Oracle Database: In-Memory Solution

Shilpa Rajpurkar
DBA Lead Consultant – Oracle

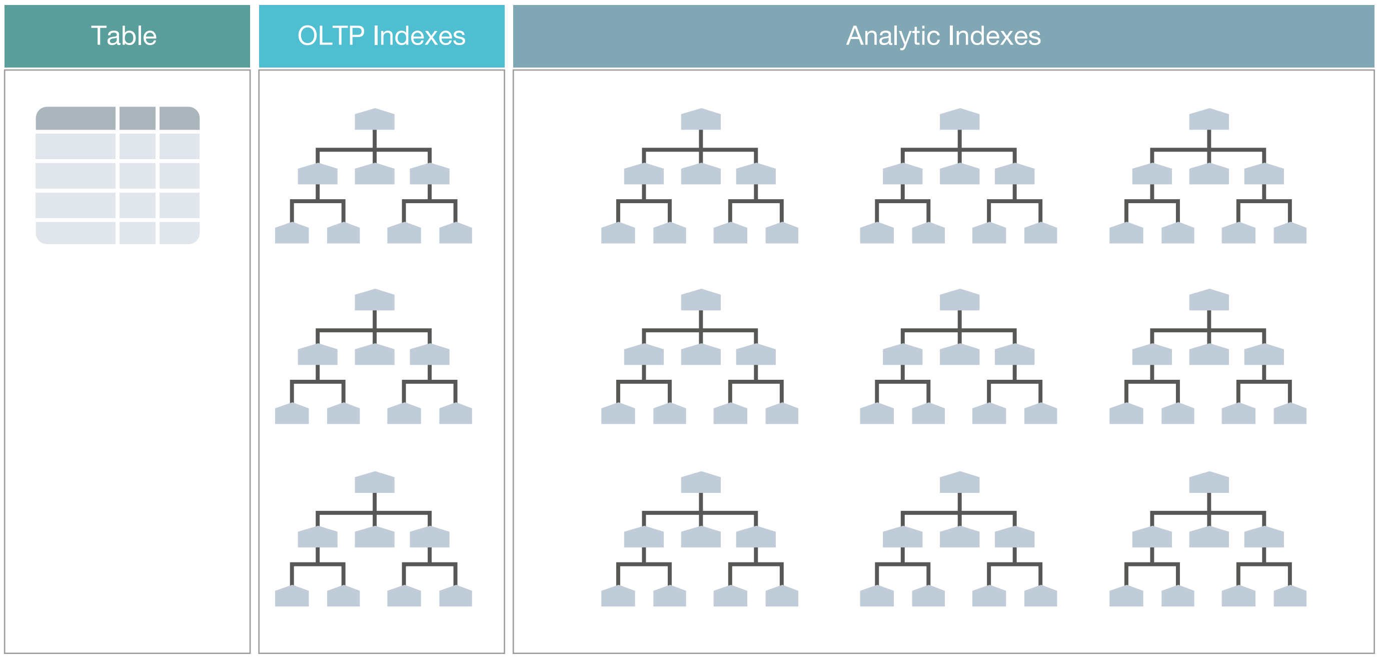
Oracle Database In-Memory Solution

Oracle Database In-Memory is a suite of features, first introduced in Oracle Database 12c Release 1 (12.1.0.2), that greatly improves performance for real-time analytics and mixed workloads. The In-Memory Column Store (IM column store) is the key feature of Database In-Memory.

Analytic Application - Challenges and Solutions

Traditionally, obtaining good performance for analytic queries meant satisfying a number of requirements. You must understand user access patterns. You must provide good performance, which typically requires creating indexes, materialized views, and OLAP cubes.

For example, if you create few indexes for a table (primary key and foreign key indexes) to improve performance for an OLTP application, additional indexes may be still required to provide good performance for analytic queries.



Additional access structures (indexes) cause performance overhead because you must create, manage, and tune them. For example, inserting a single row into a table requires an update to all indexes on this table, which increases response time.

The demand for real-time analytics means that more analytic queries are being executed in a mixed-workload database. The traditional approach is not sustainable.

