Oracle Enterprise Data Architecture

Data Lifecycle

ORACLE WHITE PAPER | JANUARY 2019

Table of Contents

Disclaimer

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.

Introduction

Oracle is continuously building innovative storage technologies that provide high performance, greater automation and increased cost savings to ensure organizations achieve the maximum benefit from their Oracle Database. These technologies allow Oracle Database to provide industry-leading solutions while also providing tremendous flexibility for the deployment of applications, whether in the cloud or on-premises.

This document discusses how the various technologies that enable Oracle to deliver a robust enterprise data architecture, benefit organizations in the areas of performance, storage management, data lifecycle automation, data transfer, compliance/versioning and more. Major performance related technologies discussed in this document, like Oracle Database In-Memory for high speed, real-time analytics, and Oracle Advanced Compression for sophisticated database compression, can significantly improve performance and reduce I/O and storage footprints.

The enterprise data architecture provides organizations the flexibility to leverage Oracle Database's sophisticated scale out and engineered systems to expand their compute footprint as their computing needs grow and change over time.

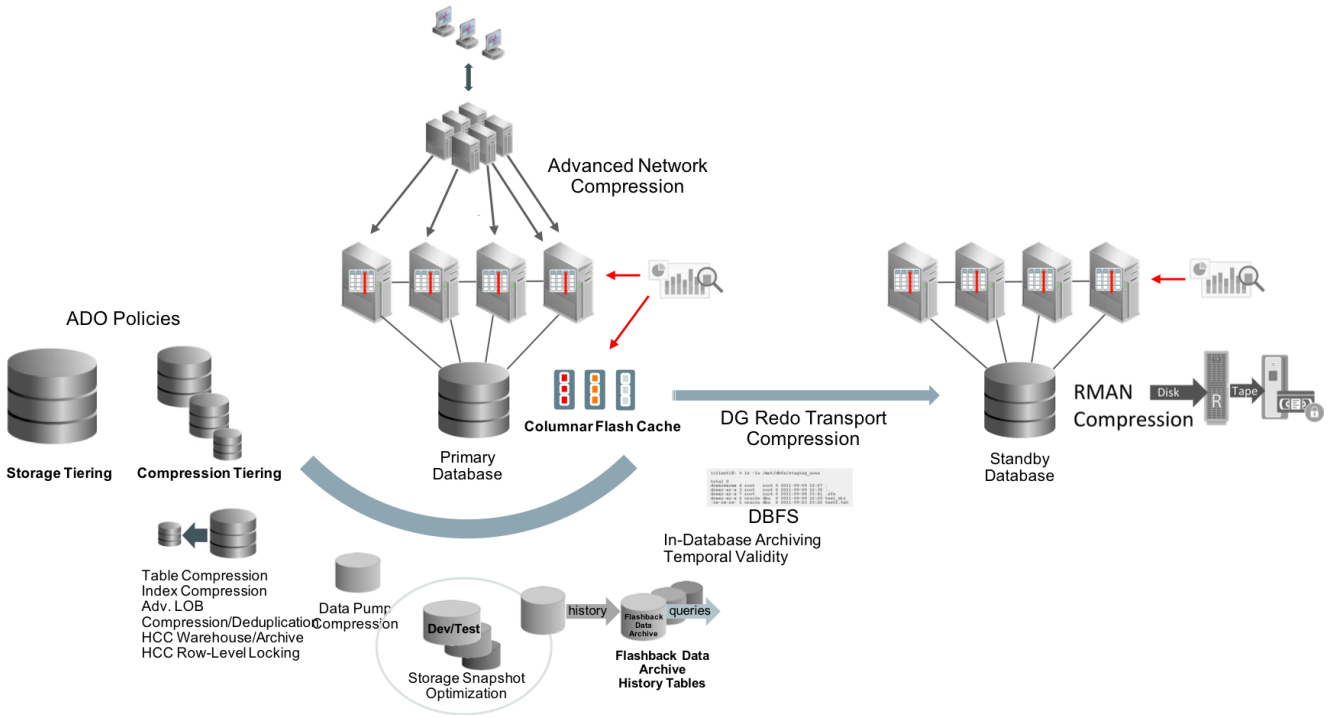

Figure 1. Oracle Enterprise Data Architecture

Enterprise Data Architecture

The various component technologies that comprise the Oracle Enterprise Data Architecture fit broadly into these general key categories:

Performance

Maintaining database performance is always a concern and is exasperated by the exponential growth in data that organizations experience. The amount of data that enterprises store and manage is growing rapidly - various industry estimates indicate that data volume is doubling every 2-3 years. This rapid growth of data presents daunting challenges for both IT and database administrators in regards to maintaining database performance.

