic level, organizations can implement a data lifecycle management strategy by partitioning data based on the age of the data, and then moving historical partitions to low-cost storage, while keeping partitions that are more active on high performance storage. Data Partitioning allows a table, index, or index-organized table (IOT) to be subdivided into pieces. Each piece of a database object is a partition. Each partition has its own name and may optionally have its own storage characteristics (including compression).

From the perspective of a database administrator, a partitioned object has multiple pieces, managed either collectively or individually. This gives the administrator considerable flexibility in managing the partitioned object. However, from the perspective of the application, a partitioned table is identical to a non-partitioned table; no modifications to application queries are necessary when accessing a partitioned table.

It is not unusual for partitioning to improve the performance of queries or maintenance operations by an order of magnitude. Moreover, partitioning can greatly reduce the total cost of data ownership, enabling a “tiered archiving” approach of keeping older but still relevant information online on lower cost storage devices.

Advanced Row Compression

Advanced Row Compression, a feature of Advanced Compression, uses a unique data compression algorithm specifically designed to work with database tables in all types of applications. The algorithm works by eliminating duplicate values within a database block, even across multiple columns. The compression ratio achieved with a given data set depends on the nature of the data being compressed.

In general, organizations can expect to reduce their storage space consumption 2x to 4x by using Advanced Row Compression. That is, the amount of space consumed by compressed data will be two to four times smaller than that of the same data without compression.

Advanced Index Compression

Indexes are used extensively in OLTP and mixed workload environments, as they are capable of efficiently supporting a wide variety of access paths to the data stored in relational tables. It is very common to find many indexes being created on a single table to support a multitude of access paths for applications. This can cause indexes to contribute a greater share to the overall storage of a database when compared to the size of the base tables alone.

Advanced Index Compression, a feature of Advanced Compression, enables the highest levels of data compression and provides enterprises with tremendous cost-savings and performance improvements due to reduced I/O. In general, organizations can expect to reduce their index storage space consumption 2x to 5x by using Advanced Index Compression, depending on which index compression level (LOW/HIGH) is implemented.

Hybrid Columnar Compression

Hybrid Columnar Compression (HCC) enables higher levels of data compression and provides enterprises with tremendous cost savings. Average compression ratios can range from 6x to 15x depending on which Hybrid Columnar Compression level is implemented – real world customer benchmarks have resulted in compression ratios of up to 50x and more.

Oracle's Hybrid Columnar Compression technology utilizes a combination of both row and columnar methods for storing data. While HCC compressed data can be modified using conventional Data Manipulation Language (DML) operations, such as INSERT and UPDATE – HCC is best suited for applications with no, or very limited DML update operations.

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.

Data Lifecycle Management Usage Example

Steps 1 to 3: Define Data Classes, Logical Storage Tiers, and Information Lifecycle
Oracle Features: *Data Partitioning, Advanced Compression and/or Hybrid Columnar Compression*

- **Define the Data Classes**

  This step involves looking at all the data in your organization. This analysis requires organizations to understand which objects are associated, with which applications, where those objects are located (on what class of storage), whether the objects have been compressed, and the granularity of the object (table vs. partition).

- **Create Logical Storage Tiers**

  This step identifies and creates logical storage tiers, utilizing higher cost high performance storage and lower cost high-capacity storage.

- **Define a Lifecycle**

  The lifecycle definition describes how data migrates across logical storage tiers during its lifetime. A lifecycle definition includes one or more lifecycle stages that select a logical

storage tier, data attributes such as compression and/or read only, and a retention period for data residing on that lifecycle stage.

The lifecycle brings together the information/activities in **STEPS 1 and 2** to allow DBAs to plan *WHERE* to store data (the logical storage tiers), *HOW* to store data (the data granularity and whether to compress the data) and *HOW LONG* data should be retained (which also helps determine how to compress the data).

