ORACLE

# Oracle Database In-Memory

## AskTOM Office Hours – 23c New Features – November 9, 2023

**Andy Rivenes**
Database In-Memory Product Manager
Twitter: @TheInMemoryGuy
Email: andy.rivenes@oracle.com

# Database In-Memory 23c New Features



# Oracle Database 23c Free

- All of the new features discussed today are available in Oracle Database 23c Free
- The 23c Free version is 23.3
- Examples were made using 23c Free in a Virtual Box environment
  - SGA_TARGET = 1400M
  - INMEMORY_SIZE = 800M (for AIM tests this was reduced to 400M)
- SSB schema from the Database In-Memory LiveLabs environment was used for most examples

- Oracle Database 23c Free is available here: https://www.oracle.com/database/free/
- Oracle Database 23c Free limitations:
  - 2 CPUs for foreground processes
  - 2GB of RAM (SGA and PGA combined)
  - 12GB of user data on disk (irrespective of compression factor)

# 23c Free Database In-Memory Features

- Changes to v$inmemory_area
- New - Selective Columns
- Automatic In-Memory enhancements
- New – Automatic In-Memory Sizing
- In-Memory Deep Vectorization enhancements
- New – In-Memory Advisor
  - In-Memory Eligibility Tool
  - In-Memory Advisor (2.0)



# Changes to v$inmemory_area

- A new pool has been added to the In-Memory area
- This new pool takes most of the structures, and space required, out of the Shared Pool

```
Connected to:
Oracle Database 23c Free Release 23.0.0.0.0 - Develop, Learn, and Run for Free
Version 23.3.0.23.09

SQL> @09_im_usage.sql
Connected.
SQL> SELECT pool, alloc_bytes, used_bytes, populate_status, con_id
  2  FROM   v$inmemory_area;

POOL                     ALLOC_BYTES          USED_BYTES POPULATE_STATUS                CON_ID
---------------- -------------------- -------------------- ------------------------- ----------
1MB POOL                 796,917,760          680,525,824 DONE                               3
64KB POOL                 25,165,824            4,128,768 DONE                               3
IM POOL METADATA          16,777,216           16,777,216 DONE                               3

SQL>
```

# Selective In-Memory Columns

## New INMEMORY Column Include/Exclude Syntax

- Now in 23c you can specify ALL columns as INMEMORY or NO INMEMORY and then selectively include or exclude columns from population:
  - ALTER TABLE <table name> NO INMEMORY(ALL) INMEMORY(col1, col2);
  - ALTER TABLE <table name> INMEMORY(ALL) NO INMEMORY(col1, col2);