To help ensure database performance continues to meet an organization's expectations (and SLAs) Oracle Enterprise Data Architecture technologies can automatically manage data across a wide variety of storage tiers and formats. This includes high-speed in-memory columnar data for high performing analytics queries to OLTP and Data Warehouse workloads that can leverage flash and multiple data and disk formats – all with an eye on improving database performance.

Oracle's compression tiering and storage tiering can also help improve application performance and ensure that performance sensitive data is on the most performant storage tier while colder, inactive data is still available, and stored on the most cost efficient tier.

Storage Management

Database related storage requirements continue to grow and have become a challenge, not just because of the sheer size of the data growth, but also because of increasing performance requirements associated with this growth. Oracle Database offers technologies to not only manage data storage growth based on size, but also based on the current usage of the data. Often, organizations do not recognize that not all data has the same access requirements.

With Oracle's data compression technologies, it is possible to utilize a "*compression tiering*" solution, where organizations can choose to use the level of compression that best matches how their data is currently being used. For example, colder data (historic/archive data) is compressed at a higher level, at the cost of slightly slower access, while active data can be compressed at a level of compression optimized for active OLTP data.

Oracle Database provides several levels of compression to ensure the compression used best fits the usage of the data as that data moves through its lifecycle - from hot/active to warm/less active to cold/historical – while still meeting the performance and availability requirements for the application.

Data Lifecycle Automation

Most data has a lifecycle of activity from inception, where it can be highly volatile, to retention "just in case" where the data may never (or rarely) be accessed, but must be retained to meet regulatory or fiduciary requirements. Even if data is no longer needed, archive and purge techniques rarely work well unless the data is truly isolated. In most applications, data has many dependencies and making some data unavailable invalidates other, related data, or renders it useless due to lack of context.

Even when data can be grouped into archivable sets, it is rarely practical to remove the data (or place the data on tape) if it will ever need to be accessed again. The idea of "unplugging" data and then, if

needed, plugging it back in to access is a complicated and rarely successful process. Data formats, the time to reload and possible version upgrade incompatibilities (i.e. both database and application) can all make it very impractical to reload data that has been removed from the database.

Oracle's solution is to retain the data in the database and use "*storage tiering*" technologies to manage older, less active data, possibly also making it read only to avoid multiple backups, and to keep the data in the database so that it is always secure and accessible. 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 slightly slower speeds – the effect on the overall application performance is minimal, due to the rarity of accessing colder data.

Data Transfer and Backup

Any time data is touched; there is an opportunity to make its transfer (network movement, data backups, etc.) more efficient. The Oracle Enterprise Data Architecture provides additional compression technologies to reduce network traffic and disaster recovery (Oracle Data Guard) redo transport traffic as well as reducing database backup and recovery times.

This is much more efficient than typical storage-based compression utilities that do not understand the data (or its format) and must rely on generic compression algorithms. While storage-based solutions are ideally suited to provide compression for a wide range of applications and uses, storage-based compression solutions can be characterized as being "database-unaware" -- meaning that they lack the application awareness needed to truly optimize the storage and use of Oracle Database data.

Compliance and Versioning

The Oracle Enterprise Data Architecture includes technologies that allow data to be archived for auditing or other compliance requirements, and versioned to simplify application level views of data at any time in its lifecycle.

Versioning and auditing enables organizations to track and store transactional changes to data over the lifetime of that data -- very useful for compliance with record stage policies and audit reports. Likewise, these technologies also enable users and applications to set the "archive state" for their data. Data that has been marked as archived remains safe in the database, but is not visible to an application unless the application wants to include this archived data in its results.

Including these technologies with Oracle database frees organizations from having to specially build this functionality into their application, which would typically require additional coding and application maintenance.

Oracle Technologies

This section discusses the specific features/capabilities of the Oracle Enterprise Architecture grouped by where the technologies are typically utilized in the data lifecycle. Most of these features work in conjunction with other Oracle Database features. They are not isolated but instead work together to enable industry leading abilities in database and application management and performance.

To understand how the various Oracle technology components interact with each other, as well as understand the benefits provided by each component, this document will follow data from the early OLTP stages of its lifecycle (where the data is actively modified) until the later stages of its lifecycle, where the

data is primarily being used for reporting/analytical purposes, and then later is maintained as archive/historic data (often for legal or regulatory or legal reasons).

