19c ORACLE Database

Oracle Advanced Compression

ORACLE WHITE PAPER | FEBRUARY 2019

ORACLE

Table of Contents

Introduction

The amount of data that enterprises are storing, and managing, is growing rapidly - various industry estimates indicate that data volume is doubling every 2-3 years. This exponential growth of data presents daunting challenges for IT. First, and foremost, are storage costs: even though the cost of storage has been declining dramatically, the enormous growth in the volume of data still makes storage one of the biggest cost elements of most IT budgets. In addition, as databases grow at accelerating rates, it is difficult to continue to meet performance requirements while staying within budget.

Oracle is a pioneer in database compression technology. Oracle Advanced Compression, and Oracle Database, together provide a robust set of compression, performance and data storage optimization capabilities that enable IT managers to succeed in this complex environment. Oracle Advanced Compression provides a comprehensive set of compression capabilities to help customers improve performance while reducing storage costs. It allows IT administrators to significantly reduce their overall database storage footprint by enabling compression for all types of data –relational (table), unstructured (file), index, network and backup data.

Although storage cost savings and optimization across servers (production, development, QA, Test, Backup and etc...) are often seen as the most tangible benefits, all of the features of Advanced Compression are designed to improve performance for all components of your IT infrastructure, including memory, network bandwidth and storage. Whether it is a cloud or an on-premises Oracle database deployment, Oracle Advanced Compression can deliver robust compression across different environments with no changes in applications. Benefits from Oracle Advanced Compression include smaller database storage footprint, savings in backups and improved system performance.

Oracle Advanced Compression provides a comprehensive set of compression features designed to reduce costs and improve performance by enabling compression for structured data, unstructured data, indexes, database backups, network traffic and for Data Guard redo. Each of these Advanced Compression capabilities is described in this document.

Data Compression

Oracle Database 11g Release 1 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. In addition, 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.

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 (needed with a global symbol table) when accessing compressed data.

Benefits of Advanced Row Compression

The compression ratio achieved in a given environment depends on the data being compressed, specifically the cardinality of the data. In general, organizations can expect to reduce their storage space consumption by a factor of 2x to 4x by using Advanced Row Compression. That is, the amount of space consumed by uncompressed data will be two to four times larger than that of the compressed data.

The benefits of Advanced Row Compression go beyond just on-disk storage savings. One significant advantage is Oracle's ability to read compressed blocks (data and indexes) directly, in memory, without uncompressing the blocks. This helps improve performance due to the reduction in I/O, and the reduction in system calls related to the I/O operations. Further, the buffer cache becomes more efficient by storing more data without having to add memory.

Minimal Performance Overhead

As described above, Advanced Row Compression has no adverse impact on read operations. Although there can be additional work performed while writing data, making it impossible to completely eliminate performance overhead for write operations. There are several optimizations that minimize this overhead for Advanced Row Compression.

A key optimization is that Oracle Database compresses blocks in batch mode rather than compressing data every time a write operation takes place. A newly initialized block remains uncompressed until data in the block reaches an internally controlled threshold. When a transaction causes the data in the block to reach this threshold, all contents of the block are compressed. Subsequently, as more data is added to the block and the threshold is again reached, the entire block is recompressed to achieve the highest level of compression.

This process repeats until Oracle determines that the block can no longer benefit from further compression. Only the transaction that performs the compression of the block will experience the slight compression overhead – the

majority of DML transactions on compressed blocks will have the exact same performance as they would with uncompressed blocks.

Some of the additional optimization for Advanced Row Compression performance includes:

Partial Compression

With Advanced Row Compression, when the block is full, it is compressed. More rows are added (since more rows can now fit into the block) and the process of recompression is repeated several times until the rows in the block cannot be compressed further. Blocks are usually compressed and reformatted in their entirety, but in some cases, the block can be partially compressed, hence resulting in CPU savings and extra compression.

The partial compression feature is used on already compressed blocks (i.e. compressed with Advanced Row Compression). It looks for uncompressed rows and transforms those into a compressed form, adding or reusing symbols from the block dictionary - this is faster than recompressing the whole block again. Full block recompression also requires that no rows are locked in the block or, that all the rows in the block are locked by the transaction inserting rows into the block. Partial compression gets around these requirements by locking and compressing only those rows that are uncompressed and unlocked - hence it can take place in the presence of other uncommitted transactions in the block.