# Selective In-Memory Columns Example

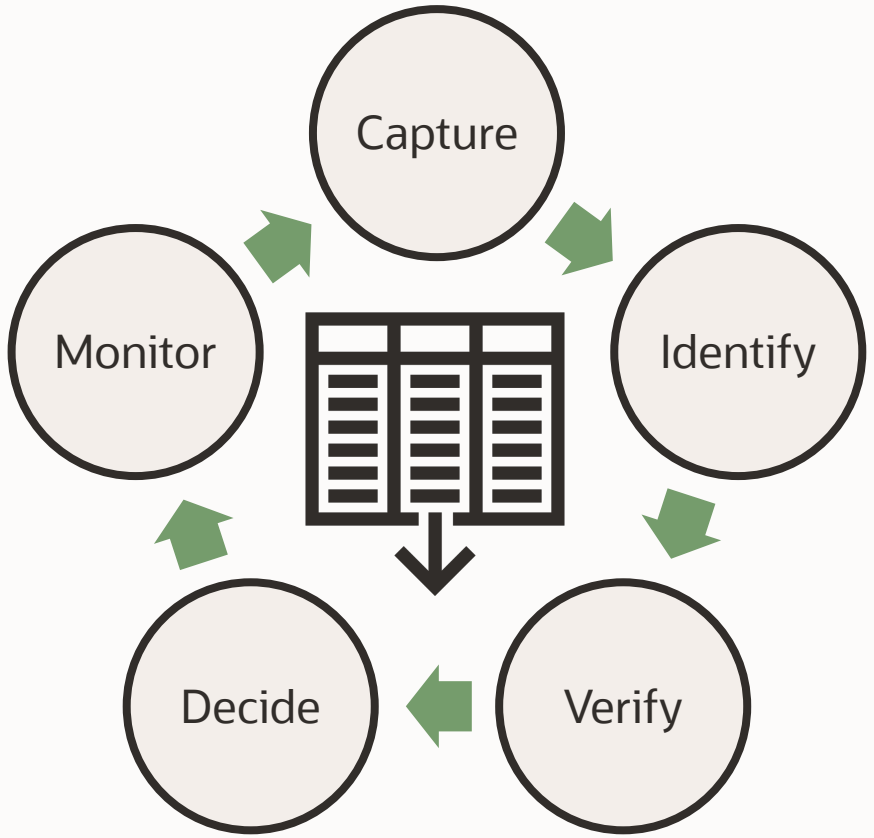
```
SQL> alter table cust_test inmemory (c_custkey,c_name) no inmemory(all);

Table altered.

SQL> SELECT TABLE_NAME, COLUMN_NAME, INMEMORY_COMPRESSION from V$IM_COLUMN_LEVEL where TABLE_NAME = 'CUST_TEST';

TABLE_NAME      COLUMN_NAME     INMEMORY_COMPRESSION
--------------- --------------- -------------------------
CUST_TEST       C_CUSTKEY       DEFAULT
CUST_TEST       C_NAME          DEFAULT
CUST_TEST       C_ADDRESS       NO INMEMORY
CUST_TEST       C_CITY          NO INMEMORY
CUST_TEST       C_NATION        NO INMEMORY
CUST_TEST       C_REGION        NO INMEMORY
CUST_TEST       C_PHONE         NO INMEMORY
CUST_TEST       C_MKTSEGMENT    NO INMEMORY

8 rows selected.

SQL>
```



# Automatic Enablement of In-Memory Features

NEW IN 23c

- In Oracle Database 23c AIM has been enhanced to add the ability to automatically:
  - Enable Join Groups where beneficial
  - Enable In-Memory Optimized Arithmetic for beneficial number columns
  - Enable higher compression levels on specific columns
  - Unpopulate unused columns
- Reduces manual effort required to leverage key performance features
- Enhanced workload analysis to better account for mixed workload environments
  - DML overhead with In-Memory (for example, fetching invalid rows in column store from buffer cache) is factored into analysis
- No application changes required



# Automatic In-Memory Features
## Control and Status

- DBMS_AUTOIM Package
  - SET_PARAMETER
    - Control feature creation (on or off)
    - Set statistics time window
  - ACTIVITY_REPORT
    - Generate AIM task activity for a specified time interval

