Pro CDB	Pro PDB
Eine Oracle-Software-Version	RMAN Point-in-time Recovery
Geplante RMAN-Backups	„Ad hoc"-RMAN-Backups
Data Guard	Möglichkeit für „Flush Shared_Pool"
Einige Parameter wie Character Set	Parameter, für die gilt „IsPDB_Modifiable = "TRUE""
Redo und Undo	

Tabelle 1

admin/noncdb_to_pdb.sql" wird nun der statische Anteil der (ehemaligen) Non-CDB endgültig entfernt, damit diese als PDB laufen kann.

Importieren eines Full Database Exports in eine leere PDB

Hierbei wird zunächst eine neue leere PDB in der 12*c*-CDB erzeugt – einschließlich des dazugehörigen Service, um diese überhaupt ansprechen zu können. Anschließend wird ein mittels „expdp … FULL=Y" erzeugter „Full Database Export" in die soeben erzeugte PDB importiert. Hierbei kann einem ein weiteres neues Feature von 12*c* behilflich sein: Data Pump unterstützt jetzt nämlich auch die Verwendung von Transportable Tablespaces, sodass das eigentliche Dump File sehr klein bleibt und schnell erzeugt ist. Hinweis: Ein Backport dieses Features ist auch in 11.2.0.3 eingeflossen, sodass es entsprechend auch für eine Migration einer 11*g*-Non-CDB in eine 12*c*-PDB verwendet werden kann.

CDB vs. PDB

Wie bereits eingangs erwähnt, verhält sich eine PDB nach außen hin grundsätzlich wie eine Non-CDB. Wegen der zugrunde liegenden Architektur mit geteilten Ressourcen aller PDBs innerhalb einer CDB existieren gleichzeitig aber auch Merkmale und Funktionen, die sich demzufolge nicht auf der Ebene einzelner PDBs abspielen. Tabelle 1 listet die wichtigsten auf.

Lizenzrechtlicher Hinweis

Wie schon aufgrund des Namens „Oracle Multitenant Option" zu erwarten ist, handelt es sich bei den in diesem Artikel vorgestellten Möglichkeiten um eine zusätzlich zur Datenbank zu lizenzierende Option. Weniger offensichtlich ist die Tatsache, dass dies für den Sonderfall von nur einer einzelnen PDB in einer CDB nicht gilt. Im Umkehrschluss bedeutet dies, dass auch ohne die entsprechende Option sämtliche 12*c*-Datenbanken mit einer einzelnen PDB betrieben werden können – auch in der Standard Edition.

Weitere Informationen

1. Umfangreiches (englischsprachiges) Whitepaper zu Oracle Multitenant: http://www.oracle.com/technetwork/database/multitenant-wp-12c-1949736.pdf
2. Aus der Reihe der deutschsprachigen „Oracle Dojos" die Ausgabe Nr. 7 zum Thema „Multitenant" – sowohl in gedruckter Form als auch online: http://www.oracle.com/webfolder/technetwork/de/community/dojo/index.html
3. Oracle Database Days: http://tinyurl.com/odd12c

Manuel Hoßfeld
manuel.hossfeld@oracle.com

Automatisierung des Information Lifecycle Managements

Ulrike Schwinn, ORACLE Deutschland B.V. & Co. KG

Was ist neu in Oracle Database 12*c* im Bereich „Information Lifecycle Management"? Was sind die sogenannten „Heat Maps"? Wie sieht die Automatisierung aus? Wie aufwändig ist das Setup? Dieser Artikel beantwortet diese und weitere Fragen und illustriert sie an einigen Beispielen.

Information Lifecycle Management (ILM) ist kein neues Schlagwort, sondern ein gängiger Begriff, um den Lebenszyklus von Daten zu beschreiben. Ziel ist es, Informationen entsprechend ihrem Wert und ihrer Nutzung optimal auf dem jeweils kostengünstigsten Speichermedium im besten Fall automatisch zu platzieren – so die Definitionen im Web.

Die Techniken zur Realisierung von ILM wie Partitionierung, Speicherplatz-Einsparung durch Komprimierung, Verlagerung von Daten und Virtual Private Database (VPD) für die unterschiedlichen Sichtweisen auf die Daten sind schon lange Bestandteil der Oracle-Datenbank und werden mit jedem Release weiterentwickelt. Allerdings musste eine Automatisierung bisher über eigene Programme oder spezielle Werkzeuge implementiert werden, nachdem man die Daten vorab manuell kategorisiert hat. Neu in Oracle Database 12*c* ist nun die vollständige Integration von ILM-Features in die Datenbank. Zur Implementierung sind keine zusätzlichen Werkzeuge oder Skripte erforderlich.