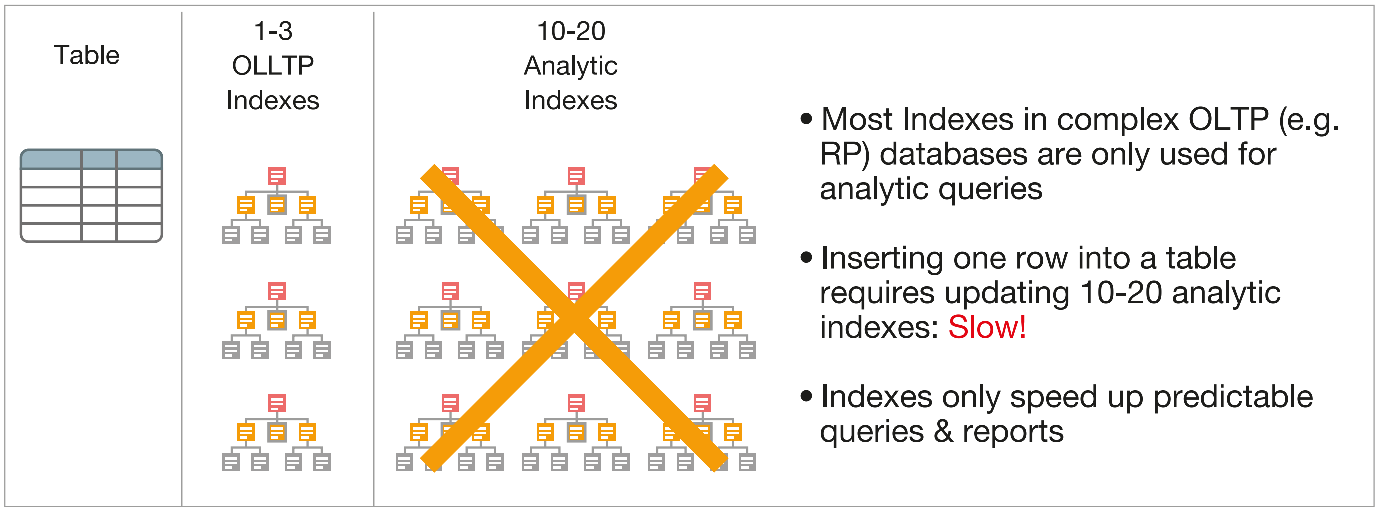


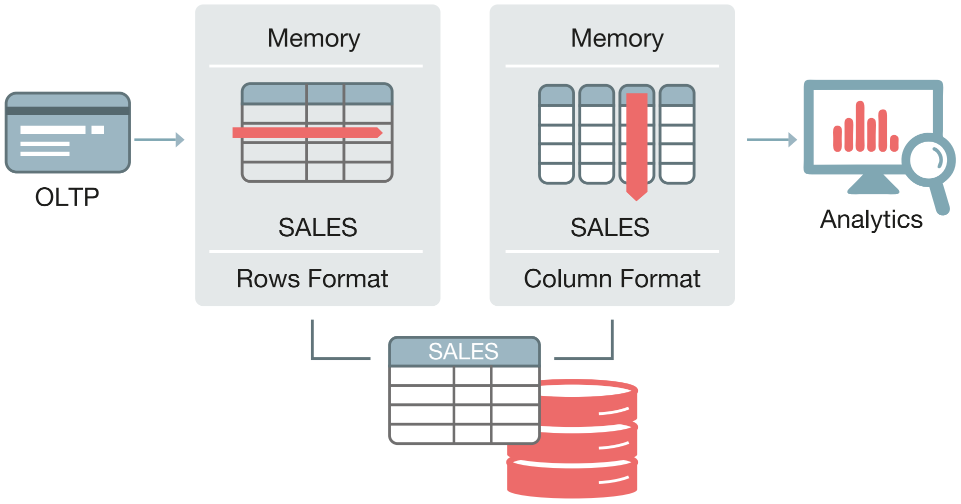
Table
1-3
OLLTP
Indexes
10-20
Analytic
Indexes
• Most Indexes in complex OLTP (e.g. RP) databases are only used for analytic queries
• Inserting one row into a table requires updating 10-20 analytic indexes: Slow!
• Indexes only speed up predictable queries & reports

Row or Columnar Format

An Oracle database stores rows contiguously in data blocks. For example, in a table with three rows, an Oracle data block stores the first row, followed by the second row, and then the third row. Each row contains all the column values for that specific row. Data stored in row format is optimized for transaction processing.