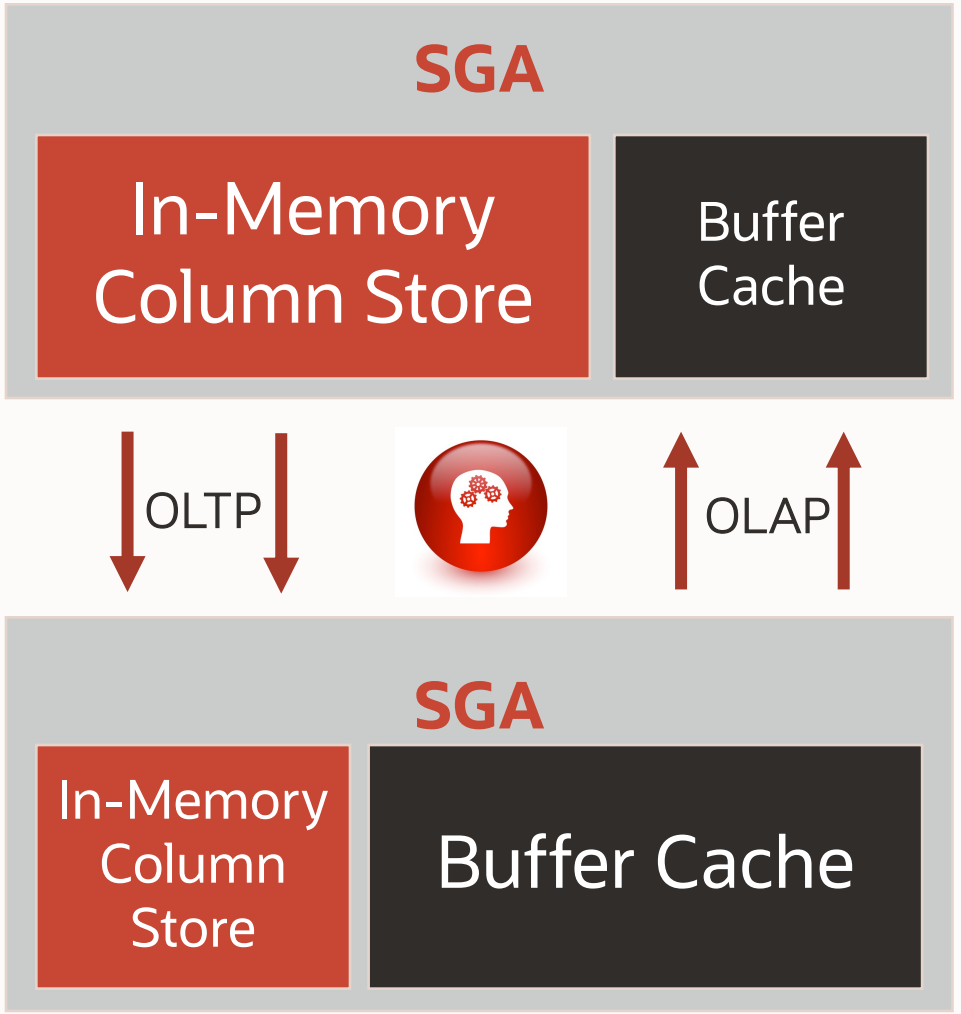
- DBA_AIM_PERF_FEATURES View
  - Determine owner, table, columns that have AIM performance features enabled
  - Features:
    - Optimized Arithmetic
    - Bloom filter optimization (cached hash values)
    - Stored using vector optimization
    - Join Group



# Automatic In-Memory Sizing

## Automatic Sizing of the In-Memory Column Store

- Auto-size column store to factor in different workload types sharing resources at the same time
  - Shrink column store and increase buffer cache if DML intensive workload predicted.
  - Grow column store and shrink buffer cache if OLAP intensive workload predicted
- Works with Automatic In-Memory (AIM)
  - Requires AIM level HIGH and ASMM to be enabled (i.e. SGA_TARGET)
- The INMEMORY_SIZE parameter sets the minimum IM column store size
- Works in conjunction with In-Memory Hybrid Exadata Scans to ensure that partially populated objects can be accessed both in the IM column store and IM columnar format in Exadata Smart Flash Cache (i.e. cell memory)



# Automatic In-Memory Sizing Example

Required parameter settings:

- heat_map = on
- sga_target
- inmemory_automatic_level = high

IM column store must be under memory pressure:

| OWNER | SEGMENT_NAME | PARTITION_NAME | POPULATE_STATUS | Disk Size | In-Memory Size | Bytes Not Populated |
|---|---|---|---|---|---|---|
| SSB | CUSTOMER | | COMPLETED | 24,928,256 | 23,199,744 | 0 |
| SSB | DATE_DIM | | COMPLETED | 122,880 | 1,179,648 | 0 |
| SSB | LINEORDER | PART_1996 | OUT OF MEMORY | 565,018,624 | 263,061,504 | 267,354,112 |
| SSB | LINEORDER | | OUT OF MEMORY | 563,470,336 | 210,501,632 | 323,739,648 |
| SSB | LINEORDER | | STARTED | 563,322,880 | 131,072 | 563,322,880 |
| SSB | PART | | COMPLETED | 56,893,440 | 16,973,824 | 0 |
| SSB | SUPPLIER | | COMPLETED | 1,769,472 | 2,228,224 | 0 |

