ORACLE 12c
DATABASE
IN-MEMORY

An Oracle White Paper
January, 2017

Oracle Database In-Memory Advisor

ORACLE

Disclaimer

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.

Intended Audience

Readers are assumed to have hands-on experience with Oracle Database technologies from the perspective of a DBA or performance specialist.

Introduction

Oracle Database 12.1.0.2 introduced Oracle Database In-Memory allowing a single database to efficiently support mixed analytic and transactional workloads. An Oracle Database configured with Database In-Memory delivers optimal performance for transactions while simultaneously supporting real-time analytics and reporting. This is possible due to a unique "dual-format" architecture that enables data to be maintained in both the existing Oracle row format, for OLTP operations, and a new purely in-memory column format, optimized for analytical processing. In-Memory also enables both datamarts and data warehouses to provide more ad-hoc analytics, giving end-users the ability to ask multiple business driving queries in the same time it takes to run just one now.

For complete details about Oracle Database In-Memory, see the Oracle Database In-Memory whitepaper and the Oracle Database In-Memory Page on oracle.com.

This paper discusses the Oracle Database In-Memory Advisor. The Advisor analyzes your workload and makes specific recommendations regarding how to size Oracle Database In-Memory and which objects would render the greatest benefit to your system when placed In-Memory.

In-Memory Advisor Description

The goal of Oracle Database In-Memory is to optimize analytical processing in the database. The In-Memory Advisor analyzes the analytical processing workload present in your database to determine an estimated benefit for the database as a whole if that analytical workload is optimized.

For transactional processing, the benefit of Database In-Memory is that it allows you to drop indexes used for analytical processing, thereby reducing the index maintenance overhead for transactions.

The In-Memory Advisor differentiates analytics processing from other database activity based upon SQL plan cardinality, Active Session History (ASH), use of parallel query, and other statistics.

The In-Memory Advisor estimates the In-Memory size of objects based upon statistics and heuristic compression factors.

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

- Elimination of user I/O waits, cluster transfer waits, buffer cache latch waits, etc.
- 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 Advisor produces a recommendation report optimized for the In-Memory size you specify. If you omit the In-Memory size, it defaults to the largest In-Memory size recommended by the Advisor based on its analysis. Independent of the In-Memory size you specify, the top of the report lists a number of In-Memory sizes with estimated performance benefits. With this information, you may choose a different In-Memory size. The rest of the report shows recommendations optimized for the In-Memory size you choose.

It is not necessary to run the Advisor again to optimize for a different In-Memory size. Just generate a second report specifying your revised In-Memory size, and the second report will provide recommendations optimized for the revised In-Memory size. This can be done any number of times without re-running the Advisor from scratch, saving a significant amount of time.

Once you have generated a report optimized for your chosen In-Memory size, the next section of the report lists the SQL statements with the highest estimated performance benefit from the specified In-Memory size optimization. Next in the report is a list of the objects which are recommended to be placed in the In-Memory column store along with a recommended compression type for each object.

Along with the report, the Advisor also produces a SQLPLUS script to modify the recommended objects to place them In-Memory with the recommended compression types.

This output is described in more detail in the 'Advisor Output' section below.

Installing the Advisor

Supported Versions and Licensing

The Advisor can be installed on Oracle Database Version 11.2.0.3 and above. The Advisor is licensed as part of the Oracle Tuning Pack. Further information can be found in the Licensing Guide.

Installation Process

It is recommended to log in as SYS to install the Advisor. When installing as a different user, the installation process will cite any missing privileges. In addition, when installing the Advisor as a user other than SYS on Oracle Database 12.1 and above, the installation process will describe additional required actions and implications.

The Advisor can be installed into either a multitenant database or non-multitenant database. If installing into a multitenant database, the Advisor can be installed in the root, CDB$ROOT, or in a pluggable database (PDB).

When installing the Advisor into a pluggable database (PDB), or a non-multitenant database, the installation procedure will create a user IMADVISOR to contain the Advisor objects. When installing into the root of a multitenant database, the CDB$ROOT, a user named C##IMADVISOR will be created.

The following procedure shows the installation from a non-multitenant database. The installation in a PDB would be similar. If connected to the CDB$ROOT of a multitenant database, the prompts below would reflect the C##IMADVISOR username.

To install the Advisor on your database (user entries are shown in bold and highlighted in yellow):

```
$ unzip imadvisor.zip
$ sqlplus sys/<password> as sysdba

SQL*Plus: Release 12.1.0.2.0 Production
Copyright (c) 1982, 2014, Oracle.  All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing
options

SQL> @instimadv
Welcome to the Oracle Database In-Memory Advisor (DBMS_INMEMORY_ADVISOR)
installation.


DBMS_INMEMORY_ADVISOR uses Active Session History (ASH), Automatic Workload
Repository (AWR) and optionally SQL Tuning Sets (STS) to determine which
tables, partitions and subpartitions to place In Memory for optimized
analytics processing performance.  DBMS_INMEMORY_ADVISOR produces a
recommendation report and a SQLPlus script to implement its recommendations.

DBMS_INMEMORY_ADVISOR users require the ADVISOR privilege.
```

```
This installation script will create user IMADVISOR and add object
definitions to the schema.  This user is created using the IDENTIFIED BY
password method with a random-generated password.  If you prefer to use
either the IDENTIFIED EXTERNALLY or IDENTIFIED GLOBALLY method, abort this
installation by pressing ^C.  Then create user IMADVISOR using your preferred
method.  Add no objects or grants to the IMADVISOR schema.  Then run
this installation script again.

User IMADVISOR requires both a permanent and temporary tablespace.
Available tablespaces:


TABLESPACE_NAME
------------------------------
SYSAUX
SYSEXT
SYSTEM (default permanent tablespace)
TEMP (default temporary tablespace)
UD1


Enter value for permanent_tablespace: SYSTEM

Permanent tablespace to be used with IMADVISOR: SYSTEM


Enter value for temporary_tablespace: TEMP

Temporary tablespace to be used with IMADVISOR: TEMP

No errors.
No errors.
.
.
.
No errors.
No errors.

All done!

DBMS_INMEMORY_ADVISOR installation successful.

Users who will use the DBMS_INMEMORY_ADVISOR package must be granted
the ADVISOR privilege.


DBMS_INMEMORY_ADVISOR installation and setup complete.

To uninstall:

SQL> @catnoimadv.sql
```

```
Disconnected from Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing
options
```

Running the Advisor

The PL/SQL interface to the components of the Advisor is described at the end of this paper. In Appendix 1, there is a sample script you can use to run the Advisor on your database. This sample script can also be modified to take advantage of the additional functionality available in the PL/SQL procedures described below.

If you are running the Advisor from the root of a multitenant database, CDB$ROOT, you must supply a PDB_NAME value in the CREATE_TASK procedure. The CREATE_TASK procedure is documented later in this paper.

An interactive script to run the Advisor and produce the report is also included in the file imadvisor.zip. This script is 'imadvisor_recommendations.sql'. If running from a CDB$ROOT, the script will prompt you for a PDB name.