To resolve performance issue relating to analytic queries, a columnar format was introduced. A columnar database stores selected columns, not rows, contiguously. For example, in a large sales table, the sales IDs reside in one column, and sales regions reside in a different column.

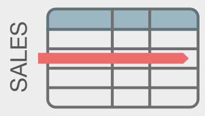
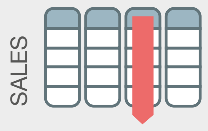
Analytical queries access few columns, but scan the entire data set. So, the columnar format is the most efficient one for analytics. As columns are stored separately, an analytical query can access only required columns, and avoid reading unnecessary data.



Database vendors typically force their customers to choose between a columnar and row-based format.

There is trade-off between the two formats, gaining the advantages of one format means losing the advantages of the other alternate format. Applications either achieve rapid analytics or rapid transactions, but not both. So, there doesn't exist as single solution that solves the problem of mixed-use database.

Database Storage: Row Format Versus Column Format

Row		**Transactions** run faster in row format Row format is best for fast processing of few rows and many columns Example: Insert or query on sales or order
Column		**Analytics** run faster in row format Column format is best for fast accessing of few columns and many rows Example: Report sales totals by region

Database In-Memory: Concept

The Database In-Memory feature includes the IM column store, advanced query optimizations, and availability solutions. These features combine to speed up the analytic queries without sacrificing OLTP performance or availability.

In-Memory Column

Oracle Database In-Memory enables data to be simultaneously populated in memory in both row format (in the buffer cache) and a new In-Memory column format. The Oracle Database query optimizer is fully aware of the column format: it automatically routes analytic queries to the column format and OLTP operations to the row format, ensuring outstanding performance and complete data consistency for all workloads without any application changes.

The database maintains full transactional consistency between the row and column formats, just as it maintains consistency between tables and indexes. In this approach, there is only a single copy of the table in storage, so there are no additional storage costs or synchronization issues.

INMEMORY clause is used in DDL statements to enable the IM column to be stored at any of the following levels:

- Column (non-virtual or virtual)
- Table, materialized view, or partition
- Tablespace

If the INMEMORY attribute is specified at the tablespace level, then by default all new tables and materialized views in the tablespace are enabled for the IM column store. Here, row-based data on disk is automatically transformed into columnar data in the IM column store. You can configure all, or a subset of a database object's columns for population in the IM column store.

In-Memory Area

Database In-Memory uses an IM column store, which is a new component of the Oracle Database System Global Area (SGA), called the In-Memory Area. The IM column store does not replace the buffer cache, but acts as a supplement, so data can now be stored in memory in both a row and a column format.

The In-Memory area is sub-divided into two pools - a 1MB pool used to store the actual column formatted data populated into memory, and a 64K pool used to store metadata about the objects that are populated into the IM column store.

The amount of available memory in each pool is visible in the V$INMEMORY_AREA view. The relative size of the two pools is determined by internal heuristics; the majority of the In-Memory area memory is allocated to the 1MB pool.

Query to find Memory allocation

```
Select pool,alloc_bytes,used_bytes,populate_status from v$INMEMORY_AREA;
```

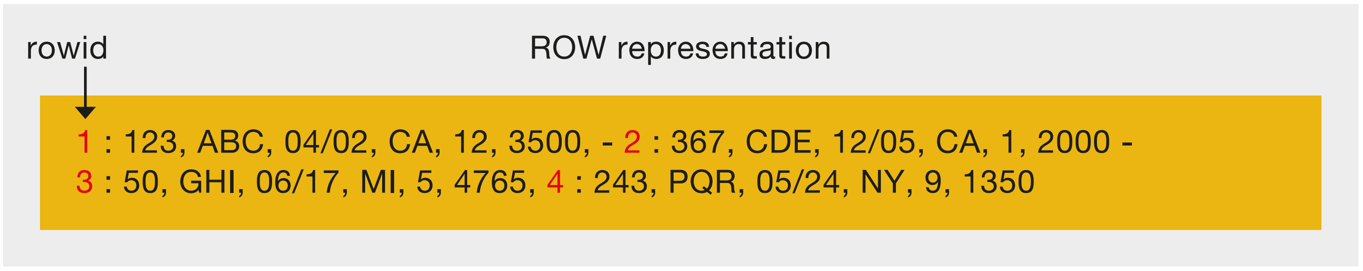
Architecture

To understand IN-Memory column store architecture, we will consider the example of sales table.