# Automatic In-Memory Sizing Example, Part 2

```
select
  component, oper_type, oper_mode, parameter, initial_size,
  target_size, final_size,
  to_char(start_time,'MM/DD/YYYY HH24:MI:SS') start_time,
  to_char(end_time,'MM/DD/YYYY HH24:MI:SS') end_time
from
  v$sga_resize_ops
where
  component in ('shared_pool','DEFAULT buffer cache','In-Memory Area')
order by
  start_time
/
```

```
COMPONENT            OPER_TYPE     OPER_MODE PARAMETER       INITIAL_SIZE TARGET_SIZE FINAL_SIZE START_TIME          END_TIME
-------------------- ------------- --------- --------------- ------------ ----------- ---------- ------------------- -------------------
In-Memory Area       STATIC                  inmemory_size              0   419430400  419430400 11/06/2023 18:11:09 11/06/2023 18:11:09
DEFAULT buffer cache INITIALIZING            db_cache_size      226492416   226492416  226492416 11/06/2023 18:11:09 11/06/2023 18:11:09
DEFAULT buffer cache STATIC                  db_cache_size              0   226492416  226492416 11/06/2023 18:11:09 11/06/2023 18:11:09
In-Memory Area       GROW          DEFERRED  inmemory_size      419430400   545259520  545259520 11/06/2023 18:25:32 11/06/2023 18:25:32
DEFAULT buffer cache SHRINK        DEFERRED  db_cache_size      226492416   100663296  100663296 11/06/2023 18:25:32 11/06/2023 18:25:32
```



# In-Memory Vectorization Framework: Multi-Level Joins and Aggregations

- In-Memory Deep Vectorization Framework was introduced on Oracle Database 21c
  - The first feature that leveraged the framework improved the performance of single level hash joins
- In Oracle Database 23c the In-Memory Deep Vectorization Framework adds more join support:
  - Multiple Join Keys
  - Semi and Outer Join
  - Fully Grouping and Aggregation
  - Multiple Levels of Joins
- This translates to faster join performance by utilizing SIMD optimizations during join processing



# Semi-Join Example

```
select /*+ MONITOR */ count(l.lo_custkey)
from lineorder l
where l.lo_partkey IN (select p.p_partkey from part p)
and l.lo_quantity <= 3;
```

```
-----------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name      | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |           |       |       |       |  5113 (100)|          |       |       |
|   1 |  SORT AGGREGATE               |           |     1 |    18 |       |            |          |       |       |
|*  2 |   HASH JOIN RIGHT SEMI        |           |  1640K|    28M|    12M|  5113   (8)| 00:00:01 |       |       |
|   3 |    TABLE ACCESS INMEMORY FULL | PART      |   800K|  3906K|       |    73   (3)| 00:00:01 |       |       |
|   4 |    PARTITION RANGE ALL        |           |  1640K|    20M|       |  2448  (15)| 00:00:01 |     1 |     3 |
|*  5 |     TABLE ACCESS INMEMORY FULL| LINEORDER |  1640K|    20M|       |  2448  (15)| 00:00:01 |     1 |     3 |
-----------------------------------------------------------------------------------------------------
```