To run the Advisor using the included script, start SQLPLUS as a user with the privilege to execute the DBMS_INMEMORY_ADVISOR package (user entries are shown in bold and highlighted in yellow; descriptions of variations in the behavior of the interactive script that depend on your environment and choices are shown in italics and highlighted in green):

```
$ sqlplus sys/<password> as sysdba
SQL> @imadvisor_recommendations


This script creates and runs an In-Memory Advisor task that analyzes
your workload to determine an optimal In-Memory configuration.


This script then generates an HTML recommendation report file in the
current working directory: imadvisor_<task_name>.html


This script also generates a sqlplus DDL script to implement the
recommendations: imadvisor_<task_name>.sql
```

If you have not yet created any tasks, the following list of existing tasks along with the explanation of creating new tasks vs. using existing tasks will be omitted.

```
NOTE: You may specify one of your existing tasks if you wish to optimize for
a different In-Memory size.

Using an existing, executed task is faster than a new task since a new task
requires statistics gathering and analysis.

But if you wish to analyze a different workload or use a different statistics
capture window or add a SQLSET, you must specify a new task.

The following is a list of your existing tasks:

TASK_NAME                      DATE_CREATED
------------------------------ --------------------
my_task1                       2016-MAR-09 15:08:09
my_task2                       2016-MAR-09 15:09:20

Default task_name (new task): im_advisor_task_20160310145038
Enter value for task_name: my_task3
```

If you specify an existing Advisor task name, all of the following information and prompts are omitted up to the point where the performance benefit / cost estimates are presented after statistics gathering and analysis – see below.

```
Advisor task name specified: my_task3

New Advisor task will be named: my_task3...
```

The following prompt will be omitted if you have not imported any augmented AWR workloads.

```
By default, the Advisor runs against a live workload on this database.
This database also has imported, augmented AWR workloads.

Press ENTER or respond NO to run against a live workload.
Respond YES to run against an augmented AWR workload.

Enter value for run_against_augmented_awr: YES

The Advisor can use the following augmented AWR imports:
```

```
Augmented AWR Import DBID
-------------------------
1502131805
1503031036
1512071241

Enter value for dbid: 1503031036
Analyzing and reporting on an augmented AWR workload with DBID=1503031036...
```

The following prompt for a PDB name is omitted unless you are running against a live workload on a CDB root or an augmented AWR workload captured from a CDB root.

```
Enter value for pdb_name: CDB1_PDB1
CDB1_PDB1
```

The following prompt for an Automatic Workload Repository (AWR) store is omitted unless you are running against a live workload on a 12.2+ PDB.

```
When analyzing a live workload on a PDB, you have the option to use the
AWR root store (default) or AWR PDB store.
Press ENTER or specify ROOT or PDB:
Enter value for awr_store: PDB
```

The following prompt for a RAC instance number is omitted unless you are running against a live workload on a multi-instance RAC or an augmented AWR workload captured from a multi-instance RAC.

```
Choose one of the following instance numbers for this workload:
INSTANCE_NUMBER DEFAULT_INSTANCE
--------------- ----------------
              1        *
              2

Enter value for instance_number: 2
The In-Memory Advisor will analyze the workload from instance number 2.
```

Unless you initially specify an existing Advisor task name, the following In-Memory size description and prompt is presented regardless of your previous choices.

```
The In-Memory Advisor optimizes the In-Memory configuration for a specific
In-Memory size that you choose.
```

```
After analysis, the In-Memory Advisor can provide you a list of performance
benefit estimates for a range of In-Memory sizes.  You may then choose the
In-Memory size for which you wish to optimize.

If you already know the specific In-Memory size you wish, please enter
the value now.  Format: nnnnnnn[KB|MB|GB|TB]

Or press <ENTER> to get performance estimates first.
Enter value for inmemory_size: <ENTER>

The In-Memory Advisor will display performance benefit estimates after
analysis.

Enter begin time for report:

--    Valid input formats:
--      To specify absolute begin time:
--        [MM/DD[/YY]] HH24:MI[:SS]
--        Examples: 02/23/03 14:30:15
--                  02/23 14:30:15
--                  14:30:15
--                  14:30
--      To specify relative begin time: (start with '-' sign)
--        -[HH24:]MI
--        Examples: -1:15  (SYSDATE - 1 Hr 15 Mins)
--                  -25    (SYSDATE - 25 Mins)
```

The default begin time with a live workload is -60 minutes.

```
Default begin time: 03/02/15 10:00:34
Enter value for begin_time: <ENTER>

Report begin time specified:
```

The default duration time with a live workload is SYSDATE-begin_time.

```
Enter duration in minutes starting from begin time:
(defaults to <latest-snapshot-end-time> - begin_time)

Enter value for duration: <ENTER>
```

```
Report duration specified:

Using 2015-MAR-02 10:00:34.000000000 as report begin time
Using 2015-MAR-03 09:00:59.000000000 as report end time
```

You may optionally specify a comma separated list of object owner
and name patterns to be considered for In Memory Placement.
Example:

GEEK_SUMMARY.%,%.GEEK_%

Press ENTER to consider all objects.

You may specify any number of object owner and name patterns, each separated by a comma (","). You may also use a percent sign ("%") as a wild card. All letters are case sensitive. In a given object pattern, separate the object owner from the object name with a period ("."). Press ENTER without specifying any patterns and all objects will be considered.

Enter value for consider_objects_like: **SR3.%BIT%,%AR3.%DATE%,SR1.PROD,ACCOUNT.%**

Considering only objects matching these patterns for In Memory placement:

SR3.%BIT%,%AR3.%DATE%,SR1.PROD,ACCOUNT.%

```
In-Memory Advisor: Adding statistics...

In-Memory Advisor: Finished adding statistics.

In-Memory Advisor: Analyzing statistics...

In-Memory Advisor: Finished analyzing statistics.
```

If you specify an existing Advisor task name at the start of this script, it will immediately go to this section with the performance benefit / cost estimates.

If you specify an In-Memory size prior to statistics gathering and analysis, the following performance / cost estimates are omitted .

```
The In-Memory Advisor estimates the following performance benefits:
```

IN-MEMORY SIZE	PERCENTAGE OF MAXIMUM SGA SIZE	ESTIMATED ANALYTICS PROCESSING TIME REDUCTION (SECONDS)*	ESTIMATED ANALYTICS PROCESSING PERFORMANCE IMPROVEMENT FACTOR*
16.73GB	1673	653954	7.8X
15.89GB	1589	26394	1.0X
15.06GB	1506	26394	1.0X
14.22GB	1422	26394	1.0X
13.38GB	1338	26394	1.0X
12.55GB	1255	26394	1.0X
11.71GB	1171	26394	1.0X
10.87GB	1087	26394	1.0X
10.04GB	1004	26394	1.0X
9.201GB	920	26394	1.0X
8.364GB	836	26394	1.0X
7.528GB	753	26394	1.0X
6.692GB	669	26394	1.0X
5.855GB	586	26394	1.0X
5.019GB	502	26394	1.0X
4.182GB	418	26394	1.0X
3.346GB	335	26394	1.0X
2.509GB	251	26394	1.0X
1.673GB	167	26394	1.0X
856.5MB	84	26394	1.0X

*Estimates: The In-Memory Advisor's estimates are useful for making In-Memory decisions. But they are not precise. Due to performance variations caused by workload diversity, the Advisor's performance estimates are conservatively limited to no more than 10.0X faster.

Choose the In-Memory size you wish for optimization (default=16.73GB):
Enter value for inmemory_size: **<ENTER>**

```
The In-Memory Advisor is optimizing for an In-Memory size of 16.73GB...

(You can re-run this task with this script and specify a different an In-
Memory size.  Re-running a task to optimize for a different In-Memory size is
faster than creating and running a new task from scratch.)

Fetching recommendation files for task: my_task3

Placing recommendation files in: the current working directory

Fetched file: imadvisor_mytask3.html
Purpose:      recommendation report primary html page

Fetched file: imadvisor_tpch.sql
Purpose:      recommendation DDL sqlplus script
```

In the example above, the output files are listed as the generated report and the generated DDL script. The contents of the output are described in the next section.

Advisor Output

The Advisor produces its output with the DBMS_INMEMORY_ADVISOR.GENERATE_RECOMMENDATIONS procedure. That procedure produces the following output:

imadvisor_TASKNAME.html

The Advisor summary report is called '`imadvisor_`*`taskname`*`.html`' where '*taskname*' is the name of the task you gave the Advisor.

This report has several sections.