Starting from the transaction that created the data, we will discuss each Oracle technology component, included in this document, as that component is utilized during the different stages of the data's lifecycle, as illustrated below.

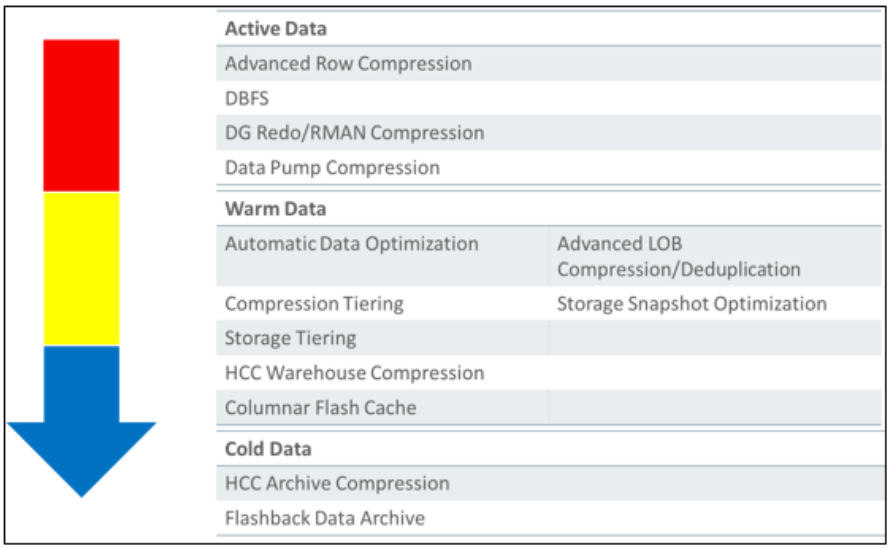

Figure 2. Data Lifecycle Flow with Enterprise Data Architecture

LIFECYCLE STAGE

ACTIVE OLTP

Action: Transaction creates new data or data is loaded from an external source – Oracle Heat Map and Automatic Data Optimization utilized to implement compression tiering for OLTP data using Advanced Row Compression

Organizations typically want to reduce the storage footprint for their ever-growing databases. Oracle Database provides several types of compression, allowing organizations to choose the type of compression best suited to the current usage of the data, as data moves through its lifecycle - from hot/active to warm/less active to cold/historical – while still meeting the performance and availability requirements for the application. We call this compression tiering.

Before compression tiering can be used to compress data to its optimal compression level (based on the data's current usage) an organization must first understand where that data is in its lifecycle. This is a task made more difficult given that IT organizations do not always have visibility into the usage of that data. This typically requires knowledge of the usage of the data and with commercial, off-the-shelf applications, or custom applications where the organization no longer has access to the developers who built the application, this can be difficult or impractical.

Fortunately, the easiest way to gather and manage data usage information is with Oracle Heat Map, an important component in the Oracle Enterprise Data Architecture.

Heat Map

Heat Map is the key to providing the heuristics to manage data (tables/partitions) based on use and usage type. 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. It can also differentiate between row-store accesses versus column-store access (i.e. Database In-Memory).

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. The figure below 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.

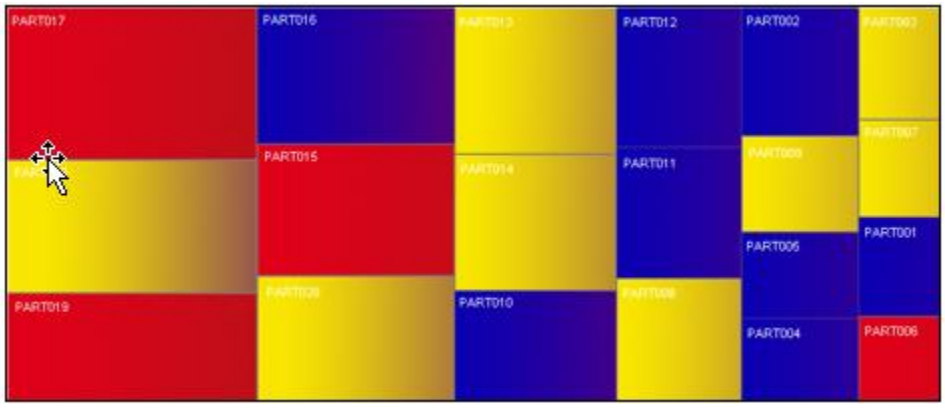

Figure 3. Heat Map Information for Partitioned Table Usage

With the data collected by Heat Map, organizations can create Automatic Data Optimization (ADO) policies to automate compression tiering across the database. Oracle Database evaluates ADO policies during the DBA-defined database maintenance window, and uses the information collected by Heat Map to determine which policies to execute.

