Heat Map, Information Lifecycle Management (ILM) and Automatic Data Optimization (ADO) in Oracle Database 12c Release 2 (12.2)

In Oracle Database 12.1 the Heat Map and Automatic Data Optimization (ADO) functionality was only available when using the non-CDB architecture. In Oracle Database 12.2 this functionality is now supported in the multitenant architecture. This article gives an overview of Heat Map, Information Lifecycle Management (ILM) and Automatic Data Optimization (ADO) in Oracle Database 12c Release 2 (12.2). The examples are based around the multitenant architecture, but the information applies equally to the non-CDB architecture in Oracle Database 12.1 and 12.2.

* [Heat Map](https://oracle-base.com/articles/12c/heat-map-ilm-ado-12cr2" \l "heat-map)
* [Automatic Data Optimization (ADO)](https://oracle-base.com/articles/12c/heat-map-ilm-ado-12cr2#ado)
* [ILM ADO Parameters](https://oracle-base.com/articles/12c/heat-map-ilm-ado-12cr2#ilm-ado-parameters)

Heat Map

The heat map functionality allows you to track data access at the segment level and data modification at the row and segment level, so you can identify the busy segments of the system. This functionality is controlled by the HEAT\_MAP parameter, that can be set at the system or session level.

Display the current setting of the HEAT\_MAP parameter at the PDB level.

CONN / AS SYSDBA

ALTER SESSION SET CONTAINER = pdb1;

SHOW PARAMETER heat\_map;

NAME TYPE VALUE

------------------------------------ ----------- ------------------------------

heat\_map string OFF

SQL>

Enable the heat map for the PDB.

ALTER SYSTEM SET heat\_map = ON;

SHOW PARAMETER heat\_map;

NAME TYPE VALUE

------------------------------------ ----------- ------------------------------

heat\_map string ON

SQL>

Notice that the heat map is still disabled at the CBD level.

CONN / AS SYSDBA

SHOW PARAMETER heat\_map;

NAME TYPE VALUE

------------------------------------ ----------- ------------------------------

heat\_map string OFF

SQL>

Once the heat map functionality is enable the database will track segment changes for all segments except for those in the SYSTEM and SYSAUX tablespaces. You can display the heat map information using the following views and pipelined table functions.

* V$HEAT\_MAP\_SEGMENT
* {USER|ALL|DBA}\_HEAT\_MAP\_SEG\_HISTOGRAM
* {USER|ALL|DBA}\_HEAT\_MAP\_SEGMENT
* {USER|ALL|DBA}\_HEATMAP\_TOP\_OBJECTS
* {USER|ALL|DBA}\_HEATMAP\_TOP\_TABLESPACES
* DBMS\_HEAT\_MAP.BLOCK\_HEAT\_MAP
* DBMS\_HEAT\_MAP.EXTENT\_HEAT\_MAP
* DBMS\_HEAT\_MAP.OBJECT\_HEAT\_MAP
* DBMS\_HEAT\_MAP.SEGMENT\_HEAT\_MAP
* DBMS\_HEAT\_MAP.TABLESPACE\_HEAT\_MAP

Do some work that will be tracked.

CONN / AS SYSDBA

ALTER SESSION SET CONTAINER = pdb1;

CREATE USER test IDENTIFIED BY test QUOTA UNLIMITED ON users;

GRANT CREATE SESSION, CREATE TABLE TO test;

CONN test/test@pdb1

CREATE TABLE t1 (

id NUMBER,

description VARCHAR2(50),

CONSTRAINT t1\_pk PRIMARY KEY (id)

);

INSERT INTO t1

SELECT level,

'Description for ' || level

FROM dual

CONNECT BY level <= 10;

COMMIT;

SELECT \*

FROM t1;

SELECT \*

FROM t1

WHERE id = 1;

We can now run some queries to see the tracked information.

