19c ORACLE Database

Implementing Information Lifecycle Management (ILM) with Oracle Database

ORACLE WHITE PAPER | FEBRUARY 2019

ORACLE

Disclaimer

Introduction

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 the growing demands being placed 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 was used to help 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 how and why data is being retained, 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 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. Oracle Database contains a rich feature set that can help implement an Information Lifecycle Management (ILM) solution and meet these new data storage demands, including Data Partitioning, Advanced Row Compression, Hybrid Columnar Compression, Automatic Data Optimization, Heat Map, Direct NFS Client, Clonedb, SecureFiles, In-Database Archiving and Database File System (DBFS).

Information Lifecycle Management

Information Lifecycle Management (ILM) is the practice of applying policies for the effective management of information throughout its useful life. ILM for Oracle Database includes every phase of information from its beginning to its end, and consists of 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.

Generally speaking, there are **FIVE STEPS** to implement an ILM 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 ILM with Oracle Database

When implementing an ILM strategy with Oracle Database, organizations typically use Advanced Compression and Data Partitioning to manually create and deploy a compression and storage tiering solution – a solution that requires 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, can 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 remains on the highest performance storage.

What has become clear to many organizations, after implementing manual storage tiering solutions, is that the ideal ILM solution is automated and is not reliant upon the organization's best guess of data access and usage patterns, but instead uses data usage information maintained by the database. The ideal automation solution would provide policy-based classification of data based on usage, greatly simplifying the ILM implementation process.

Heat Map and Automatic Data Optimization

Two features of Oracle Advanced Compression help organizations automate ILM: *Heat Map* and *Automatic Data Optimization (ADO).*

Heat Map

Heat Map automatically tracks usage information at the row and segment levels.[1] 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.

Automatic Data Optimization

Automatic Data Optimization (ADO) allows organizations to create policies for data compression and data movement, and to implement automatic tiering of compression and storage. Oracle Database evaluates ADO policies during the DBA-defined database maintenance window, and uses the information collected by Heat Map to determine which operations to execute. All ADO operations are executed automatically and in the background, with no user intervention required.

ADO policies can be specified at the segment or row level for tables and table partitions. 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. 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. Custom conditions can also be created by the DBA, allowing other factors to be used to determine when to move or compress data.

Compression Tiering

ADO supports both row-level compression tiering and segment-level compression tiering. Policies can be created to compress inactive rows at the block level (all rows on a block must meet the ADO policy condition before the block is compressed), or to compress entire segments either with Advanced Row Compression or Hybrid Columnar Compression. All of the compression actions implemented by ADO are fully online, and do not block any queries or OLTP transactions.

Storage Tiering

ADO supports storage tiering at the tablespace level in the database. As a tablespace reaches a *percent used value* (set by the DBA) any segments that have qualifying ADO policies will be moved to a target tablespace as specified by the policy; the oldest segments will be moved first. Once the space in the original tablespace reaches a *percent free value* (set by the DBA) no more segments are moved. 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.

Implementing ILM with Oracle Database

At the core of an Oracle Database ILM solution is the ability to define multiple data classes and tiers of storage, and assign different portions of data to different tiers based on the desired cost, performance and security for each portion. This is enabled using Data Partitioning, Advanced Row Compression and Hybrid Columnar Compression, which are briefly described below.

Data Partitioning

At the most basic level, IT administrators can implement an Information Lifecycle Management (ILM) 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 called a partition. Each partition has its own name, and may optionally have its own storage characteristics. From the perspective of a database administrator, a partitioned object has multiple pieces, which can be managed either collectively or individually. This gives the

[1] 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.

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 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.

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 storage savings 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.

HCC Compression Levels

HCC *Warehouse compression* (also referred to as Query compression) has been optimized on Exadata storage to increase scan query performance by taking advantage of the smaller number of blocks on disk. HCC *Archive Compression* is optimized to maximize storage savings, typically achieving a compression ratio of 15:1 (15x).

Hybrid Columnar Compression is available for use on Exadata, SuperCluster, Pillar Axiom, FS1, The Oracle Database Appliance (ODA) and Sun ZFS Storage Appliance (ZFSSA) storage hardware.

Oracle Database ILM Implementation Example

The remainder of this document will discuss how the features of Oracle Database can implement the **FIVE STEPS** of an ILM strategy that are defined earlier in this document. These steps will be reviewed as to how and where they fit into an Oracle Database ILM solution – as well as which Oracle Database features can be utilized.

Steps 1 to 3: Define Data Classes, Logical Storage Tiers and Information Lifecycle

Related Oracle Features:
Data Partitioning, Advanced Row 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**

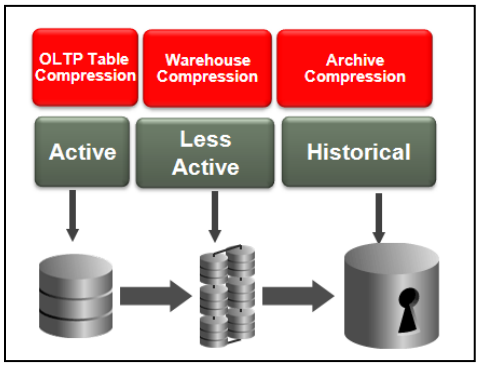
  The Lifecycle definition describes how data migrates across logical storage tiers during its lifetime. A lifecycle definition includes one or more lifecycle stages that select a logical storage tier, data attributes such as compression and/or read only, and a retention period for data residing on that lifecycle stage.