Based on Heat Map data (i.e. segment access, row modification or time since creation), ADO policies can be created to move a table/partition through various levels of compression, including Advanced Row Compression and Hybrid Columnar Compression. The level used can be determined based upon the current usage of the data as identified by the Heat Map.

Automatic Data Optimization Compression Tiering

ADO provides for the implementation of an automated Information Lifecycle Management (ILM) strategy -- recognizing that data has different access patterns during its existence can lead to usage that is much more efficient and performant, rather than treating data as a "one size fits all".

Heat Map provides the usage information and ADO enables the creation of the policies needed to control the format and location of data based on its actual usage characteristics. ADO policies are specified at the segment or row level for tables and table partitions. ADO policies specify *what* conditions (of data access)

[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).

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.

In this 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 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). Row-level ADO policies can 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. 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;
```

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.

Since the data we are following in this document is being actively modified, an organization will want to use Advanced Row Compression (as in the examples above) to compress their active data. Advanced Row Compression uses a unique compression algorithm specifically designed to work with OLTP/DW applications to improve performance while reducing storage costs.

Although storage cost savings and optimization across servers (production, development, QA, Test, Backup and etc...) is often seen as the most tangible benefit, all of the features of Oracle Advanced Compression are designed to improve performance for all components of your IT infrastructure, including memory, network bandwidth and storage.

Advanced Row Compression

Advanced Row Compression can significantly reduce a tables/partitions storage requirements, 2x to 4x compression ratios are typical, but the challenge is to maintain the performance of data access when using compression.

Advanced Row Compression uses a unique compression algorithm specifically designed to work with OLTP/DW applications. The algorithm works by eliminating duplicate values within a database block, even across multiple columns. Compressed blocks contain a structure called a symbol table that maintains compression metadata. When a block is compressed, duplicate values are eliminated by first adding a single copy of the duplicate value to the symbol table. Each duplicate value is then replaced by a short reference to the appropriate entry in the symbol table.

Through this innovative design, compressed data is self-contained within the database block, as the metadata used to translate compressed data into its original state is stored in the block header. When compared with competing compression algorithms that maintain a global database symbol table, Oracle's approach offers significant performance benefits by not introducing additional I/O when accessing the compressed data's metadata.

Oracle's unique technology allows for significant compression without a "decompression" penalty. Oracle can operate directly on the compressed data. This reduces the I/O and memory footprints because the data is queried, in a compressed format, and is stored, in memory, in the same compressed format (i.e. in the Oracle Database buffer cache). This actually improves performance for all types of applications and is often contrary to what most developers and administrators initially think about compression, instead assuming that it will require significant CPU resources to compress and decompress the data.

Oracle Database 11g Release 1 first introduced OLTP Table Compression, now called Advanced Row Compression, which maintains compression during all types of data manipulation operations, including conventional DML such as INSERT and UPDATE. Advanced Row Compression minimizes the overhead of write operations on compressed data, making it suitable for transactional / OLTP environments as well as Data Warehouses, extending the benefits of compression to all application workloads.

LIFECYCLE STAGE

ACTIVE OLTP
ANALYTICS

Action: Transactional data populated in Database In-Memory to facilitate real-time analytics directly on the source data – without having to wait for the data to be transferred to a reporting platform

As the amount of data under management continues to grow it becomes evident to organizations that their data needs to serve multiple use cases -- including support for transactional applications and reporting/analytical applications. Historically, in a traditional enterprise data architecture transactional data would be maintained in a transactional/OLTP database system while the data, or a copy of the data, needed for reporting/analytics would typically be transferred (and managed) in a separate data warehouse database. While this traditional configuration to the multi-use case of data issue has been the approach often taken by organizations, the Oracle Enterprise Data Architecture can instead offer a much more streamlined and better performing solution for those organizations seeking a better solution to their dual-use case requirements.

Oracle Database In-Memory provides a unique dual-format architecture that enables tables to be simultaneously optimized for both transactional and reporting/analytics uses, but requiring only one database.

Figure 4. Oracle's Unique Dual-Format Architecture