Wichtige Voraussetzung für den Einsatz dieser neuen Technologie ist die Lizenzierung der Advanced-Compression-Option. Im Wesentlichen handelt es sich dabei um zwei neue Features: Heat Map und automatische Daten-Optimierung (Automatic Data Optimization). Heat Map verfolgt Veränderungen und Abfragen auf Zeilen- und Segment-Ebene und gibt einen detaillierten Überblick über den Zugriff auf die Daten. Die automatische Daten-Optimierung verlagert und/oder komprimiert die Daten gemäß Nutzer-definierter Regeln (Policies), basierend auf den Informationen, die sich aus der Heat Map ergeben.

Wichtig zu wissen ist, dass in Release 1 noch einige Einschränkungen existieren, die unbedingt bei der Nutzung zu berücksichtigen sind. Die ILM-Funktionen stehen beispielsweise momentan nur in einer Nicht-Multitenant-Architektur (NON CDB) zur Verfügung.

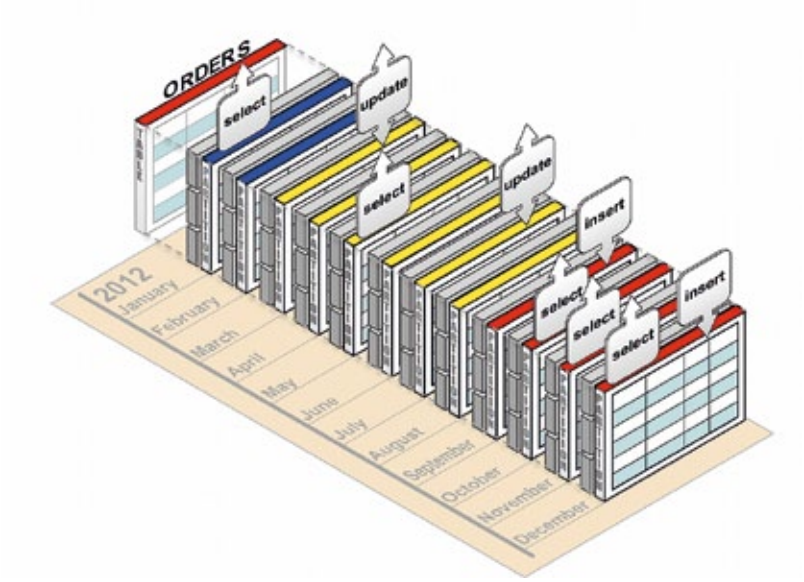

Abbildung 1: Heat Map – Zugriffe auf ein Segment

Automatische Klassifizierung von Daten

Bevor über eine Automatisierung der optimalen Datenablage nachgedacht werden kann, sind die Daten zu kategorisieren. Im Klartext bedeutet dies, dass sie je nach Zugriffsstatistik in unterschiedliche Kategorien eingeteilt werden – natürlich automatisch ohne Interaktion durch den Nutzer oder DBA. In Oracle Database 12*c* ist diese Funktion über das sogenannte „Heat-Map-Feature" implementiert. Heat Maps geben einen Überblick über die Aktivitäten auf den unterschiedlichen Objekten. Dabei wird nicht nur die Segment-Ebene berücksichtigt, sondern sogar die aktuellste Veränderung auf Block-Ebene dokumentiert (siehe Abbildung 1). Die Aktivitäten bestehen aus Lese- und Schreib-Operationen. Sogar Table beziehungsweise Index Lookups werden mitgeschrieben. Damit kein verfälschtes Bild entsteht, sind operative Eingriffe wie Statistik-Management, Verlagerungen etc. nicht vermerkt.

Eingeschaltet wird das Heat-Map-Feature über einen einzigen dynamischen Initialisierungs-Parameter „HEAT_MAP" („ON" bzw. „OFF"). Die Default-Einstellung ist „OFF": „SQL> ALTER SYSTEM set HEAT_MAP=ON;". Danach werden automatisch alle Zugriffe über einen speziellen In-Memory-Zugriff geloggt. Zugriffe auf Objekte im SYSTEM- und SYSAUX-Tablespace werden dabei ausgelassen.

Der Zugriff auf die Heat Map kann anschließend über Standard-Schnittstellen wie Data-Dictionary-Views, Fixed Tables oder PL/SQL-Packages erfolgen. Eine grafische Schnittstelle in Cloud Control 12*c* ist geplant (Stand August 2013). Die nachfolgenden Beispiele geben einen kleinen Einblick in die verschiedenartigen Verwendungsweisen. Einen ersten aktuellen Einblick liefert die View „V$HEAT_MAP_SEGMENT". Sie zeigt Realtime-Informationen über die Segment-Zugriffe. Listing 1 zeigt ei-

```
SELECT h.object_name, o.object_type, h.track_time, h.segment_write
write, h.segment_read read, h.full_scan, h.lookup_scan
FROM v$heat_map_segment h join dba_objects o
ON (o.object_id=h.obj#);

OBJECT_NAME       OBJECT_TYPE     TRACK_TIME          WRI REA FUL LOO
----------------- --------------- ------------------- --- --- --- --
PRODUCTS          TABLE           11.07.2013 15:19 YES NO  NO  NO
DEPARTMENTS       TABLE           11.07.2013 15:19 NO  NO  NO  NO
CONFIG$           TABLE           11.07.2013 15:19 NO  NO  NO  NO
TAB_300_300       INDEX           11.07.2013 15:19 NO  NO  NO  NO
TAB_300           TABLE           11.07.2013 15:19 NO  NO  YES NO
TAB_300_OLTP      TABLE           11.07.2013 15:19 NO  NO  NO  NO
TAB_300_OLTP      TABLE           11.07.2013 15:19 NO  NO  NO  NO
...
```

Listing 1

```
SELECT object_name, segment_write_time write, full_scan, lookup_scan FROM dba_heat_map_segment
WHERE owner ='SH' AND object_name!='SALES' ORDER BY 2,3;

OBJECT_NAME       WRITE            FULL_SCAN        LOOKUP_SCAN
----------------- ---------------- ---------------- -------------
TAB_300_OLTP      10.07.2013 17:21 10.07.2013 17:21
TAB_300           10.07.2013 17:21 11.07.2013 15:01
TABLE_300_SELECT  10.07.2013 18:22
PRODUCTS          11.07.2013 15:01 26.06.2013 12:30 25.06.2013 12:48
PRODUCTS_PK       11.07.2013 15:01                  11.07.2013 15:01
CUSTOMERS_PK                                        25.06.2013 22:48
…
```

Listing 2

```
SELECT tablespace_name, segment_count, allocated_bytes, min_writetime, max_writetime
FROM dba_heatmap_top_tablespaces;

TABLESPACE_NAME SEGMENT_COUNT ALLOCATED_BYTES MIN_WRITE MAX_WRITE
--------------- ------------- --------------- --------- ---------
ADOTEST                    16       135266304
EXAMPLE                   334       190775296 18-JUN-13 05-JUL-13
USERS                      20      5235408896 21-JUN-13 10-JUL-13
```

Listing 3

nen Auszug zur Track-Zeit „15:19 Uhr am 11. Juli“. Die Objekte, deren Spalten den Wert „YES“ beinhalten, sind aktuell im Zugriff.

Mit „ALL_, DBA_“ und „USER_HEAT_MAP_SEGMENT“ kann man den letzten Zugriff auf die Segmente anzeigen. Folgendes Code-Beispiel listet die Zugriffe auf einige der Objekte des Users „SH“ auf. Wie zu erkennen ist, sind Lese- und Schreib-Zugriffe dokumentiert und darüber hinaus die Full Table Scans beziehungsweise die zugehörigen Index Scans aufgelistet (siehe Listing 2).

Aus Gründen der Übersichtlichkeit ist die Spalte „SUBOBJECT_NAME“ nicht selektiert. Im Fall von Zugriffen auf partitionierte Tabellen wie zum Beispiel bei der Tabelle „SALES“ ist dies natürlich zur genauen Segment-Analyse unbedingt erforderlich.

Interessiert man sich nur für die Objekte und Tablespaces, die am häufigsten im Zugriff waren, kann man die Views „DBA_HEATMAP_TOP_OBJECTS“ beziehungsweise „DBA_HEATMAP_TOP_TABLESPACES“ verwenden. In Listing 3 sind die drei Top-Tablespaces unter Angabe der jeweiligen Objektanzahl und des verwendeten Speicherplatzes aufgelistet.