Background Compression

Typically, Advanced Row Compression block compression is only triggered using direct load operations or by DML insert/update operations. Starting with Database 12c Release 2**,** background space tasks can compress blocks if these find them as compression candidates. A block can be marked as a candidate to be compressed based on previous DML operations or by other space tasks evaluating the space usage of a block. Compression done in the background is the same as Advanced Row Compression, but triggered by background processes instead of an active SQL session. Another example of background-triggered compression is by using ADO (Automatic Data Optimization) row-level compression policies.

Array Inserts

Prior to Oracle Database 12c Release 2**,** array inserts caused multiple recompressions per block: A batch of rows is inserted into a block and the block is compressed, the next batch is inserted and the block is recompressed and so on. Starting with Database 12c Release 2**,** Oracle Database estimates the number of rows that would fit into a compressed block. All these rows are buffered, compressed and a full block image is generated. This mean that typically compression occurs only once or twice (one compression to estimate compression ratio) per block, as opposed to occurring potentially many times as in pre-Database 12c Release 2.

This optimization provides a significant benefit in elapsed time. Oracle Database also obtains a much better compression ratio because Oracle Database can compress many more rows together (hence being able to extract common symbols more effectively). The algorithms also adaptively vary the number of buffered rows and increases the number of rows buffered depending on running estimates of how many

compressed rows would fit into a block. With this enhancement, tables with Advanced Row Compression not only enable much faster scans, but also faster inserts than uncompressed tables, because of reduced logical and physical block gets.

Migration and Best Practices

For new tables and partitions, enabling Advanced Row Compression is easy: simply CREATE the table or partition and specify "ROW STORE COMPRESS ADVANCED". See the example below:

> **CREATE TABLE** emp (emp_id NUMBER, first_name VARCHAR2(128), last_name VARCHAR2(128)) **ROW STORE COMPRESS ADVANCED;**

For existing tables and partitions, there are a number of recommended approaches to enabling Advanced Row Compression:

1. **ALTER TABLE … ROW STORE COMPRESS ADVANCED**
   - This approach will enable Advanced Row Compression for all future DML -- however, the existing data in the table will remain uncompressed.
2. **Online Redefinition (DBMS_REDEFINITION)**
   - This approach will enable Advanced Row Compression for future DML and will compress existing data. Using DBMS_REDEFINITION keeps the table online for both read/write activity during the migration. Run DBMS_REDEFINITION in parallel for best performance.
   - Online redefinition will clone the indexes to the interim table during the operation. All the cloned indexes are incrementally maintained during the sync (refresh) operation so there is no interruption in the use of the indexes during, or after, the online redefinition. The only exception is when online redefinition is used for redefining a partition -- any global indexes are invalidated and need to be rebuilt after the online redefinition.
3. **ALTER TABLE … MOVE ROW STORE COMPRESS ADVANCED**
   - This approach will enable Advanced Row Compression for future DML and will compress existing data. While the table is being moved, it is online for read activity but has an exclusive (X) lock – so all DML will be blocked until the move command completes. Run ALTER TABLE…MOVE in parallel for best performance.
   - ALTER TABLE… MOVE will invalidate any indexes on the partition or table; those indexes will need rebuilt after the ALTER TABLE… MOVE. For partition moves, the use of ALTER TABLE… MOVE PARTITION with the UPDATE INDEXES clause will maintain indexes (it places an exclusive (X) lock so all DML will be blocked until the move command completes) – not available for non-partitioned tables.
   - The ALTER TABLE... MOVE statement allows you to relocate data of a non-partitioned table, or of a partition of a partitioned table, into a new segment, and optionally into a different tablespace. ALTER TABLE…MOVE ROW STORE COMPRESS ADVANCED compresses the data by creating new extents for the compressed data in the tablespace being moved to -- it is important to note that the positioning of the new segment can be anywhere within the data file, not necessarily at the tail of the file or head of the file. When the original segment is

released, depending on the location of the extents, it may or may not be possible to shrink the data file.

4. **ALTER TABLE … MOVE TABLE/PARTITION/SUBPARTITION … ONLINE ROW STORE COMPRESS ADVANCED**

   This approach will enable Advanced Row Compression for future DML and will compress existing data. ALTER TABLE ... MOVE TABLE/PARTITION/SUBPARTITION … ONLINE allows DML operations to continue to run uninterrupted on the table/partition/subpartition that is being moved. Indexes are maintained during the move operation, so a manual index rebuild is not required.

