

ORACLE®



# Get the Best Out of Oracle Database Compression



Gregg Christman  
Database Performance  
Product Manager

MAY 16 & 17, 2018

CLEVELAND PUBLIC AUDITORIUM, CLEVELAND, OHIO

[WWW.NEOOUG.ORG/GLOC](http://WWW.NEOOUG.ORG/GLOC)

# Safe Harbor Statement

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.

# Program Agenda

- 1 ➤ Advanced Row Compression
- 2 ➤ Advanced Index Compression
- 3 ➤ Hybrid Columnar Compression
- 4 ➤ Other Compression Stuff...
- 5 ➤ More Information



# Advanced Row Compression Insights

Oracle Database 18c Database Storage Optimization

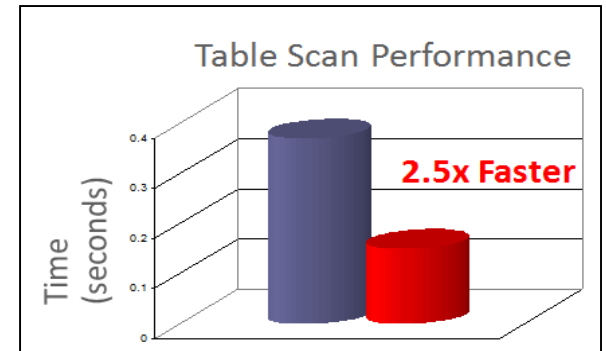
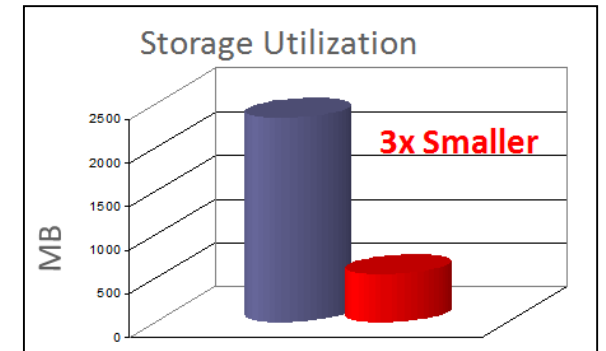
# When to Use Advanced Row Compression

- **Ideal for OLTP/DW Data**

- *Customers experience: 2x to 4x compression ratios*

- **Queries run directly on compressed data -- scans and reporting run faster...**

- Compressed data/indexes stay compressed in memory
  - Helps improve performance due to the reduction in I/O and the reduction in system calls related to the I/O operations
  - Maintains compression across all DML operations and bulk load operations
- 2.5x better query performance is typical
  - Everything is faster: table scans, backups, database cloning, etc
  - Buffer cache becomes more efficient by storing more data without having to add memory



# When Compression Occurs....

## Insert Direct Load Compression

- Triggered when the data is inserted using the bulk load mechanisms **such as insert with an append hint or using a SQL loader**
  - Data is inserted **above the segment high water mark** (a virtual last used block marker for a segment) and can be written out to data blocks very efficiently
  - The compression engine is able to **buffer, compress and write** out compressed rows to the data block. As a result, the *space savings are immediate*
    - *Rows are never written in an uncompressed format* for Insert Direct Load compression



# When Compression Occurs....

## Recursive Compression

- **Invoked on conventional DML operations**, such as single row or array inserts and updates
  - These DML compression operations ***write out the rows in uncompressed format***, and when the **data blocks reach internal block fullness threshold compression is invoked**
  - Under such scenario, Oracle is able to compress the data block in a recursive transaction which is committed immediately after compression
    - The space saved due to compression is immediately released and can be used by any other transaction
    - **Compression is triggered by the user DML operation** (user transaction), but actual compression of data happens in a recursive transaction
    - The fate of compression is not tied to the fate of the user transaction



