Improving Performance With 12c In-Memory Option

Prepared by: Fong Zhuang & Sergiy Smyrnov
Fong Zhuang

- Database Architect and Team Lead
- 19+ years with Walgreens

My Specialty & Interest

- Oracle Performance Tuning and High Availability
- Oracle E-Business Suit (EBS) and Oracle Business Intelligence Enterprise Edition (OBIEE)
- Oracle Engineered System
- Data Warehouse Implementation
- Oracle Certified Professional (OCP)
Sergiy Smyrnov

- Lead Database Architect
- 20+ years of experience with Database Technologies
- My Areas of expertise
  - Oracle Engineered Systems
  - High Availability/DR implementations for database technologies
  - Data Warehouse/DSS and OLTP/E-commerce database systems
  - Database automation

https://www.linkedin.com/in/sergiy-smyrnov-a2248728/
Walgreens

- Included in the Retail Pharmacy USA Division of Walgreens Boots Alliance, Inc., the first Global Pharmacy-Led Health and Wellbeing Enterprise
- One of the nation’s largest drugstore chains
- More than 8,175 stores
- Approximately 400 Walgreens stores offer Healthcare Clinic or other provider retail clinic services
- Focused on enhancing our customers’ beauty shopping experience
Agenda

- Row-major Oriented vs Column Oriented DBMS
- Introduction of Oracle Dual Format SGA Architecture
- Case Studies in Walgreens
  - Case 1 OLTP EBS environments (Exadata)
  - Case 2 OLAP OBIEE Data Warehouse (ODA vs. Exadata)
- Release 12.2 In-Memory enhancement
- Walgreens roadmap to use In-Memory features
CHALLENGE: Accelerate the performance of both OLAP and OLTP in the same database.

Traditional Row-major DBMS → OLTP workloads.

Columnar Oriented DBMS → A proven practice for OLAP / Data Warehouse

Oracle innovative OLTAP solution
Innovative OLTAP Solution - Dual Format Architecture

- New In-Memory Area option in Oracle Database 12c Release 1 (12.1.0.2)
- Static SGA pool in special columnar format for OLAP.
- Speeding up analytical queries
- Buffer Cache in row format for OLTP
- Same data stored in both Buffer Cache & In-Memory Area in different formats.
Transactionally Consistent

- Data stored in disk in Row format
- DML \(<---\) Buffer Cache row store.
- Read-only \(\rightarrow\) In-Memory Column Store.
- Transaction (inserts, updates, and deletes) committed \(\rightarrow\) Both stores.
- The population includes a transformation from row to columnar format and it consumes CPU.
In-Memory Area Structure

- **1MB-Pool IMCU (In-Memory Compression Unit)**
  - Logical unit of storage
  - Roughly equivalent to an extent within a tablespace.

- **64KB-Pool SMU (Snapshot Metadata Unit)**
  - Metadata about the IMCU & transactional info
  - Records min and max values
  - Serves as an In-Memory Storage Index

- Each IMCU maps to a SMU.
- NO Least-Recently-Used-Mechanism (LRU)
Case Study in Walgreens Financial System

- **Case 1 in EBS**
  - OLTP mix-workload
  - Relational entity model in 3NF.

- **Case 2 in OBIEE Data Warehouse**
  - OLAP
  - Entity model star schema

- **Lower environments in ODAs**

- **Performance test and PROD environments in Exadata**
Case Study Details

Please refer to following paper for more details:

❖ Published in IOUG Select:

“Improving Performance With 12c In-Memory Option”
Case 1 OLTP - Walgreens Oracle EBS (Exadata)

Database Evaluation:

- POC Performance Test Environment
- Database host: Exadata X-3 Quarter Rack
- RAC (2 nodes)
- Host memory: 256GB / node
- 16 cores / node
- DB version 12.1.0.2
Leverage In-Memory Advisor

Use Active Session History (ASH), Automatic Workload Repository (AWR) and optionally SQL Tuning Sets (STS).

Determine which tables, partitions and sub-partitions to place In Memory.

Licensed as part of the Database Tuning pack.

Minimum DB version to run In-Memory Advisor is 11.2.0.3.

Implemented recommendations in Database 12.1.0.2 and above.
Where to get it and how to install it?