Below are some best practices, and considerations, regarding the features of Advanced Compression:

- The general recommendation is to compress all of the application related tables in the database with one exception: if the table is used as a queue. That is, if the rows are inserted into the table, then later most or all of the rows are deleted, then more rows are inserted and then again deleted. This type of activity is not a good use case for compression due to the overhead to constantly compress rows that are transient in nature
- The best test environment for each Advanced Compression capability is where you can most closely duplicate the production environment– this will provide the most realistic (pre- and post- compression) performance and functionality comparisons.
- Space usage reduction with Advanced Row Compression gives the best results where the most duplicate data is stored (low cardinality). This is especially true for backups -- greater compression will result in less data backed up and hence shorter recovery time. Sorting data (on the columns with the most duplicates) prior to bulk loads may increase the compression ratio.
- Prefix compression (index) is included with Oracle Database Enterprise Edition.
- Although CPU overhead is typically minimal, implementing Advanced Row Compression is ideal on systems with available CPU cycles, as compression will have additional, although minor overhead for some DML operations.
- Compression Advisor is a PL/SQL package that is used to estimate potential storage savings, for Advanced Row Compression, based on analysis of a sample of data. It provides a good estimate of the actual compression ratio after implementing Advanced Row Compression. A version of Compression Advisor, which supports Oracle Database 9*i* Release 2 through 11g Release 1, is available free on the Advanced Compression page on Oracle.com. Compression Advisor (DBMS_COMPRESSION) is included with Oracle Database 11g Release 2 and above.
- Advanced Row Compression is NOT supported for use with tables that have LONG data types.
- Larger blocks do not always ensure higher Advanced Row Compression ratios. Testing with your own data is suggested if you want to determine if larger/smaller block sizes will have an impact on your Advanced Row Compression ratio.
- It is recommended that LOBs over 4K in size be managed using SecureFiles. The Advanced LOB Compression and Deduplication features of Advanced Compression reduce the amount of storage required for LOBs.

- Data Pump compression is completely independent of Advanced Row Compression. The Data Pump dumpfile is uncompressed inline during the import process, and the data is then imported into the target table based on the compression characteristics of the table.
- Before Oracle Database 12c Release 2, blocks containing many types of chained rows could not be compressed. This limitation has been removed starting with Oracle Database 12c Release 2.
- Index-Organized Tables (IOT's) are essentially indexes, so they cannot be compressed with Advanced Row or Basic Compression. IOT's can be compressed with Prefix Compression.
- Index Key (prefix) Compression can be very beneficial when the prefix columns of an index are repeated many times within a leaf block. However, if the leading columns are very selective, or if there are not many repeated values for the prefix columns, then Index Key Compression would not be beneficial
- Advanced Row Compression works well with tablespace-level encryption. Tables are compressed before encryption, so the compression ratio is not affected by the encryption. With column-level encryption, the encryption is done before compression, which will negatively impact the compression ratio
- If users need more flexibility in tailoring an ADO policy, they can do so with a custom ADO policy. Custom ADO policies utilize a user provided function to evaluate each applicable segment.
- It is sometimes necessary to move data as quickly as possible from one tier to another and it is not possible to wait until the next maintenance window. The ability to execute ADO policies immediately provides the ability to move or compress data on demand, regardless of any existing policies.

Unstructured Data Compression

SecureFiles offers a 'best-of-both-worlds' architecture for storing unstructured content such as documents, images, spreadsheets and XML files and is specifically engineered to deliver high performance for file data equal to or better than traditional file systems, while retaining the advantages of Oracle Database.

SecureFiles is designed as a superset of the ANSI standard LOB data type and offers easy migration from existing BasicFiles LOBs, the precursor to SecureFiles. With SecureFiles, organizations can manage all relational data and associated file data with Oracle Database using a single security/audit model, a unified backup & recovery process and perform seamless retrievals across all information.

Advanced Compression includes Advanced LOB Compression and Deduplication features that can dramatically reduce the storage footprint of SecureFiles data, while also improving performance.

Advanced LOB Deduplication

It is extremely common for applications to store exact replicas of files. A typical example is an email application where multiple users may receive the same attachment. Advanced LOB Deduplication eliminates duplicate copies of SecureFiles data. Oracle Database stores one image of the SecureFiles data and replaces the duplicate copies with references to this image.