CONN / AS SYSDBA

ALTER SESSION SET CONTAINER = pdb1;

COLUMN object\_name FORMAT A20

SELECT track\_time,

object\_name,

n\_segment\_write,

n\_full\_scan,

n\_lookup\_scan

FROM v$heat\_map\_segment

ORDER BY 1, 2;

TRACK\_TIME OBJECT\_NAME N\_SEGMENT\_WRITE N\_FULL\_SCAN N\_LOOKUP\_SCAN

-------------------- -------------------- --------------- ----------- -------------

25-FEB-2017 18:25:31 T1 1 2 1

25-FEB-2017 18:25:31 T1\_PK 1 0 1

SQL>

COLUMN owner FORMAT A20

COLUMN object\_name FORMAT A20

SELECT track\_time,

owner,

object\_name,

segment\_write,

full\_scan,

lookup\_scan

FROM dba\_heat\_map\_seg\_histogram

ORDER BY 1, 2, 3;

TRACK\_TIME OWNER OBJECT\_NAME SEG FUL LOO

-------------------- -------------------- -------------------- --- --- ---

25-FEB-2017 18:26:15 TEST T1 YES YES YES

25-FEB-2017 18:26:15 TEST T1\_PK YES NO YES

SQL>

SET LINESIZE 100

COLUMN owner FORMAT A10

COLUMN segment\_name FORMAT A20

COLUMN tablespace\_name FORMAT A20

SELECT owner,

segment\_name,

segment\_type,

tablespace\_name,

segment\_size

FROM TABLE(DBMS\_HEAT\_MAP.object\_heat\_map('TEST','T1'));

OWNER SEGMENT\_NAME SEGMENT\_TYPE TABLESPACE\_NAME SEGMENT\_SIZE

---------- -------------------- -------------------- -------------------- ------------

TEST T1 TABLE USERS 65536

TEST T1\_PK INDEX USERS 65536

SQL>

The heat map information can be really useful for identifying the busy and quiet segments in your database.

Automatic Data Optimization (ADO)

Enabling the heat map functionality also enables Automatic Data Optimimzation (ADO), part of Information Lifecycle Management (ILM). This allows the database to control compression and storage tiering of segments based on usage patterns. Although it can be used with regular table segments, it only really makes sense with partitioning, as it is unlikely you will have whole tables that are not accessed for long periods of time, whereas it can be very likely to have partitions for low-use data.

Create some tablespaces to represent the storage tiers. The following syntax uses Oracle Managed Files (OMF), hence no datafile names are needed.

CONN / AS SYSDBA

ALTER SESSION SET CONTAINER = pdb1;

CREATE TABLESPACE fast\_storage\_ts DATAFILE SIZE 1M AUTOEXTEND ON NEXT 1M;

CREATE TABLESPACE medium\_storage\_ts DATAFILE SIZE 1M AUTOEXTEND ON NEXT 1M;

CREATE TABLESPACE slow\_storage\_ts DATAFILE SIZE 1M AUTOEXTEND ON NEXT 1M;

A table can be created with an ADO ILM policy. The following example creates a partitioned invoices table. It manually allocates partitions to different storage tiers, and includes a tier policy on a partition basis to migrate unused segments to tablespaces on slower storage. There is a compression policy at the table-level, that is inherited by all partitions.

CONN test/test@pdb1

DROP TABLE invoices PURGE;

CREATE TABLE invoices (

invoice\_no NUMBER NOT NULL,

invoice\_date DATE NOT NULL,

comments VARCHAR2(500)

)

PARTITION BY RANGE (invoice\_date)

(

PARTITION invoices\_2016\_q1 VALUES LESS THAN (TO\_DATE('01/04/2016', 'DD/MM/YYYY')) TABLESPACE slow\_storage\_ts,

PARTITION invoices\_2016\_q2 VALUES LESS THAN (TO\_DATE('01/07/2016', 'DD/MM/YYYY')) TABLESPACE slow\_storage\_ts,

PARTITION invoices\_2016\_q3 VALUES LESS THAN (TO\_DATE('01/09/2016', 'DD/MM/YYYY')) TABLESPACE medium\_storage\_ts

**ILM ADD POLICY TIER TO slow\_storage\_ts READ ONLY SEGMENT AFTER 6 MONTHS OF NO ACCESS**,

PARTITION invoices\_2016\_q4 VALUES LESS THAN (TO\_DATE('01/01/2017', 'DD/MM/YYYY')) TABLESPACE medium\_storage\_ts