Given below is the visual representation of the Sales table that has 4 rows and 6 columns -

PROD ID	CUST ID	TIME ID	STATE	QTT	AMOUNT
123	ABC	04/02	CA	12	3500
367	CDE	12/05	CA	1	2000
50	GHI	06/17	MI	5	4765
243	PQR	05/24	NY	9	1350

Below diagram shows the row representation of table. Each entry starts with rowid followed by values for each columns -


rowid
ROW representation
1 : 123, ABC, 04/02, CA, 12, 3500, - 2 : 367, CDE, 12/05, CA, 1, 2000 -
3 : 50, GHI, 06/17, MI, 5, 4765, 4 : 243, PQR, 05/24, NY, 9, 1350

In columnar representation the way the data is stored is different from above. Each entry has the values from one column and with each value there is a rowid for identifying the row that column value belongs to.

Cloumnar representation

123 : 1, 367 : 2, 50 : 3, 243 : 4 - ABC : 1, CDE : 2, GHI : 3, PQR : 4 -
04/02 : 1, 12/05, :2, 06/17 : 3, 05/24 : 4 - CA : 1 ;2, Mi : 3, NY : 4 -
12 : 1, 1 : 2, 5 : 3, 9 : 4 - 3500 : 1, 2000 : 2, 4765 : 3, 1350 : 4

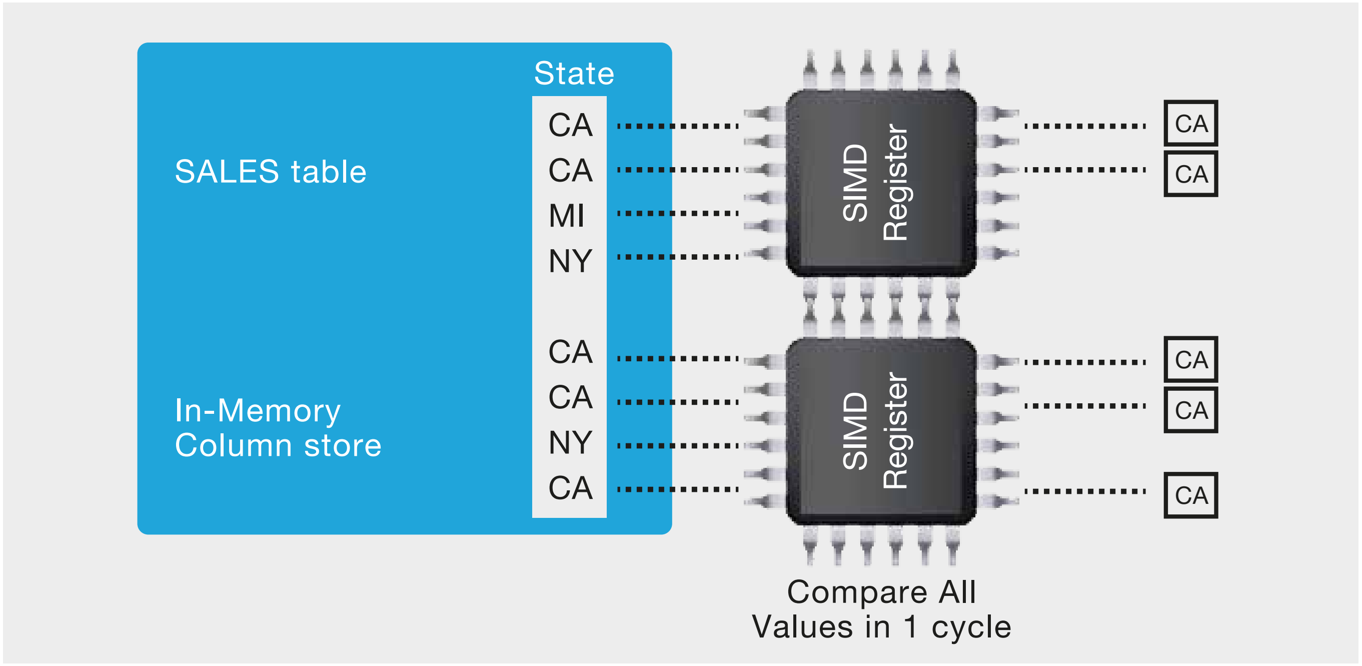
One of the advantages of this representation is the possibility to compress data. Row 1 and 2 has same value for state columns i.e., 'CA'. Below columnar representation compresses both values as shown in the circle.