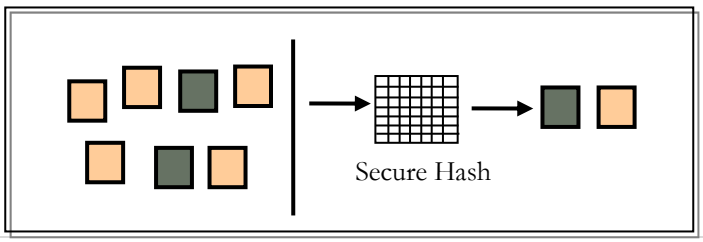

Figure 3 Advanced LOB De-duplication

Consider an email application where 10 users receive an email with the same 1MB attachment. Without Advanced LOB Deduplication, the system would store one copy of the file for each of the 10 users – requiring 10MB of storage. If the email application in our example uses Advanced LOB Deduplication, it will store the 1MB attachment just once. That is a 90% savings in storage requirements.

In addition to the storage savings, Advanced LOB Deduplication also increases application performance. Specifically, write and copy operations are much more efficient since only references to the SecureFiles data are written. Further, read operations may improve if duplicate SecureFiles data already exists in the buffer cache.

Advanced LOB Compression

Advanced Compression provides another mechanism to control the size of your SecureFiles data. Advanced LOB Compression utilizes industry standard compression algorithms to further minimize the storage requirements of SecureFiles data. With Advanced LOB Compression, files such as documents or XML files typically experience a 2x to 3x compression ratio.

Advanced LOB Compression automatically avoids compressing data that would not benefit from compression – for instance a document that was compressed via a 3rd party tool before being inserted into the database as a SecureFiles file. Applications are still able to perform random reads and writes on compressed SecureFiles data since the compressed data is internally broken down into small chunks of data. This can vastly improve performance when compared with compressing entire files before inserting them into the database.

There are three levels of Advanced LOB Compression: LOW, MEDIUM, and HIGH. By default, Advanced LOB Compression uses the MEDIUM level, which typically provides good compression with a modest CPU overhead of 3%-5%. Advanced LOB Compression LOW is optimized for high performance. Advanced LOB Compression LOW maintains about 80% of the compression achieved through MEDIUM, while utilizing typically 3x less CPU. Finally, Advanced LOB Compression HIGH achieves the highest storage savings but incurs the most CPU overhead.

Backup Compression

In addition to compressing data stored inside the database, Advanced Compression also includes the capability to compress backed up data. Recovery Manager (RMAN) and Data Pump are the two most commonly used tools to backup the data stored inside an Oracle Database.

RMAN makes a block-by-block backup of the database data, also known as a “physical” backup, which can be used to perform database, tablespace or block level recovery. Data Pump is used to perform a “logical” backup by offloading data from one or more tables into a flat file.

Advanced Compression includes the capability to compress the backup data generated by both of these tools.

Recovery Manager (RMAN) Compression

The continuous growth in enterprise databases creates an enormous challenge to database administrators. The storage requirements for maintaining database backups and the performance of the backup procedures are directly impacted by database size. Advanced Compression includes RMAN compression technology that can dramatically reduce the storage requirements for backup data.

Due to RMAN's tight integration with Oracle Database, backup data is compressed before it is written to disk or tape and does not need to be uncompressed before recovery – providing an enormous reduction in storage costs and a potentially large reduction in backup and restore times.

There are three levels of RMAN Compression included with Advanced Compression: LOW, MEDIUM, and HIGH. Generally speaking, the three levels can be categorized as such:

- **HIGH** - Best suited for backups over slower networks where the limiting factor is network speed
- **MEDIUM** -Recommended for most environments. Good combination of compression ratios and speed
- **LOW** - Least impact on backup throughput and suited for environments where CPU resources are the limiting factor.

As indicated above, if you are I/O-limited but have idle CPU, then HIGH could work best, as it uses more CPU, but saves the most space and thus gives the biggest decrease in the number of I/O's required to write the backup files. On the other hand, if you are CPU-limited, then LOW or MEDIUM probably makes more sense - less CPU is used, and about 80% of the space savings will typically be realized (compared to Basic).

Data Pump Compression

The ability to compress the metadata associated with a Data Pump job was first provided in Oracle Database 10g Release 2. In Oracle Database 11g, this compression capability was extended so that table data can be compressed on export; this extended capability is a feature of Advanced Compression.

