ADO

(Automatic Data Optimization)

Author	**김지성**
Creation Date	**2019.08**
Last Updated	
Version	**1.0**

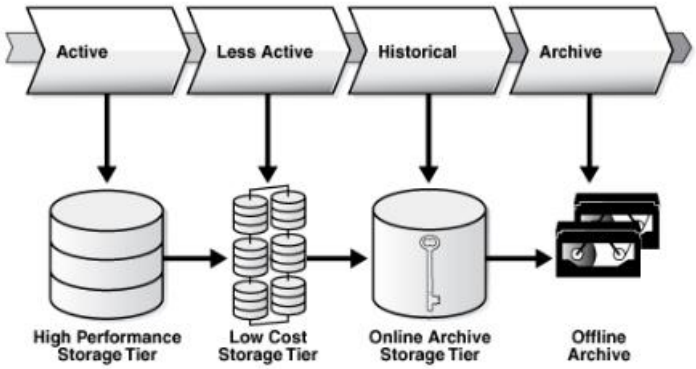
Version	변경일자	변경자(작성자)	주요내용

Contents

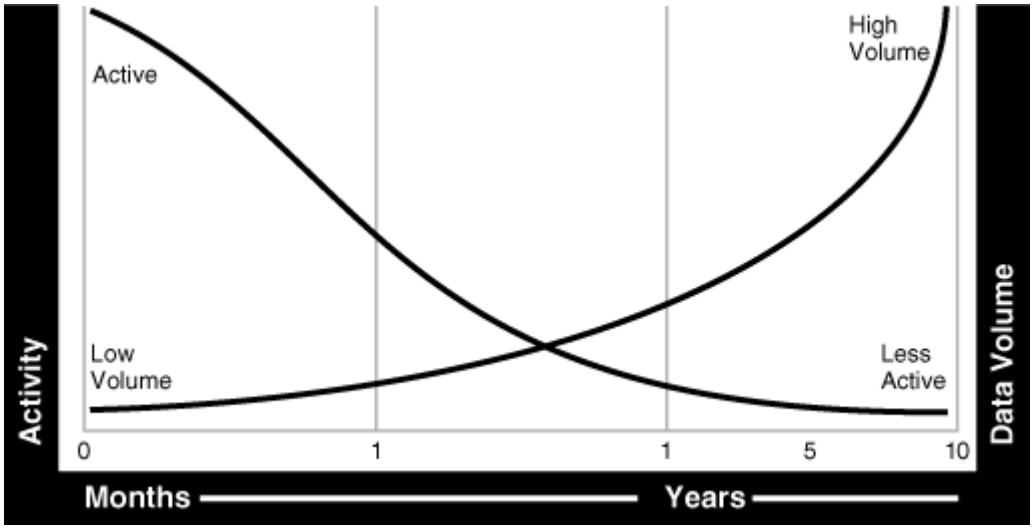
1. ILM (Information Lifecycle Management)

1.1. Overview

오늘날 OLTP(Online Transaction Processing) 시스템의 Data는 Email, 사진, 주문과 같이 다양한 방법으로 제공되며, 구조화되지 않은 많은 Data들이 빠른 속도로 성장하고 있습니다.
최근 기업이 당면한 과제 중 하나는 급증하는 Data를 저비용으로 유지관리하며 최적의 성능을 보장하는 것입니다. 시간 경과에 따라 Data가 어떻게 발전하고 사용 하는지를 모니터링 하여 해당 Data의 수명 주기를 파악 하고 자주 사용되지 않는 Data를 걸러내어 Application의 성능 감소 & Storage 비용 등 관리 포인트를 감소시킬 수 있는 전략을 세워야 합니다.



Oracle의 ILM(Information Lifecycle Management)은 Process, Policy, Software & Hardware를 결합하여 이러한 문제를 해결함으로써 Data의 수명 주기의 각 단계에서 적절한 기술을 사용할 수 있도록 설계되었습니다.



1.2. Oracle ILM 특징

- Application Transparency

ILM 에서는 Application 을 사용자가 정의하지않고 해당 데이터를 사용하는 Application 에 아무런 영향을 미치지 않기때문에 다양한 변경 사항을 데이터에 적용할 수 있습니다. Data 는 Life Cycle 의 여러 단계에서 쉽게 이동가능하며 Application 투명성이 기존 Application 에 아무런 영향을주지 않고 새로운 정책 & 요구 사항에 신속하게 적응하는 유연성을 제공합니다.

- Fine-grained data

정교한 수준의 Data 를 볼 수 있으며 관련 Data 를 Group 화 할 수 있습니다.

- Storage 비용감소

Data 의 Access 패턴에 따라서 자주 사용하지 않는 대량의 Data 를 Low-Cost Storage 에 저장하는것이 ILM 을 구현하는 핵심입니다. Oracle 은 다양한 유형의 저장 장치를 활용할 수 있기 때문에 최소한의 비용으로 최대한의 Data 를 보관할 수 있습니다.

- Enforceable Compliance Policies

규정 준수를 위해 정보를 보관하는 경우 규정에 따라 데이터를 보관하고 관리한다는 것을 규제 기관에 제시하는 것이 중요합니다. Oracle Database 에서 데이터에 대한 모든 액세스를 시행하고 기록하는 보안 및 감사 정책을 정의 할 수 있습니다.

Advanced Compression New Feature

Oracel Database 11G	Oracle Database 12C
OLTP Compression	Advanced Row Compression
Secure Files Compression	Advanced LOB Compression
Secure Files De-duplication	Advanced LOB Deduplication
Hybrid Columnar Compression	Hybrid Columnar Compression
NEW FEATURE=>	HEAT MAP (Object And Row level)
NEW FEATURE=>	Automatic Data Optimization(ADO)
NEW FEATURE=>	Temporal Optimization