# When Compression Occurs....

## Inline Compression

- Also designed for conventional DML operations, such as single row or array inserts and updates, which write out the rows in uncompressed format first
  - When the data blocks reach an internal block fullness threshold, compression is invoked, compressing all the uncompressed rows in the data block
- The main difference between Recursive compression and Inline compression is that Inline compression is designed to **compress data when all the rows in the block are actively modified or locked by the current user transaction**
  - **Compression is in-line with the user transaction triggering it.** This is done to ensure that there is no loss in the concurrency
  - **The space saved by Inline compression can be used by the compressing transaction anytime, but will be available for other transactions only when this transaction commits**
  - Once it commits, the space is released for other transactions. Additionally, if the user transaction rolls back, the compression of this data block is also rolled back

# Compression During Updates (more details)

- **When are Updates Compressed?**

- After compression has taken place, if a row that has been compressed is updated, then the resulting row can be re-compressed only if another compression is triggered
- If all free space has been exhausted or subsequent inserts don't trigger another compression then the rows that have been updated will stay uncompressed
- Subsequent updates will cause row migrations once all space in the block is exhausted (i.e. the reserved space from the **PCTFREE** setting)

# Partial Compression

- **Partial Compression on Already Compressed Blocks**

- With Advanced Row Compression, when the block is full, it is compressed. More rows are then 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 in **Oracle Database 12c Release 2 and above** the block can be partially compressed -- resulting in CPU savings and extra compression
  - The partial compression feature 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
  - Partial compression locks and compresses only those rows that are uncompressed and unlocked - hence it can take place in the presence of other uncommitted transactions in the block



# Other Changes to Compression

See: <https://blogs.oracle.com/DBStorage/>  
(for New Features Discussion)

- **Better Chained Row Support**

- Before Oracle Database 12cR2, blocks containing many types of chained rows could not be compressed. This limitation has been removed in Oracle Database 12c Release 2 and above

- **Background Compression**

- Background space tasks can now compress blocks if these find them as 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

- **Array Insert Compression**

- Before Database 12c Release 2, array inserts (such as INSERT ... AS SELECT ) caused multiple recompressions per block
  - In Database 12c Release 2 and above, Oracle now estimates the number of rows that would fit into a compressed block. All these rows are then buffered, compressed, and a full block image is generated
  - Compression occurs only once or twice -- as opposed to occurring potentially many times

# Tracking Compression with AWRs

## Advanced Row Compression

- HSC OLTP positive compression + HSC OLTP negative compression** = Total number of attempted compressions and re-compressions (non-direct load)
- HSC OLTP Compressed Blocks** = Incremented first time a block a compressed
- HSC IDL Compressed Blocks** = Block compressions above the HWM (such as in CTAS or insert append)
- ((HSC OLTP positive compression + HSC OLTP negative compression) - (HSC IDL Compressed Blocks + HSC OLTP Compressed Blocks))** = number of attempted compressions and re-compressions re-compressions (including direct loads)

See: <https://blogs.oracle.com/DBStorage/> (critical OLTP Statistics)

Statistic Name	Description
HSC Compressed Segment Block Changes	Total number of block changes to Tables/Heaps Segments (Compressed only).
HSC Heap Segment Block Changes	Total number of block changes to Tables/Heaps Segments (Compressed or Non-Compressed).
HSC IDL Compressed Blocks	Number of blocks compressed during Insert Direct Load, actual number of blocks.
HSC OLTP Compressed Blocks	Number of blocks compressed during DML (inserts and updates) activity - actual number of blocks.
HSC OLTP Non Compressible Blocks	Blocks marked as Final (Not to be compressed again unless substantial changes to the data in the block).
HSC OLTP Space Saving	Number of bytes saved in total using OLTP Table Compression. Take the delta on every compression and adds.
HSC OLTP inline compression	Number of Inline Compressions. Inline Compression is compression of a block inline in the user transaction. Space released is associated (held reserved) with the user transaction and can only be used by this transaction till the user transaction commits. Once the user transaction commits, the space is released for other transactions.  Total number of block compressions = Inline + Recursive + IDL Compressions
HSC OLTP recursive compression	Number of Recursive Compressions. Recursive Compression is compression of a block in a recursive transaction which is immediately committed. Space released is not associated (not reserved) with the user transaction triggering compression and can be used by any transaction immediately.  Total number of block compressions = Inline + Recursive + IDL Compressions
HSC OLTP positive compression	Number of times compression was beneficial (post compression block had more free space).
HSC OLTP negative compression	Number of times compression was negative (post compression block had less free space) and was reverted back.
HSC OLTP Drop Column	Number of compression attempts due to drop column.
HSC OLTP Compression skipped rows	Number of rows that are skipped for compression (could be deleted or non-compressible etc.).
Heap Segment Array Inserts	Number of array inserts into Heap Segments.
Heap Segment Array Updates	Number of array updates into Heap segments.