- MOS note: 1965343.1
- Download imadvisor.zip from Oracle, copy to DB server--->unzip it
- Install in SQLPLUS with sysdba privilege
  - SQL> @instimadv.sql
- Run the IM advisor recommendation script with appropriate period
  - SQL> @imadvisor_recommendations.sql
- Review two outputs generated from the script
Case 1 OLTP - Walgreens Oracle EBS

In-Memory Advisor report highlights:

• A report based on 1 week of workload in EBS
• 67% DB time is doing analytics processing
• With 535.2GB In-Memory Size, the Estimated Analytics Processing Time Reduction is 79 hours.
• With 20GB of In-Memory, the potential time reduction is 14 hours.
Size **SGA** and **PGA** properly

Check MOS Note 1903683.1

- **Existing Single Instance:**
  \[ \text{SGA\_TARGET} = \text{Existing SGA\_TARGET} + \text{INMEMORY\_SIZE} \]

- **Existing RAC:**
  \[ \text{SGA\_TARGET} = \text{Existing SGA\_TARGET} + \text{INMEMORY\_SIZE} \times 1.1 \]

- If you are using parallel execution to reduce the possibility of spilling to TEMP on large joins and aggregations
  \[ \text{PGA\_AGGREGATE\_TARGET} = \text{PARALLEL\_MAX\_SERVERS} \times 2 \text{G} \]

**Note:**

- Additional space is required for the shared_pool as IMCS allocates additional locks from the shared pool

- Avoid “ORA – 4031” - unable to allocate string bytes of shared memory
Enable In-Memory Column Store

- Controlled by the parameter INMEMORY_SIZE
- Minimum size of 100MB
- In RAC DB, add additional 10% overhead to INMEMORY_SIZE.
  
  SQL> alter system set sga_max_size = 70G scope=spfile sid='*';
  SQL> alter system set sga_target = 70G scope=spfile sid='*';
  SQL> alter system set inmemory_size = 30G scope=spfile sid='*';

- Restart the database to take effect.

```sql
SQL> show sga
Total System Global Area 7.5162E+10 bytes
  Fixed Size             7652376 bytes
  Variable Size          1.7985E+10 bytes
  Database Buffers       2.4696E+10 bytes
  Redo Buffers           2.0000E+05 bytes
  In-Memory Area         3.2212E+10 bytes
SQL> show parameter inmemory_size
+-----------------------+------------+-------------+
| NAME                  | TYPE       | VALUE       |
+-----------------------+------------+-------------+
| inmemory_size         | big integer| 30G         |
```
In-Memory Area Structure

- The pool information in `v$inmemory_area`

```
SQL> select * from v$inmemory_area;

<table>
<thead>
<tr>
<th>POOL</th>
<th>ALLOC_BYTES</th>
<th>USED_BYTES</th>
<th>POPULATE_STATUS</th>
<th>CON_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1MB POOL</td>
<td>2.5744E+10</td>
<td>0</td>
<td>DONE</td>
<td>0</td>
</tr>
<tr>
<td>64KB POOL</td>
<td>6422724608</td>
<td>0</td>
<td>DONE</td>
<td>0</td>
</tr>
</tbody>
</table>
```

- By default, 20% of the In-memory size is allocated to the 64KB pool.
- Most of the chances, 1MB pool exhausted but 64KB pool plenty of space.
- No mechanism automatically resize 1MB pool and 64KB pool.
- Option of modifying "_inmemory_64k_percent" parameter to reduce the size of the 64K pool with Oracle support’s approval.
Case 1 OLTP - Walgreens Oracle EBS (Cont.)

Target the bottleneck:

- App team’s request: One EBS batch job was taking 14 hours & 39 min. Any solutions?

![Table showing job details and time taken](image)
Case 1 OLTP - Walgreens Oracle EBS (Cont.)

In-Memory advisor html report analysis:

If you add the objects to IMCS with 94.8MB in size, the time reduction is 4.7 hours.

<table>
<thead>
<tr>
<th>In-Memory Size</th>
<th>Percentage of Maximum SOA Size (50.00GB)</th>
<th>Estimated Analytics Processing Time Reduction (Seconds)</th>
<th>Estimated Analytics Processing Performance Improvement Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.00GB</td>
<td>60%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>1.857GB</td>
<td>4%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>1.739GB</td>
<td>4%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>1.667GB</td>
<td>4%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>1.572GB</td>
<td>4%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>1.481GB</td>
<td>4%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>1.399GB</td>
<td>4%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>1.304GB</td>
<td>4%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>1.204GB</td>
<td>4%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>1.11GB</td>
<td>4%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>1.018GB</td>
<td>4%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>948.1MB</td>
<td>0%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>852.3MB</td>
<td>0%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>758.5MB</td>
<td>0%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>660.7MB</td>
<td>0%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>568.8MB</td>
<td>0%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>474.6MB</td>
<td>0%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>379.2MB</td>
<td>0%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>282.4MB</td>
<td>0%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>189.6MB</td>
<td>0%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
<tr>
<td>94.81MB</td>
<td>0%</td>
<td>17290</td>
<td>1.5X</td>
</tr>
</tbody>
</table>
### Case 1 OLTP - Walgreens Oracle EBS (Cont.)

**In-Memory advisor html report analysis (Cont.)**

Only 850 sec out of 51,520 sec (14.3 hours) of Total Analytics Processing Time

Per AWR, this SQL took avg 0.04 sec to run. It was executed 27,072 times.

<table>
<thead>
<tr>
<th>SQL Id</th>
<th>SQL Text</th>
<th>Analytics Processing Time Used (Seconds)</th>
<th>Estimated Analytics Processing Time Reduction (Seconds) With Unlimited Memory</th>
<th>Estimated Analytics Processing Performance Improvement Factor With Unlimited Memory</th>
<th>Estimated Analytics Processing Time Reduction (Seconds) With 30,000GB</th>
<th>Estimated Analytics Processing Performance Improvement Factor With 30,000GB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SELECT XGL.LOCATION NUMBER, STORE NO, NVGL.SALES TAX OUT, roundup(PKG.GET_STATE_CODE_F(XGL.STATE)) STATE...</td>
<td>850</td>
<td>7</td>
<td>1.0x</td>
<td>7</td>
<td>1.0x</td>
</tr>
</tbody>
</table>

With 30,000GB, The One SQL Statement With Analytics Processing Benefit

Only 7 sec of reduction, Seriously?
10 tables are recommended to populate into IMCS.

- Table Population Examples in Engineered System:

```sql
SQL> ALTER TABLE "GL"."GL_CODE_COMBINATIONS" INMEMORY
duplicate all priority critical MEMCOMPRESS FOR QUERY LOW;
```

- Eligible Segments: Table, Partition, Subpartitions, Materialized Views, and tablespace.

- Tune parameter `inmemory_max_populate_servers` for segment population performance, but watch over the CPU usage carefully.
  - The default value is the less value: 1/2 X CPU thread count OR `PGA_AGGREGATE_TARGET / 512M`
The IM column store compression Levels

SQL> ALTER TABLE "GL"."GL_CODE_COMBINATIONS" INMEMORY
duplicate all priority critical MEMCOMPRESS FOR QUERY LOW;
In-Memory Population Priority

SQL> ALTER TABLE "GL"."GL_CODE_COMBINATIONS"
INMEMORY duplicate all priority critical
MEMCOMPRESS FOR QUERY LOW;

• CRITICAL
• HIGH
• MEDIUM
• LOW
• NONE (Default)
When “In Memory” meets RAC

```sql
ALTER TABLE "GL"."GL_CODE_COMBINATIONS" INMEMORY
duplicate all priority critical MEMCOMPRESS FOR QUERY LOW;
```

- For Non Engineered System, use “distribute” subclause
  - Distribute by rowid range
  - Distribute by partition
  - Distribute by subpartition
- Enable Auto DOP
  - Parallel_degree_policy=auto
- In Engineered System, option of using subclause “duplicate” or “duplicate all” to duplicate IMCUs into other nodes (mirroring).
Check the execution results after adding tables to In-Memory:

- The job completed in 1 hour and 9 minutes.
- Compare to 14 hours and 39 minutes of the previous timing, 12X performance improvement.
Check the application execution results after adding tables to In-Memory (Cont.)

- Where was the performance improvement coming from?

<table>
<thead>
<tr>
<th>SQL Id</th>
<th>SQL Text</th>
<th>Analytics Processing Time Used (Seconds)</th>
<th>Estimated Analytics Processing Time Reduction (Seconds) With Unlimited Memory</th>
<th>Estimated Analytics Processing Performance Improvement Factor With Unlimited Memory</th>
<th>Estimated Analytics Processing Time Reduction (Seconds) With 30.00GB</th>
<th>Estimated Analytics Processing Performance Improvement Factor With 30.00GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>3213653653639</td>
<td><code>SELECT XGL.LOCATION_NUMBER, XGL.SALES TAX OUTBOUND_PKG.GET_STATE_CODE_F(XGL.STATE) AS ...</code></td>
<td>7</td>
<td>1.0X</td>
<td>7</td>
<td>1.0X</td>
<td>1.0X</td>
</tr>
</tbody>
</table>

Reality: Per base AWR, after adding tables to In-Memory, it took 0.012 sec each time.

Reality: It was executed 27,072 times.

Reality: The total processing time reduction is 758 Sec (12.4 Min).

Reality: The performance improvement factor is 3.3X.
Identify the key time saving SQL from AWR Report

SQL ordered by Elapsed Time

<table>
<thead>
<tr>
<th>Elapsed Time (s)</th>
<th>Executions</th>
<th>Elapsed Time per Exec(s)</th>
<th>% Total</th>
<th>% CPU</th>
<th>% IO</th>
<th>SQL Id</th>
<th>SQL Module</th>
<th>SQL Text</th>
</tr>
</thead>
</table>
| 53.129.99        | 0          | 97.22                    | 99.02   | 0.21  |     | 44513hm880cwm | XXGLSALESTAXOUT | BEGIN XXGL_SALES_TAX_OUTBOUND...
| 50.562.91        | 538        | 94.02                    | 92.56   | 99.23 | 0.00 | d5y6120ykn39r | XXGLSALESTAXOUT | SELECT DISTINCT FFV.FLEX_VALUE...
| 1,223.84         | 1          | 1,223.84                 | 98.47   | 0.57  |     | 05a3358nnn6yrt | XXGLSALESTAXOUT | begin dbms_feature_usage_inter...
| 1,221.63         | 1          | 1,221.63                 | 98.54   | 0.55  |     | dxwqcppggunz26r | XXGLSALESTAXOUT | BEGIN DBMS_FEATURE_AWR();feature...
| 1,221.60         | 1          | 1,221.60                 | 98.54   | 0.55  |     | 6tw49mdqppop2cr | XXGLSALESTAXOUT | SELECT /* DS_SVC *//*dynam...
| 1,057.61         | 27,072     | 0.04                     | 97.56   | 1.94  | 1.58 | 1223hn53h9hn9r | XXGLSALESTAXOUT | SELECT XGL.LOCATION_NUMBER SLO...
| 397.77           | 349        | 1.14                     | 94.28   | 3.32  |     | dffkcnofystw | MMON_SLAVE | WITH MONITOR_DATA AS (SELECT I...
| 365.76           | 349        | 1.05                     | 93.67   | 3.72  |     | fh6upapxs5cswz | MMON_SLAVE | BEGIN sys.dbms_auto_report_int...
| 355.77           | 349        | 1.02                     | 93.79   | 3.74  |     | 6wz26ak6t6gq98 | MMON_SLAVE | SELECT XMTTYPE (DBMS_REPORT.GET...
| 339.89           | 1          | 339.89                   | 99.17   | 0.07  |     | 73629tmf0izrh | DBMS_SCHEDULER | DECLARE job BINARY_INTEGER := ...
### Top SQL performance evaluation in Wag Sales Tax Outbound job