Oracle Database In-Memory's dual-format architecture enables organizations to run reports and ad hoc queries in addition to OLTP transactions, using the same data. The key advantage of Oracle's in-memory capabilities is the possibility of "real-time analytics" (the ability to run analytic queries directly on the source data) without having to wait for that data to be transferred or ETL'd to a reporting platform. This provides organizations with trends and summaries of their data that can be acted on in real time. In fact, real-time analytics opens up the possibilities of asking business questions, which were not being asked before, because of the resource requirements or time delay in accessing data.

In addition, Database In-Memory eliminates the need for most analytic indexes by delivering performance similar to having an index on every column, but with much less transactional overhead. Removing analytic indexes speeds up OLTP operations since the analytic indexes are no longer needed by every transaction -- further reducing the storage requirements of the database

Database In-Memory

Database In-Memory uses an In-Memory column store (IM column store), which is a component of the Oracle Database System Global Area (SGA), called the In-Memory Area. Data in the IM column store does not reside in the traditional row format used by the Oracle Database; instead, it uses a new columnar format.

Oracle Database has traditionally stored data in a row format. In a row format database, each new transaction or record stored in the database is represented as a new row in a table. That row is made up of multiple columns, with each column representing a different attribute about that record. A row format is ideal for online transaction systems, as it allows quick access to all of the columns in a record since all of the data for a given record are kept together in-memory and on-storage. A column format database stores each of the attributes about a transaction or record in a separate column structure. A column format is ideal for analytics, as it allows for faster data retrieval when only a few columns are selected but the query accesses a large portion of the data set.

Supported Database In-Memory workloads include:

- **OLTP and Mixed Workloads:** Database In-Memory enables orders of magnitude faster analytics without impacting OLTP workload. In fact, as long as there are sufficient CPU resources to support the analytic workload, there is virtually no impact to the OLTP workload.
- **Data Warehouse Workloads:** Data Warehouse workloads have been traditionally dominated by analytic reporting requirements. Data Warehouses are usually time based historical repositories that integrate data from one or more disparate sources. Organizations tend to run analytic reports on the most current data while retaining older data for historical data retention. While this description is a simplistic view of Data Warehousing, the reality is that most analytic reporting is performed on relatively recent data. Database In-Memory is ideally suited to this type of workload. Database In-Memory enables all types of analytic workload by leveraging columnar formatted data, along with specialized technology, to make scanning of in-memory columnar data orders of magnitude faster than similar row-based queries.

But what happens when a DML operation (insert, update or delete) occurs on each format? A row format is incredibly efficient for processing DML as it manipulates an entire record in one operation (i.e. insert a row, update a row or delete a row). A column format is not as efficient at processing DML, to insert or delete a single record in a column format all the columnar structures in the table must be changed. That could require one or more I/O operations per column. Database systems that support only one format suffer the tradeoff of either sub-optimal OLTP or sub-optimal analytics performance. Database In-Memory provides the best of both worlds by

allowing data to be simultaneously populated in both an in-memory row format (the buffer cache) and a new in-memory columnar format (the IM column store): a dual-format architecture.

It is important to remember that the IM column store does not replace the buffer cache, but acts as a supplement, so that data can now be stored in memory in both a row and a columnar format. In addition to supporting DRAM in the SGA for the IM column store, Database In-Memory columnar formatted data is supported on Exadata flash -- this enables organizations to have Terabytes of columnar formatted data to run high-speed analytic processing.

Not all of the objects in an Oracle database need to be populated in the IM column store. This is an advantage over so-called “pure” in-memory databases that require the entire database to be memory-resident. With Oracle Database In-Memory, the IM column store should be populated with the most performance-critical data in the database. Less performance-critical data can reside on lower cost flash or disk. If your database is small enough, you can populate all of your tables into the IM column store.

Database in-Memory on Exadata enables organizations to have objects populated in the columnar format that makes the best use of memory and flash resources. On Exadata systems there is much more Flash available than DRAM and this makes a good secondary tier for columnar data. As data is ingested into the Data Warehouse and has significant analytic value, it is an ideal candidate for population into the in-memory column store. As it ages, and is less actively queried, it may be a candidate to be evicted from the IM column store and accessed in Flash storage in columnar format and/or converted to hybrid columnar compression (see below) format for storage savings while retaining columnar query capabilities. Ultimately, the data may be retained just for archive purposes and Hybrid Columnar Compression may provide the most economical method of storing that data while still having it available to query if necessary.