Utilizing the planning from **STEP 3**, the most active data can be located on a high-performance tier, and the less active / historical data on lower-cost tiers (and begins to associate the appropriate compression levels to the various storage tiers). Using Oracle Data Partitioning, the most active data partitions can be placed on faster, higher performance storage, while less active and historical data can be placed on lower cost storage.

Data/index compression and index optimization can also be applied as desired on a partition-by-partition basis. With this combination of features, the organization is meeting all its performance, reliability, and security requirements, but at a significantly lower cost than in a configuration where all data is located on one tier of storage.

With OLTP applications, organizations can use Advanced Row Compression for the most active tables/partitions, to ensure that newly inserted, or updated data, will be compressed as DML operations are performed against the active tables/partitions. For cold or historic data (tables/partitions with no or limited DML update activity) within the OLTP application, organizations can use either Query Level or Archive Level Hybrid Columnar Compression (assuming they are using storage that supports HCC).

Reducing the space requirement for indexes, without sacrificing performance, requires data lifecycle management actions like Automatic Data Optimization for data segments. Using the Index compression and optimization capabilities, the same ADO infrastructure can also automatically optimize index storage. Like ADO for data segments, the automatic index compression and optimization capability achieves data lifecycle management on indexes by enabling organizations to set policies that automatically optimize indexes.

When using ADO for indexes, an OPTIMIZE clause enables ADO to optimize the index whenever the policy condition is met. The optimization process includes actions such as compressing, merging, or rebuilding indexes.

- Compresses portions of the key values in an index segment
- Merges the contents of index blocks where possible to free blocks for reuse
- Rebuilds index to improve space usage and access speed

When the optimize clause is specified, Oracle Database automatically determines which action is optimal for the index and implements that action as part of the optimize clause, you do not have to specify which action is taken.

Prior to automating data lifecycle management with Oracle Database, organizations implemented both the storage tiering and compression tiering of their data/indexes manually, based upon their knowledge of the database. With Oracle Database, storage tiering, compression tiering and storage optimization can be automated, reducing the requirement for organizations to have deep insights into their data access/usage patterns.

Steps 4 and 5: Assign a Lifecycle to Tables/Partitions and Define and Enforce Compliance Policies

Oracle Features: *Data Partitioning, Advanced Compression and/or Hybrid Columnar Compression, Automatic Data Optimization and Heat Map*

Implementing an automated compression tiering and storage tiering solution using Automatic Data Optimization and heat map is straightforward, as the example below will show.

In this example, we have a table named "orders" that was initially created without any compression. We have turned on heat map and are tracking the usage of this table over time. It is the intention, of our organization, to wait until most the post-load activities, that are performed initially on the table, complete and then the table be compressed, using Advanced Row Compression, without moving the table (meaning the table will be compressed in place).

Once the tables cool down (with no or few DML inserts/updates) and begins to be primarily used for reports/queries (LESS ACTIVE tier), we will then compress the table with HCC QUERY HIGH. When the table has become colder and is only occasionally queried (used for reporting purposes), we will then compress it even further with HCC ARCHIVE HIGH.

The example uses the ADO condition "***no modification***".

The ADO policy below enables Advanced Row Compression, and since we specified "row" versus "segment" level compression, the tables' blocks will be individually compressed when all the rows on the block meet the ADO compression policy that is specified (that being AFTER 2 DAYS OF NO MODIFICATION)

ALTER TABLE orders **DATA LIFECYCLE MANAGEMENT ADD POLICY**
ROW STORE COMPRESS ADVANCED ROW
AFTER 2 DAYS OF NO MODIFICATION;

This policy allows the post-load activity to subside on the table before compression is enabled. For organizations with SLAs around the load times, this allows the table to be created and populated as quickly as possible, before implementing compression.

Compression can be specified at the "row" level or the "segment" level. Row level allows the table to be compressed in place, block-by-block, as all the rows on a block meet the ADO policy condition. Tables/partitions can also be compressed at the segment level; this means the entire segment is compressed at the same time.