**ILM ADD POLICY TIER TO slow\_storage\_ts READ ONLY SEGMENT AFTER 6 MONTHS OF NO ACCESS**,

PARTITION invoices\_2017\_q1 VALUES LESS THAN (TO\_DATE('01/04/2017', 'DD/MM/YYYY')) TABLESPACE fast\_storage\_ts

**ILM ADD POLICY TIER TO medium\_storage\_ts READ ONLY SEGMENT AFTER 3 MONTHS OF NO ACCESS**,

PARTITION invoices\_2017\_q2 VALUES LESS THAN (TO\_DATE('01/07/2017', 'DD/MM/YYYY')) TABLESPACE fast\_storage\_ts

**ILM ADD POLICY TIER TO medium\_storage\_ts READ ONLY SEGMENT AFTER 3 MONTHS OF NO ACCESS**

)

**ILM ADD POLICY ROW STORE COMPRESS BASIC SEGMENT AFTER 3 MONTHS OF NO ACCESS**;

We can see the policies have been applied using the USER\_ILMOBJECTS view.

SET LINESIZE 200

COLUMN policy\_name FORMAT A20

COLUMN object\_owner FORMAT A15

COLUMN object\_name FORMAT A15

SELECT policy\_name,

object\_owner,

object\_name,

object\_type,

inherited\_from,

enabled,

deleted

FROM user\_ilmobjects

ORDER BY 1;

POLICY\_NAME OBJECT\_OWNER OBJECT\_NAME OBJECT\_TYPE INHERITED\_FROM ENA DEL

-------------------- --------------- --------------- ------------------ -------------------- --- ---

P13 SYS INVOICES TABLE POLICY NOT INHERITED YES NO

P13 SYS INVOICES TABLE PARTITION TABLE YES NO

P13 SYS INVOICES TABLE PARTITION TABLE YES NO

P13 SYS INVOICES TABLE PARTITION TABLE YES NO

P13 SYS INVOICES TABLE PARTITION TABLE YES NO

P13 SYS INVOICES TABLE PARTITION TABLE YES NO

P13 SYS INVOICES TABLE PARTITION TABLE YES NO

P14 SYS INVOICES TABLE PARTITION POLICY NOT INHERITED YES NO

P15 SYS INVOICES TABLE PARTITION POLICY NOT INHERITED YES NO

P16 SYS INVOICES TABLE PARTITION POLICY NOT INHERITED YES NO

P17 SYS INVOICES TABLE PARTITION POLICY NOT INHERITED YES NO

SQL>

We can also add policies to an existing table. The following example repeats what we saw earlier by creating the table, then aplying the ADO ILM policies.

CONN test/test@pdb1

DROP TABLE invoices PURGE;

CREATE TABLE invoices (

invoice\_no NUMBER NOT NULL,

invoice\_date DATE NOT NULL,

comments VARCHAR2(500)

)

PARTITION BY RANGE (invoice\_date)

(

PARTITION invoices\_2016\_q1 VALUES LESS THAN (TO\_DATE('01/04/2016', 'DD/MM/YYYY')) TABLESPACE slow\_storage\_ts,

PARTITION invoices\_2016\_q2 VALUES LESS THAN (TO\_DATE('01/07/2016', 'DD/MM/YYYY')) TABLESPACE slow\_storage\_ts,

PARTITION invoices\_2016\_q3 VALUES LESS THAN (TO\_DATE('01/09/2016', 'DD/MM/YYYY')) TABLESPACE medium\_storage\_ts,

PARTITION invoices\_2016\_q4 VALUES LESS THAN (TO\_DATE('01/01/2017', 'DD/MM/YYYY')) TABLESPACE medium\_storage\_ts,

PARTITION invoices\_2017\_q1 VALUES LESS THAN (TO\_DATE('01/04/2017', 'DD/MM/YYYY')) TABLESPACE fast\_storage\_ts,

PARTITION invoices\_2017\_q2 VALUES LESS THAN (TO\_DATE('01/07/2017', 'DD/MM/YYYY')) TABLESPACE fast\_storage\_ts

);

ALTER TABLE invoices MODIFY PARTITION invoices\_2016\_q3

ILM ADD POLICY TIER TO slow\_storage\_ts READ ONLY SEGMENT AFTER 6 MONTHS OF NO ACCESS;