At the top is a summary of the Total Database Time analyzed in the report, and which percentage of that Database Time can be ascribed to Analytics Processing.

The section labeled 'In-Memory sizes' contains a table giving possible In-Memory sizes and the estimated benefit from each of those In-Memory sizes. Based on this information, you may wish to generate another report with a different In-Memory size. You can choose to optimize for any In-Memory you choose – it need not be included in the list.

Below this section there is a table summarizing the performance benefit of the SQL statements with the highest estimated performance improvements with the optimization for the In-Memory size you specified. Click the "All SQL Statements" link at the bottom of this table to view the estimated SQL performance benefits of all SQL statements. (The Advisor omits this link when the number of SQL statements is 10 or less.)

Below this section there is a table summarizing the recommendations of the top objects to place in memory with their recommended compression type and estimated benefit with the optimization for the In-Memory size you specified. Click the "All Objects" link at the bottom of this table to view all recommended objects. (The Advisor omits this link when the number of objects is 10 or less.) Click the "Rationale Summary" link to view a rationale summary for the recommendations.

Finally, at the bottom of the report there is a table of information about the database for which the Advisor optimized and a table describing the Advisor's analysis methods. Click the "DDL Script" link to view DDL script that will implement the recommendations (see file imadvisor_TASKNAME.sql).

imadvisor_TASKNAME.sql

In addition to the html report, a script file is generated that contains the SQL which can be run on the target database to modify the objects recommended to be placed In-Memory along with the recommended compression types. If you wish, you can modify the DDL to fine tune non-default parameters. However, Oracle recommends you do not remove or add objects or change the recommended compression types as these have been optimized for the In-Memory size you specified. Changing the list of recommended objects or the recommended compression types can be less than optimal. For an optimal In-Memory configuration for a different In-Memory size, generate another Advisor report specifying the In-Memory size you wish.

Running the Advisor Against AWR Data From A Different Database

You can capture Automatic Workload Repository (AWR) data from a database using the $ORACLE_HOME/rdbms/admin/awrextr.sql script. This operation captures the AWR data into a Data Pump dump file. You can subsequently load it onto a different database using the $ORACLE_HOME/rdbms/admin/awrload.sql script. Refer to the Transporting Automatic Workload Repository Data section in the Oracle Database Performance Tuning Guide for more information about extracting AWR data from one database and loading it onto another.

Loaded AWR data lacks some information required by the Advisor. To use the Advisor with loaded AWR data, you must augment the AWR data using the Advisor's AWR augment feature: To capture the augment data, use the imadvisor_awr_augment_export.sql script on the same database where you use the $ORACLE_HOME/rdbms/admin/awrextr.sql script. It is best to use the imadvisor_awr_augment_export.sql in the same timeframe as you use the $ORACLE_HOME/rdbms/admin/awrextr.sql script, preferably immediately afterward. To load the augment data, you can use the imadvisor_awr_augment_import.sql script on the same database as you use the $ORACLE_HOME/rdbms/admin/awrload.sql script. After loading the AWR augment, you can query which workloads are augmented with the DBA_IMA_AWR_AUGMENTS view.

Refer to Appendix 2 for information on the data contained in the AWR augment.

Adjusting Advisor Parameter Settings

The Advisor employs a number of heuristic parameter settings to estimate in-memory performance, a threshold for reasonable recommendations and determine in-memory eligibility based on minimum size. Refer to the GET_PARAMETER, SET_PARAMETER PL/SQL packages below for the interface that allows you to obtain and change parameter settings. The table in Appendix 4: Advisor Parameters lists the available parameter settings.

Deinstalling the Advisor

If you wish to deinstall the Advisor from your database, execute the following statement with DBA privileges:

```
SQL> @catnoimadv.sql
```

NOTE: Do not use the In-Memory Advisor version 1.0.0.0.0 or version 1.0.0.0.1 deinstall method. The previous deinstall method will not work with version 2.0.0.1.0 or later. However, the version 2.0.0.1.0 deinstall method will work with all versions.

PL/SQL Interface – DBMS_INMEMORY_ADVISOR

This section contains documentation on the PL/SQL interface to the components of the Oracle Database In-Memory Advisor.

CREATE_TASK Procedure

CREATE_TASK creates a DBMS_INMEMORY_ADVISOR task.

Syntax:

```
PROCEDURE create_task (
   task_name          IN VARCHAR2,
   task_desc          IN VARCHAR2   := NULL,
   dbid               IN  NUMBER    := NULL,
   instance_number    IN  NUMBER    := NULL,
   pdb_name           IN VARCHAR2   := NULL,
   awr_store_source   IN VARCHAR2   := NULL);
```

Parameters:

PARAMETER	DESCRIPTION
task_name	A unique name up to 30 characters in length for the DBMS_INMEMORY_ADVISOR task.
task_desc	Optional description of the task up to 256 characters in length.
dbid	Database identifier from which statistics are to be captured. By default or if you specify NULL, statistics are captured from the local database to which the invoker of CREATE_TASK is connected. Alternatively, you can specify the database identifier for a database from which an Automatic Workload Repository (AWR) data has been loaded onto the local database (for example, using the admin/awrload.sql script). In this case, you must also augment the AWR data using the imadvisor_awr_augment_import.sql script. See Running The Advisor Against AWR Data From A Different Database for more details.
instance_number	Database instance number from which statistics are to be captured. By default or if you specify NULL with the local database, statistics are captured from the instance to which the invoker of CREATE_TASK is connected. If you specify a database identifier (dbid) from which AWR was loaded, the default database instance is the instance upon which you ran the AWR augment with the imadvisor_awr_augment_export.sq script.

pdb_name	The name of the PDB for which you desire in-memory recommendations. This parameter only applies to a multitenant (CDB) database. When executing this procedure against a workload in the current database (using the default DBID): • If you are executing from within a CDB root, you must specify a pdb_name and it must match one of the PDBs contained in the current multitenant database. • If you are executing from within a PDB, pdb_name defaults to the connected PDB name. If you specify pdb_name, it must match the connected PDB name. When running against an augmented AWR import (using a DBID corresponding to that import): • An augmented AWR export can be imported onto the same database version or newer. It cannot be imported onto an older database version. • An augmented AWR export from a non-CDB, a CDB root or a PDB can be imported onto a non-CDB, a CDB root or PDB – the database type of the export need not match the import. Exports and imports with PDBs are supported with 12.2+. • With an augmented AWR import from a non multitenant database, you must not specify pdb_name. • With an augmented AWR import from a CDB root, you must specify pdb_name to match one of the PDBs contained by the multitenant database from which it was exported.
awr_store_source	The Automatic Workload Repository (AWR) store to be used: 'ROOT' or 'PDB'. Prior to Oracle Database 12.2, all AWR data is stored on the root. Starting with Multitenant Oracle Database 12.2, AWR data can be stored on the root or on PDBs or a combination of both. A choice of the AWR root store or a PDB AWR store is only applicable with a live workload on a 12.2+ PDB. The In Memory Advisor sets the awr_store_source default accordingly with all other environments. Therefore, only set this parameter with a live workload on a 12.2+ PDB. With a live workload on a 12.2+ PDB, the default is 'ROOT'. You may specify 'PDB'. Be aware that automatic AWR snapshots are disabled by default on the PDB store: prior to running the In Memory Advisor against a live workload on a 12.2 PDB store, manually capture AWR snapshots onto your PDB store or enable automatic AWR snapshots on your PDB store. Starting with Oracle Database 12.2, you can import AWR data onto your PDB store. With an imported AWR workload on a 12.2+ PDB, the In Memory Advisor defaults to 'PDB', which is where the AWR is placed when imported on a PDB. AWR data imported onto the root is not accessible from a PDB. Thus as indicated above, you need not set this parameter with an imported

AWR workload on a 12.2+ PDB as it will correctly default.