Cloumnar representation

123 : 1, 367 : 2, 50 : 3, 243 : 4 - ABC : 1, CDE : 2, GHI : 3, PQR : 4 -
04/02 : 1, 12/05, : 2, 06/17 : 3, 05/24 : 4 - CA : 1 ; 2 , Mi : 3, NY : 4 -
12 : 1, 1 : 2, 5 : 3, 9 : 4 - 3500 : 1, 2000 : 2, 4765 : 3, 1350 : 4

The other advantage of this representation is vector processing using Single Instruction processing and Multiple Data value (SIMD) processing.

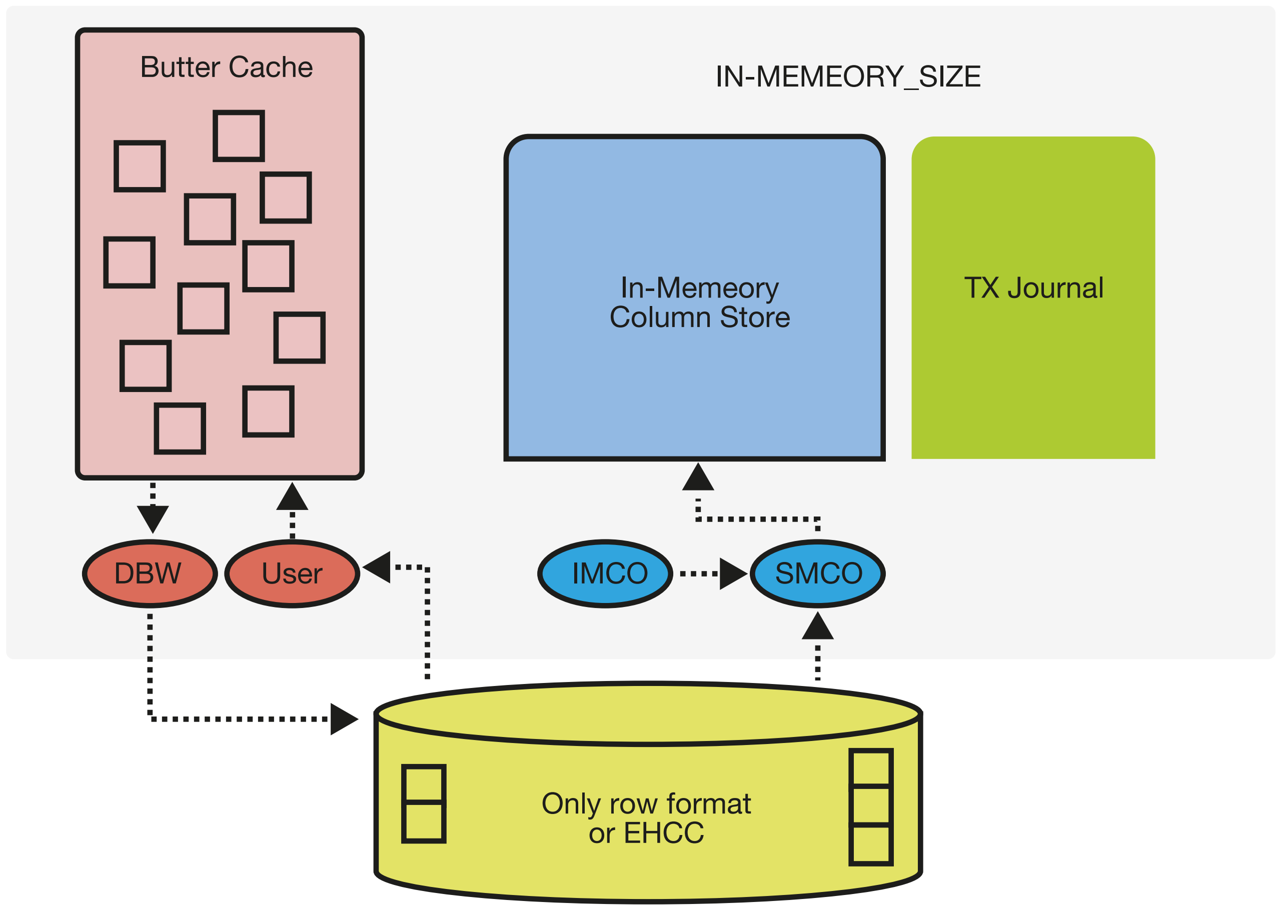
SELECT count (*) FROM sales WHERE state='CA'