2. ADO(Automatic Data Optimization)

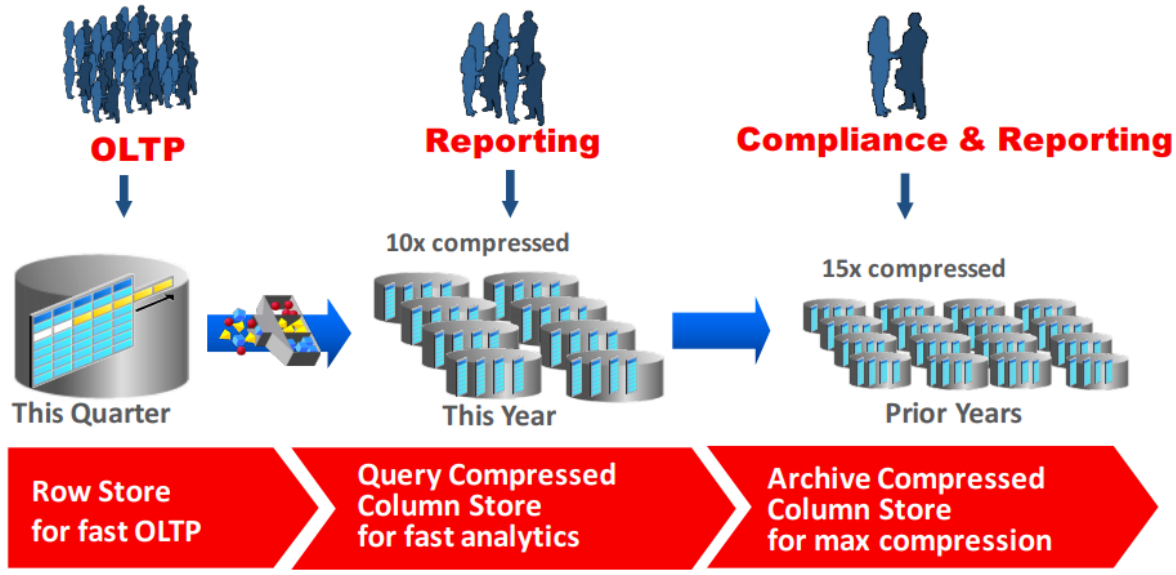
2.1. Overview

Automatic Data Optimization 을 사용하면 데이터 압축 (Smart Compression) 및 데이터 이동에 대한 정책을 생성하여 저장 및 압축 계층화를 구현할 수 있습니다. Smart Compression 은 Heat Map 정보를 활용하여 압축 정책과 압축 수준을 실제 데이터 사용량과 연관시키는 기능을 말합니다.

Oracle Database 는 유지 관리 기간 동안 ADO 정책을 평가하고 Heat Map 에서 수집 한 정보를 사용하여 실행할 작업을 결정합니다. 모든 ADO 작업은 사용자 개입없이 자동으로 백그라운드에서 실행됩니다. ADO 정책은 테이블 및 테이블 파티션의 세그먼트 또는 행 수준에서 지정할 수 있습니다. 정책 기준이 충족 될 때 정책이 백그라운드에서 자동으로 실행되거나 필요할 때 실행할 수 있습니다.

PL / SQL 함수를 사용하여 사용자 정의 조건을 생성하고 ADO 의 유연성을 확장하고 데이터를 이동하거나 압축 할시기를 결정할 수 있습니다.

Heat Map 및 ADO 에는 Advanced Compression 옵션이 필요합니다.



2.2. License

Feature / Option / Pack	SE2	EE	EE-EXA	DBCSSE	DBCSEE	DBCSEE-HP	DBCSEE-HP	EXACS	Notes
Heat Map	N	Y	Y	N	N	Y	Y	Y	EE and EE-Exa: Requires the Oracle Advanced Compression option or the Oracle Database In-Memory option
Automatic Data Optimization	N	Y	Y	N	N	Y	Y	Y	EE and EE-Exa: Requires the Oracle Advanced Compression option or the Oracle Database In-Memory option

2.3. Heat Map

Heat Map 은 Row 및 Segment 레벨에서 데이터 사용 통계를 저장하는 Oracle Database 12c 기능입니다. 스토리지 비용을 줄이고 성능을 향상 시키며 Oracle Database 스토리지를 최적화하기 위해 데이터 압축 및 이동을 자동화하는 데 사용됩니다. Heat Map 은 자동 데이터 최적화와 함께 사용되어 데이터 사용에 따라 압축 및 스토리지 정책을 자동화 할 수 있습니다. Segment 레벨 Heat Map 은 테이블 및 파티션의 마지막 수정 및 액세스 시간을 추적합니다. 행 레벨 Heat Map 은 행 레벨에서 수정 시간을 표시합니다. 한 번 활성화 된 Heat Map 은 데이터 수명주기 동안 자동으로 유지되는 압축 및 저장 정책을 정의하는 데 사용할 수있는 세그먼트 및 행 수준의 통계를 자동으로 수집합니다.

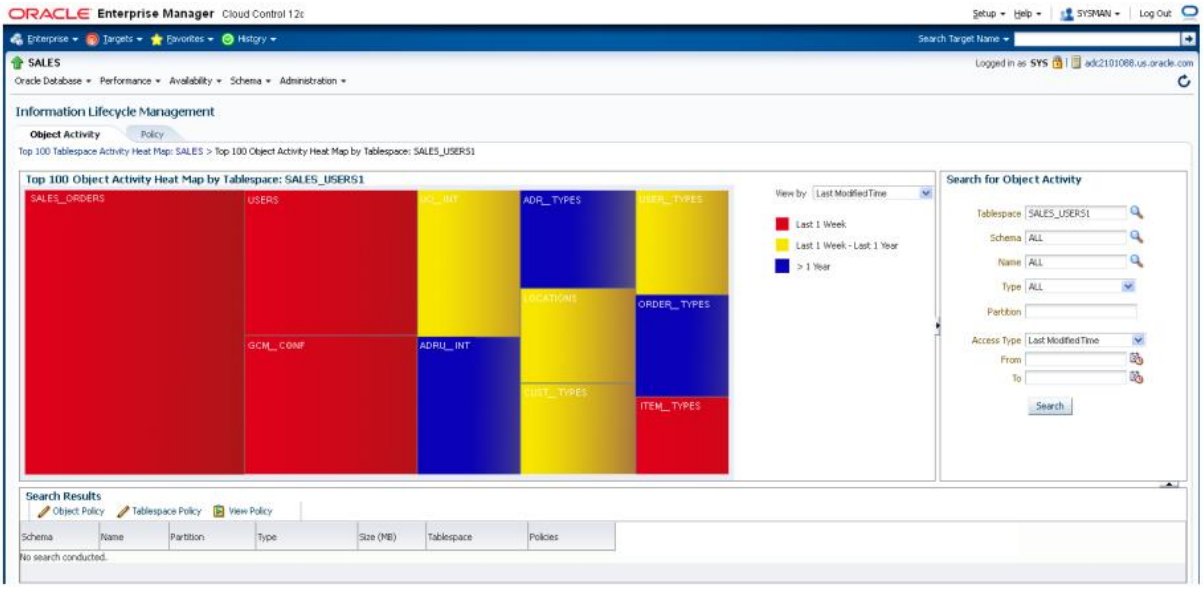
Heat Map 은 다음과 같이 활성화 할 수 있습니다.

```
ALTER SYSTEM SET HEAT_MAP = ON;
```

기본적으로 값은 OFF 입니다.

V$HEAT_MAP_SEGMENT 뷰는 실시간 세그먼트 액세스 정보를 표시합니다.

ALL_, DBA_ 및 USER_HEAT_MAP_SEGMENT 뷰는 사용자에게 표시되는 모든 세그먼트에 대한 최신 세그먼트 액세스 시간을 표시합니다. ALL_, DBA_ 및 USER_HEAT_MAP_SEG_HISTOGRAM 뷰는 사용자에게 표시되는 모든 세그먼트에 대한 세그먼트 액세스 정보를 표시합니다. DBA_HEAT MAP_TOP_OBJECTS 뷰는 상위 1000 개 개체에 대한 Heat Map 정보를 표시합니다. DBA_HEAT MAP_TOP_TABLESPACES 뷰는 상위 100 개의 Tablespace 공간에 대한 Heat Map 정보를 표시합니다.



Heat Map 은 먼저 Memory 에 수집된 정보를 유지하다가 Segment Level Policy 인경우 Scheduler job 을 통해 한시간에 한번씩 Data 를 HEAT_MAP_STAT$ Table 에 저장합니다. Row Level policy 일 경우 MMON Background Process 가 HEAT_MAP_STATS$로부터 통계정보를 계산하여 15 분에 한번씩 평가를 수행합니다.

```
##Segment Level Scheduler 확인
SELECT LOG_DATE, OWNER, JOB_NAME, STATUS
FROM DBA_SCHEDULER_JOB_RUN_DETAILS
WHERE JOB_NAME LIKE 'ILMSEGFLS%'
ORDER BY LOG_DATE DESC;

LOG_DATE                                  OWNER  JOB_NAME        STATUS
----------------------------------------- -----  --------------- ------------
19/07/03 16:07:36.115037 +09:00           SYS    ILMSEGFLS2      SUCCEEDED
```

2.4. Managing Heat Map Data

```
[oracle:YOUNHA:/home/oracle> ss

SQL*Plus: Release 12.2.0.1.0 Production on Sun May 26 15:55:40 2019

Copyright (c) 1982, 2016, Oracle.  All rights reserved.



Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

##Heat_Map Enable 확인
SQL> show parameter heat_map

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
heat_map                             string      OFF

##Heat_Map Enable
SQL> alter system set heat_map=on;

System altered.

##Heat Map에 포착된 Segment 정보 확인
SQL> SELECT SUBSTR(OBJECT_NAME,1,20), SUBSTR(SUBOBJECT_NAME,1,20), TRACK_TIME,
            SEGMENT_WRITE,FULL_SCAN, LOOKUP_SCAN
      FROM V$HEAT_MAP_SEGMENT;

no rows selected

SQL> select * from GD_EMPLOYEE;

(생략)

10000 rows selected.

##FULL Scan(Lookup_scan = Index)
```

```
SQL> SELECT OBJECT_NAME,SUBOBJECT_NAME,TRACK_TIME,SEGMENT_WRITE,FULL_SCAN,LOOKUP_SCAN
     FROM V$HEAT_MAP_SEGMENT;

OBJECT_NAME          SUBOBJECT_NAME  TRACK_TI SEG FUL LOO
-------------------- --------------- -------- --- --- ---
GD_EMPLOYEE                          19/05/26 NO  YES NO

SQL> insert into gd_employee values
(00010001,'KIMJISUNG','JISUNG.KIM@GOODUSDATA.COM','010-6336-6363','19/05/26','Welcome
Oracle');

1 row created.

SQL> commit;

Commit complete.

##Segment Write
SQL> SELECT OBJECT_NAME,SUBOBJECT_NAME,TRACK_TIME,SEGMENT_WRITE,FULL_SCAN,LOOKUP_SCAN
     FROM V$HEAT_MAP_SEGMENT;

OBJECT_NAME          SUBOBJECT_NAME  TRACK_TI SEG FUL LOO
-------------------- --------------- -------- --- --- ---
GD_EMPLOYEE                          19/05/26 YES YES NO

##User Level Segment Access 정보
SQL> SELECT OBJECT_NAME, SUBOBJECT_NAME, SEGMENT_WRITE_TIME, SEGMENT_READ_TIME,
            FULL_SCAN, LOOKUP_SCAN
       FROM USER_HEAT_MAP_SEGMENT;

OBJECT_NAME          SUBOBJECT_NAME  SEGMENT_ SEGMENT_ FULL_SCA LOOKUP_S
-------------------- --------------- -------- -------- -------- --------
GD_EMPLOYEE                          19/05/26          19/05/26



##상위 1000 개의 Heat MAP Object
```

```
SQL> SELECT OWNER, OBJECT_NAME, OBJECT_TYPE, TABLESPACE_NAME, SEGMENT_COUNT
       FROM DBA_HEAT MAP_TOP_OBJECTS
  ORDER BY SEGMENT_COUNT DESC;

OWNER      OBJECT_NAME          OBJECT_TYPE      TABLESPACE_NAME                SEGMENT_COUNT
---------- -------------------- ---------------- ------------------------------ -------------
GD_TEST    GD_ORDER             TABLE            GD_TEST_DATA                              15
GD_TEST    GD_CUSTOMER          TABLE            GD_TEST_DATA                               2
GD_TEST    GD_WAREHOUSE         TABLE            GD_TEST_DATA                               2
GD_TEST    GD_PRODUCT           TABLE            GD_TEST_DATA                               2
GD_TEST    GD_ORDER_ITEM        TABLE            GD_TEST_DATA                               2
GD_TEST    GD_LOCATION          TABLE            GD_TEST_DATA                               2
GD_TEST    GD_PRODUCT_CATEGORY  TABLE            GD_TEST_DATA                               2
GD_TEST    GD_REGION            TABLE            GD_TEST_DATA                               2
GD_TEST    GD_COUNTRY           TABLE            GD_TEST_DATA                               2
GD_TEST    GD_INVENTORY         TABLE            GD_TEST_DATA                               2
GD_TEST    GD_CONTACT           TABLE            GD_TEST_DATA                               2
GD_TEST    GD_EMPLOYEE          TABLE            GD_TEST_DATA                               2

##상위 100 개의 Tablespace 에 대한 Segment 정보
SQL> SELECT TABLESPACE_NAME, SEGMENT_COUNT
       FROM DBA_HEAT MAP_TOP_TABLESPACES
  ORDER BY SEGMENT_COUNT DESC;

TABLESPACE_NAME      SEGMENT_COUNT
-------------------- -------------
GD_TEST_DATA                    37
TEST2                            0
TEST                             0

##GD_TEST Schema 의 GD_ORDER Segment 의 Access 시간 & Size 확인
```

```
SQL> SELECT OWNER, SEGMENT_NAME, PARTITION_NAME, TABLESPACE_NAME,
            SEGMENT_TYPE, SEGMENT_SIZE
       FROM TABLE(DBMS_HEAT_MAP.OBJECT_HEAT_MAP('GD_TEST', 'GD_ORDER'));
```

OWNER	SEGMENT_NAME	PARTITION_NAME	TABLESPACE_NAME	SEGMENT_TYPE	SEGMENT_SIZE
GD_TEST	PK_GD_ORDER		GD_TEST_DATA	INDEX	293601280
GD_TEST	GD_ORDER	GD_ORDER_2012	GD_TEST_DATA	TABLE PARTITION	41943040
GD_TEST	GD_ORDER	GD_ORDER_2006	GD_TEST_DATA	TABLE PARTITION	100663296
GD_TEST	GD_ORDER	GD_ORDER_2018	GD_TEST_DATA	TABLE PARTITION	41943040
GD_TEST	GD_ORDER	GD_ORDER_2008	GD_TEST_DATA	TABLE PARTITION	41943040
GD_TEST	GD_ORDER	GD_ORDER_2017	GD_TEST_DATA	TABLE PARTITION	41943040
GD_TEST	GD_ORDER	GD_ORDER_2010	GD_TEST_DATA	TABLE PARTITION	41943040
GD_TEST	GD_ORDER	GD_ORDER_2009	GD_TEST_DATA	TABLE PARTITION	41943040
GD_TEST	GD_ORDER	GD_ORDER_2007	GD_TEST_DATA	TABLE PARTITION	41943040
GD_TEST	GD_ORDER	GD_ORDER_2013	GD_TEST_DATA	TABLE PARTITION	41943040
GD_TEST	GD_ORDER	GD_ORDER_2011	GD_TEST_DATA	TABLE PARTITION	41943040
GD_TEST	GD_ORDER	GD_ORDER_2015	GD_TEST_DATA	TABLE PARTITION	41943040
GD_TEST	GD_ORDER	GD_ORDER_2019	GD_TEST_DATA	TABLE PARTITION	16777216
GD_TEST	GD_ORDER	GD_ORDER_2016	GD_TEST_DATA	TABLE PARTITION	41943040
GD_TEST	GD_ORDER	GD_ORDER_2014	GD_TEST_DATA	TABLE PARTITION	41943040

2.5. Restriction

- ADO 및 Heat Map 은 다중 점유 컨테이너 데이터베이스 (CDB)에서 지원되지 않습니다. [즉, CDB + 하나 이상의 PDB]
- 정책이 테이블 공간 수준에서 기본값 인 경우 ADO 에 대한 사용자 지정 정책 (사용자 정의 함수)은 지원되지 않습니다.
- ADO 는 Storage Tiering 를 사용할 때 대상 Tablespace Fres Space 에 대한 검사를 수행하지 않습니다.
- ILM 정책은 21 일의 배수로 지정할 수 있습니다.
- Object Type 이나 Materialized View 테이블에서는 ADO 가 지원되지 않습니다.
- ADO 는 IOT Table 또는 클러스터에서 지원되지 않습니다.
- ADO 동시성 (ADO 에 대한 동시 정책 작업 수)은 Oracle 스케줄러의 동시성에 따라 다릅니다. ADO 에 대한 작업이 두 번 이상 실패하면 작업이 사용 안 함으로 표시되고 작업은 나중에 수동으로 활성화해야합니다.
- ADO 정책은 Oracle Scheduler 유지 관리 창에서만 실행됩니다. 유지 관리 창 외부에서는 모든 정책이 중지됩니다.
- ADO 에는 테이블과 테이블 파티션 이동과 관련된 제한 사항이 있습니다.

3. Smart Compression & Tiering

3.1. Overview

ILM 정책을 수립하여 ADO 를 사용하여 데이터베이스 내의 다른 계층의 Storage 간에 Data Compress 및 이동을 자동화 할 수 있습니다. Data Optimization 을 자동화 하려면 SYSTEM 수준에서 HEAT_MAP 을 활성화해야합니다.

SQL CREATE 및 ALTER TABLE 명령으로 ILM 절을 추가하여 ADO 에 대한 정책을 작성, DISABLE, ENABLE 할 수 있습니다. ILM 정책 절은 Compress 또는 저장소 계층화 정책을 설정하고 정책 작업이 발생할 때 조건을 지정하는 AFTER and ON 절과 같은 추가 절을 포함 합니다.

전체 테이블 또는 테이블 파티션에 정책을 추가 할 수 있습니다.

```
##Sample Table 생성
CREATE TABLE GD_ORDER_T
(ORDER_ID VARCHAR2(10)
, CUSTOMER_ID VARCHAR2(8)  NOT NULL
, STATUS      VARCHAR2(5) NOT NULL
, EMPLOYEE_ID VARCHAR2(8)
, ORDER_DATE  DATE NOT NULL
)  PARTITION BY RANGE (ORDER_DATE)
(
  PARTITION GD_ORDER_2006 VALUES LESS THAN (TO_DATE('20070101', 'YYYYMMDD'))
, PARTITION GD_ORDER_2007 VALUES LESS THAN (TO_DATE('20080101', 'YYYYMMDD'))
, PARTITION GD_ORDER_2008 VALUES LESS THAN (TO_DATE('20090101', 'YYYYMMDD'))
, PARTITION GD_ORDER_2009 VALUES LESS THAN (TO_DATE('20100101', 'YYYYMMDD'))
, PARTITION GD_ORDER_2010 VALUES LESS THAN (TO_DATE('20110101', 'YYYYMMDD'))
, PARTITION GD_ORDER_2011 VALUES LESS THAN (TO_DATE('20120101', 'YYYYMMDD'))
, PARTITION GD_ORDER_2012 VALUES LESS THAN (TO_DATE('20130101', 'YYYYMMDD'))
, PARTITION GD_ORDER_2013 VALUES LESS THAN (TO_DATE('20140101', 'YYYYMMDD'))
, PARTITION GD_ORDER_2014 VALUES LESS THAN (TO_DATE('20150101', 'YYYYMMDD'))
, PARTITION GD_ORDER_2015 VALUES LESS THAN (TO_DATE('20160101', 'YYYYMMDD'))
, PARTITION GD_ORDER_2016 VALUES LESS THAN (TO_DATE('20170101', 'YYYYMMDD'))
, PARTITION GD_ORDER_2017 VALUES LESS THAN (TO_DATE('20180101', 'YYYYMMDD'))
, PARTITION GD_ORDER_2018 VALUES LESS THAN (TO_DATE('20190101', 'YYYYMMDD'))
, PARTITION GD_ORDER_2019 VALUES LESS THAN (TO_DATE('20200101', 'YYYYMMDD'))
, PARTITION GD_ORDER_MAX  VALUES LESS THAN (MAXVALUE))
ILM ADD POLICY ROW STORE COMPRESS ADVANCED SEGMENT AFTER 12 MONTHS OF NO ACCESS;


Table created.
```

```
##ILM 정책 확인
SQL> SELECT policy_name, policy_type, enabled
        FROM USER_ILMPOLICIES;

POLICY_NAME          POLICY_TYPE   ENA
-------------------- ------------- ---
P1                   DATA MOVEMENT YES
```

```
## 위에서 생성한 GD_ORDER_T Table 에 ALTER 문을 사용하여 ILM 정책을 ADD 할 수 있다.

[ 30 일간 수정이 없는 테이블 GD_ORDER_T 에 대하여 Row-level Compress 적용 ]
ALTER TABLE GD_ORDER_T ILM ADD POLICY ROW STORE COMPRESS ADVANCED ROW
AFTER 30 DAYS OF NO MODIFICATION;

[ 30 일간 수정이 없는 테이블 GD_ORDER_T 에 대하여 Segment-level Compress 적용 ]
ALTER TABLE GD_ORDER_T MODIFY PARTITION GD_ORDER_2006
ILM ADD POLICY ROW STORE COMPRESS ADVANCED SEGMENT
AFTER 30 DAYS OF NO MODIFICATION;

[ 6 개월간 수정이 없는 테이블 GD_ORDER_T 에 대하여 Segment-level Compress 적용 ]
ALTER TABLE GD_ORDER_T MODIFY PARTITION GD_ORDER_2006
ILM ADD POLICY COMPRESS FOR ARCHIVE HIGH SEGMENT
AFTER 6 MONTHS OF NO MODIFICATION;

[ 12 개월간 접근이 없는 테이블 GD_ORDER_T 에 대하여 Segment-level Compress 적용 ]
ALTER TABLE GD_ORDER_T MODIFY PARTITION GD_ORDER_2006
ILM ADD POLICY COMPRESS FOR ARCHIVE HIGH SEGMENT
AFTER 12 MONTHS OF NO ACCESS;

[ Low-Cost Storage 로 테이블 이동을 위해 Storage Tier 정책 적용 ]
ALTER TABLE GD_ORDER_T MODIFY PARTITION GD_ORDER_2008
ILM ADD POLICY TIER TO TEST;

SQL> SELECT policy_name, policy_type, enabled
       FROM USER_ILMPOLICIES;
```

```
POLICY_NAME          POLICY_TYPE   ENA
-------------------- ------------- ---
P1                   DATA MOVEMENT YES
P2                   DATA MOVEMENT YES
P3                   DATA MOVEMENT YES
P4                   DATA MOVEMENT YES
P5                   DATA MOVEMENT YES
P6                   DATA MOVEMENT YES
```

3.2. Storage Tiering

Storage 계층화 정책을 생성할 때 12C 에서는 Segment 또는 Tablespace 두 가지 level 에서 계층화를 할 수 있습니다. Row level 은 지원하지 않습니다.

Storage 이동을 위한 정책 설정시 TBS_PERCENT_USED, TBS_PERCENT_FREE 두 파라미터를 활용할 수 있습니다. 두 파라미터의 Default 값은 각 85,25 입니다. Source Tablespace 의 공간이 85% 이상이되면, Tiering 정책에 의해서 Segment 를 Target Tablespace 로 이동시키고 FREE Space 25%를 유지합니다.

```
SQL> select * from dba_ilmparameters;

NAME                           VALUE
------------------------- ----------
ENABLED                             1
RETENTION TIME                     30
JOB LIMIT                           2
EXECUTION MODE                      2
EXECUTION INTERVAL                 15
TBS PERCENT USED                   85
TBS PERCENT FREE                   25
POLICY TIME                         0
```

NAME	Description
ENABLED	ENABLED 매개 변수는 ADO 백그라운드 평가 및 실행을 제어합니다. 기본값은 1 or True 입니다.
RETENTION TIME	RETENTION TIME 은 완료된 ADO Task 의 데이터가 제거되기 전에 유지되는 시간을 지정합니다. 기본값은 30 일입니다.
JOB LIMIT	JOB LIMIT 은 ADO 작업의 최대 수를 제어합니다. 동시 ADO 작업의 최대 수는 (JOB LIMIT) * (인스턴스 수) * (각 인스턴스의 CPU 수)로 계산됩니다. 기본값은 10 입니다.
EXECUTION MODE	ADO 가 온라인 또는 오프라인 모드에서 실행되는지 여부를 제어합니다. 기본값은 온라인입니다.
EXECUTION INTERVAL	ADO 가 백그라운드 평가를 시작하는 빈도를 결정합니다. 기본값은 15 분입니다.
TBS PERCENT USED	테이블 공간이 가득 찬 것으로 간주 될 때 테이블 공간 할당량의 백분율을 지정합니다. 기본값은 85 퍼센트입니다.
TBS PERCENT FREE	테이블 공간에 대한 목표 여유 백분율을 지정합니다. 기본값은 25 퍼센트입니다.

3.2.1. Storage tiering 실습

Heat_Map = ON 으로 설정된 상태에서 High Tablespace(Source)와 Low Tablespace(Target)을 생성 후 특정 Table을 High Tablespace에 생성합니다.
TBS_PERCENT_USED, TBS_PERCENT_FREE Parameter Value를 각각 5, 95로 설정하여 Storage tiering을 확인해보겠습니다.

```
##HIGH / LOW 10M Tablespace 생성
SQL> create tablespace high datafile '/oradata/gd/high01.dbf' size 10m autoextend off;

Tablespace created.

SQL> create tablespace low datafile '/oradata/gd/low01.dbf' size 10m autoextend off;

Tablespace created.

##User 생성 및 권한 추가
SQL> create user gd identified by gd default tablespace high;

User created.

SQL> alter user gd quota unlimited on high;

User altered.

SQL> alter user gd quota unlimited on low;

User altered.

SQL> grant dba to gd;

Grant succeeded.




##Table 생성 후 5000 Rows Insert
```

```
SQL> conn gd/gd

Connected.

SQL> CREATE TABLE EMP
                (EMPNO NUMBER(4) NOT NULL,
                 ENAME VARCHAR2(10),
                 JOB VARCHAR2(9),
                 MGR NUMBER(4),
                 HIREDATE DATE,
                 SAL NUMBER(7, 2),
                 COMM NUMBER(7, 2),
                 DEPTNO NUMBER(2));

Table created.

SQL> begin
for i in 1..5000 loop
    INSERT /*+ APPEND */ INTO EMP VALUES (i, 'SMITH','CLERK',7902,TO_DATE('17-12-1980',
'DD-MM-YYYY'),800, NULL, 20);
end loop;
commit;
end;
/

PL/SQL procedure successfully completed.
```

##Tablespace Free Space 조회

```
SQL> set pagesize 66
SQL> set lines 200
column  pct_used format 999.9       heading "%|Used"
column  name    format a16      heading "Tablespace Name"
column  bytes   format 9,999,999,999,999    heading "Total Bytes"
column  used    format 9,999,999,999,999   heading "Used"
column  free    format 999,999,999,999  heading "Free"
column  auto    format a5 heading "auto"
```

```
SQL> select a.tablespace_name name,
sum(b.bytes)/count( distinct a.file_id||'.'||a.block_id )  bytes,
sum(b.bytes)/count( distinct a.file_id||'.'||a.block_id ) -
sum(a.bytes)/count( distinct b.file_id ) used,
sum(a.bytes)/count( distinct b.file_id ) free,
100 * ( (sum(b.bytes)/count( distinct a.file_id||'.'||a.block_id )) -
(sum(a.bytes)/count( distinct b.file_id ) )) / (sum(b.bytes)/count( distinct
a.file_id||'.'||a.block_id )) pct_used, b.autoextensible auto
from sys.dba_free_space a, sys.dba_data_files b
where a.tablespace_name = b.tablespace_name
and b.tablespace_name in ('LOW','HIGH')
group by a.tablespace_name, b.tablespace_name, b.autoextensible
order by a.tablespace_name;
```

Tablespace Name	Total Bytes	Used	Free	% Used	auto
HIGH	10,485,760	1,310,720	9,175,040	12.5	NO
LOW	10,485,760	1,048,576	9,437,184	10.0	NO

##EMP Table Location 조회

```
Col owner for a10
Col segment_name for a12
Col tablespace_name for a15
SQL> select owner,segment_name,tablespace_name from dba_segments where
segment_name='EMP' and owner='GD';
```

OWNER	SEGMENT_NAME	TABLESPACE_NAME
GD	EMP	HIGH

##Storage Tiering Policy 추가

```
SQL> ALTER TABLE EMP ILM ADD POLICY TIER TO LOW;

Table altered.
```

##ILM Move Policy 확인

```
col policy_name for a11
col COMPRESSION_LEVEL for a17
col TIER_TABLESPACE for a15
SQL> select policy_name,action_type, scope,
compression_level,tier_tablespace,condition_type, condition_days
from dba_ilmdatamovementpolicies
where tier_tablespace = 'LOW'
order by policy_name;

POLICY_NAME ACTION_TYPE SCOPE   COMPRESSION_LEVEL TIER_TABLESPACE CONDITION_TYPE         CONDITION_DAYS
----------- ----------- ------- ----------------- --------------- --------------------- --------------
P21         STORAGE     SEGMENT                   LOW                                             0
```

##ILM Object 확인

```
col object_owner for a12
col object_name for a11
col subobject_name for a14
col object_type for a11
SQL> select *
from dba_ilmobjects
where object_owner='GD';

POLICY_NAME OBJECT_OWNER OBJECT_NAME SUBOBJECT_NAME OBJECT_TYPE INHERITED_FROM       TBS_INHERITED_FROM          ENA DEL
----------- ------------ ----------- -------------- ----------- -------------------- ------------------------------ --- ---
P21         GD           EMP                        TABLE       POLICY NOT INHERITED           YES NO
```

##ILM Parameter Value 확인

```
Col name for a20
Col value for 999
SQL> select * from dba_ilmparameters where NAME LIKE 'TBS%';

Tablespace Name      VALUE
-------------------- -----
TBS PERCENT USED        85
TBS PERCENT FREE        25
```

##ILM Parameter Value 변경

```
SQL> EXEC dbms_ilm_admin.customize_ilm(DBMS_ILM_ADMIN.TBS_PERCENT_FREE,95);

PL/SQL procedure successfully completed.

SQL> EXEC dbms_ilm_admin.customize_ilm(DBMS_ILM_ADMIN.TBS_PERCENT_USED,5);

PL/SQL procedure successfully completed.

Col name for a20
Col value for 999
SQL> select * from dba_ilmparameters where NAME LIKE 'TBS%';

Tablespace Name      VALUE
-------------------- -----
TBS PERCENT USED         5
TBS PERCENT FREE        95

SQL> select owner,segment_name,tablespace_name from dba_segments where
segment_name='EMP' and owner='GD';

OWNER      SEGMENT_NAME TABLESPACE_NAME
---------- ------------ ---------------
GD         EMP          HIGH

##Manual Execute ILM Policy
set serveroutput on
declare
v_var number;
begin
DBMS_ILM.EXECUTE_ILM (owner=>'GD',object_name=>'EMP',task_id=>v_var,
policy_name=>'P21');
end;
/
PL/SQL procedure successfully completed.
```

```
SQL> select owner,segment_name,tablespace_name from dba_segments where
segment_name='EMP' and owner='GD';

OWNER      SEGMENT_NAME TABLESPACE_NAME
---------- ------------ ---------------
GD         EMP          LOW

SQL> select a.tablespace_name name,
sum(b.bytes)/count( distinct a.file_id||'.'||a.block_id )  bytes,
sum(b.bytes)/count( distinct a.file_id||'.'||a.block_id ) -
sum(a.bytes)/count( distinct b.file_id ) used,
sum(a.bytes)/count( distinct b.file_id ) free,
100 * ( (sum(b.bytes)/count( distinct a.file_id||'.'||a.block_id )) -
(sum(a.bytes)/count( distinct b.file_id ) )) / (sum(b.bytes)/count( distinct
a.file_id||'.'||a.block_id )) pct_used, b.autoextensible auto
from sys.dba_free_space a, sys.dba_data_files b
where a.tablespace_name = b.tablespace_name
and b.tablespace_name in ('LOW','HIGH')
group by a.tablespace_name, b.tablespace_name, b.autoextensible
order by a.tablespace_name;

                                                                    %
Tablespace Name         Total Bytes              Used             Free   Used auto
---------------- ------------------ ------------------ ---------------- ------ -----
HIGH                    10,485,760          1,048,576        9,437,184   10.0 NO
LOW                     10,485,760          1,441,792        9,043,968   13.8 NO
```

##ILM Task Operation 확인

```
alter session set nls_date_format='yyyy-mm-dd hh24:mi:ss';
col task_owner for a10
col completion for a40
SQL> SELECT *
FROM dba_ilmtasks;

TASK_ID TASK_OWNER STATE     CREATION_TIME                       START_TIME          COMPLETION_TIME
--------- --------- -------- ----------------------------------- ----------------------------------- ----------
-----------------------------------------------------------

361 GD        COMPLETED 19/06/26 22:48:30.094871              19/06/26 22:48:30.094871            19/06/26
```

```
22:48:30.094871

SQL> SELECT * FROM user_ilmevaluationdetails;

TASK_ID POLICY_NAME OBJECT_OWNER OBJECT_NAME SUBOBJECT_NAME OBJECT_TYPE
SELECTED_FOR_EXECUTION                    JOB_NAME        COMMENTS
---------- ---------- ------------ ---------- -------------- ---------- -------------
----------------------------- --------------- --------------------------------

       361 P21          GD            EMP                            TABLE       POLICY
DISABLED
```

3.3. Compression Tiering

ADO 와 Heat_Map 을 이용한 Compression Tiering 을 사용하면 Storage 비용 절감 & 운영 비용 절약이 가능합니다. Compression Tiering 은 Row, Segment, Group, Tablespace 단위로 가능하며 Segment 단위의 정책은 Table & Partition 에 적용 됩니다. 기본적으로 ADO Policy 이 적용되는 Default Level 은 Tablespace 입니다.

	No modification	Creation	Low Access	No Access	Tablespace fullness	Custom policy
Compression (Row)	✓					
Compression (Segment)	✓		✓	✓		✓
Compression (Group)	✓		✓	✓		
Compression (Tablespace)	✓	✓	✓	✓		
Storage Tiering (Segment)					✓	✓
Storage Tiering (Tablespace)					✓	

3.4. Row level Compression

Row Compress 는 Insert, Update, Bulk-loaded 시에 발생하며 압축 TYPE 은 4 가지로 구성 됩니다.

- ROW STORE COMPRESS BASIC

  Basic Table Compression 을 사용하며 Direct-Path Insert 시 압축이 가능합니다.

- ROW STORE COMPRESS ADVANCED

  Advanced Row Compression(OLTP Table Compression)은 테이블의 모든 DML 문에 대해서 압축을 진행합니다. OLTP 환경에서 권장하는 Type 입니다.

- COLUMN STORE COMPRESS FOR QUERY LOW/HIGH

  HCC(Exadata Hybrid Columnar Compression)를 사용하는 방식으로 Row Store 보다 압축률이 높습니다. Load 성능이 중요하거나 DML 구문이 적을 때 사용됩니다. DW 환경에서 권장하는 Type 입니다. Default Vale HIGH

- COLUMN STORE COMPRESS FOR ARCHIVE LOW/HIGH

  HCC(Exadata Hybrid Columnar Compression)를 사용하는 방식으로 가장 높은 수준의 압축률을 제공하며 거의 Access 가 없는 데이터 or Read-only 데이터를 대상으로 권장됩니다. Default Value LOW

3.4.1. Segment Level Compress

세그먼트 레벨 정책은 한 번만 실행됩니다. 정책이 성공적으로 실행 된 후에는 정책이 실행 중지되고 다시 평가되지 않습니다. 그러나 정책을 다시 명시적으로 활성화 할 수 있습니다. 행 수준 정책은 계속 실행되고 성공적인 실행 후에는 비활성화되지 않습니다.

```
SQL> ALTER TABLE GD ILM ADD POLICY
COLUMN STORE COMPRESS FOR QUERY HIGH
SEGMENT AFTER 90 DAYS OF NO MODIFICATION;
```

3.4.2. Tablespace Level Compress

ADO 정책의 디폴트인 TABLESPACE 단위로 row store compress advanced 압축 정책을 적용하는 예입니다.
아래처럼 AFTER 30 DAYS OF LOW ACCESS 라고 지정할 경우에, 지난 30 일 동안의 통계 정보들이 LOW ACCESS metric 을 결정합니다.
정책은 접근 시간과, 특정 기간동안의 통계들을 기반으로 LOW ACCESS 를 평가합니다. RowID 에 접근한 후, full table scan 이 수행된 수, DML 의 발생 빈도등이 참고되는 정보들입니다.

```
SQL> ALTER TABLESPACE GOODUSDATA DEFAULT ILM ADD POLICY
ROW STORE COMPRESS ADVANCED SEGMENT AFTER 30 DAYS OF LOW ACCESS;
```

3.4.3. GROUP Level Compression

Group Level Compress 는 Policy 에 적용할 Table & Partition 에 의존하는 Secure File LOB 도 압축이 적용되며 Global Index 도 유지됩니다.

그룹 정책에 적용 할 수있는 압축의 기본 매핑은 다음과 같습니다.

- COMPRESS ADVANCED : 힙 테이블의 경우 인덱스 및 LOWLOB 세그먼트의 표준 압축에 매핑됩니다.
- COMPRESS FOR QUERY LOW / HIGH : 인덱스 및 MEDIUM LOB 세그먼트의 표준 압축에 매핑됩니다.
- COMPRESS FOR ARCHIVE LOW / HIGH : 인덱스 및 HIGH LOB 세그먼트의 표준 압축에 매핑됩니다 .

압축 매핑은 변경할 수 없습니다. GROUP 세그먼트 수준 정책에만 적용될 수 있습니다. 스토리지 계층화 정책은 세그먼트 수준에서만 적용되며 행 수준에서 지정할 수 없습니다.

90 일동안 데이터의 수정이 없을 시 자동적으로 테이블 GD 와 GD 에 dependent 한 SecureFiles LOB 모두를 row store compress advanced level 로 압축한다.

```
SQL > ALTER TABLE GD ILM ADD POLICY
ROW STORE COMPRESS ADVANCED
GROUP AFTER 90 DAYS OF NO MODIFICATION;
```

6 달 동안 데이터의 접근이 없을 시 파티션 p2019 와 해당 파티션에 관련된 SecureFiles LOB 를 HCC archive high level 로 압축한다.

```
SQL > ALTER TABLE tab2 MODIFY PARTITION p1 ILM ADD POLICY
COLUMN STORE COMPRESS FOR ARCHIVE HIGH
GROUP AFTER 6 MONTHS OF NO ACCESS;
```

3.4.4. Compression Tiering 실습

ROW STORE COMPRESSION 을 사용하여 ADO policy 생성 및 관리방법에대해 알아보겠습니다.
지난 60 일 동안 수정이 없는 Table 의 Block 들을 Compression Tiering 을 적용하기 위해 EMP Table 에 ROW STORE COMPRESSION ADO Policy 를 생성하겠습니다.

##Heat map tracking start time 을 100 일 뒤로 돌려서 설정 이후 수집되는 통계들이 ADO Policy 에 사용될 수 있는 설정을 수행합니다.

```
SQL> exec dbms_ilm_admin.set_heat_map_start(start_date => sysdate - 100);

PL/SQL procedure successfully completed.

SQL> DROP TABLE EMP PURGE;
SQL> CREATE TABLE EMP
                (EMPNO NUMBER(4) NOT NULL,
                 ENAME VARCHAR2(10),
                 JOB VARCHAR2(9),
                 MGR NUMBER(4),
                 HIREDATE DATE,
                 SAL NUMBER(7, 2),
                 COMM NUMBER(7, 2),
                 DEPTNO NUMBER(2));

Table created.

SQL> begin
for i in 1..5000 loop
    INSERT /*+ APPEND */ INTO EMP VALUES (i, 'SMITH','CLERK',7902,TO_DATE('17-12-1980',
'DD-MM-YYYY'),800, NULL, 20);
end loop;
commit;
end;
/

PL/SQL procedure successfully completed.
```

```
SQL> alter session set nls_date_format='yyyy-mm-dd hh24:mi:ss';

Session altered.
```

##EMP Table Heat MAP Tracking 유무 확인

```
SQL> select OBJECT_NAME,SEGMENT_WRITE_TIME , SEGMENT_READ_TIME, FULL_SCAN
FROM dba_heat_map_segment
WHERE OBJECT_NAME='EMP'
AND OWNER = 'GD';

OBJECT_NAME SEGMENT_WRITE_TIME  SEGMENT_READ_TIME   FULL_SCAN
----------- ------------------- ------------------- -------------------
EMP         2019-06-27 01:04:29

SQL> select count(*) from EMP;

COUNT(*)
----------
5000

SQL> select OBJECT_NAME,SEGMENT_WRITE_TIME , SEGMENT_READ_TIME, FULL_SCAN
FROM dba_heat_map_segment
WHERE OBJECT_NAME='EMP'
AND OWNER = 'GD';

OBJECT_NAME SEGMENT_WRITE_TIME  SEGMENT_READ_TIME   FULL_SCAN
----------- ------------------- ------------------- -------------------
EMP         2019-06-27 01:04:42                     2019-06-27 01:04:42

SQL> select object_name, track_time,
segment_write, full_scan, lookup_scan
from DBA_HEAT_MAP_SEG_HISTOGRAM
WHERE OBJECT_NAME='EMP'
AND OWNER = 'GD';

OBJECT_NAME TRACK_TIME          SEG FUL LOO
----------- ------------------- --- --- ---
```

```
EMP          2019-06-27 01:07:15 YES YES NO
EMP          2019-06-26 22:33:38 YES NO  NO
```

##ADO Policy를 설정하기 전에 EMP Table에 Compression 속성 확인

```
col table_name for a10
col compress for a10
SQL> select owner,table_name,compression,compress_for from dba_tables where
table_name='EMP' and owner='GD';

OWNER      TABLE_NAME COMPRESS COMPRESS_FOR
---------- ---------- -------- ------------------------------
GD         EMP        DISABLED

SQL> select b.name, b.subname,
a.AVGROWSIZE_NC,a.AVGROWSIZE_C,
a.NBLK_NC, a.NBLK_ADVANCED, a.NBLK_EHCC,
a.NROWS_NC, a.NROWS_ADVANCED, a.NROWS_EHCC
from sys.compression_stat$ a, sys.obj$ b
where a.obj# = b.obj# and
b.owner# = userenv('SCHEMAID');

no rows selected
```