The following combinations are NOT supported and will produce errors:

- awr_store_source=>'PDB' on Oracle Database 11.2 or 12.1.
- awr_store_source=>'PDB' on a non-CDB or CDB root.
- awr_store_source=>'ROOT' on a 12.2+ PDB with imported AWR data.

Example:

EXEC dbms_inmemory_advisor.create_task ('MYTASK');

ADD_SQLSET Procedure

ADD_SQLSET adds a captured SQL Tuning Set to the specified task. Since ADD_STATISTICS captures the SQL workload from the Automatic Workload Repository (AWR), addition of any SQL tuning sets is optional. However, the AWR SQL workload contains only the hottest SQL statements (those that use the greatest amount of resources). Therefore, a number of SQL statements may be omitted from the AWR SQL workload. Addition of SQL Tuning Sets helps produce more accurate results when the AWR SQL workload is missing a large percentage of the of the SQL workload. This is most likely to happen with a large number of distinct SQL statements or with ad hoc queries.

After you have executed ADD_STATISTICS, you can call ASH_SQL_COVERAGE_PCT to determine if you need to start over and add SQL tuning sets with broader SQL workload coverage.

Syntax:

```
PROCEDURE add_sqlset (
   task_name           IN  VARCHAR2,
   sqlset_name         IN  VARCHAR2,
   sqlset_owner        IN  VARCHAR2 := NULL);
```

Parameters:

PARAMETER	DESCRIPTION
task_name	Name of the task to which the statistics are to be added.
sqlset_name	Name of the SQL Tuning Set to be added
sqlset_owner	Name of the owner of the SQL Tuning Set to be added.

Usage Notes:

1. Procedure CREATE_TASK must be executed prior to executing this procedure.
2. Execution of procedure ADD_SQLSET is optional. If ADD_SQLSET to be executed, it must be done prior to ADD_STATISTICS.
3. DBMS_INMEMORY_ADVISOR.ADD_STATISTICS captures the SQL data from the Automatic Workload Repository (AWR). Therefore, adding a SQL Tuning Set from AWR (using DBMS_SQLTUNE.SELECT_WORKLOAD_REPOSITORY) is not useful or recommended. Instead, use DBMS_SQLTUNE.CAPTURE_CURSOR_CACHE_SQLSET or another means to create a SQL Tuning Set that contains a broader representation of the SQL workload.

Example:

```
EXEC dbms_inmemory_advisor.add_sqlset ('MYTASK', 'MYSQLSET', USER);
```

ADD_STATISTICS Procedure

ADD_ STATISTICS adds ASH and AWR and other statistics from the specified capture window to the specified task. With a live workload (as opposed to an imported, augmented AWR workload), ADD_STATISTICS also adds live ASH and other statistics for the specified capture window to the specified task. One statistic window must be added prior to use of DBMS_INMEMORY_ADVISOR . EXECUTE_TASK.

Syntax:

```
PROCEDURE add_statistics (
   task_name              IN  VARCHAR2,
   capture_window_start   IN  TIMESTAMP   := NULL,
   capture_window_end     IN  TIMESTAMP   := NULL);
```

Parameters:

PARAMETER	DESCRIPTION
task_name	Name of the task to which the statistics are to be added.
capture_window_start	The starting date and time for the capture window of ASH and AWR statistics to be added to the task. By default or if you specify NULL, the window starts with the oldest statistics available.
capture_window_end	The ending date and time for the capture window of ASH and AWR statistics to be added to the task. By default or if you specify NULL, the window ends with the newest statistics available.

Usage Notes:

1. Procedure CREATE_TASK must be executed prior to executing this procedure.
2. Execution of procedure ADD_SQLSET is optional. If ADD_SQLSET is to be executed, it must be done prior to ADD_STATISTICS.
3. Statistics must be added once using either ADD_STATISTICS or ADD_HIST_STATISTICS prior to executing EXECUTE_TASK. ADD_STATISTICS uses timestamps to define the capture window. ADD_HIST_STATISTICS uses AWR snapshot identifiers to define the capture window.
4. A large capture window provides the Advisor with more data, which can result in more accurate recommendations. However, more data takes more time to process. With a very large capture window, the Advisor may run for a very long time.

Example:

```
EXEC dbms_inmemory_advisor.add_statistics ('MYTASK', SYSTIMESTAMP-60, SYSTIMESTAMP);
```

ADD_HIST_STATISTICS Procedure

ADD_ HIST_STATISTICS adds ASH and AWR and other statistics from the specified capture window to the specified task. One statistic window must be added prior to use of DBMS_INMEMORY_ADVISOR . EXECUTE_TASK.

Syntax:

```
PROCEDURE add_hist_statistics (
   task_name              IN  VARCHAR2,
   start_snap_id          IN  NUMBER := NULL,
   end_snap_id            IN  NUMBER := NULL);
```

Parameters:

PARAMETER	DESCRIPTION
task_name	Name of the task to which the statistics are to be added.
start _snap_id	The starting AWR snapshot identifier for the capture window of ASH and AWR statistics to be added to the task. By default or if you specify NULL, the window starts with the oldest statistics available.
end_snap_id	The ending AWR snapshot identifier for the capture window of ASH and AWR statistics to be added to the task. By default or if you specify NULL, the window ends with the newest statistics available.

Usage Notes:

1. Procedure CREATE_TASK must be executed prior to executing this procedure.
2. Execution of procedure ADD_SQLSET is optional. If ADD_SQLSET is to be executed, it must be done prior to ADD_HIST_STATISTICS.
3. Statistics must be added once using either ADD_STATISTICS or ADD_HIST_STATISTICS prior to executing EXECUTE_TASK. ADD_STATISTICS uses timestamps to define the capture window. ADD_HIST_STATISTICS uses AWR snapshot identifiers to define the capture window.
4. AWR snapshot identifiers and association time information are available in DBA_HIST_SNAPSHOT.
5. A large capture window provides the Advisor with more data, which can result in more accurate recommendations. However, more data takes more time to process. With a very large capture window, the Advisor may run for a very long time.

Example:

```
EXEC dbms_inmemory_advisor.add_hist_statistics ('MYTASK', 177, 282);
```

ASH_SQL_COVERAGE_PCT Function

ASH_SQL_COVERAGE_PCT returns the percentage (0-100) of ASH samples that are covered by SQL statistics. If less than 50%, you may wish to start over with more SQL statistics by adding SQL tuning sets. If you have already added SQL tuning sets, you may wish to do so again using longer capture windows..

Syntax:

```
FUNCTION ash_sql_coverage_pct (
    task_name  IN VARCHAR2) RETURN NUMBER;
```

Parameters:

PARAMETER	DESCRIPTION
task_name	A unique name up to 30 characters in length for the DBMS_INMEMORY_ADVISOR task.
Return value:	Percentage (0-100) of ASH samples that are covered by SQL statistics. .

EXECUTE_TASK Procedure

EXECUTE_TASK analyzes the available data and statistics that have been added to the specified task.

Syntax:

```
PROCEDURE execute_task (
   task_name              IN VARCHAR2,
   consider_objects_like  IN VARCHAR2 := NULL);
```

Parameters:

PARAMETER	DESCRIPTION
task_name	Name of the task to analyze.
consider_objects_like	Optional list of object name patterns to consider as candidates. By default, all objects are candidates. You may optionally specify a comma separated list of object name patterns to be considered exclusively as candidates. The object name pattern is as follows: OBJECT_OWNER.OBJECT_NAME, ... Separate the owner from the name with a period (“.”). You may specify multiple object owner and name patterns separated by commas (","). You may use the percent sign ("%") as a wild card. The object owner and name patterns are case sensitive. When you specify one or more object name patterns, only objects which match at least one of the patterns will be considered.