# Advanced Index Compression Insights

Oracle Database 18c Database Storage Optimization

# When to Use Prefix (Index Key) Compression

- **Included with Oracle Database Enterprise Edition**

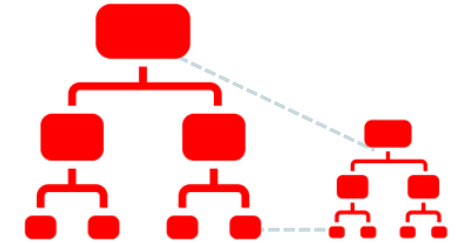
- Eliminates duplicate copies of pre-defined number of index prefix columns at the index leaf block level
  - Effective way to permanently reduce the index size, both on disk and in cache.
  - ***Supports Index Organized Tables (IOT's)***
- The number of prefix columns to consider for compression is specified by the DBA at the index create time (or rebuild time) and is constant for all index leaf blocks
  - Compression can be very beneficial when the prefix columns of an index have many repeated rows within a leaf block
- ***ANALYZE INDEX*** will give advice on whether / how many columns to choose
- *Customers experience: 2x compression ratios*

**Example:**

```
CREATE INDEX idxname ON tablename(col1, col2, col3)  
COMPRESS;
```



# When to Use Advanced Index Compression



- **The optimal compression algorithms will be computed on a block by block bases**
  - Produces the best compression ratio possible
  - Now possible to have different index leaf blocks compressed differently, or not to be compressed at all
- **Advanced High (12.2.0.1+)**
  - Heavyweight compression of data and Oracle metadata using a variety of compression techniques
  - Higher compression ratio than Advanced Low
    - *Customer Experience: 4-5x, highly compressible indexes 15-20x*
- **Advanced Low (12.1.0.1+)**
  - LOW must be specified explicitly in 12.2+, as in:  
CREATE INDEX i on t(c1..cn) **COMPRESS ADVANCED LOW;**
  - *Customers experience: 2x+ compression ratios*

## Example:

```
CREATE INDEX idxname ON  
tablename(col1, col2, col3) COMPRESS  
ADVANCED HIGH;
```

# Index Compression Tips and Insights

- **Prefix Compression**

- Relies on DBA to choose the most optimal prefix column count
  - Specified prefix column count may not be optimal to produce the best compression ratio for every block in the index
- Requires running ANALYZE INDEX to obtain the optimal prefix column count, which produces the optimal count for the index as a whole
  - This is not at the granularity of a block, so may not yield the best compression ratio

- **Advanced Index Compression**

- Advanced Index Compression is not supported on Bitmap Indexes
- Cannot compress Index Organized Tables (IOTs) with Advanced Index Compression
  - IOTS can be compressed with free Prefix Compression
- Cannot compress Functional Indexes with Advanced Index Compression

See: <https://blogs.oracle.com/DBStorage/> (for Index Compression discussion)

# Hybrid Columnar Compression Insights

Oracle Database 18c Database Storage Optimization

# When to Use Hybrid Columnar Compression (HCC)

- **Ideal for DW and Some OLTP Applications**

- Prior to 12.2, primarily intended for data with no/few modifications
  - DML INSERTS/UPDATES would reduce compression ratio
- With 12.2 and above 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
- *Customers experience: 6x to 50x compression ratios (some as high as 200x)*
- HCC compressed data stays compressed in memory