Database In-Memory can also reduce the need for analytic reporting indexes. Database In-Memory queries scan the entire column(s) of a segment and do not make use of traditional Oracle Database indexes. Typically, organizations have needed to create large numbers of analytic reporting indexes in order to run their analytic reports. This requires significant amounts of disk space, in many cases as much as the base object(s), and requires overhead to maintain these indexes during DML operations. Since indexes are persistent objects, they generate undo and redo and have to be backed up. This also increases the overhead of maintaining them.

Oracle Database In-Memory inherits all the proven functionality of Oracle Database, including the sophisticated and robust high availability solutions embodied in Oracle’s popular Maximum Availability Architecture (MAA). Oracle Database In-Memory is fully compatible with Oracle’s Multitenant database architecture, allowing consolidated databases to take advantage of a combination of fast in-memory and low-cost storage technologies

Application SQL does not have to be changed to take advantage of Oracle Database In-Memory. The Oracle SQL Optimizer automatically routes analytic queries to the column format and OLTP queries to the row format based on cost, transparently delivering best-of-both-worlds performance. Oracle Database also automatically maintains full transactional consistency between the row and the column formats, just as it maintains consistency between tables and indexes today. The new column format is a pure in-memory format and is not persistent on disk, so there are no additional storage costs or storage synchronization issues.

Another advantage of Database In-Memory is that it can significantly reduce reporting oriented I/O workload. Database In-Memory queries only use CPU and memory resources and typically run in a fraction

of the time that row based analytic queries take to run. This can actually reduce the overall analytic workload and provide the ability to run even more analytics or ad hoc drill down type queries.

Automatic In-Memory (AIM)
The Automatic In-Memory capability enables Database In-Memory to automatically evict segments from the IM column store based solely on Heat Map and other usage criteria. This does not require the creation of policies ahead of time. This allows for the automatic management of the IM column store when there is not enough room to populate all objects (i.e. the column store is under memory pressure). With AIM enabled, the most active data will populated as the least active data is evicted.

ADO policies also support the addition to, the compression of, and the removal of segments from the IM column store. Again, this is supported for Heat Map based or time-based policies. All ADO operations are executed automatically and in the background, with no user intervention required.

LIFECYCLE STAGE

Action: Automatic Data Optimization Compression Tiering – Inactive Data automatically compressed using Oracle Hybrid Columnar Compression as the data continues to become less active (can include data evicted from the IM column store) and is primarily used for reporting/query-only purposes

A key benefit of the Enterprise Data architecture is that Heat Map can track and monitor the usage of tables and partitions over the lifecycle of those database objects. As discussed earlier, when data is created and is being actively modified, organizations typically enable Advanced Row Compression to ensure the best possible compression ratios, and performance, for actively modified data. However, at this stage of the data lifecycle, even though the data volume is still growing exponentially, much of the data has begun to “cool” down and is used primarily for queries and reporting purposes (read mostly/only data).

The information tracked by Heat Map reflects the changing usage patterns of the data we have been following, and together with existing ADO policies can be used to implement the organization's compression tiering strategy. In our example a compression tiering strategy makes use of ADO to move the aging (colder data) out of its current Advanced Row Compression level to the compression level best suited for the current usage of the table/partition while also significantly reducing the object's storage footprint. Since the table/partition has cooled down (with no or few DML updates or inserts), the ADO polices in place will automatically, and transparently, move the compression level to Hybrid Columnar Warehouse Compression.

Oracle’s Hybrid Columnar Compression (HCC) technology is a new method for organizing data within a database block. As the name implies, this technology utilizes a combination of both row and columnar formats while avoiding the performance shortfalls of a pure columnar format. A logical construct, called the compression unit (CU), stores a set of hybrid columnar compressed rows. When data is loaded, column values for a set of rows are grouped together and compressed. After the column data for a set of rows has been compressed, it is stored in a compression unit.

One of the key benefits of the hybrid columnar approach is that it provides both the compression and performance benefits of columnar storage without sacrificing the robust feature set of Oracle Database.

Warehouse (Query) Compression (HCC)

HCC Warehouse (also called Query Compression) provides significant storage savings by leveraging Hybrid Columnar Compression technology. Query Compression typically provides a 10:1 (10x) data compression ratio, delivering roughly five times the industry average savings. For example, enabling Query Compression on an uncompressed 100 terabyte data warehouse would reduce the storage requirements to only 10 terabytes. Query Compression would return 90 terabytes of storage back to the enterprise for other uses.

While Query Compression is a database storage optimization feature, the implementation of Hybrid Columnar Compression on Exadata is optimized to improve I/O scan performance during typical data warehouse queries. The I/O required to scan a Query-compressed table typically decreases by the compression ratio achieved. Therefore, scan-oriented queries that access a table that has a compression ratio of 10:1 will likely have a reduction in I/O of up to 10x. Total query performance will also likely improve, however it will depend on the available CPU resources.