Usage Notes:

Procedure CREATE_TASK and ADD_STATISTICS must be executed prior to executing this procedure

Examples:

```
EXEC dbms_inmemory_advisor.execute_task ('MYTASK');
EXEC dbms_inmemory_advisor.execute_task ('MyTask2', 'GEEK_SUMMARY.%,%.GEEK_%');
```

GENERATE_RECOMMENDATIONS Procedure

Generates a number of output files based upon the analysis of the specified task:

1. Recommendation report, html format which can include multiple files.
2. SQLPlus script with DDL commands to implement the recommendations.

Syntax:

```
PROCEDURE generate_recommendations (
   task_name              IN VARCHAR2,
   directory_name         IN VARCHAR2  :=  'DATA_PUMP_DIR',
   inmemory_size          IN NUMBER   := NULL,
   single_page_report     IN BOOLEAN  := FALSE);
```

Parameters:

PARAMETER	DESCRIPTION
task_name	Name of the task from which to generate recommendations.
directory_name	The name of the Oracle directory object that is to be used for placement of the recommendation files. An Oracle directory object defines a portal from the database to a directory on the database server host. The default is DATA_PUMP_DIR. If you specify NULL, the recommendations are not placed in database server files. However, the recommendations are always stored in the In-Memory Advisor's task data. See below for instructions on how to use the imadvisor_fetch_recommendations.sql script to transfer the recommendations from the In-Memory Advisor's task data to files in a client directory.
inmemory_size	The size to be used for in-memory storage in bytes. The Oracle Database In-Memory Advisor uses a default size as the largest recommended In-Memory size. The top page of the generated report contains information to help you choose the most suitable in-memory size. You can therefore first generate the report using the default in-memory size. If you then decide to use a different in-memory size, you can call GENERATE_RECOMMENDATIONS again, specifying the in-memory size you wish.
single_page_report	If set to TRUE, the report will be generated as a single html file. By default, the report will be generated as multiple html files to which the top html page connects via html links. Regardless of a single page or multiple html pages, the name of the html file to open with your browser is imadvisor_*task-name*.html, where *task-name* is the name of your Advisor task.

Usage Notes:

1. Procedures CREATE_TASK and EXECUTE_TASK must be executed prior to executing this procedure. In addition, procedure ADD_STATISTICS or ADD_HIST_STATISTICS must also be executed prior to executing this procedure.
2. You can either place the recommendation files in a database server directory using the directory_name parameter or you can fetch the recommendations into the current working directory on the client using the imadvisor_fetch_recommendations.sql script after executing GENERATE_RECOMMENDATIONS. To do the latter, you must have write privileges on the client's current working directory.
3. The recommendation report files are named in the following format:
   imadvisor_*<*task_name*>.html
   The recommendation implementation script is named:
   imadvisor_<*task_name*>.sql

Example 1:

```
-- Place the recommendations in the current working directory on the
-- client.
EXEC dbms_inmemory_advisor.generate_recommendations
  ('MYTASK', NULL, 50000000000);
@imadvisor_fetch_recommendations
Enter value for task_name: MYTASK
HOST firefox imadvisor_MYTASK.html
@imadvisor_MYTASK.sql
```

Example 2:

```
-- Place the recommendations in directory /scratch/my_oracle_dir on
-- database server host orasvr.  For this to work, /net/orasvr must
-- be NFS mounted on the client host and directory /scratch/my_oracle_dir
-- must be created on the database server host with write access to
-- the database server and read access to the client user.
CREATE DIRECTORY  my_directory  '/scratch/my_oracle_dir/';
EXEC dbms_inmemory_advisor.generate_recommendations
  ('MYTASK', 'my_directory', 50000000000);
HOST firefox /net/orasvr/scratch/my_oracle_dir/imadvisor_MYTASK.html
@/net/orasvr/scratch/my_oracle_dir/imadvisor_MYTASK.sql
```

DROP_TASK Procedure

DROP_TASK drops a DBMS_INMEMORY_ADVISOR task.

Syntax:

```
PROCEDURE drop_task (
   task_name           IN VARCHAR2,
   force               IN BOOLEAN := FALSE);
```

Parameters:

PARAMETER	DESCRIPTION
task_name	A unique name up to 30 characters in length for the DBMS_INMEMORY_ADVISOR task.
force	By default, drop_task will not drop an active task. However, if a task hangs or if the database is shut down while the task is running, the task can be left an active state. To drop a task in this state, specify FORCE=>TRUE.

Example:

```
EXEC dbms_inmemory_advisor.drop_task ('MYTASK');
```

SET_PARAMETER Procedure

Sets an Advisor parameter. The available parameters are described in Appendix 4.

Syntax:

```
PROCEDURE set_parameter (
   parameter_name      IN VARCHAR2,
   parameter_value     IN NUMBER,
   task_name           IN VARCHAR2  := NULL);
```

Parameters:

PARAMETER	DESCRIPTION
parameter_name	The name of the parameter for which the setting is to be changed.
parameter_value	The value for the new parameter setting.
task_name	Name of the task for which the parameter setting is to be applied. If you omit task_name or specify it as NULL, the parameter setting will be applied to all Oracle Database In-Memory Advisor tasks that have not had any task-specific settings for the specified parameter name.

Usage Notes:

1. Procedure SET_PARAMETER must be executed prior to executing procedure ADD_STATISTICS or ADD_HIST_STATISTICS in order for the parameter setting to have effect.

Example:

```
DBMS_INMEMORY_ADVISOR.SET_PARAMETER
  ('INMEMORY_READ_PERF_FACTOR', 15.0); -- set for all tasks
DBMS_INMEMORY_ADVISOR.SET_PARAMETER
  ('INMEMORY_CPU_PERF_FACTOR', 3.0, 'my_task'); -- set for 'my_task' only
```

GET_PARAMETER Function

Returns an Advisor parameter value. The available parameters are described in Appendix 4.

Syntax:

```
FUNCTION get_parameter (
   parameter_name        IN VARCHAR2,
   task_name             IN VARCHAR2  := NULL);
```

Parameters:

PARAMETER	DESCRIPTION
parameter_name	The name of the parameter for which the current setting is to be obtained.
task_name	Name of the task from which the parameter setting is to be obtained. If you omit task_name or specify it as NULL, the parameter setting will be obtained as the default value for all Oracle Database In-Memory Advisor tasks that have not had any task-specific settings for the specified parameter name.

Example:

```
irpf := DBMS_INMEMORY_ADVISOR.GET_PARAMETER
   ('INMEMORY_READ_PERF_FACTOR'); -- obtain default value for all tasks
icpf := DBMS_INMEMORY_ADVISOR.SET_PARAMETER
   ('INMEMORY_CPU_PERF_FACTOR', 'my_task'); -- obtain value from 'my_task'
```

RESET_PARAMETERS Procedure

Resets all Oracle Database In-Memory Advisor parameters to their original default settings.

Syntax:

```
PROCEDURE reset_parameters (
    task_name              IN VARCHAR2  := NULL);
```

Parameters:

PARAMETER	DESCRIPTION
task_name	Name of the task for which the parameter settings are to be reset. If you omit task_name or specify it as NULL, the default parameter settings will be reset for all Oracle Database In-Memory Advisor tasks that have not had task-specific parameter settings.

Usage Notes:

1. Procedure RESET_PARAMETERS must be executed prior to executing procedure ADD_STATISTICS or ADD_HIST_STATISTICS in order for the parameter resetting to have effect.