The lifecycle brings together the information/activities in **STEPS** 1 and 2 to allow DBA's 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 diagram below (figure 1) shows how 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 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 compression can also be applied as desired on a partition-by-partition basis. With this combination of features, 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 one tier of storage.

With OLTP applications, organizations can use Advanced Row Compression (previously named OLTP Table 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 Warehouse or Archive Hybrid Columnar Compression (assuming they are using storage that supports HCC).



Define multiple tiers of storage, and assign different portions of your data to different tiers based on the desired cost, performance and security for each portion

Data is compressed with Advanced Row Compression and/or HCC across storage tiers -- 3x to 15x compression is typical depending on the type of compression used

Figure 1. Compression and storage tiering.

Prior to Oracle Database 12*c*, organizations implemented both the storage tiering and compression tiering of their data manually, based upon their knowledge of the database. With Oracle Database, storage tiering and compression can be automated, reducing the requirement for organizations to have deep insights into their data access/usage patterns. This information can now be provided by the database itself, using the Advanced Compression Heat Map capability, and ILM policies can be enforced automatically by the database using Automatic Data Optimization (ADO).

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

Related Oracle Features:

Data Partitioning, Advanced Row Compression and/or Hybrid Columnar Compression, Automatic Data Optimization and Heat Map

Implementing an automated compression 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 the majority of 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 cools down (with no or few DML 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 ILM 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 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.

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 ILM 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 realized when ADO automatically compresses the data to the highest level possible (HCC ARCHIVE HIGH) with Oracle. In this example, this data is still needed for query purposes, but is no longer being actively modified (no or few DML 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 being made to the data, this ADO policy will be applied.

```
ALTER TABLE orders ILM 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) and allows the historic data, which remains online, to still be accessed by applications as needed and ensures a smaller footprint for the historic data (LESS ACTIVE tier).

This example uses the “best practice” approach of compressing using both Advanced Row Compression and Hybrid Columnar Compression. Advanced Row Compression (as well as Heat Map and ADO) are features of Advanced Compression. While HCC does not require Advanced Compression, it does have other requirements[2], please see here for the Oracle HCC White Paper. 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.

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 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 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 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 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. Of course this only applies to tables and partitions that have a "TIER TO" ADO policy defined (see example 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)

```
ALTER TABLE orders ILM ADD POLICY TIER TO lessactivetbs;
```

In this simple ILM example, Oracle Database automatically evaluated the ADO policies to determine when data was eligible to be moved to a higher compression level, and when data was eligible to be moved 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.

Additional Features for Implementing ILM with Oracle Database

Oracle Database contains a rich set of features to enhance and optimize an Information Lifecycle Management (ILM) solution, including:

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

➢ **Direct NFS Client (dNFS)**

Standard NFS client software, provided by the operating system, is not optimized for Oracle Database file I/O access patterns. With Oracle Database 11*g* and later releases, organizations can configure Oracle Database to access NFS V3 (or V4 with Oracle Database 12*c*) NAS devices directly using Oracle Direct NFS Client, rather than using the operating system kernel NFS client. Oracle Database will access files stored on the NFS server directly through the integrated Direct NFS Client, eliminating the overhead imposed by the operating system kernel NFS. These files are also accessible via the operating system kernel NFS client thereby allowing seamless administration.

➢ **CloneDB**

CloneDB allows the creation of databases using "copy on write" technology to quickly create database clones. CloneDB requires that each clone database use the Direct NFS Client and that a backup of the source database be available on Direct NFS Client-mounted storage. There must also be enough storage for that backup and for each database block that changes in each of the clone database(s). The time to create the clone is very quick because the source database's data is not copied to create the clone – only file header information is copied – and when creating multiple clones from a single source database the total storage required can be significantly less than with traditional full size database copies. This allows the organization to easily keep development, test and QA databases available and in sync quickly while using significantly less storage.

➢ **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. SecureFiles was first introduced in Oracle Database 11*g*, and enables a major paradigm shift for storing and managing files. 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, and enabling all the other features of Oracle Database for both structured and unstructured data. See also Database File System below.

➢ **Database File System (DBFS)**

Database File System (DBFS) implements a standard file system interface for files stored in Oracle Database. With this interface, storing files in the database is no longer limited to programs specifically written to use BLOB and CLOB programmatic interfaces - files in the database can now be transparently accessed using any operating system (OS) program that acts on files. DBFS makes it easy to store all your files in your database.

➢ **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 an ILM strategy at the segment level.

Conclusion

Information Lifecycle Management (ILM) enables organizations to understand how their data is accessed over time, and manage the data compression and storage tiering 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 Advanced Compression, with Oracle Database, provide comprehensive and automated ILM capabilities that minimize costs while maximizing performance. In combination with its comprehensive compression features, Oracle Database provides an ideal platform for implementing Information Lifecycle Management for all of your database data.

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