Data Pump compression is an inline operation, so the reduced dump file size means a significant savings in disk space. Unlike operating system or file system compression utilities, Data Pump compression is fully inline on the import side as well, so there is no need to decompress a dump file before importing it. The compressed dump file sets are automatically decompressed during import without any additional steps by the Database Administrator.

Full Data Pump functionality is available using a compressed file. Any command that is used on a regular file will also work on a compressed file. The following options are used to determine which parts of a dump file set should be compressed:

- **ALL -** Enables compression for the entire export operation.
- **DATA-ONLY** - Results in all data being written to the dump file in compressed format.
- **METADATA-ONLY** - Results in all metadata being written to the dump file in compressed format. This is the default.
- **NONE** - Disables compression for the entire export operation.

An expdp command-line option for Oracle Data Pump Export can be used to control the degree of compression used (BASIC, LOW, MEDIUM or HIGH) for an Oracle Data Pump dump file – the same options can also be specified to the PL/SQL DBMS_DATAPUMP package.

The higher the degree of compression, the higher the latency incurred but the better compression ratio achieved. That is, the HIGH option will likely incur more overhead, but should compress the data better. These options enable the DBA to trade off time spent compressing data against the size of the Oracle Data Pump dump file.

The reduction in dump file size will vary based on data types and other factors. Note that when importing using Data Pump, the CREATE TABLE statements will have compression clauses that match the definition in the export file. If a compression clause is missing, then the table inherits the COMPRESSION attributes of the tablespace where the table is stored.

Index Compression

Indexes are used extensively inside OLTP databases since 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 a large number of indexes created on a single table to support the multitude of access paths for OLTP 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 is a form of index block compression. Creating an index using Advanced Index Compression reduces the size of all supported unique and non-unique indexes -- while still providing efficient access to the indexes. Advanced Index Compression works well on all supported indexes, including those indexes that are not good candidates (indexes with no duplicate values, or few duplicate values for given number of leading columns of the index) with the existing Prefix Compression feature.

Advanced Index Compression works at the block level to provide the best compression for each block, this means that users do not need knowledge of data characteristics – Advanced Index Compression automatically chooses the right compression per block. The “HIGH” level of Advanced Index Compression provides significant space savings while also improving performance for queries that are executed using indexes.

The HIGH compression level offers the following advantages over the LOW compression level:

- Gives higher compression ratios in most cases.
- Employs more complex compression algorithms than advanced low compression.
- Stores data in a compression unit, which is a special on-disk format.

Network Compression

Advanced Network Compression, also referred to as SQL Network Data Compression, compresses the network data transmitted at the sending side and then uncompresses it at the receiving side to reduce the network traffic. Advanced Network Compression 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.

The benefits of Advanced Network Compression include:

- **Increased effective network throughput:** Compression allows transmission of large data in less time. SQL query response becomes faster due to the reduced

transmission time. Constrained bandwidth environments can utilize this to reduce query response time.

- **Reduced bandwidth utilization:** Compression saves bandwidth by reducing the data transmitted, allowing other applications to use the freed-up bandwidth. This also helps in reducing the cost of providing network bandwidth.

Advanced Network Compression not only makes SQL query responses faster but also saves bandwidth. On narrow bandwidth connections, with faster CPU, it could significantly improve performance. The compression is transparent to client applications.

Data Guard Redo Transport Compression

Oracle Data Guard provides the management, monitoring and automation software infrastructure to create, maintain and monitor one or more standby databases to protect enterprise data from failures, disasters, errors and data corruptions. Data Guard maintains synchronization of primary and standby databases using redo data (the information required to recover a transaction). As transactions occur in the primary database, redo data is generated and written to the local redo log files.

Data Guard Redo Transport Services transfer redo data to the standby site(s). With Advanced Compression, redo data may be transmitted in a compressed format to reduce network bandwidth consumption and in some cases reduce transmission time of redo data. Redo data can be transmitted in a compressed format when the Oracle Data Guard configuration uses either synchronous redo transport (SYNC) or asynchronous redo transport (ASYNC).

Information Lifecycle Management

Business requirements are not the same for all data in a database.

Data goes through various stages in its Information lifecycle: It starts out as active data, when the data is first created and then frequently queried and modified; this data is an ideal candidate for Advanced Row Compression. After some period-of-time, data typically becomes less active – a time period when it is queried often, for example for report generation, however it is modified rarely; this data is an ideal candidate for Hybrid Columnar Warehouse (Query) Compression. In the final stage, data becomes more or less dormant – it is no longer updated and is queried very infrequently, if at all, but it must be kept for compliance and regulatory purposes; this data can be compressed using Hybrid Columnar Archive Compression.