Example:

```
DBMS_INMEMORY_ADVISOR.RESET_PARAMETERS
  ('my_task'); -- reset for ‘my_task’
DBMS_INMEMORY_ADVISOR.RESET_PARAMETERS(); -- reset for all tasks
```

Appendix 1: Sample PL/SQL Interface Script to Run the Advisor

This sample script can be used to run the Advisor's PL/SQL interface and produce the recommendations for a live workload running on the same database.

```
SET SERVEROUTPUT ON;
DECLARE
  sql_coverage_pct NUMBER;
BEGIN
  BEGIN
    dbms_inmemory_advisor.drop_task ('my_task', force=>TRUE);
  EXCEPTION
    WHEN OTHERS THEN NULL;
  END;
  dbms_inmemory_advisor.create_task ('my_task');
  dbms_inmemory_advisor.add_statistics ('my_task');
  sql_coverage_pct := dbms_inmemory_advisor.ash_sql_coverage_pct
('my_task');
  dbms_output.put_line ('sql_coverage_pct='||sql_coverage_pct);
  dbms_inmemory_advisor.execute_task ('my_task');


  dbms_inmemory_advisor.generate_recommendations ('my_task');
END;
/

DEFINE task_name='my_task';
@imadvisor_fetch_recommendations.sql
HOST firefox imadvisor_my_task.html
@imadvisor_my_task.sql
PROMPT All done!
```

Appendix 2: Advisor Views

DBA_IMA_AWR_AUGMENTS

View `dba_ima_awr_augments` lists the Automatic Workload Repository (AWR) workload augments that have been imported with `imadvisor_awr_augment_import.sql`:

Column Name	Description	Data Type
augment_schema	Name of the schema on the local database where the AWR augment data is stored.	VARCHAR2(128)
augment_date	The date and time at which the AWR augment was captured on the source database.	DATE
augment_enabled	'Y' indicates the AWR augment is enabled for use by the In-Memory Advisor on the local database. 'N' indicates it is not enabled. If you load multiple augments for with same DBID, you must disable all but one of the AWR augments for the DBID you intend to use with the In-Memory Adivisor. To disable an augment: UPDATE <augment_schema> . imadvisor_awr_augment SET augment_enabled = 'N'; COMMIT; where <augment_schema> is the value found in the augment_schema column in this view. To re-enable an augment: UPDATE <augment_schema> .	

	imadvisor_awr_augment SET augment_enabled = 'Y'; COMMIT;	
imadvisor_awr _augment_version	The version of the In-Memory Advisor that was used to capture the augment on the source database.	VARCHAR2(30)
imadvisor_awr_augment_buildnum	The build number of the In-Memory Advisor that was used to capture the augment on the source database.	NUMBER
awr_loaded	'Y' indicates that the AWR workload has been loaded. 'N' indicates that the AWR workload has not been load and therefore this augment cannot be used with the In-Memory Advisor. To load the AWR workload, use $ORACLE_HOME/rdbms/admin/awrload.sql.	VARCHAR2(1)
local_db_augment	'Y' indicates that this augment was captured from this local database. That is, it has the same DBID as the local database. 'N' indicates that this augment was captured from a different database than the current, local database. A local augment is not required and the In-Memory Advisor does not use local augments.	VARCHAR2(1)
dbid	Database identifier from the database where the augment was captured. (v$database . dbid)	NUMBER
db_name	Name of the database where the augment was captured (v$database . name).	VARCHAR2(30)
db_created	Creation date of the database where the augment was captured (v$database . created).	DATE
instance_number	Instance number of the database instance where the augment was captured (v$instance . instance_number).	NUMBER
instance_name	Name of the database instance where the augment was captured (v$instance . instance_name).	VARCHAR2(30)
instance_version	Version of the database instance where the augment was captured (v$instance.version).	VARCHAR2(30)
instance	Version banner of the database instance where the augment was captured (v$version.banner WHERE	VARCHAR2(80)

_version_banner	banner LIKE 'Oracle Database %').	
compatible _version	Compatible version setting on the database instance where the augment was captured (v$parameter.value WHERE name='compatible'; set to instance_version when not found).	VARCHAR2(30)
service_name	Service name of the database instance where the augment was captured (v$listener_network.value WHERE type='SERVICE NAME').	VARCHAR2(1024)
sga_max_size	Maximum SGA size on the database instance where the augment was captured (v$parameter.value WHERE name = 'sga_max_size'; set to NULL when not found).	NUMBER
inmemory_size	In-Memory size setting for the database instance where the augment was captured (v$parameter.value WHERE name = 'inmemory_size'; set to 0 when not found).	NUMBER
inmemory _unused_space	Unused in-Memory space on the database instance where the augment was captured (inmemory_size - v$inmemory_area . SUM(used_bytes); set to 0 when not found).	NUMBER
ash_sample _interval	Active Session History sample interval on the database instance where the augment was captured (v$parameter.value WHERE name = '_ash_sample_interval'; set to 1000 when not found).	NUMBER
ash_diskfilter _ratio	Active Session History disk filter ratio (v$parameter.value WHERE name = '_ash_disk_filter_ratio'; set to 10 when not found).	NUMBER
db_supports _inmemory	'Y' indicates that the database instance where the augment was captured supports the In-Memory feature. 'N' indicates that the In-Memory feature was not supported. Note: This column does **not** indicate whether or not the license for the In-Memory feature has been purchased.	VARCHAR2(1)

	('Y' if any views named LIKE 'V%$INMEMORY%' exist, 'N' otherwise.)	
cdb_root	'Y' indicates that the augment was captured from a CDB root. 'N' indicates that the augment was captured from a non-CDB.	VARCHAR2(1)

DBA_IMA_BENEFIT_COST

View `dba_ima_benefit_cost` shows the estimated performances improvements for a list of In-Memory sizes:

Column Name	Description	Data Type
owner	The owner of the In-Memory Advisor task to which the listed performance estimates apply.	VARCHAR2(30)
task_name	The name of the task to which the listed performance estimates apply.	VARCHAR2(30)
inmemory_size_bytes	Number of bytes for the In-Memory size.	NUMBER
inmemory_size	The In-Memory size shown with a size unit (example: 17.00GB = 18,253,611,008 bytes).	VARCHAR2(10)
is_user_specified_size	'Y' indicates this is the In-Memory size specified with the most recent call to GENERATE_RECOMMENDATIONS. 'N' indicates this In-Memory size is not user specified.	VARCHAR2(1)
is_incremental_step_size	'Y' indicates this In-Memory size is one of the 5% incremental steps of the largest recommended In-Memory size. 'N' indicates is not one of the 5% incremental steps. Both the is_user_specified_size and the is_incremental_step_size columns can be 'Y'. But only one of these columns can be 'N'.	VARCHAR2(1)
pct_max_recommended_imsize	The percentage of the largest recommended In-Memory size.	NUMBER
pct_sga_max_size	The percentage of the maximum SGA size parameter	NUMBER

	setting.	
est_reduced_analytics_secs	The number of reduced processing seconds estimated with the In-Memory size.	NUMBER
est_perf_improvement_factor	The performance improvement factor estimated with the In-Memory size.	VARCHAR2(1000)

DBA_IMA_RECOMMENDATIONS

Column Name	Description	Data Type
task_id	The task identifier.	NUMBER
task_owner	The task owner who created the task.	VARCHAR2(30)
task_name	The task name as specified by the task owner.	VARCHAR2(30)
object_id	The object identifier for the In-Memory placement candidate.	NUMBER
table_owner	The owner of the table which is or which contains the In-Memory placement candidate.	VARCHAR2(128)
table_name	The name of the table which is or which contains the In-Memory placement candidate.	VARCHAR2(128)
table_partitioning_level	The partitioning level of the table: NONE: An unpartitioned table. STANDARD: A table comprised of standard partitions. COMPOSITE: A table comprised of partitions comprised of subpartitions.	VARCHAR2(10)
partition_name	The name of the partition which is or which contains the In-Memory placement candidate. NULL if the candidate is a full table.	VARCHAR2(128)
subpartition_name	The name of the subpartition which is the In-Memory placement candidate. NULL if the candidate is a full partition or a full table.	VARCHAR2(128)
partition_type	The type of partitioning at the partition level. NONE with unpartitioned tables	VARCHAR2(10)
subpartition_type	The type of partitioning at the subpartition level. NONE with unpartitioned and standard partitioned	VARCHAR2(10)

	tables.	
non_im_dbtime_secs	The number of analytic processing seconds applied to this candidate.	NUMBER
uncompressed_bytes	The statistics based size of this candidate: NUM_ROWS * AVG_ROW_LEN.	NUMBER
compression_type	The candidate In-Memory compression type: 8192: COMP_INMEMORY_NOCOMPRESS 16384: COMP_INMEMORY_DML 32768: COMP_INMEMORY_QUERY_LOW 65536: COMP_INMEMORY_QUERY_HIGH 131072: COMP_INMEMORY_CAPACITY_LOW 262144: COMP_INMEMORY_CAPACITY_HIGH Each candidate object is considered with each In-Memory compression type.	NUMBER
est_compression_factor	The estimated compression factor for the candidate compression type. These estimated compression factor can be changed with the following parameters: COMP_FACTOR_NOCOMPRESS, default=1 COMP_FACTOR_DML, default=3 COMP_FACTOR_QUERY_LOW, default=4 COMP_FACTOR_QUERY_HIGH, default=6 COMP_FACTOR_CAPACITY_LOW, default=8 COMP_FACTOR_CAPACITY_HIGH, default=10	NUMBER
est_compressed_bytes	The estimated In-Memory size for the candidate object with the candidate compression type.	NUMBER
est_reduced_dbtime_secs	The estimated number of reduced analytic processing seconds with In-Memory placement of this candidate object with this candidate compression type.	NUMBER
est_reduced_dbtime_pct	The estimated percentage of total analytic processing time reduction with In-Memory placement of this candidate object with this candidate compression type.	NUMBER
recommended_with_im_size_gte	This candidate is recommended for In-Memory placement only with an In-Memory size greater than or equal the In-Memory size in this column. When the status column equals 'rejected', this column IS NULL.	NUMBER

recommended_with_im_size_lt	When this column IS NOT NULL, this candidate is recommended for In-Memory placement only with an In-Memory size less than the In-Memory size in this column. When this column IS NULL and the status column equals 'accepted', there is no upper limit to the In-Memory size for this recommendation.	NUMBER
rationale	The rationale for the recommendation (where the status column equals 'accepted') or rejection (where the status column equals 'rejected') of this candidate object with this candidate compression type.	VARCHAR2(4000)
rank	The rank of this candidate object with this candidate compression type. Where rank equals 0, the candidate object was rejected due to object-specific attributes and statistics. Where rank is greater than 0 and status equals 'accepted', the lower the rank, the better the estimated-performance-improvement-benefit / estimated-In-Memory-size-cost ratio.	NUMBER
status	The status of this candidate object with this candidate compression type: 'accepted' or 'rejected'.	VARCHAR2(8)

DBA_IMA_RECOMMENDATION_FILES

Column Name	Description	Data Type
owner	The task owner who created the task.	VARCHAR2(30)
task_name	The task name as specified by the task owner.	VARCHAR2(30)
file_name	The name of the html format recommendation file.	VARCHAR2(100)
file_contents	The content of the html format recommendation file.	CLOB

DBA_IMA_RECOMMENDATION_LINES

Column Name	Description	Data Type
owner	The task owner who created the task.	VARCHAR2(30)
task_name	The task name as specified by the task owner.	VARCHAR2(30)
file_name	The name of the html format recommendation file.	VARCHAR2(100)

file_line_number	The line number of the html format recommendation line.	NUMBER
file_line	The content of the html format recommendation file line.	VARCHAR2(4000)

DBA_IMA_TASK_INFORMATION

Column Name	Description	Data Type
owner	The task owner who created the task.	VARCHAR2(30)
task_name	The task name as specified by the task owner.	VARCHAR2(30)
description	The description of this task as specified by the task owner.	VARCHAR2(4000)
dbid	The database identifier for which this task optimized.	NUMBER
instance_number	The RAC instance number for which this task optimized.	NUMBER
pdb_name	The Pluggable Database name for which this task optimized. NULL if not a Pluggable Database.	VARCHAR2(128)
state	The current state of this task: nascent: Task is either in the process of being created or has been created and not yet run. active: Task is current running a public interface. inactive: Task has previously run a public interface but is currently inactive. terminal: Task is in the process of being dropped. crashed: Task crashed unexpectedly during the run of a public interface.	VARCHAR2(30)
last_action	The last public interface that was run with this task: create_task add_sqlset add_statistics add_hist_statistics ash_sql_coverage_pct execute_task generate_recommendations	VARCHAR2(30)

stats_added	This column indicates whether statistics have been added to this task. 'Y' indicates they have. 'N' indicates they have not.	CHAR(1)
sqlset_added	This column indicates whether any SQLSETs have been added to this task. 'Y' indicates at least one. 'N' indicates none.	CHAR(1)
executed	This column indicates whether the task has been executed. 'Y' indicates it has. 'N' indicates it has not.	CHAR(1)
report_generated	This column indicates whether the task has generated any recommendation reports. 'Y' indicates at least one. 'N' indicates none.	CHAR(1)
date_created	This column shows the date and time this task was created.	DATE
date_modified	This column shows the date and time of the most recent public interface run with this task.	DATE
statistics_window_start	This column shows the statistics capture window start timestamp for this task.	TIMESTAMP(3)
statistics_window_end	This column shows the statistics capture window end timestamp for this task.	TIMESTAMP(3)
total_dbtime_secs	This column shows the total database activity seconds during the statistics capture window.	NUMBER
total_analytics_secs	This column shows the total analytic database activity during the statistics capture window.	NUMBER

USER_IMA_BENEFIT_COST

This view is the same as DBA_IMA_BENEFIT_COST except it only shows tasks owned by the current user and it does not include the OWNER column.

USER_IMA_RECOMMENDATIONS

This view is the same as DBA_IMA_RECOMMENDATIONS except it only shows tasks owned by the current user and it does not include the TASK_OWNER column.

USER_IMA_RECOMMENDATION_FILES

This view is the same as DBA_IMA_RECOMMENDATION_FILES except it only shows tasks owned by the current user and it does not include the OWNER column.

USER_IMA_RECOMMENDATION_LINES

This view is the same as DBA_IMA_RECOMMENDATION_LINES except it only shows tasks owned by the current user and it does not include the OWNER column.

USER_IMA_TASK_INFORMATION

This view is the same as DBA_IMA_TASK_INFORMATION except it only shows tasks owned by the current user and it does not include the OWNER column.

Appendix 3: AWR Augment Tables

Database 11.2

dba_objects (owner, object_name, subobject_name, object_type, object_id, data_object_id, temporary)

dba_sqlset (owner, name, id, created, last_modified, statement_count)

dba_sqlset_plans (sqlset_id, sql_id, force_matching_signature, plan_hash_value, id, cardinality, operation, options, object_instance, object_owner, object_name, object_type, optimizer)

dba_tables (owner, table_name, blocks, avg_row_len, num_rows, empty_blocks, compression, compress_for)

dba_part_tables (owner, table_name, partitioning_type, subpartitioning_type, partition_count, def_subpartition_count, def_compression, def_compress_for)

dba_tab_partitions (table_owner, table_name, partition_name, avg_row_len, num_rows, blocks, composite, subpartition_count, empty_blocks, compression, compress_for)

dba_tab_subpartitions (table_owner, table_name, partition_name, subpartition_name, avg_row_len, num_rows, blocks, empty_blocks, compression, compress_for)

dba_users (user_id, username)

Database 12.1

cdb_objects (owner, object_name, subobject_name, object_type, object_id, data_object_id, temporary, con_id)

cdb_sqlset (owner, name, id, created, last_modified, statement_count, con_dbid,con_id)

cdb_sqlset_plans (sqlset_id, sql_id, force_matching_signature, plan_hash_value, id, cardinality, operation, options, object_instance, object_owner, object_name, object_type, optimizer, con_dbid, con_id)

cdb_tables (owner, table_name, blocks, avg_row_len, num_rows, empty_blocks, compression, compress_for, con_id)

cdb_part_tables (owner, table_name, partitioning_type, subpartitioning_type, partition_count, def_subpartition_count, def_compression, def_compress_for, con_id)

cdb_tab_partitions (table_owner, table_name, partition_name, avg_row_len, num_rows, blocks, composite, subpartition_count, empty_blocks, compression, compress_for, con_id)

cdb_tab_subpartitions (table_owner, table_name, partition_name, subpartition_name, avg_row_len, num_rows, blocks, empty_blocks, compression, compress_for, con_id

dba_users (user_id, username, con_id)

```
 cdb|dba}_sqlset (
       '||clone_column ('cdb_sqlset', 'con_id', 'NUMBER')||',
```

'||clone_column ('cdb_sqlset', 'con_dbid', 'NUMBER')||',
owner, name, id, created, last_modified, statement_count)

{cdb|dba}_sqlset_plans (
'||clone_column ('cdb_sqlset_plans', 'con_id', 'NUMBER')||',
'||clone_column ('cdb_sqlset_plans', 'con_dbid', 'NUMBER')||',
sqlset_name, sqlset_owner, sqlset_id, sql_id, force_matching_signature, plan_hash_value,
id, cardinality, operation, options, object_instance, object_owner, object_name, object_type,
optimizer)

{cdb|dba}_sqlset_statements (
'||clone_column ('cdb_sqlset_statements', 'con_id', 'NUMBER')||',
'||clone_column ('cdb_sqlset_statements', 'con_dbid', 'NUMBER')||',
sqlset_name, sqlset_owner, sqlset_id, sql_id, force_matching_signature, sql_text,
parsing_schema_name, parsing_schema_id, plan_hash_value, module, action, elapsed_time, cpu_time,
buffer_gets, disk_reads, direct_writes, rows_processed, fetches, executions, end_of_fetch_count,
optimizer_cost, command_type, first_load_time, stat_period, active_stat_period, plan_timestamp)

Appendix 4: Advisor Parameters

The Advisor employs a number of heuristic parameter settings to estimate in-memory performance, a threshold for reasonable recommendations and determine in-memory eligibility based on minimum size. Refer to the GET_PARAMETER, SET_PARAMETER PL/SQL packages for the interface that allows you to obtain and change parameter settings. The following table lists the available parameter settings:

Parameter Name	**Description**	**Default Value**
INMEMORY_READ_PERF_FACTOR	Estimate that database time that had been spent on read activity on a given object will be N x faster with in-memory placement. A value of 10.0 implies performance with what had been read activity will be as much as 10 x faster.	10.0
INMEMORY_WRITE_PERF_FACTOR	Estimate that database time spent on write activity on a given object will be N x slower with in-memory placement, where N<1. A value of 0.9 implies performance with write activity will be at least 10% slower.	0.9
INMEMORY_CPU_PERF_FACTOR	Estimate that database time that had been spent on CPU activity on a given object will be N x faster with in-memory placement. A value of 2.0 implies performance with CPU activity will be as much as 2 x faster.	2.0
MIN_OVERALL_BENEFIT_FACTOR	Use a threshold to determine whether recommendations with the largest recommendation size can produce any noticeable difference in performance. A value of 1.05 implies that if the estimated, total performance improvement (including both analytics and non-analytics processing) is no more than 5%, it is unlikely the performance improvement will be noticeable. In this case, the no recommendations are produced.	1.05
MIN_INMEMORY_OBJECT_SIZE	Use a threshold for the on-disk size of objects to determine eligibility of placement in memory.	65536

	A value of 65536 implies that objects smaller than 64KB (NUM_ROWS * AVG_ROW_LEN) on disk are ineligible for in-memory placement. (Note: on-disk compression is not included in the on-disk size computation.)	
FAVOR_HIGH_COMPRESSION	By default, the Advisor optimizes the Oracle Database In-Memory configuration for the highest reduction in analytics database time. When you specify an in-memory size that is relatively small in comparison to the largest recommended in-memory size, the optimal in-memory configuration with the greatest reduction in analytics time will often consist of a small number of high-benefit objects with low compression. This is because low compression can provide greater performance benefit. While such an in-memory configuration is optimal in terms of analytics time reduction, due to the small number of objects placed in memory, the performance benefits may be limited to certain components in your database application. Therefore, you may desire an in-memory configuration that uses higher compression and places more objects in memory. Such a configuration may have somewhat less of a reduction in analytics time. But its benefits may also be applied to more of your database application components. To obtain an in-memory configuration using higher compression types and with placement of more objects in-memory with a small in-memory size, set the FAVOR_HIGH_COMPRESSION parameter to 1. By default, FAVOR_HIGH_COMPRESSION is set to 0, which favors higher reduction in analytics time rather than higher compression.	0
LOB_BENEFIT_REDUCTION	The Oracle Database In-Memory feature does not support placement of LOB data type columns in-memory when the LOB value is stored outside the table row. Therefore, tables that have columns of LOB data types (CLOB, BLOB, BFILE) with their values stored outside the table row can have lower performance benefit with queries that reference the LOB columns. The Advisor computes a heuristic estimate for the	1

	benefit reduction. The higher the number of LOB columns in tables with storage outside the row, the higher the degree of benefit reduction. You can adjust this estimate with this parameter. The default value is 1. With this value, the benefit reduces to zero when all columns in the table are LOB columns with storage outside the row. With a value of 0, no benefit reduction will be applied. With a value between 0 and 1, a smaller degree of benefit reduction will be applied in comparison to the default value of 1. For example, with a value of 0.5, the benefit reduces to half its original value when all columns in the table are LOB columns with storage outside the row. With values greater than 1, a higher degree of benefit reduction will be applied in comparison to the default value of 1. For example, with a value of 2, the benefit reduces to zero when half the columns in the table are LOB columns with storage outside the row. Using values less than 1 can be useful when the LOB columns are not frequently accessed. Using values higher than 1 can be useful when the LOB columns are heavily accessed.	
SUPPRESS_LIVE_STATISTICS	ADD_STATISTICS combines both live and Automatic Workload Repository (AWR) data when running against a live workload on the local database. This provides the Advisor with more data which can result in more accurate recommendations. However, more data requires more processing time. If The Advisor takes too long to run with a live workload, you can reduce the data the Advisor processes by setting SUPPRESS_LIVE_STATISTICS to 1.	0

Appendix 5: Required Privileges

Interface	**Required Privileges**
PL/SQL interface: DBMS_INMEMORY_ADVISOR	ADVISOR
Interactive script: imadvisor_recommendations.sql	ADVISOR CREATE SESSION DBA
Interactive script: imadvisor_ash_sql_coverage.sql	ADVISOR CREATE SESSION
Interactive script: imadvisor_fetch_recommendations.sql	ADVISOR CREATE SESSION DBA
Interactive script: imadvisor_rationale.sql	ADVISOR CREATE SESSION DBA
Interactive script: imadvisor_awr_augment_export.sql	ADVISOR CREATE ANY TABLE CREATE PROCEDURE CREATE SESSION DBA SELECT ANY DICTIONARY UNLIMITED TABLESPACE (or similar)
Interactive script: imadvisor_awr_augment_import.sql	ADVISOR CREATE ANY TABLE CREATE SESSION DBA SELECT ANY DICTIONARY UNLIMITED TABLESPACE (or similar)

Oracle Database In-Memory Advisor
January, 2017

Author: Jack Raitto, Kurt Engeleiter
Contributing Authors:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com