Further improving performance is Exadata's Smart Scan technology, which greatly reduces the amount of data sent from storage to the database server by offloading much of the scan activities to the Exadata storage. Exadata Smart Scan works directly on Hybrid Columnar compressed data on Exadata. With this level of improvement in I/O scan performance, Query Compression reduces costs by both decreasing the amount of storage required and by eliminating the need to increase the number of disk drives and related hardware to meet performance objectives.

Query Compression provides two levels of compression: LOW and HIGH. Query Compression HIGH typically provides a 10x reduction in storage, while Query Compression LOW typically provides a 6x reduction. Both levels are optimized on Exadata to increase scan query performance by taking advantage of the fewer number of blocks on disk. To maximize the storage savings and query performance benefits of Query Compression, the default level is HIGH.

While HCC compressed data can be modified using conventional Data Manipulation Language (DML) operations, such as UPDATE and DELETE - HCC is best suited for applications with no, or very limited DML operations.

LIFECYCLE STAGE

Action: Automatic Data Optimization Compression Tiering – Cold Data compressed to a higher level of Oracle Hybrid Columnar Compression as the data needs to be primarily maintained for historic/legal or regulatory reasons but may still need to be accessed for query/reporting purposes

Again, fortunately for the organization, the information tracked by Heat Map reflects another change in the lifecycle flow of the data and together with existing Automatic Data Optimization policies implements a change to the compression tiering to further compress this now cold data. The organization's compression tiering strategy

will now automatically, and transparently, move the table/partition out of its current HCC Warehouse Compression level to a compression level even better suited for archive/historic data -- HCC Archive Compression.

Archive Compression (HCC)

HCC Archive Compression provides significant storage savings by leveraging Hybrid Columnar Compression technology. Archive Compression is optimized to maximize storage savings, typically achieving a compression ratio of 15:1 (15x). That is, an uncompressed table or partition would require 15x more storage than a table or partition using Archive Compression.

In contrast to Warehouse Compression, Archive Compression is a pure storage saving technology. Tables or partitions utilizing Archive Compression will typically experience a slight decrease in performance - a factor of the compression algorithm being optimized for maximum storage savings. Therefore, Archive Compression is intended for tables or partitions that store data that is rarely accessed. Data Warehouses on Exadata will typically store frequently queried data in partitions compressed with Warehouse Compression (for performance), while historical data is stored in partitions compressed with Archive Compression (for storage savings).

In many applications, historical data is responsible for consuming up to 80% of the allocated storage. It is no wonder that IT administrators are implementing information lifecycle strategies that archive much of this historical data to tape. However, this approach has several inherent flaws. Once data is archived to tape, the application can no longer access this data directly. In order to access the archived data, IT administrators must first restore the data from tape and load it back into the database. This can take a tremendous amount of time and does not meet the requirements of today's fast-paced businesses.

Complicating things further, data archived to tape can become out-of-sync with structural changes to the database schema, such as the addition of columns and constraints. Therefore, restoring this data back into the database requires not only a significant amount of time but also a significant amount of resources to correctly restore the data and make it accessible by the application. Of course, requests for this data are usually extremely urgent and any delays affect management's ability to make critical business decisions. As you can see, this approach to storing historical data can actually be quite costly to the business.

Archive Compression provides the storage savings benefits of archiving data to tape while keeping this data online for immediate access and modification. Further, as the application evolves, all the historical data will evolve with the database schema modifications, such as new columns, constraints, etc. Therefore, when an application user needs to access historical data, the application will be able to seamlessly service queries without any need to involve IT administrators or application developers.

While query performance against tables or partitions with Archive Compression is slightly slower than tables or partitions with Warehouse Compression or Advanced Row Compression, they are orders of magnitude faster than queries that must access data archived to tape.

Summary of the different compression levels that can be implemented as part of compression tiering.

Table Compression Method	Compression Level	CPU Overhead	Applications	Notes
Basic table compression	High	Minimal	DSS	None.
Advanced row compression	High	Minimal	OLTP, DSS	None.
Warehouse compression (Hybrid Columnar Compression)	Higher	Higher	DSS	The compression level and CPU overhead depend on compression level specified (LOW or HIGH).
Archive compression (Hybrid Columnar Compression)	Highest	Highest	Archiving	The compression level and CPU overhead depend on compression level specified (LOW or HIGH).

Figure 5. Summary of Oracle Database Data Compression Levels

LIFECYCLE STAGE

Action: Automatic Data Optimization Storage Tiering - Once data becomes "cold" it can be tiered to secondary (tier 2) storage to ensure space for more frequently accessed data on primary (tier 1) storage