- **Compression Levels Balance Query Performance and Storage Reduction**

- **QUERY** compression level best for scan query performance -- 6x to 10x compression ratios typical
- **ARCHIVE** compression best for long term data management and data archival -- 10x to 15x compression ratios typical

*HCC Requires: Exadata, SuperCluster, Pillar Axiom, ZFSSA storage or FS1*

# Tracking Compression with AWRs

## HCC Compression

- **EHCC Rows Compressed = Total number of rows compressed using HCC compression**
- **EHCC Rows Not Compressed = Total number of rows not compressed with HCC**
- **EHCC Total Rows Decompressed = Total number of rows decompressed counting every decompression**
- **EHCC CUs Compressed = Total number HCC CU's compressed**

Statistic Name	Description		
EHCC Analyzer Calls	Number of times HCC Analyzer was called to pick the optimal compression algorithms	EHCC Compressed Length	Total number of bytes compressed using HCC compression
EHCC Archive High CUs compressed	Number of HCC CUs compressed at Archive High level	EHCC Decompressed Length	Total number of bytes decompressed. Every decompression is counted
EHCC Archive High CUs decompressed	Number of HCC CUs decompressed at Archive High level. Every decompression is counted	EHCC Check CUs Decompression	Number of times CUs decompressed by block checking
EHCC Archive Low CUs compressed	Number of HCC CUs compressed at Archive Low level	EHCC Dump CUs Decompressed	Number of times CUs decompressed for block dumps
EHCC Archive Low CUs decompressed	Number of HCC CUs decompressed at Archive Low level. Every decompression is counted	EHCC ROWID CUs Decompressed	Number of times ROWID CUs were decompressed
EHCC Query High CUs compressed	Number of HCC CUs compressed at Query High level	Cell CUS sent compressed	Number of CUs sent compressed from Exadata cells to server to be processed
EHCC Query High CUs decompressed	Number of HCC CUs decompressed at Query High level. Every decompression is counted	Cell CUs sent head uncompressed	Number of CUs sent uncompressed from Exadata cells to server to be processed
EHCC Query Low CUs compressed	Number of HCC CUs compressed at Query Low level	Cell CUs sent head piece	Number of CUs sent head piece from Exadata cells to server to be processed
EHCC Query Low CUs decompressed	Number of HCC CUs decompressed at Query Low level. Every decompression is counted	EHCC CUs no rows pass minmax	Number of CUs for which can be pruned since no row passes min-max values
EHCC CUs compressed	Total number of HCC CUs compressed	EHCC Columns Decompressed	Number of column CUs decompressed due to selective decompression
EHCC CUs decompressed	Total number of HCC CUs decompressed. Every decompression is counted	EHCC Pieces Buffered for Decompression	Number of CUs buffered for decompression
EHCC Rows Compressed	Total number of rows compressed using HCC compression algorithms	EHCC Used on ZFS Tablespace	Tracks HCC usage on ZFSSA storage
EHCC Rows Not Compressed	Total number of rows not compressed with HCC	EHCC Used on Pillar Tablespace	Tracks HCC usage on Pillar storage
EHCC Total Rows Decompressed	Total number of rows decompressed counting every decompression		

See: <https://blogs.oracle.com/DBStorage/> (critical HCC Statistics)

# More [**interesting**] Compression Stuff

Encryption, Enabling Compression and More....

# Compression and Encryption

## Tablespace Encryption

- *Data and index compression are done before encryption* -- this ensures the maximum space and performance benefits from compression, while also receiving the security of encryption at rest

## Column Encryption

- *Compression is done after encryption.* This means that compression will have minimal effectiveness on encrypted columns
- There is one notable exception: if the column is a Secure Files LOB, and the encryption is implemented with Secure Files LOB Encryption, and the compression (and possibly deduplication) are implemented with Secure Files LOB Compression & Deduplication, then compression will be done before encryption

## Querying Encrypted/Compressed Data

- When selecting data, it is first decrypted from the result set, then the result set is decompressed



# Enabling Compression: Now More *Online* Choices...

## ALTER TABLE ... MOVE TABLE/PARTITION/SUBPARTITION ... **ONLINE**

- Enables Advanced Row Compression/HCC for future DML and also 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
  - New with Oracle Database 12c Release 2 and above, move *tables* online as well as *partitions/subpartitions*
  - Using the **UPDATE INDEXES** clause maintains **both local and global indexes** during the move -- so a manual index rebuild is not required

**Tip:** *Enabling compression online, or offline, doesn't impact the compression ratio...*

See: <https://blogs.oracle.com/DBStorage/> (for enabling compression discussion)

# Common Compression Questions

- **Do I have to compress all the tables/partitions in my database?**
  - No – You can pick and choose which tables/partitions you want to compress and you don't have to compress everything at one time
- **What is the overhead associated with Advanced Row Compression?**
  - Approximately **3% to 5% CPU** is typically reported by customers. CPU overhead offset partially by reduced IO
- **Who is the typical Advanced Compression customer?**
  - Oracle compression is used by financial, government, education, healthcare, utilities, insurance, retail, manufacturing and more....
- **My storage has compression, why not just use that?**
  - Oracle compression is built into the database, this allows Oracle to keep data and indexes compressed in memory – this isn't possible with storage-based compression
- **After I enable compression, is there any other admin work we need to do to maintain compression?**
  - Nope. Once compression is enabled there isn't any maintenance required to maintain the compression of your tables and partitions

# Other Things You Should Watch For...

- **Row Chaining**

- Make sure PCTFREE is adequate for amount of updates – default is 10
- Running out of PCTFREE could lead to row chaining which impacts performance

- **Conventional Path Bulk Loads**

- Direct path is optimized for compression (provides better load performance):
  - When performing bulk loads, specify *insert /\*+ append \*/* for better performance
  - Use direct path bulk loads (CTAS, Append Hint, SQL Loader and etc...) when possible

- **Un-Realistic Test Cases**

- Best test environment for each Advanced Compression capability is where you can most closely duplicate the production environment
- This provides the most realistic (pre- and post- compression) performance and functionality comparisons

# More Other Things You Should Watch For...

- **Prefix Compression Bloat**

- When the leading columns are very selective, or if there are not many repeats for the prefix columns, it is possible to make indexes larger than their uncompressed equivalent due to the overhead to store the prefix entries
- Important to compress the right indexes and with correct number of prefix columns
  - Advanced Index Compression won't compress if there is no advantage in doing so

- **Pre-11.2 Update Bloat w/ Compression**

- With 11gR1 it was possible that updates would cause tables/partitions to grow in size
- Optimizations in 11gR2 removed update bloat issue – make sure early versions have ACO patches

- **Compressed LOB Compression**

- LOBS that may not have compressed well in-line may compress better as Secure Files LOBS
- If LOB was already compressed unlikely Oracle will get much, if any, additional compression

# More Information

**Just In case I didn't Answer All Your Questions....**

# Proof-of-Concept (PoC)

## Prepare Yourself for a Successful PoC



ORACLE ADVANCED COMPRESSION BEST PRACTICES

### Oracle Advanced Compression Proof-of-Concept (POC) Insights and Best Practices

**FEATURES TYPICALLY EVALUATED**

- **Advanced Row Compression**  
Enables table data to be compressed during all types of data manipulation operations
- **Index Key Compression and/or Advanced Index Compression**  
Reduces the size of all supported unique and non-unique indexes
- **Backup Data Compression**  
Compression for backup data when using Oracle Recovery Manager (RMAN) or Oracle DataPump
- **Advanced LOB Compression and Deduplication**  
Provides compression and deduplication for LOBs managed by SecureFiles

**Oracle Advanced Compression**

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

The Oracle Advanced Compression option 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.

Whether it is a cloud or an on-premise 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.