So wird beispielsweise der erste beziehungsweise letzte Schreibzugriff auf Objekte im entsprechenden Tablespace mitgeschrieben. Die Funktion „BLOCK_HEAT_MAP“ des Package „DBMS_HEAT_MAP“ listet die zuletzt erfolgten Schreibzugriffe sogar bis auf Block-Ebene genau auf. Listing 4 zeigt, welche Blöcke der Tabelle „TAB_300“ des Users „COMP_TEST“ zuletzt geändert worden sind.

```
SELECT tablespace_name, relative_fno fno, block_id, to_
char(writetime, 'MM/DD/YY HH24:MI.SS' )
FROM TABLE(dbms_heat_map.block_heat_map('COMP_TEST','TAB_300'));

       FNO   BLOCK_ID TO_CHAR(WRITETIME
---------- ---------- -----------------
         5        251 07/11/13 16:20.33
         5        252 07/11/13 16:20.33
         5        253 07/11/13 16:20.33
         5        254 07/11/13 16:20.33
         5        255 07/11/13 16:20.33
         5       6656 07/11/13 16:20.33
```

Listing 4

Automatische Daten-Optimierung

Um zu erklären, was mit automatischer Daten-Optimierung (Automatic Data Optimization) gemeint ist, skizziert folgendes Beispiel einen möglichen Anwendungsfall (siehe Abbildung 2). Zuerst werden Daten mit Bulk-Load-Operationen oder anderen konventionellen Methoden in die Ta-

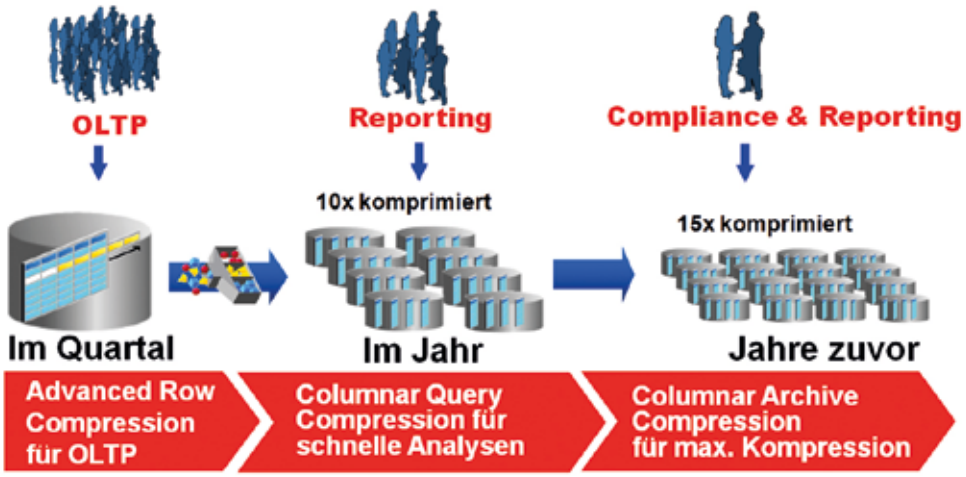

Abbildung 2: Beispiel für automatische Daten-Optimierung

bellen geladen und unterliegen dabei starken Veränderungen. Zu diesem Zeitpunkt ist eine Komprimierung noch nicht erwünscht, somit werden die Daten unkomprimiert gespeichert. Nach einer gewissen Zeit erfolgen nur noch vereinzelte Veränderungen über OLTP-Transaktionen. Nun kann es sinnvoll sein, die Daten aus Platzgründen in das OLTP-Komprimierungsformat zu konvertieren. Nach einiger Zeit werden die Daten nur noch selten genutzt; nun könnten sie auf ein anderes Speichermedium ausgelagert werden – vielleicht sogar auf ein Speichermedium wie ZFS oder Pillar, die eine höhere Komprimierungsrate durch HCC-Komprimierung erlauben, oder einfach nur auf ein beliebiges anderes Low-Cost-Storage-Medium.

```
ALTER TABLE sh.sales
ILM ADD POLICY ROW STORE COMPRESS
ADVANCED SEGMENT
AFTER 20 DAY OF NO MODIFICATION;
```

Listing 5

```
ALTER TABLE sh.customers
ILM ADD POLICY ROW STORE COMPRESS
ADVANCED ROW
AFTER 1 DAY OF NO MODIFICATION;
```

Listing 6

```
ALTER TABLE sh.sales
ILM ADD POLICY TIER TO adotest;
```

Listing 7

```
SELECT SUBSTR(name,1,32) name,
value
FROM dba_ilmparameters WHERE name
LIKE ‚TBS%‘;

NAME                          VALUE
-------------------- ----------
TBS PERCENT USED              70
TBS PERCENT FREE              30
```

Listing 8

Dieses Szenario lässt sich ab Oracle Database 12*c* einfach in der Datenbank abbilden und kann sogar in automatisierter Form ablaufen. Wichtiges Hilfsmittel zur Implementierung von automatischer Daten-Optimierung sind die sogenannten „Policies“. Eine Policy-Spezifikation beinhaltet dabei eine Aktion wie zum Beispiel Compression oder Storage Tiering und zusätzlich eine Bedingung, unter der die Aktion ausgeführt werden soll. Policies können auf Segment- oder Row-Ebene implementiert sein. Listing 5 zeigt eine Segment-Level-Policy (Schlüsselwort „SEGMENT“), die automatisch eine Tabelle in das OLTP-Komprimierungsformat umwandelt, nachdem zwanzig Tage lang keine Veränderungen (Schlüsselwort „NO MODIFICATION“) erfolgt sind. „ROW STORE COMPRESS ADVANCED“ ist dabei die neu eingeführte Syntax für OLTP-Compression in 12*c*.