In Oracle Database, several features of Advanced Compression enhance the Information Lifecycle capabilities of Oracle Database. Heat Map automatically tracks modification and query timestamps, providing detailed insights into how data is accessed. Automatic Data Optimization (ADO) automatically moves and compresses data based on the information collected by Heat Map. Together, these capabilities help to implement automated Information Lifecycle Management (ILM) strategies.

Heat Map

Heat Map is an Advanced Compression feature that collects usage information at the block and segment levels. By using Heat Map in conjunction with Automatic Data Optimization - see the Automatic Data Optimization section

below - Oracle Database can automate compression and storage policies based on the usage of the data, reducing storage costs, improving performance and optimizing storage.

At the segment level, Heat Map tracks the timestamps of the most recent modification and query of each table and partition in the database. At the block level, Heat Map tracks the most recent modification timestamp. These timestamps are used by Automatic Data Optimization to define compression and storage policies, which will be automatically maintained throughout the lifecycle of the data. Heat Map ignores internal operations done for system tasks -- automatically excluding Stats Gathering, DDLs, Table Redefinitions and similar operations. In addition, Heat Map can be disabled at the session level, allowing DBA's to exclude manual maintenance, avoiding pollution of Heat Map data.

With the data collected by Heat Map, Oracle Database can automatically compress each partition/table independently based on Heat Map data -- implementing compression tiering to ensure that the current compression level best suits the current usage of the table/partition. This compression tiering can use all forms of Oracle data compression, including: Advanced Row Compression and all levels of Hybrid Columnar Compression (HCC) if the underlying storage supports HCC. Oracle Database can also compress individual database blocks based on Heat Map data.

Automatic Data Optimization

Automatic Data Optimization (ADO) allows organizations to create policies that implement data compression and storage tiering automatically. ADO policies define conditions and corresponding actions to be applied to specific objects. Utilizing the information maintained by Heat Map, Oracle Database executes the registered ADO actions, for the requested objects, to move them to the desired state transparently and automatically.

- ADO policies can be specified at the segment level, or the row level for tables and partitions.
- Segment and row level ADO policies are evaluated and executed automatically in the background during maintenance windows, or they can be executed on demand.
- Storage tiering (Tier To) is specified at the segment level, and is triggered by space pressure in the tablespace where the segment currently resides.

An ADO policy includes specification of the following:

- Which condition will initiate compression -- such as *no access* or no *modification*. Custom conditions can be created based upon specific business requirements.
- When the policy will take effect -- for example, after "x" days (or months or years) of no modification, or "x" days after table/partition creation, or when the tablespace containing the object meets the pre-defined tablespace fullness threshold.

Example ADO Policies:

In this first example, a segment-level ADO policy is created to automatically compress the entire table after there have been no modifications for at least 30 days, using Advanced Row Compression:

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

In this next example, a row-level ADO policy is created to automatically compress blocks in the table, after no rows in the block have been modified for at least 3 days, using Advanced Row Compression:

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

Other ADO policy actions can include data movement to other storage tiers (TIER TO), including lower cost storage tiers or storage tiers with other compression capabilities such as Hybrid Columnar Compression (HCC). HCC requires the use of Oracle Storage – Exadata, SuperCluster, Pillar Axiom, Sun ZFS Storage Appliance (ZFSSA), FS1 or the Oracle Database Appliance (ODA).

In this example, a tablespace-level ADO policy automatically moves the table to a different tablespace when the tablespace currently containing the object meets a pre-defined tablespace fullness threshold:

ALTER TABLE employee **ILM ADD POLICY tier to lowcost;**

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

Additional Capabilities

These are additional capabilities included with Advanced Compression.

Optimization for Flashback Data Archive History Tables

Flashback Data Archive provides the ability to track and store transactional changes to a table over its lifetime. Flashback Data Archive is useful for compliance with record stage policies and audit reports. The Flashback Data Archive feature in Oracle Database (previously known as the Total Recall feature in Oracle Database 11g), provides a mechanism for tracking changes to production databases that is secure, efficient, easy to use and application transparent. Data archived by Flashback Data Archive is maintained in history table(s). The optimization for Flashback Data Archive history tables feature allows users to specify that the history tables be compressed.