# Semi-Join Example, Part 2

## SQL*Plus way to determine Deep Vector usage (similar to Join Groups)

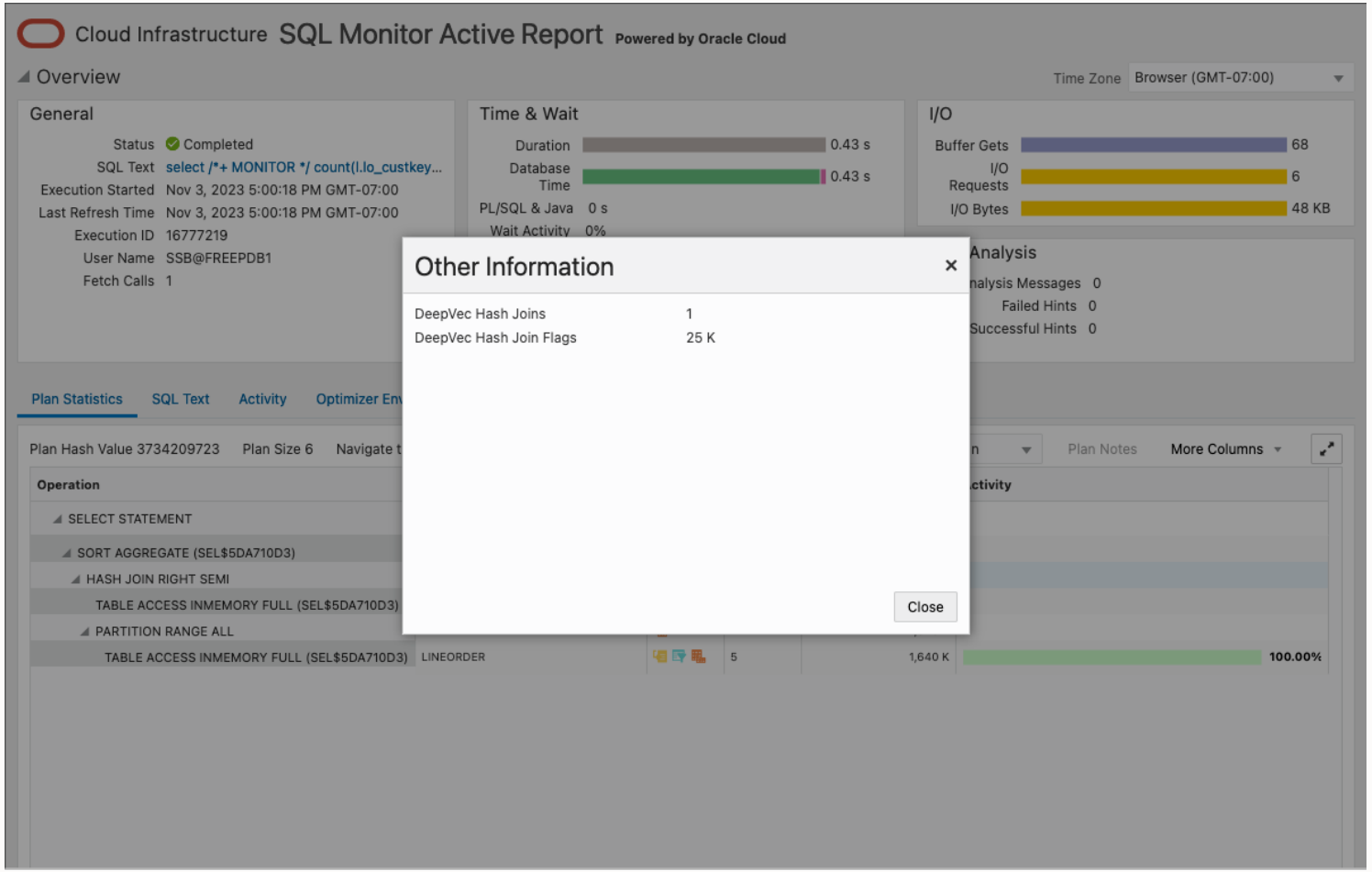
```
SELECT
  '   ' || deepvec.rowsource_id || ' - ' row_source_id,
    CASE
      WHEN deepvec.deepvec_hj IS NOT NULL
      THEN
        'deep vector hash joins used: ' || deepvec.deepvec_hj || ', deep vector hash join flags: ' || deepvec.deepvec_hj_flags
      ELSE
        'deep vector HJ was NOT leveraged'
    END deep_vector_hash_join_usage_info
FROM
  (SELECT EXTRACT(DBMS_SQL_MONITOR.REPORT_SQL_MONITOR_XML, q'#//operation[@name='HASH JOIN' and @parent_id]#') xmldata
   FROM   DUAL) hj_operation_data,
  XMLTABLE('/operation'
    PASSING hj_operation_data.xmldata
    COLUMNS
     "ROWSOURCE_ID"        VARCHAR2(5) PATH '@id',
     "DEEPVEC_HJ"          VARCHAR2(5) PATH 'rwsstats/stat[@id="11"]',
     "DEEPVEC_HJ_FLAGS"    VARCHAR2(5) PATH 'rwsstats/stat[@id="12"]') deepvec;
```

```
Deep Vectorization Usage:
-------------------------
   2 -      deep vector hash joins used: 1, deep vector hash join flags: 24576
```