Das nächste Beispiel zeigt eine „ROW“-Policy. Oracle evaluiert in regelmäßigen Abständen die Blöcke der „ORDERS“-Tabelle. Jeder Block, der den Anforderungen – in diesem Fall „1 DAY OF NO MODIFICATION“ – genügt, wird komprimiert, um wieder ausreichend Platz zu schaffen (siehe Listing 6).

Diese Policy erlaubt höchst mögliche Performance beim Laden von Daten und zusätzlich die Vorteile der Platzersparnis durch die Komprimierung. Im Gegensatz zum Segment-Policy-Beispiel muss nicht darauf gewartet werden, bis eine ganze Partition den Anforderungen genügt.

Nun stellt sich die Frage, wann diese Policies evaluiert werden. Da die Aktionen automatisch im Hintergrund ablaufen sollen, bietet sich das Maintenance Window des „Default Maintenance“-Plans zur Evaluierung und Ausführung an. Möchte der DBA eingreifen, ist dies natürlich auch über die entsprechenden Kommandos möglich.

Zusätzlich zu den gerade illustrierten Aktionen über Komprimierung gibt es auch die Möglichkeit der Verlagerung auf einen anderen Storage (Storage Tiering). Die Implementierung ist einfach, wie Listing 7 zeigt.

Diese Art von Policy wird durch den Füllgrad des entsprechenden Tablespace getriggert. Ist der Quell-Tablespace (hier „USERS“) annähernd voll, werden die Daten in den Tablespace „ADOTEST“ verlagert. Der DBA kann dabei die Eigenschaft „voll“ beeinflussen. Im Beispiel ist der Tablespace bei 70 Prozent voll, wie die eingestellten Parameter zeigen (siehe Listing 8).

Policies können natürlich ein- und ausgeschaltet beziehungsweise auch gelöscht werden. Ein Monitoring ist wie beim Heat-Map-Feature über die entsprechenden Views wie „DBA_ILMDATAMOVEMENTPOLICIES“ und „DBA_ILMOBJECTS“ möglich. Das abschließende Beispiel zeigt einen Ausschnitt aus den im Artikel genutzten Policies (siehe Listing 9).

Fazit

Die neuen ILM-Features ermöglichen eine automatische Datenkomprimierung beziehungsweise automatisierte

Verlagerung der Daten auf ein anderes Speichermedium. Wichtige Voraussetzung ist dabei die Nutzung beziehungsweise das Einschalten des Heat-Map-Features auf System-Ebene. Die Operationen erfolgen dabei im Hintergrund und können im Fall von Partitionen sogar vollständig online ausgeführt werden.

Der Artikel hat nur einen kleinen Ausschnitt der Funktionalität demonstriert. So ist es zum Beispiel auch möglich, eigene Funktionen zu definieren, um den Zeitpunkt des Storage Tiering oder der Komprimierung festzulegen. Wer mehr darüber erfahren möchte, kann Handbücher, Blogs und White Paper konsultieren.

```
OBJECT_NAME   SUBOBJECT_NAME  ACTION_TYPE SCOPE   TIER_TABLESPACE
------------ --------------- ----------- ------- ---------------
CONDITION_TYPE         DAYS
---------------------- ----
CUSTOMERS                    COMPRESSION ROW
LAST MODIFICATION TIME    1
SALES         SALES_1995     STORAGE     SEGMENT ADOTEST
                           0
SALES         SALES_1996     STORAGE     SEGMENT ADOTEST
                           0
SALES         SALES_H1_1997  STORAGE     SEGMENT ADOTEST
                           0
```

Listing 9

Weitere Informationen

1. Tipps in der DBA Community: http://blogs.oracle.com/dbacommunity_deutsch
2. VLDB and Partitioning Guide
3. SQL Language Reference (Syntax wie „ALTER TABLE")
4. PL/SQL-Packages and Types-Reference
5. Automatic Data Optimization with Oracle Database 12c
6. Deutschsprachige Veranstaltungen zum Thema: tinyurl.com/odd12c

Ulrike Schwinn
ulrike.schwinn@oracle.com