The next policy, that was specified by the DBA, will be automatically enforced by the database (at the segment level) when heat map determines there has been no data modifications for 90 days. The policy changes the compression level of the table to a higher level of compression (HCC QUERY HIGH) when the data is being used primarily for queries/reporting.

ALTER TABLE orders **DATA LIFECYCLE MANAGEMENT ADD POLICY**
COLUMN STORE COMPRESS FOR QUERY HIGH SEGMENT
AFTER 90 DAYS OF NO MODIFICATION;

Changing the compression from Advanced Row Compression to Hybrid Columnar Compression (HCC QUERY HIGH), occurs during a maintenance window after the specified ADO policy criteria has been met.

When this table further "cools down" additional storage and performance gains can also be realized when ADO automatically compresses the data to the highest level possible (HCC ARCHIVE

HIGH) with Oracle Hybrid Columnar Compression. In this example, this data is still needed for query purposes, but is no longer being actively modified (no or few DML inserts/updates) and only occasionally queried or used for reporting. This cold/historic data is an ideal candidate for HCC ARCHIVE HIGH compression.

After 180 days of no modification to the data, this ADO policy will be applied.

ALTER TABLE orders **DATA LIFECYCLE MANAGEMENT ADD POLICY**
COLUMN STORE COMPRESS FOR ARCHIVE HIGH SEGMENT
AFTER 180 DAYS OF NO MODIFICATION;

With the final ADO compression tiering policy criteria being satisfied, the data is now compressed to the HCC ARCHIVE HIGH level and could be moved to lower cost storage (Tier 2). This allows active data to remain on higher performance tiers (ACTIVE Tier 1) and allows the historic data, which remains online, to be accessed by applications as needed and ensures a smaller footprint for the historic data (LESS ACTIVE Tier 2).

This example uses the “best practice” approach of compressing using both Advanced Row Compression and Hybrid Columnar Compression. Advanced Row Compression (as well as ADO) are features of Advanced Compression. While HCC does not require Advanced Compression, it does have other requirements[1], please see the Oracle HCC Technical Brief on the Advanced Compression page on Oracle.com.

While compression tiering, best practice does include the use of HCC, if an organization does not have access to HCC, then they would use only Advanced Row Compression in their ADO policies.

In the above example ADO has been used to implement compression tiering of the data, but in a similar manner using the OPTIMIZE clause for indexes provides an opportunity for ADO to optimize an index whenever the policy condition is met. The optimal action invoked by ADO is determined automatically by Oracle Database -- this could include compress, merge, or rebuild (as described earlier).

ALTER INDEX orders_idx **DATA LIFECYCLE MANAGEMENT ADD POLICY**
OPTIMIZE AFTER 3 DAYS OF NO MODIFICATION;

Using the example above, as with ADO for data segments, the heat map framework is used to collect activity statistics on the index as it goes through data lifecycle stages. When the index ADO policy qualifies for execution, the database automatically determines which index optimization to implement (merge, compress or rebuild index). In the example, the ADO policy automatically optimizes the index based on the recommendation of the database when the policy conditions (no modification for 3 days) are met and the policy is executed.

ADO-based storage tiering (Tier To) is not based upon the ADO condition clause (i.e., after “x” days of NO MODIFICATION) as is compression tiering/storage optimization 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 users 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 exception to the storage pressure

[1] *Note that Hybrid Columnar Compression is only available with Oracle Database on Exadata or with specific Oracle Storage.*

requirement are storage tiering policies with the 'READ ONLY' option, these are triggered by a heat map-based condition clause.