This solution is simple to implement, requires no application changes and also supports all High Availability features.

IM Column store is faster than scanning of row based data due to:

- No buffer cache overhead: In IM columns store, data is not stored in data file it is purely available in, IN Memory columnar format. No redo or physical read is generated.
- Scanning is done only for the columns necessary for the query rather than entire row of data. Additionally, the database uses storage indexes and an internal dictionary to read only the necessary IMCUs for a specific query. For example, if a user request all sales for a state with a state ID in specific range, then the database can use IMCU pruning to eliminate IMCUs that do not contain this value.
- Here, compression improves the scanning process. The volume of data that the database must scan in the IM column store is less than the corresponding volume in the database buffer cache.



Background Process

The IM column store is populated by a set of background processes referred to as worker processes (ora_w001_orcl). While this occurs the database is fully available, however it is not the case with a pure in-memory database. Each worker process is given a subset of database blocks from the object to populate into the IM column store. Population is a streaming mechanism, which columnizes and compresses the data simultaneously. The IM column store is made up of multiple In-Memory Compression Units (IMCUs) similar to extent in tablespace. Each worker process allocates its own IMCU and populates its subset of database blocks in it. It is read in the same order as it appears in the row format, no sorting or ordering of data is done during population.

Objects are populated into the IM column store either in a prioritized list immediately after the database is opened or after they are scanned (queried) for the first time. The order in which objects are populated is controlled by the keyword PRIORITY, which has five levels. The default PRIORITY is NONE, which means an object is populated only after it is scanned for the first time. All objects at a given priority level must be fully populated before the population for any objects at a lower priority level can commence. However, the population order can be superseded if an object without a PRIORITY is scanned, triggering its population into IM column store.

```
ALTER TABLE customers INMEMORY PRIORITY CRITICAL
```

Priority can be -

Critical: Object is populated immediately after the database is opened.

High: Object is populated after all CRITICAL objects have been populated, if space remains available in the IM column store.

Medium: Object is populated after all CRITICAL and HIGH objects have been populated, and space remains available in the IM column store.

Low: Object is populated after all CRITICAL, HIGH, and MEDIUM objects have been populated, and if space remains available in the IM column store.

None: Objects are only populated after they are scanned for the first time (Default), if space is available in the IM column store.

Limitation

Below objects and datatypes are not supported by this format:

- Object owned by the SYS user and stored in the SYSTEM or SYSAUX tablespace
- Index Organized Tables (IOTs)
- Clustered Tables
- LONGS (deprecated since Oracle Database 8i)
- Out of line LOBS

Reference

https://www.oracle.com/database/database-in-memory/index.html

https://www.youtube.com/watch?v=IZ7UMoQxtLo

Author

Shilpa Rajpurkar
DBA Lead Consultant – Oracle

Shilpa Rajpurkar is an Oracle DBA. Now, working with Mphasis as DBA Lead consultant, since Oct - 2009. She has total experience of 14 years in IT industry. She has worked in Insurance, Retail and telecom domain. She is currently working on 12c and had worked on most of the version of oracle.

About Mphasis