# Semi-Join Example, Part 3

## SQL Monitor way to determine Deep Vector usage



# Semi-Join Example, Part 4

## Execution Plan Output

```
Plan hash value: 3734209723

-------------------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name      | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |           |       |       |       |  5113 (100)|          |       |       |
|   1 |  SORT AGGREGATE               |           |     1 |    18 |       |            |          |       |       |
|*  2 |   HASH JOIN RIGHT SEMI        |           |  1640K|    28M|    12M|  5113   (8)| 00:00:01 |       |       |
|   3 |    TABLE ACCESS INMEMORY FULL | PART      |   800K|  3906K|       |    73   (3)| 00:00:01 |       |       |
|   4 |    PARTITION RANGE ALL        |           |  1640K|    20M|       |  2448  (15)| 00:00:01 |     1 |     3 |
|*  5 |     TABLE ACCESS INMEMORY FULL| LINEORDER |  1640K|    20M|       |  2448  (15)| 00:00:01 |     1 |     3 |
-------------------------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("L"."LO_PARTKEY"="P"."P_PARTKEY")
   5 - inmemory("L"."LO_QUANTITY"<=3)
       filter("L"."LO_QUANTITY"<=3)
25 rows selected.

SQL>
SQL> set echo off
Deep Vectorization Usage:
-------------------------

   2 -      deep vector hash joins used: 1, deep vector hash join flags: 24576
```



# New Database Embedded In-Memory Advisor

- Previous version
  - Standalone package that had to be installed in the database
  - Analyzed existing database workload using AWR/ASH data to determine benefit that Database In-Memory might provide
  - Generated an HTML report with estimates for overall benefit, benefit for top SQL statements and objects based on an in-memor size

- New in 23c, the In-Memory Advisor is now part of Oracle Database
- Relies on Heat Map data for analysis
  - Heat Map is now available as part of Oracle Database Enterprise Edition (no separate license)
- A new Eligibility Tool has been added to quickly identify databases where Database In-Memory would NOT be useful – Backported to 19c (19.20)
- A comprehensive advisor analysis based on a workload timeframe and an object benefit analysis

# In-Memory Advisor Example
## Eligibility Tool

```
SQL> variable inmem_eligible BOOLEAN
SQL> variable analysis_summary VARCHAR2(4000)
SQL> exec dbms_inmemory_advise.is_inmemory_eligible(26, 30, :inmem_eligible, :analysis_summary);

PL/SQL procedure successfully completed.

SQL> print inmem_eligible

INMEM_ELIGI
-----------
TRUE

SQL> print analysis_summary

ANALYSIS_SUMMARY
--------------------------------------------------------------------------------------------------
Observed Analytic Workload Percentage is 100% is greater than target Analytic Workload Percentage 20%


SQL>
```



# In-Memory Advisor Example, Part 1

## New Native In-Memory Advisor

```
SQL> variable taskid NUMBER;
SQL> exec dbms_inmemory_advise.start_tracking(:taskid);

PL/SQL procedure successfully completed.

SQL> print taskid

    TASKID
----------
         1

SQL>

*** Do analytic work here ***

SQL> exec dbms_inmemory_advise.stop_tracking;

PL/SQL procedure successfully completed.

SQL>
```



# In-Memory Advisor Example, Part 2

## New Native In-Memory Advisor

```
SQL> exec dbms_inmemory_advise.generate_advise;

PL/SQL procedure successfully completed.

SQL> select round(INMEMORY_SIZE/1024/1024,3) as inmemory_size_mb,
 trunc((ESTIMATED_DB_TIME_HIGH/60),2) as estimated_db_time_minutes,
 recommended_obj_list as recommended_objects
from dba_inmemory_advisor_recommendation
where task_id = 4;
  2    3    4    5
INMEMORY_SIZE_MB      ESTIMATED_DB_TIME_MINUTES RECOMMENDED_OBJECTS
---------------- ------------------------------ --------------------------------------------------------------------------------
               0                          10.18
             302                           6.73 Owner: SCOTT Table: ANA_1 ; Owner: SCOTT Table: ANA_5 ; Owner: SCOTT Table: ANA_
                                                4 ;

             528                           5.06 Owner: SCOTT Table: ANA_1 ; Owner: SCOTT Table: ANA_5 ; Owner: SCOTT Table: ANA_
                                                4 ; Owner: SCOTT Table: ANA_3 ;

             754                            3.4 Owner: SCOTT Table: ANA_1 ; Owner: SCOTT Table: ANA_5 ; Owner: SCOTT Table: ANA_
                                                4 ; Owner: SCOTT Table: ANA_3 ; Owner: SCOTT Table: ANA_2 ;
```

# Where Can You Get More Information?



# https://blogs.oracle.com/in-memory/dbim-resources