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 data lifecycle management ADO parameters with the CUSTOMIZE_DATA LIFECYCLE MANAGEMENT procedure in the DBMS_DATA LIFECYCLE MANAGEMENT_ADMIN PL/SQL package, for example:

```
BEGIN
DBMS_DATA LIFECYCLE MANAGEMENT_ADMIN.CUSTOMIZE_DATA LIFECYCLE
MANAGEMENT(DBMS_DATA LIFECYCLE MANAGEMENT_ADMIN.TBS_PERCENT_USED,85):
DBMS_DATA LIFECYCLE MANAGEMENT_ADMIN.CUSTOMIZE_DATA LIFECYCLE
MANAGEMENT(DBMS_DATA LIFECYCLE MANAGEMENT_ADMIN.TBS_PERCENT_FREE,25):
END;
```

In this example, 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 (see examples below). This frees up space on your tier 1 storage (ACTIVE tier) 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 (LESS ACTIVE/COLD Tier).

For example:

ALTER TABLE orders **DATA LIFECYCLE MANAGEMENT ADD POLICY TIER TO** lessactivetbs;
ALTER INDEX orders_idx **DATA LIFECYCLE MANAGEMENT ADD POLICY TIER TO** lessactivetbs;

In this simple TIER TO example, Oracle Database automatically evaluated the ADO policies (and tablespace fullness) to determine when data and/or index segments are eligible to move to a different tablespace. This ensures data accessibility and performance, while reducing the storage footprint even further – with no additional burden placed on database administrators or storage management staff.

When setting storage tiering policies, you can specify SEGMENT to create a segment-level storage tiering policy. This type of policy instructs the database to migrate tables segments to the specified tablespace. Specify GROUP to create a group-level storage tiering policy. This type of policy instructs the database to migrate the table and its dependent objects, such as indexes and SecureFiles LOBS, to the specified tablespace.

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 index optimization policy, and a storage tiering policy both qualify for execution, the database will execute both operations in a single segment reorganization step.

Additional Data Lifecycle Management Features

Database contains a rich set of features to enhance and optimize a data lifecycle management solution, including:

SecureFiles

SecureFiles is designed to deliver high performance for files stored in Oracle Database, comparable to that of traditional file systems, while retaining the advantages of Oracle Database.

Traditionally, relational data is stored in a database while unstructured data is stored as files in file systems; with SecureFiles, you can store relational and file data together in Oracle Database and deliver high performance while also implementing a unified security model, a unified backup and recovery infrastructure.

Database File System (DBFS)

The Oracle Database File System (DBFS) creates a standard file system interface on top of files and directories that are stored in database tables. DBFS is like NFS in that it provides a shared network file system that looks like a local file system. Like NFS, there is a server component and a client component. In DBFS, the server is the Oracle Database. Files are stored as SecureFiles LOBs in a database table.

A set of PL/SQL procedures implement the file system access primitives such as create, open, read, write, and list directory. The implementation of the file system in the database is called the DBFS Content Store. The DBFS Content Store allows each database user to create one or more file systems that can be mounted by clients. Each file system has its own dedicated tables that hold the file system content.

In-Database Archiving

In-Database Archiving allows applications to archive rows within tables by marking them as inactive. This feature can meet compliance requirements for data retention while hiding archived data from current application usage.

Archived rows can be displayed using SQL statements that specifically make them visible, and the rows can be re-activated if needed because they still reside in the original table. These archived rows can also be compressed to reduce their storage usage and can be incorporated into a data lifecycle management strategy at the segment level.

Flashback Time Travel

Flashback Time Travel (formerly Flashback Data Archive) provides the ability to track and store transactional changes to a table over its lifetime. Flashback Time Travel is useful for compliance with record stage policies and audit reports. The Flashback Time Travel feature in Oracle Database provides a mechanism for tracking changes to production databases that is secure, efficient, easy to use and application transparent.

Conclusion

Data lifecycle management enables organizations to understand how their data is accessed over time and manage the data compression/storage optimization and storage tiering accordingly. However, most data lifecycle management 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, along with Oracle Database compression, provide comprehensive and automated data lifecycle management capabilities that minimize costs while maximizing performance.

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle