Optimize Oracle Business Intelligence Analytics with Oracle 12c In-Memory Database Option

Kai Yu, Senior Principal Architect
Dell Oracle Solutions Engineering
Dell, Inc.

Abstract:
By adding the In-Memory columnar store feature, the recently released In-Memory Database option is designed to significantly improve query performance for OLAP application. This presentation will explore how to leverage this In-Memory feature on the Oracle OBIEE analytics environment. This presentation will use a financial analytics project as a case study to explore the best practices, lesson learned, performance studies of applying In-Memory database in the business analytic applications.

Oracle 12c In-Memory Option

Oracle 12c Database introduced the In-Memory Database option. This option aimed to accelerates analytics and also speed up mixed-workload OLAP/OLTP. This feature is also transparent to applications.

The Oracle 12c Database In-Memory option introduced a dual-format of architecture which consists:
1. Oracle traditional row based format. This row format data is stored in storage as well as in buffer cache in SGA. This row format is good for OLTP (insert/update/delete) operations.
2. The In-memory option is introduced with Oracle 12.1.0.2. With this option, Oracle added a column format In-Memory column store, a new component of Oracle Database SGA. This memory structure coexists with the database buffer cache (row format). This option is designed to speed OLAP application, and also supports OLTP/OLAP mixed workloads.

The Dual Format Architecture can be illustrated as Figure 1.

Let’s take a look at this In-Memory Column Store. As we mentioned, this is a new component in SGA also called In-Memory Area. To turn on this option, you need to set this Oracle initialization parameter: inmemory_size. Since this parameter is not a dynamic parameter, you can first set it in spfile with this statement:
SQL> ALTER SYSTEM SET INMEMORY_SIZE = 100G SCOPE=SPFILE;
Then restart the database instance to take effect.
When you restart the database, the startup process will show the size of the In-Memory area size like this:

\[
\begin{array}{|l|c|}
\hline
\text{Total System Global Area} & 2.6521E+11 \text{ bytes} \\
\text{Fixed Size} & 7662672 \text{ bytes} \\
\text{Variable Size} & 2.7380E+10 \text{ bytes} \\
\text{Database Buffers} & 1.2992E+11 \text{ bytes} \\
\text{Redo Buffers} & 529207296 \text{ bytes} \\
\text{In-Memory Area} & 1.0737E+11 \text{ bytes} \\
\hline
\end{array}
\]

Database mounted.
Database opened.

We can populate the In-Memory column store by specifying objects on several ways:

**Tablespace level:** This will upload all the objects of the tablespace to In-Memory area:

SQL> alter tablespace data INMEMORY;

**Table level:** This loads the table to In-Memory area.

SQL> alter table sales INMEMORY PRIORITY CRITICAL;

You can specify different levels of the priority of loading this table by specifying the key words such as CRITICAL,

You also can exclude certain columns when you load the table to the In-Memory area, such as

SQL> alter table sales INMEMORY NO INMEMORY(prod_id)

The Oracle database instance has a set of background process dedicated to populate the In-Memory store. This list shows a set of background processes named like ora-wxxx_instance_name

```
oracl...14739 1 0 14:30 ? 00:00:17 ora_w004_poedbl
oracl...14759 1 0 14:30 ? 00:00:15 ora_w005_poedbl
oracl...14763 1 0 14:30 ? 00:00:12 ora_w006_poedbl
oracl...14765 1 0 14:30 ? 00:00:12 ora_w007_poedbl
oracl...17515 1 0 14:38 ? 00:00:06 ora_w008_poedbl
oracl...19364 1 0 14:43 ? 00:00:06 ora_w009_poedbl
oracl...19366 1 0 14:44 ? 00:00:06 ora_w010_poedbl
oracl...112632 1 0 15:26 ? 00:00:22 ora_w020_poedbl
oracl...112634 1 0 15:26 ? 00:00:22 ora_w021_poedbl
```

Oracle 12c provides several new features to accelerate query execution by leveraging this in-memory option: In-Memory Scan, In-Memory Storage Index, SIMD Vector Processing, In-Memory Joins, In-Memory Aggregation.

Another big advantage of the In-Memory Option is that it is application transparent. There is no need to modify application to take advantage this option. The Oracle query analyzer will automatically select the new query plan if the table is In-Memory store. You can see whether the In-Memory option takes effect by looking the INMEMORY key word in query plan such as :

```
PLAN_TABLE_OUTPUT
  18 TABLE ACCESS INMEMORY FULL  EDAPINHDR_BASE
  19 PARTITION LIST JOIN-FILTER  EDAPILN_BASE
  21 TABLE ACCESS INMEMORY FULL  EDAPITQ_BASE
```
Oracle Business Intelligence Enterprise Edition 11g

Oracle Business Intelligence Enterprise Edition (OBIEE) is a business intelligence and Analytics Platform and common infrastructure for reports, scorecards, dashboards, ad-hoc analysis, OLAP analysis. It consists of the following:

- OBIEE 11g Interactive Dashboards solution for Interactive Dashboards
- Ad hoc Analysis and Interactive Reporting
- Oracle BI Mobile for Mobile Analytics
- Oracle Business Intelligence Publisher for Enterprise Reporting.

It can be installed on Windows or Linux OS. The following shows the GUI interface of the OBIEE:

![OBIEE GUI](image)

Oracle Exalytics In-Memory Machine

Oracle Engineered System for Extreme Analytic is designed to deliver extreme in-memory analytics performance. It is made of two main components together:

- Optimized Oracle Business Intelligence Foundation Suite, which consists of Oracle Business Intelligence Enterprise Edition and Oracle Essbase.
- Oracle TimesTen In-Memory Database for Exalytics

Oracle Exalytics In-Memory Machine features the following hardware and software components:

- An x86-64 server: 4 X Intel Xenon E7-4800 processors, 2 TB RAM, 2 QDR 40Gb/s Infiniband Ports, 2X 10Gbps Ethernet ports, 6 X 400G Flash PCI-e
- Oracle Business Intelligence Foundation Suite including Oracle Essbase
- Oracle TimesTen In-Memory Database for Exalytics
- Exalytics In-Memory Software

People may be confused with these two In-Memory databases: TimesTen In-Memory Database and Oracle 12c In-Memory. Actually these are two different products. Oracle TimesTen In-Memory Database for Exalytics is a full memory database designed to run Analytics. It usually runs on the same server as OBIEE as a way to cache all the business analytics data. While Oracle 12c In-Memory is a feature added to general Oracle Database, and it works both OLAP and OLTP mixed workload. Oracle OBIEE usually runs on different machine from the machine that runs Oracle database. OBIEE connects to Oracle database using ODBC or JDBC through SQL*Net on network. It is common that Oracle Exalytics In-Memory Machine that runs both Oracle Business Intelligence Foundation Suite and Oracle TimesTen In-Memory Database for Exalytics have the Oracle Database in backend to keep the database persistent.
Oracle OBIEE with Oracle 12c In-Memory Database Option

As a business intelligence and analytics platform, Oracle OBIEE Server connects to a physical database such as Oracle Database through ODBC/JDBC and present a logic schema view independent of physical database. When report users run the reports, the OBIEE server translates or map the logic SQLs from the report presentation layer to physical SQL queries and send the SQLs to the Oracle database, then the Oracle Database executes the physical SQL queries and fetch the query data and send them back to OBIEE server to make the report. The Oracle BI Administration tools shows the three layers: Presentation Business Model and Mapping, Physical.

We are using the following example to discuss how to use Oracle In-Memory option to accelerate Oracle BI reports. Usually BI analytics processing requires a large set of the data from the database. On the BI report physical level, you may be able to see many full table scan on some of very large tables and complex join operations on these big tables. Essentially the IO operations for these large full table scan make a big part of the report execution. A reduction of these IO operations can significantly improve the report performance. With the Oracle In-Memory option, we can load a partial or the entire of those full scan tables into the Oracle In-Memory store, and with these tables in database memory, we can significantly reduce the physical SQL query time.

With these ideas, we can have the following process:

- Start with the slow report and find the presentation layer of the report.
- Through the mapping from presentation layer to the physical layer to identify the physical SQL for the report
- Through the physical SQL to identify the underneath full table scan operation.
- Add some of the underline tables to the In-Memory column store of the

The rest of the paper will use an example to go through this process:

1. Identify the physical SQL layer for the report

With a particular report to tune, we can take a look at the mapping of the report to the physical layer. For example we take a report called EDI Queue which runs over 9 and half minutes. We would like to cut the run time of this report with Oracle In-Memory database option.
Through the presentation layer to find the Business Model and mapping of Fact EDI Queue, we identified the physical database view FACT_EDI_QUEUE_V is the physical SQL query for this report as shown below:

Figure 4: Three Layers and their mapping

By reviewing the definition of the physical View FACT_EDI_QUEUE_V, we found four big physical tables on which this view is built: EDAPIHDR_BASE, EDAPIQ_BASE, EDAPIQ_BASE, VEN_LOC_BASE. Through the view, the physical SQL query on this view results to the full table scan operations on these four tables. We decided to put the four tables into Oracle Database In-Memory area:

To Populate In-Memory Column area with these four tables, we ran these four table alter statements:

```
SQL> alter table APD_BASE.EDAPILIN_BASE inmemory priority high;
SQL> alter table APD_BASE.EDAPIQ_BASE inmemory priority high;
SQL> alter table APD_BASE.EDAPIHDR_BASE inmemory priority high;
SQL> alter table APD_BASE.VEN_LOC_BASE inmemory priority high;
```

We can confirm that these tables are already cached in the In-Memory Column area:

```
SQL> select segment name, inmemory_size from $sim_segments;
SEGMENT_NAME INMEMORY_SIZE
VEN_LOC_BASE       1279648
EDAPIQ_BASE           291168512
EDAPIHDR_BASE       961496576
VEN_LOC_BASE         1279648
EDAPILIN_BASE         930710528
VEN_LOC_BASE         1279648
```

Let's compare the query plan on the view before populating the In-Memory Column area:
This is the query plan before using In-Memory Column area:

```
SQL> explain plan for select count(*) from APD_STAR.FACT_EDI_QUEUE_V;
explain.
SQL> select plan_table_output
>  from table(dbsms_xplan.display('plan_table',null,'basic'));

PLAN_TABLE_OUTPUT
plan hash value: 1571279316

<table>
<thead>
<tr>
<th>Id</th>
<th>operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SORT AGGREGATE</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>PART JOIN FILTER CREATE</td>
<td>:BF0000</td>
</tr>
<tr>
<td>4</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>TABLE ACCESS FULL</td>
<td>DIM_PERIOD</td>
</tr>
</tbody>
</table>

PLAN_TABLE_OUTPUT

<table>
<thead>
<tr>
<th>Id</th>
<th>operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>HASH JOIN</td>
<td>DIM_REGION_CCN</td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS FULL</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PART JOIN FILTER CREATE</td>
<td>:BF0001</td>
</tr>
<tr>
<td>10</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>PART JOIN FILTER CREATE</td>
<td>:BF0002</td>
</tr>
<tr>
<td>12</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>TABLE ACCESS FULL</td>
<td>DIM_VENDOR</td>
</tr>
<tr>
<td>14</td>
<td>PARTITION LIST ALL</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>TABLE ACCESS INMEMORY FULL</td>
<td>VEN_LOC_BASE</td>
</tr>
<tr>
<td>16</td>
<td>PARTITION LIST JOIN-FILTER</td>
<td></td>
</tr>
</tbody>
</table>

PLAN_TABLE_OUTPUT

<table>
<thead>
<tr>
<th>Id</th>
<th>operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>TABLE ACCESS INMEMORY FULL</td>
<td>EDAPIHDR_BASE</td>
</tr>
<tr>
<td>18</td>
<td>PARTITION LIST JOIN-FILTER</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>TABLE ACCESS INMEMORY FULL</td>
<td>EDAPILN_BASE</td>
</tr>
<tr>
<td>20</td>
<td>PARTITION LIST JOIN-FILTER</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>TABLE ACCESS INMEMORY FULL</td>
<td>EDAPIQ_BASE</td>
</tr>
</tbody>
</table>
```

After populating the In-Memory store with these tables, we had a new query plan for this view:

```
SQL> explain plan for select count(*) from APD_STAR.FACT_EDI_QUEUE_V;
explain.
SQL> select plan_table_output
>  from table(dbsms_xplan.display('plan_table',null,'basic'));

PLAN_TABLE_OUTPUT
plan hash value: 1571279316

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SORT AGGREGATE</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>PART JOIN FILTER CREATE</td>
<td>:BF0000</td>
</tr>
<tr>
<td>4</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>TABLE ACCESS FULL</td>
<td>DIM_PERIOD</td>
</tr>
</tbody>
</table>

PLAN_TABLE_OUTPUT

<table>
<thead>
<tr>
<th>Id</th>
<th>operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>HASH JOIN</td>
<td>DIM_REGION_CCN</td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS FULL</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PART JOIN FILTER CREATE</td>
<td>:BF0001</td>
</tr>
<tr>
<td>10</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>PART JOIN FILTER CREATE</td>
<td>:BF0002</td>
</tr>
<tr>
<td>12</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>TABLE ACCESS FULL</td>
<td>DIM_VENDOR</td>
</tr>
<tr>
<td>14</td>
<td>PARTITION LIST ALL</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>TABLE ACCESS INMEMORY FULL</td>
<td>VEN_LOC_BASE</td>
</tr>
<tr>
<td>16</td>
<td>PARTITION LIST JOIN-FILTER</td>
<td></td>
</tr>
</tbody>
</table>

PLAN_TABLE_OUTPUT

<table>
<thead>
<tr>
<th>Id</th>
<th>operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>TABLE ACCESS INMEMORY FULL</td>
<td>EDAPIHDR_BASE</td>
</tr>
<tr>
<td>18</td>
<td>PARTITION LIST JOIN-FILTER</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>TABLE ACCESS INMEMORY FULL</td>
<td>EDAPILN_BASE</td>
</tr>
<tr>
<td>20</td>
<td>PARTITION LIST JOIN-FILTER</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>TABLE ACCESS INMEMORY FULL</td>
<td>EDAPIQ_BASE</td>
</tr>
</tbody>
</table>
```
Notice that the full table scans on these four tables were replaced with the TABLE ACCESS INMEMORY FULL, which indicated that the query read the data from the IN-Memory column store, instead of the storage.

As the results of the query plan change, the query execution time has been reduced from 4 minutes 2 seconds to 2 minutes 15 seconds, as shown here:

Not In-Memory

```
12:35:36 SQL> set timing on
12:35:36 SQL> set time on
12:35:36 SQL> select count(*) from APD_STAR.FACT_EDT_QUEUE_V;
   COUNT(*)
-------------
      51638519
elapsed: 00:04:02.80
```

In-Memory

```
14:51:55 SQL> select count(*) from APD_STAR.FACT_EDT_QUEUE_V;
   COUNT(*)
-------------
      51638519
elapsed: 00:02:15.11
```

We found that with this reduction of the physical SQL query time, the whole report run time was reduced from 9 minutes 31 seconds to 7 minutes 50 seconds.

**Leverage In-Memory Advisor**

Oracle In-Memory Advisor is designed to help to answer the questions: which tables and/or partitions should you mark for In-Memory column store. It is an Oracle new feature, licensed as part of the Database Tuning pack. Check MOS note: 1965343.1 for more details about Oracle In-Memory Advisor.

There are two whitepapers: Oracle Database In-Memory Advisor and Oracle Database In-Memory Advisor Best practices, published in February 2015.

The In-Memory Advisor estimates analytic processing performance improvement factors based upon the following:

- Eliminating user I/O waits, cluster transfer waits, buffer cache latch waits,
- Certain query processing advantages related to specific compression types.
- Decompression cost heuristics per specific compression types.
- SQL plan selectivity, number of columns in the result set, etc.

The process to use In-Memory Advisor:

1. Step 1: Download the In-Memory Advisor per instruction in MOS note: 1965343.1
2. Step 2: Through the mapping from presentation layer to the physical layer to identify the physical SQL for the report.
Step 3: Through the physical SQL to identify the underneath full table scan operation.

The rest article uses the EDI Queue report as an example to use the process.

- **Download and Install In-Memory Advisor**
  - Download `imadvisor.zip` from Oracle, copy to DB server and unzip it
  - Installed in SQLPLUS with sysdba privilege
  
  ```sql
  SQL> @instimadv.sql
  
  ❑ Do you currently have a valid Oracle Tuning Pack license with this database (Y/N)?
  ❑ Create a new user called IM ADVISOR
  ❑ Need to provide the connection string (from TNSNAME entry)
  ❑ Need to specify the users that will use this tool for tuning:
    Please enter a comma separated list of Oracle Database users to whom you wish EXECUTE on the DBMS_INMEMORY_ADVISOR package to be GRANTed? such as APD_STAR user
    You can GRANT EXECUTE ON DBMS_INMEMORY_ADVISOR to additional users as needed.

- **Running In-Memory Advisor**
  - Run script `imadvisor_analyze_and_report.sql` as a user with the privilege to execute the DBMS_INMEMORY_ADVISOR package:
    ```sql
    SQL> @imadvisor_analyze_and_report
    
    Specify the IM task name
    The IM Advisor generates a report as `imadvisor_<taskname>.html` file in the current working directory
    The sql file is generated as `imadvisor_sql_<taskname>.sql`
    Enter value for `im_task_name`: test
    IM Task name Specified: test
    Enter begin time for report: …
    Enter value for `begin_time`: -1:30
    Report begin time specified: -1:30
    …
    Enter duration in minutes starting from begin time:
    Defaults to SYSDATE - begin_time
    Enter value for duration: 60
    Report duration specified: 60
    Using 2015-MAR-13 09:33:13.000000000 as report begin time
    Using 2015-MAR-13 10:33:13.000000000 as report end time
    IM Advisor: Adding Statistics..
Output of In-Memory Advisor

- imadvisor_taskname.html
  - summary of the Total Database Time analyzed
  - percentage for Database Time for Analytics Processing
  - In-Memory sizes vs the estimated benefit
  - Recommending the top objects to place in memory
  - And compression type and estimated benefit

- imadvisor_taskname.sql
  A script file that contains the SQL which can be run on the target
to modify the objects recommended to be placed In-Memory
You can modify the SQL DDLs to fine tune

Acknowledgement

I would like to thank my colleague Rodrigo Radtke for his great helps on Oracle OBIEE and Dashboard applications.