ALTER TABLE invoices MODIFY PARTITION invoices\_2016\_q4

ILM ADD POLICY TIER TO slow\_storage\_ts READ ONLY SEGMENT AFTER 6 MONTHS OF NO ACCESS;

ALTER TABLE invoices MODIFY PARTITION invoices\_2017\_q1

ILM ADD POLICY TIER TO medium\_storage\_ts READ ONLY SEGMENT AFTER 3 MONTHS OF NO ACCESS;

ALTER TABLE invoices MODIFY PARTITION invoices\_2017\_q2

ILM ADD POLICY TIER TO medium\_storage\_ts READ ONLY SEGMENT AFTER 3 MONTHS OF NO ACCESS;

ALTER TABLE invoices

ILM ADD POLICY ROW STORE COMPRESS BASIC SEGMENT AFTER 3 MONTHS OF NO ACCESS;

We can disable, delete or modify policies using the following commands.

-- Table-level.

ALTER TABLE <table-name> ILM DISABLE POLICY <policy-name>;

ALTER TABLE <table-name> ILM DELETE POLICY <policy-name>;

ALTER TABLE <table-name> ILM DISABLE\_ALL;

ALTER TABLE <table-name> ILM DELETE\_ALL;

-- Partition-level.

ALTER TABLE <table-name> MODIFY PARTITION <partition-name> ILM DISABLE POLICY <policy-name>;

ALTER TABLE <table-name> MODIFY PARTITION <partition-name> ILM DELETE POLICY <policy-name>;

ALTER TABLE <table-name> MODIFY PARTITION <partition-name> ILM DISABLE\_all;

ALTER TABLE <table-name> MODIFY PARTITION <partition-name> ILM DELETE\_ALL;

The following views are available to display policy details.

* {DBA|USER}\_ILMDATAMOVEMENTPOLICIES
* {DBA|USER}\_ILMTASKS
* {DBA|USER}\_ILMEVALUATIONDETAILS
* {DBA|USER}\_ILMOBJECTS
* {DBA|USER}\_ILMPOLICIES
* {DBA|USER}\_ILMRESULTS
* DBA\_ILMPARAMETERS

ILM ADO Parameters

The full list of ILM ADO Parameters are documented [here](http://docs.oracle.com/database/122/VLDBG/ilm-strategy-heatmap-ado.htm#GUID-1C87CC45-14E2-4AE0-849C-3324DF94A4EA__CHDFBBCB). They can be displayed using the following query.

CONN / AS SYSDBA

ALTER SESSION SET CONTAINER = pdb1;

COLUMN name FORMAT A20

SELECT name, value

FROM dba\_ilmparameters

ORDER BY name;

NAME VALUE

-------------------- ----------

ENABLED 1

EXECUTION INTERVAL 15

EXECUTION MODE 2

JOB LIMIT 2

POLICY TIME 0

RETENTION TIME 30

TBS PERCENT FREE 25

TBS PERCENT USED 85

SQL>

These parameters can be altered using the DBMS\_ILM\_ADMIN.CUSTOMIZE\_ILM procedure. There is a constant defined in the package for each parameter, with the name matching the parameter name with the whitespaces replaced by "\_".

BEGIN

DBMS\_ILM\_ADMIN.customize\_ilm(DBMS\_ILM\_ADMIN.retention\_time, 60);

END;

/

For more information see:

* [Implementing an ILM Strategy With Heat Map and ADO](http://docs.oracle.com/database/122/VLDBG/ilm-strategy-heatmap-ado.htm#VLDBG14183)
* [HEAT\_MAP](http://docs.oracle.com/database/122/REFRN/HEAT_MAP.htm#REFRN10342)
* [DBMS\_HEAT\_MAP](http://docs.oracle.com/database/122/ARPLS/DBMS_HEAT_MAP.htm#ARPLS74860)
* [DBMS\_ILM](http://docs.oracle.com/database/122/ARPLS/DBMS_ILM.htm#ARPLS73513)
* [DBMS\_ILM\_ADMIN](http://docs.oracle.com/database/122/ARPLS/DBMS_ILM_ADMIN.htm#ARPLS74887)

Hope this helps. Regards Tim...