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>SQL Text</th>
<th>Executions</th>
<th>Average Elapsed Time per Exec (s)</th>
<th>Average Elapsed Time per Execution w/ tables In-Memory (s)</th>
<th>Time Reduced/Execution (s)</th>
<th>Performance Improvement Factor (Old Timing / New Timing)</th>
<th>Total time reduction(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1223h53fh9ht9</td>
<td><code>SELECT XGL.LOCATION_NUMBER STO...</code></td>
<td>27,072</td>
<td>0.04</td>
<td>0.012</td>
<td>0.028</td>
<td>3.33X</td>
<td>758 (s) OR 12.6 (m)</td>
</tr>
<tr>
<td>d5v6120yk39r</td>
<td><code>SELECT DISTINCT FFV.FLEX_VALUE FROM APPS.FND_FLEX_VALUE_CHILDREN_V FFV</code></td>
<td>538</td>
<td>94.02</td>
<td>2.16</td>
<td>91.86</td>
<td>47X</td>
<td>49496 (s) OR 825 (m) OR 13.7 (h)</td>
</tr>
</tbody>
</table>
Top SQL2 Execution Plan Comparison from SQL Monitor Output

**Column Store: In-Memory**

**Row Store: Buffer Cache**
Test results with good performance improvement on major EBS jobs:

<table>
<thead>
<tr>
<th>EBS Job Name</th>
<th>Table Disk Size</th>
<th>In-Memory Size</th>
<th>Processing Time w/o In-Memory</th>
<th>Processing Time with In-Memory</th>
<th>Time Reduced</th>
<th>Performance Improvement Factor (Old Timing / New Timing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outbound Interface Program</td>
<td>11GB</td>
<td>7GB</td>
<td>9 hours</td>
<td>5 hours</td>
<td>4 hours</td>
<td>1.8 X</td>
</tr>
<tr>
<td>Sales Tax Outbound Interface</td>
<td>8.5GB</td>
<td>1.8GB</td>
<td>14 hours and 39 minutes</td>
<td>1 hour 10 minutes</td>
<td>13 hours</td>
<td>12 X</td>
</tr>
</tbody>
</table>

Note: There were 36 indexes on 10 tables. The total index disk size is 2.3 GB. Analytic indexes can be dropped after adding tables into IMCS.
## Sample test results without performance improvement:

<table>
<thead>
<tr>
<th>EBS Job Name</th>
<th>Disk Size</th>
<th>In-Memory Size</th>
<th>Processing Time w/o In-Memory</th>
<th>Processing Time with In-Memory</th>
<th>Time Reduced</th>
<th>Performance Improvement Factor (Old Timing / New Timing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warehouse Billing Journal Inbound Interface</td>
<td>89GB</td>
<td>15GB +</td>
<td>1 hour and 20 minutes</td>
<td>1 hour 28 minutes</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Reasons with no improvement:

- One major table with 89GB which couldn’t be fully populated to In-Memory.
- The In-Memory compression ratio is relatively high.
Case 1 Summary:

- Observed time reduction from 1.8X to 12X on various jobs.
- In-Memory Advisor is efficient in identifying the candidate segments.
- The real performance could be much better than the In-Memory Advisor estimation.
- Analytic indexes can be dropped for potential space saving and DML contention reduction.
- No performance change on segments with high compression ratio.
Case 2 OLAP Oracle OBI Data Warehouse (ODA)

Database Evaluation:

- Proof of Concept Environment
- Database host:
  Oracle Database Appliance (ODA X3-2)
- RAC (2 nodes)
- Host memory: 256GB/node
- 16 cores / node
- ODA is shared by multiple DBs
- DB version 12.1.0.2
- Plan to improve with INMEMORY_SIZE 50GB
Case 2 OLAP Oracle OBI Data Warehouse (ODA)

Sample test results without performance improvement:

<table>
<thead>
<tr>
<th>Test Description</th>
<th>Disk Size</th>
<th>In-Memory Size</th>
<th>Tables NOT In Memory (HH:MM:SS.00)</th>
<th>Tables In Memory (HH:MM:SS.00)</th>
<th>Performance Improvement Factor (Old Timing / New Timing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count Fact Table W_GL_BALANCE_F</td>
<td>58GB</td>
<td>16GB</td>
<td>00:01:23.42</td>
<td>00:00:00.61</td>
<td>137 X</td>
</tr>
<tr>
<td>A Report Accesses Fact Table W_GL_BALANCE_F and 4 Small Dimension Tables with Many aggregates</td>
<td>58.6GB</td>
<td>16GB</td>
<td>00:02:48.88</td>
<td>00:00:09.12</td>
<td>18.5 X</td>
</tr>
<tr>
<td>Count Fact Table W_GL_OTHER_F</td>
<td>128GB</td>
<td>17GB</td>
<td>00:04:42.39</td>
<td>00:00:32.55</td>
<td>8.7 X</td>
</tr>
</tbody>
</table>
## Case 2 OLAP Oracle OBI Data Warehouse (ODA)

### Sample test results without performance improvement:

<table>
<thead>
<tr>
<th>Test Description</th>
<th>Disk Size</th>
<th>In-Memory Size</th>
<th>Tables NOT In Memory (HH:MM:SS.00)</th>
<th>Tables In Memory (HH:MM:SS.00)</th>
<th>Performance Improvement Factor (Old Timing / New Timing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Report Accesses Fact Table W_GL_OTHER_F and 7 Small Dimension Tables</td>
<td>128.6GB</td>
<td>17GB</td>
<td>00:01:33.04</td>
<td>00:01:29.80</td>
<td>No Significant Change</td>
</tr>
</tbody>
</table>

### Reasons with no improvement:

- One major table has 128.6 GB and the In-Memory Size is 17GB.
- The In-Memory compression ratio is relatively high.
Case 2 OLAP Oracle OBI Data Warehouse (Exadata)

Database Evaluation:

- Performance Test Environment
- Database host: Exadata X-3 Quarter Rack
- RAC (2 nodes)
- Host memory: 256GB/node
- 16 cores / node
- DB version 12.1.0.2
- Plan to improve with INMEMORY_SIZE 30GB
## Case 2 Same SQL Performance Comparison Exadata vs ODA

<table>
<thead>
<tr>
<th>Test Description</th>
<th>Disk Size</th>
<th>In-Memory Size</th>
<th>Tables NOT In Memory (HH:MM:SS.00)</th>
<th>Tables In Memory (HH:MM:SS.00)</th>
<th>Performance Improvement Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count Table W_GL_BALANCE_F</td>
<td>50.62GB</td>
<td>13.65GB</td>
<td>00:00:4.57</td>
<td>00:00:01.08</td>
<td>4.23 X</td>
</tr>
<tr>
<td>A Report Accesses Fact Table W_GL_BALANCE_F</td>
<td>51.2GB</td>
<td>14GB</td>
<td>00:00:22.47</td>
<td>00:00:07.34</td>
<td>3.06 X</td>
</tr>
<tr>
<td>Count Table W_GL_OTHER_F</td>
<td>128GB</td>
<td>13.56GB</td>
<td>00:00:04.22</td>
<td>00:00:01.03</td>
<td>4.09 X</td>
</tr>
<tr>
<td>A Report Accesses Fact Table W_GL_OTHER_F</td>
<td>128.6GB</td>
<td>13.60GB</td>
<td>00:00:05.19</td>
<td>00:00:06.80</td>
<td>No Significant Change</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Description</th>
<th>Disk Size</th>
<th>In-Memory Size</th>
<th>Tables NOT In Memory (HH:MM:SS.00)</th>
<th>Tables In Memory (HH:MM:SS.00)</th>
<th>Performance Improvement Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count Table W_GL_BALANCE_F</td>
<td>58GB</td>
<td>16GB</td>
<td>00:01:23.42</td>
<td>00:00:00.61</td>
<td>137 X</td>
</tr>
<tr>
<td>A Report Accesses Fact Table W_GL_BALANCE_F</td>
<td>58.6GB</td>
<td>16GB</td>
<td>00:02:48.88</td>
<td>00:00:09.12</td>
<td>18.5 X</td>
</tr>
<tr>
<td>Count Table W_GL_OTHER_F</td>
<td>128GB</td>
<td>17GB</td>
<td>00:04:42.39</td>
<td>00:00:32.55</td>
<td>8.7 X</td>
</tr>
<tr>
<td>A Report Accesses Fact Table W_GL_OTHER_F</td>
<td>128.6 GB</td>
<td>17GB</td>
<td>00:01:33.04</td>
<td>00:01:29.80</td>
<td>No Significant Change</td>
</tr>
</tbody>
</table>
Case 2 Summary:

• Promising performance improvement in both ODA and Exadata
• Exadata provides large Flash Cache, Smart Scan, and HCC.
• Recommend to keep the very hot data in In-Memory column store, keep the warm data to Exadata Flash Cache, and compress the cold data with Exadata HCC.
• For either environment, we expect to add more memory for further performance improvement.
Is that all? Can we do more with In-Memory?

Yes, we can (in 12cR2)!!!

- ILM (Information Lifecycle Management) for automated data policy management for IMCS in Exadata + 12cR2
- IMCS FastStart Area in 12cR2
- IMCS in Oracle Active Data Guard in 12cR2
- Improvements of IMCS for RAC services in 12cR2
- In-Memory Expressions in 12cR2
The Oracle Information Lifecycle Management (ILM) has the capability of distinguishing between “hot” (frequently accessed) and “cold” (infrequently accessed) data and set policy on tables and segments.

In 12.2, DBAs can set policies to mark an object as a candidate to be populated into In-Memory column store (INMEMORY) and in Exadata additionally in Flash-Cache (CELLMEMORY) to be able to use ADO policies for WARM/COLD data.

This significantly enhance and automate further the lifecycle of your core data within a database.

Note that in-memory ADO is only supported at the segment level.
Use *ILM ADD POLICY* syntax with

- SET INMEMORY
- MODIFY INMEMORY
- NO INMEMORY

Alter *MEMCOMPRESS* for *INMEMORY* level

- After *<time>* of no access
- After *<time>* of creation
- After *<time>* of no modification
- On *<user defined Boolean function>*

Examples:

```
ALTER TABLE rx_fill ILM ADD POLICY NO INMEMORY after 3 month of creation;
ALTER TABLE rx_fill ILM MODIFY POLICY NO INMEMORY after 7 days of no access;
ALTER TABLE rx_fill ILM ADD POLICY MEMOCOMPRESS FOR QUERY LOW AFTER 30 days of creation;
ALTER TABLE rx_fill ILM ADD POLICY MEMOCOMPRESS FOR QUERY HIGH AFTER 45 days of no modification;
```
**Exadata CELLMEMORY option**

CELLMEMORY works similar to a 12.1.0.2 columnar cache but when used with INMEMORY on Exadata is enabled by default to extend columnar cache format.

You can overwrite (enable, disable, change compression level) default and use CELLMEMORY feature on tables, partitions, sub partitions, and materialized views.

The MEMCOMPRESS clause supports FOR [QUERY|CAPACITY] [LOW| HIGH].

```sql
ALTER TABLE member CELLMEMORY;
ALTER TABLE member NO CELLMEMORY;
ALTER TABLE sales PARTITION Q1_2017 CELLMEMORY MEMCOMPRESS FOR QUERY LOW;
```

Note that this feature requires the 12.2 release of Exadata Storage Server software, but works with the 12.1 release of the database. Use of the 12.2 release of the database enables additional performance optimizations for offloaded CELLMEMORY scan processing.
**In-Memory FastStart Area in 12cR2**

A FastStart area is a designated tablespace where IM FastStart stores and manages data for INMEMORY objects.

Tablespace should be 2x size of INMEMORY_SIZE area

Oracle Database manages the FastStart tablespace without DBA intervention.

It can relieve the CPU overhead of population at the cost of additional disk space.

- In-memory Fast Start syntax:

  ```sql
  exec DBMS_INMEMORY_ADMIN.FASTSTART_ENABLE('fstart_tbs_name');
  ```

- To check the size/status of Fast Start tablespace usage use view

  ```sql
  V$INMEMORY_FASTSTART_AREA
  ```

Note: the actual FastStart IM data is written in LOB segment in that tablespace.
In 12cR2 you now have option to enable IM column store on the standby Active Data Guard database (commonly a lot of applications usually take advantage of Active Data Guard for reporting purposes), or on both the primary and standby databases.

There are three configuration options to consider:

- Identical column stores in both primary and standby databases
- Column store in just the standby database
- Different column store configurations between the primary and standby databases

Those configurations are managed by the combination of 3 settings:

- `INMEMORY_SIZE` (primary, standby)
- `INMEMORY_ADG_ENABLED` is set to true on the standby instance (if needed)
- The `INMEMORY` with additional `DISTRIBUTE FOR SERVICE` clauses for IN-MEMORY objects designated for the standby role services
In Oracle 12c R2 it become much easier to selectively populate objects to specific IM column stores in a RAC environment by taking advantage of the new DISTRIBUTE FOR SERVICE subclause when defining INMEMORY option.