##EMP TABLE ADO Policy 추가

```
SQL> alter table EMP ILM ADD POLICY ROW STORE COMPRESS ADVANCED SEGMENT
AFTER 60 DAYS OF NO MODIFICATION;

Table altered.
```

##ILM Policy 정책 확인

```
SQL> select policy_name, action_type, scope, compression_level, condition_type,
condition_days
from dba_ilmdatamovementpolicies
order by policy_name;

POLICY_NAME ACTION_TYPE SCOPE   COMPRESSION_LEVEL CONDITION_TYPE         CONDITION_DAYS
----------- ----------- ------- ----------------- ---------------------- --------------
P42         COMPRESSION SEGMENT ADVANCED          LAST MODIFICATION TIME             60

col object_owner for a12
col object_name for a11
col subobject_name for a14
col object_type for a11
SQL> select *
from dba_ilmobjects
where object_owner='GD';

POLICY_NAME OBJECT_OWNER OBJECT_NAME SUBOBJECT_NAME OBJECT_TYPE INHERITED_FROM      TBS_INHERITED_FROM         ENA DEL
----------- ------------ ----------- -------------- ----------- ------------------ -------------------------- --- ---
P42         GD           EMP                        TABLE       POLICY NOT INHERITED          YES NO

SQL> select * from dba_ilmpolicies;

POLICY_NAME POLICY_TYPE   TABLESPACE                      ENA DEL
----------- ------------- ------------------------------- --- ---
P42         DATA MOVEMENT                                 YES NO

##PL/SQL 을 사용하여 ADO Policy가 발생할 수 있는 시점으로 통계정보를 변경합니다
SQL> conn / as sysdba

Connected.

SQL> alter session set nls_date_format='yyyy-mm-dd hh24:mi:ss';

Session altered.

SQL> declare
```

```
v_object_id number;
v_dataobject_id number;
v_ts number;

begin
select object_id,data_object_id into v_object_id,v_dataobject_id
from dba_objects
where object_name='EMP'
and owner='GD';

select ts# into v_ts
from sys.ts$ a, dba_segments b
where a.name=b.tablespace_name
and b.segment_name='EMP'
and b.owner='GD';

##임의로 Heat Map Tracking Time 설정
insert into sys.heat_map_stat$
(obj#, dataobj#, track_time, segment_access, ts#) values
(v_object_id, v_dataobject_id, sysdate - 60, 1, v_ts );

commit;

end;
/

PL/SQL procedure successfully completed.

SQL> select OBJECT_NAME,SEGMENT_WRITE_TIME , SEGMENT_READ_TIME, FULL_SCAN
FROM dba_heat_map_segment
WHERE OBJECT_NAME='EMP'
AND OWNER = 'GD';

OBJECT_NAME SEGMENT_WRITE_TIME  SEGMENT_READ_TIME   FULL_SCAN
----------- ------------------- ------------------- -------------------
EMP         2019-06-27 01:57:01
```

```
SQL> select * from sys.heat_map_stat$;

OBJ#    DATAOBJ#         TS# TRACK_TIME           SEGMENT_ACCESS       FLAG      SPARE1
SPARE2
---------- ---------- ---------- ------------------- -------------- ---------- ---------
- ----------
SPARE3
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
-------------------------
N_WRITE       N_FTS    N_LOOKUP
---------- ---------- ----------
74119       74119          12 2019-04-28 01:56:38              10          0          0

##변경된 날짜를 확인하기 위해 Restart 수행
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.

SQL> startup
ORACLE instance started.

Total System Global Area 1577058304 bytes
Fixed Size                  8621136 bytes
Variable Size             687866800 bytes
Database Buffers          872415232 bytes
Redo Buffers                8155136 bytes
Database mounted.
Database opened.

SQL> alter session set nls_date_format='yyyy-mm-dd hh24:mi:ss';

Session altered.

SQL> select OBJECT_NAME,SEGMENT_WRITE_TIME , SEGMENT_READ_TIME, FULL_SCAN
FROM dba_heat_map_segment
```

```
WHERE OBJECT_NAME='EMP'
AND OWNER = 'GD';

OBJECT_NAME SEGMENT_WRITE_TIME  SEGMENT_READ_TIME   FULL_SCAN
----------- ------------------- ------------------- -------------------
EMP         2019-04-28 01:56:38

SQL> select * from sys.heat_map_stat$;

OBJ#   DATAOBJ#        TS# TRACK_TIME           SEGMENT_ACCESS       FLAG     SPARE1
SPARE2
---------- ---------- ---------- ------------------- -------------- ---------- ---------
- ----------
SPARE3
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
-------------------------
N_WRITE      N_FTS   N_LOOKUP
---------- ---------- ----------
74119      74119         12 2019-04-28 01:56:38              1
0          0          0

SQL> declare
v_var number;
begin
DBMS_ILM.EXECUTE_ILM(ILM_SCOPE=>dbms_ilm.scope_schema,
execution_mode=>dbms_ilm.ilm_execution_offline,
task_id=>v_var);
end;
/

PL/SQL procedure successfully completed.
```

##ILM Task 확인

```
SQL> select task_id, start_time as start_time from dba_ilmtasks;

TASK_ID    START_TIME
---------- ---------------------------------------------------------------------------
364          19/06/27 03:20:44.553142

SQL> select task_id, job_name, job_state, completion_time completion from
dba_ilmresults;

TASK_ID    JOB_NAME            JOB_STATE                      COMPLETION
---------- ------------------- ------------------------------ ------------------------------
364        ILMJOB781           COMPLETED SUCCESSFULLY         19/06/27 03:21:29.143259 AM

SQL> select task_id, policy_name, object_name, selected_for_execution, job_name
from dba_ilmevaluationdetails;

TASK_ID    POLICY_NAME          OBJECT_NAM SELECTED_FOR_EXECUTION
JOB_NAME
---------- -------------------- ---------- ---------------------------------------- --
------------------
364        P42                  EMP        SELECTED FOR EXECUTION
ILMJOB781

SQL> analyze table EMP compute statistics;

Table analyzed.

SQL> select compression, compress_for FROM dba_tables where table_name = 'EMP' and
owner='GD';

COMPRESS COMPRESS_FOR
-------- ------------------------------
ENABLED  ADVANCED
```

4. References

- https://docs.oracle.com/en/database/oracle/oracle-database/18/vldbg/manage-data-db-ilm.html#GUID-15F44B33-4C30-4B34-B733-FEF927282A24
- Information Lifecycle Management (ILM), Heat Map, Automatic Data Optimization (ADO) (문서 ID 1612385.1)
- Example for ILM ADO Storage Tiering Policy on Table Partition (문서 ID 1966394.1)
- Example for ILM ADO Storage Tiering Policy Using Custom PL/SQL Policy Function on Table Partition (문서 ID 1967038.1)