Storage Snapshot Optimization

While Recovery Manager (RMAN) remains the most popular method to perform Oracle Database backups, another method for taking database backups is to create a storage snapshot of all of the files in the database (data files, control files, and online redo logs), mount that snapshot on a different system than the one that runs the production database, and copy the data to tertiary storage, such as tape, from that second system.

Snapshots taken this way are “crash-consistent”, provided the storage product adheres to specific guidelines outlined in Oracle documentation. If such a snapshot is restored, Oracle cannot distinguish between that and a database that crashed at the moment the snapshot was taken. Crash-consistent backups can be opened and used after undergoing standard crash (i.e. instance) recovery. However, they cannot be reliably used for point-in-time recovery, as the redo logs do not contain sufficient information for media recovery to remove the data files' inconsistency (since the database was open for writes when snapshot was taken). To perform point-in-time recovery, one must adhere to the requirements and strictly follow the manual procedures as outlined in Support Note 604683.1.

Alternatively, snapshots taken in backup mode, i.e. ALTER DATABASE [BEGIN | END] BACKUP, remove the point-in-time recovery restriction, as additional information is written into the redo logs to remove the data files' inconsistency upon recovery. Since each database needs to be placed in this mode before snapshot is taken and taken out of this mode when the snapshot completes - this complexity is magnified when this procedure must be done for tens, hundreds, or thousands of databases. In addition, during this mode, whole block images are written to redo logs as they are changed, inducing additional I/O activity - this overhead is magnified when you have many databases running on the same array.

With the Oracle Database RECOVER .. SNAPSHOT TIME command, available with Advanced Compression, storage snapshots taken without the database in backup mode can be recovered in one-step, whether to the current time or a specific point-in-time after the snapshot was taken, without any additional procedures. By supporting all types of recovery operations using these snapshots, this optimization effectively eliminates the need for backup mode and its associated complexity and overhead, freeing the DBA's time to focus on more critical production tasks.

Hybrid Columnar Compression Row Level Locking

Oracle's Hybrid Columnar Compression (HCC) technology is a method for organizing data within a set of database blocks. HCC utilizes a combination of both row and columnar methods for storing data. A logical construct, called the Compression Unit (CU), stores a set of HCC-compressed data. When data is loaded, groups of rows are stored in columnar format, with the values for a given column stored and compressed together. After the column data for a set of rows has been compressed, it is fit into the compression unit.

Hybrid Columnar Compression uses one lock per CU. Optionally; users can choose to enable Row Level Locking for Compression Units. The default with HCC is NO ROW LEVEL LOCKING, ROW LEVEL LOCKING is explicitly specified during a CREATE TABLE or ALTER TABLE MOVE operation.

Exadata Flash Cache Compression

Exadata Flash Cache compression dynamically increases the logical capacity of the flash cache by transparently compressing user data as it is loaded into the flash cache. This allows much more data to be kept in flash, and decreases the need to access data on disk drives. The I/Os to data in flash are orders of magnitude faster than the I/Os to data on disk. The compression and decompression operations are completely transparent to the application and database, and have no performance overhead, even when running at rates of millions of I/Os per second.

Depending on the user data compressibility, Oracle Exadata Storage Server Software dynamically expands the flash cache size up to two times. Compression benefits vary based on the redundancy in the data. Tables and indexes that are uncompressed have the largest space reductions. Tables and indexes that are OLTP Table compressed have significant space reductions. Tables that use Hybrid Columnar Compression have minimal space reductions.

Online Move Table/Partition/Subpartition (to a compressed format)

ALTER TABLE ... MOVE TABLE/PARTITION/SUBPARTITION … ONLINE enables tables, partitions and/or subpartitions to be moved as an online operation without blocking any concurrent DML operations. The partition/subpartition move operation supports automatic index maintenance as part of the move, so no index rebuild is required. The "online" feature of this MOVE operation is included with Advanced Compression.

Conclusion

The massive growth in data volume, experienced by enterprises, introduces significant challenges. Companies must quickly adapt to the changing business landscape without influencing the bottom line. IT managers need to efficiently manage their existing infrastructure to control costs, yet continue to deliver extraordinary application performance.

Advanced Compression, along with Oracle Database together provide a robust set of compression, performance and data storage optimization capabilities that enable IT managers to succeed in this complex environment.

Using Advanced Compression, enterprises can efficiently manage their increasing data requirements throughout all components of their data center – minimizing CapEx and OpEx costs while continuing to achieve the highest levels of application performance.

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.

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