This is a secondary use of that sub-clause after INMEMORY on Active Dataguard.

• Example:

```
srvctl add service -db db1 -service rx_srv –preferred “inst1”
srvctl add service -db db1 -service catalog_srv -r preferred “inst2”

ALTER TABLE rx_fill INMEMORY PRIORITY MEDIUM DISTRIBUTE FOR SERVICE rx_srv;

ALTER TABLE drugs INMEMORY PRIORITY HIGH DISTRIBUTE FOR SERVICE catalog_srv;
```
In-Memory Join Groups in 12cR2

With Oracle Database 12.2., it is possible to declare that two tables will be joined on a set of columns using a new DDL construct known as a Join Group.

This enables join processing optimizations based on updated common dictionary so that joins can be performed on compact dictionary codes instead of on values.

Note that IM segments need to be re-populated after creation of Join Groups.

• Examples:

```sql
CREATE INMEMORY JOIN GROUP rx_fill_jg1(rx_fill(patient_id),member(patient_id));
CREATE INMEMORY JOIN GROUP rx_fill_jg2(rx_fill(drug_id),drug(drug_id));
CREATE INMEMORY JOIN GROUP rx_fill_jg3(rx_fill(rx_id),invoice(rx_id));
```

You can query USER_JOINGROUPS view to see Join Group definitions.

In order to determine if Join Group(s) are actually in use you need to check a SQL Monitor Active Report and click on the “eye glasses” icon for hash joins and check that "Columnar Encodings Leveraged" is showing value “1.”
In-Memory Expressions give ability to materialize simple expressions and store them in the In-Memory column store so that they only have to be calculated once.

They are treated like any other column in the IM column store so the database can scan and filter those columns and take advantage of all Database In-Memory query optimizations.

There are two types of IM expressions:

- user-defined In-Memory virtual column
- automatically detected IM expressions created by Expressions Statistics Store (ESS) which monitors workloads and store it as a hidden virtual column when captured:
  - DBMS_INMEMORY.IME_CAPTURE to capture the repeating expressions
  - DBMS_INMEMORY.IME_POPULATE to create the in-memory virtual columns.
Walgreens roadmap to use In-Memory features

Consolidation of Oracle DBs on Exadata Cloud Machine:
- Taking advantage of Large Memory (720 Gb per node)
- Taking advantage of all DB options included (in-Memory, Multitenant, Active Data Guard)
- Taking advantage of fast storage and HCC compression options
Oracle DB consolidation on Exadata Cloud Machine platform

Significantly higher DB Consolidation Density and Datacenter Footprint (Multiple VMs with multi-tenant DB consolidation per network zone/environment/business unit)

3x reduction in overall power usage (14.5 kVA vs 44.8 kVA)

Spare on-demand capacity to migrate remaining ODA/Exadatas in the future or provision DBs for new applications almost instantaneously.
Acknowledgement

- Special Thanks to director Jim Hope and manager Glenn Campbell for all the support.

- Special Thanks to senior DBA Karen Moe who participated in the In-Memory POC in her system and provided feedback to us.

- Special Thanks to senior DBA Raghav Kalakota for setting up the POC environment in ODA.
References

- IOUG SELECT: Improving Performance With 12c In-Memory Option
  http://select.iou.org/blog/improving-performance-with-12c-in-memory-option by Fong Zhuang
- Oracle White Paper: ORACLE DATABASE 12 C IN-MEMORY OPTION - The Top Tier of a Multi-tiered Database Architecture
- Oracle blogs: https://blogs.oracle.com/In-Memory/entry/what_is_an_in_memory
- Oracle White Paper: When to Use Oracle Database In-Memory
- Oracle white paper: Oracle Database In-Memory Advisor
- Oracle white paper: Oracle Database In-Memory
- IOUG presentation by Kai Yu: Optimize OLAP & Business Analytics Performance with Oracle 12c In-Memory Database Option
- Oracle Database In-Memory Powering the Real-Time Enterprise