The data we have been tracking since the beginning of this document has gone from very active to colder, historic/archive data. The design of the Enterprise Data Architecture recognizes that organizations (or even a single application) do not access all of its data equally: as typically the most critical or frequently accessed data will need the best available performance and availability. To provide this best access quality to all data would be costly, inefficient and is often architecturally impossible.

Instead, to meet this challenge an IT organization can 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. 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. The colder data is still available, but at slightly slower speeds – the effect on the overall application performance is minimal, due to the rarity of accessing colder data.

ADO storage tiering policies also support the ability to "tier" segments to different storage layers. Based on Heat Map data or time, policies can be created to move segments online to different tablespaces once the source tablespace reaches a certain utilization threshold.

Automatic Data Optimization Storage Tiering

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 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 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.

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 space. 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 (table/partition) was eligible to move to a different tablespace (tier 2 storage) – with no additional burden placed on database administrators or storage management staff.

Summary

Oracle's Enterprise Data Architecture enables organizations to understand how their data is accessed over time, and optimize their data usage (reporting, analytics or transactional), compression level and storage tiering to best meet the needs of the organization.

Additional Oracle Technologies

The Oracle Technologies above discussed how various components of the Enterprise Data Architecture provide an integrated solution to manage data as that data progresses through its lifecycle. Yet, that is only part of the power of the Enterprise Data Architecture.

These are additional Oracle Technologies' designed to provide further storage savings, streamlined data access as well as data auditing. These capabilities can be used in conjunction with the capabilities discussed earlier and can be implemented at any point of the enterprise data lifecycle flow.

LOB Storage, Compression and Deduplication		
Feature	**Description**	**Benefit**
Secure Files	Manages unstructured and file data using a single security/audit model and a unified backup and recovery process	Improves performance and optimizes storage for unstructured data inside Oracle database
Advanced LOB Compression	Determines if the SecureFiles data is compressible and will compress using industry standard compression algorithms	Significant savings in storage -- improves performance by reducing I/O, buffer cache requirements and redo generation
Advanced LOB Deduplication	Eliminates multiple, redundant copies of SecureFiles data and is completely transparent to applications	Automatically detect multiple, identical SecureFiles data and store only one copy, thereby saving storage space.
Index Compression		
Feature	**Description**	**Benefit**
Advanced Index Compression	Automates index compression	Enables the highest levels of index storage cost-savings and performance improvements (due to reduced I/O)
Prefix Compression	Reduces the overall size of indexes	One of the most useful index optimization features available to DBAs for effectively managing the space used by indexes
Database File System		
Feature	**Description**	**Benefit**
Database File System (DBFS)	Creates a standard file system interface on top of files and directories that are stored in database tables	The DBFS Content Store allows each database user to create one or more file systems that can be mounted by clients
Backup and Data Transfer Compression		
Feature	**Description**	**Benefit**
RMAN Compression	Compresses RMAN backups	Provides a reduction in storage costs and a potentially large reduction in backup and restore times
Data Pump Data Compression	Enables table data to be compressed on export	Reduced dump file size means a significant savings in disk space during backups
Data Guard Redo Transport Compression	Data Guard Redo Transport Services are used to transfer redo data to the standby site(s).	Redo data may be transmitted in a compressed format to reduce network bandwidth consumption and in some cases reduce transmission time of redo data
Advanced Network Compression	Compresses database network data	Reduces the size of the session data unit (SDU) transmitted over a data connection. Reducing the size of data reduces the time required to transmit the SDU
Compliance and Versioning		
Feature	**Description**	**Benefit**
Flashback Data Archive	Provides a mechanism for tracking changes to production databases	Provides a centralized, secure and seamlessly queryable historical data store
In-Database Archiving	Allows users and applications to set the archive state for individual rows	Rows that have been marked as archived stay where they are, but are not visible, unless the session is enabled to see archived data
Temporal Validity	Enables users to track time-periods for real world validity using Valid Time (VT) and Transaction Time (TT)	Using valid time attributes, a query could just show rows that are currently valid, while not showing rows that contain facts that are not currently valid

More Information

For more information regarding the features/capabilities discussed above, please see the following links.

Database In-Memory
Advanced Compression/Basic Compression
Hybrid Columnar Compression
SecureFiles LOB Compression/Deduplication
Index Compression
Heat Map/Automatic Data Optimization

ORACLE

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