**What is Useful to Know for Your Compression POC**

This document isn't meant to be a step-by-step guide to performing a compression POC. Instead, this document is intended to provide some best practices learned from customer POC's and to provide insights to help plan your compression POC, as well as help you understand the results of your POC.

**Enabling Advanced Row Compression**

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;
```

There are numerous ways to enable Advanced Row Compression for existing tables. While a complete discussion of each method is beyond the scope of this document, this document does provide an overview of the methods typically used by customers. For more information on these methods, please see the Oracle Database documentation.

**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.

## Key Points:

- Discusses the topics that we've covered in this session in greater detail
- Provides a basic (very simple) template for planning PoC



<http://www.oracle.com/ocom/groups/public/@otn/documents/webcontent/3411538.pdf>

# Compression Advisor

Get “Hands-On” Right Away

## Use Database Cloud to Create a Sandbox for Testing Compression

- Instant access to fully integrated infrastructure
- Scale up and scale out easily and rapidly for functional & performance validation testing
- Fully compatible for easy migration back to on-premise or cloud deployment

## Suggest the **Free Compression Advisor** to Estimate Compression Ratios

- Free to use, no additional license required
- Starts customers thinking about storage savings
- Use with Oracle Database 11g Release and later
- Works with data, indexes and LOBS



# Compression Advisor Tips and Insights

- **When Estimating Compression Ratios**

- If get this type of message when estimating HCC compression ratios:

- ORA-12801: error signaled in parallel query server P002

- ORA-64307: Exadata Hybrid Columnar Compression is not supported for tablespaces on this storage type




- Solution:** Disable parallel processing for the session (set parallel\_max\_servers=0)

- Tables residing in uniform tablespaces can be compressed. However, the compression adviser has the restriction that the scratch tablespace cannot be uniform
- Advisor creates two temp tables for Advanced Row Compression estimates and four temp tables for HCC estimates
- In earlier releases Oracle did require 1M rows in a table for estimating HCC compression ratios with advisor – this restriction is lifted in 12.1.0.2.
  - **Note:** Outside advisor there are no restrictions with Hybrid Columnar Compression in regards to the minimal amount of data needed (in tables/partitions) with HCC

# Additional Resources



## Join the Conversation

-  [https://twitter.com/aco\\_gregg](https://twitter.com/aco_gregg)
-  <https://blogs.oracle.com/DBStorage/>
-  <http://www.oracle.com/database/advanced-compression/index.html>

## Advanced Compression Case Studies

- [Goodman Fielder \(SAP user\)](#)
- [Suguna Foods \(EBS user\)](#)

## Related White Papers

- [Oracle Advanced Compression White Paper](#)
- [Advanced Compression Helps Fortune 500 Company](#)
- [Automating Compression Tiering and Storage Tiering](#)
- [Oracle E-Business Suite with Advanced Compression](#)

## Additional Information

- [Oracle Index Compression](#)
- [Database Storage Optimization \(Oracle.com page\)](#)
- [Advanced Compression Savings Tool](#)
- [Compression Advisor Information](#)
- [Hybrid Columnar Compression](#)

## Any Additional Questions

- [Oracle Storage Optimization Blog](#)

# More Information....

## MOS notes....

- Advanced Compression Master Note (Doc ID 1223705.1)
- How to compress a table that is online (Doc ID 1353967.1)
- **Advanced Compression critical patches** (Doc ID 1061366.1)
- Redo Transport compression with Data Guard (Doc ID 729551.1)
- **How to see if rows are compressed in a table** (Doc ID 1477918.1)
- A complete understanding of RMAN compression (Doc ID 563427.1)
- How to determine if Advanced Compression is used by DataPump (Doc ID 1993134.1)
- Index Organized Tables (IOTs) and compression (Doc ID 1555637.1)
- **How to find the optimal Index Key COMPRESS level for Indexes** (Doc ID 601690.1)

# Summary of Data Compression Types

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).

# Integrated Cloud

## Applications & Platform Services

ORACLE®