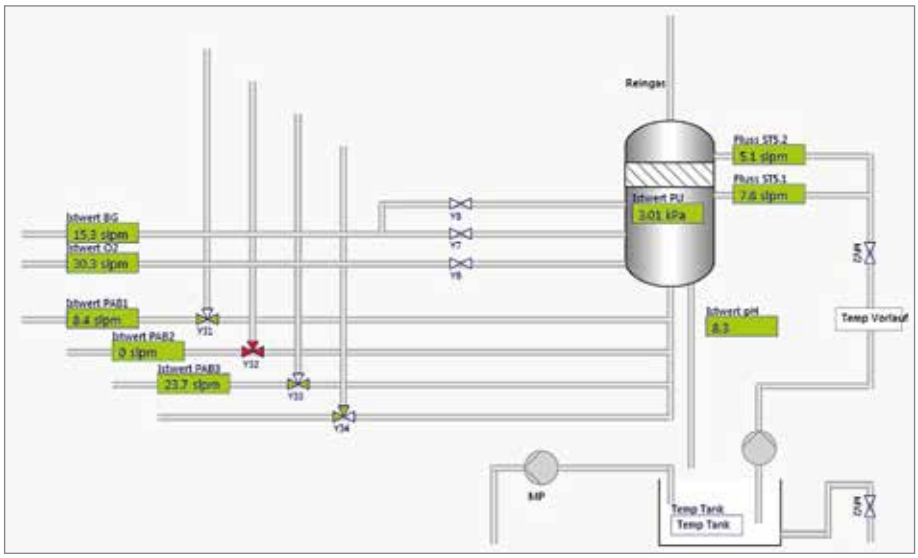

Abbildung 9: Prozessvisualisierung mit SVG

Grafiken waren keine Performance-Probleme bei der Darstellung in verschiedensten Browsern und Endgeräten feststellbar. Wer Interesse hat, findet über das Projekt unter *„https://tfmportable.ais-automation.com/apex/f?p=400:700"* eine Live-Demo.

Frank Schubert
frank.schubert@ais-automation.com

Angela Wobar
angela.wobar@ais-automation.com

Heatmap und ADO in der Praxis

Florian Feicht, Trivadis GmbH

Aufgrund der rasant wachsenden Datenmengen gewinnt der Lebenszyklus der Daten in der Datenbank immer mehr an Bedeutung. Müssen die Archivdaten auf teurem SAN liegen? Können weniger genutzte Daten komprimiert werden? Gibt es einfachere Möglichkeiten? Die Oracle-Antwort in der Version 12*c* ist ein in die Datenbank integriertes Information-Lifecycle-Management-System (ILM). Dazu wurden die Features Heat Map und Automatic Data Optimization (ADO) eingeführt.

Seit der Version 9*i* bietet die Oracle-Datenbank die Möglichkeit, Daten zu komprimieren. Dadurch benötigen diese im Idealfall weniger Speicherplatz und der Lesezugriff wird beschleunigt. Leider ist es aufgrund des Performance-Overheads nicht möglich, Komprimierung für die gesamte Datenbank auf Knopfdruck zu aktivieren. Die Entscheidung darüber, wo Komprimierung sinnvoll ist, erfordert gute Kenntnisse der Applikation. Dazu musste bisher eine aufwändige Auswertung erstellt werden. Außerdem ist es oft nicht sinnvoll, ganze Tabellen zu komprimieren, sondern nur die selten benutzten Einträge. Eine Methode, verschiedene Storage-Klassen am Datenbankserver automatisiert zu verwalten, stand nicht zur Verfügung.

12c bringt die Übersicht

Mit Einführung der Version 12c wurde ein Information Lifecycle Management implementiert. Eine automatisierte Verwaltung der Daten durch ihren „Lebenszyklus" ist dadurch sehr einfach möglich. Um den aktuellen Lifecycle-Stand zu überwachen, wird die Heat Map genutzt *(siehe Abbildung 1)*. Mit diesem Feature werden die Zugriffe auf die Datenbankobjekte zentral protokolliert. Dadurch entsteht eine feingranulare Übersicht auf folgende Ebenen:

- Block
- Extent

- Segment
- Objekt
- Tablespace

Schreib- und Lesezugriffe werden separat protokolliert. Somit entsteht eine Karte der am häufigsten verwendeten Objekte. Im Zusammenhang mit der Heat Map wird von Daten gesprochen, die „hot", „warm" und „cold" sind. In der View „V$HEAT_MAP_SEGMENT" *(siehe Listing 1)* ist dies online im Speicher protokolliert. Die Informationen werden periodisch in die Datenbank geschrieben. Der Zugriff erfolgt über die entsprechenden „DBA/USER/ALL"-Views. Hier sind unter anderem „DBA_HEATMAP_TOP_OBJECTS" und „DBA_HEAT_MAP_TOP_TABLESPACES" hilfreich. Gespeichert sind die Daten der Heat Map im „SYSAUX"-Tablespace.

Zusätzlich steht das Package „DBMS_HEAT_MAP" zur Verfügung, mit dem verschiedene Auswertungen erstellt werden können *(siehe Listing 2)*. Zu beachten ist, dass Objekte im „SYSTEM"- und „SYSAUX"-Tablespace von der Überwachung ausgenommen sind. Außerdem sind interne Systemjobs, DDLs, Redefinitions und die Sammlung der Statistiken nicht in der Heat Map protokolliert.

Die Heat Map aktivieren

Die Heat Map ist ein fest im Kernel integriertes Feature. Sie wird mit dem Befehl „ALTER SYSTEM SET heat_map=ON;" aktiviert. Der Parameter kann auf System- oder auf Session-Ebene gesetzt sein, wodurch eine einfache Deaktivierung für Wartungsarbeiten ermöglicht wird. Durch Aktivierung der Heat Map entsteht ein Overhead abhängig von der Gesamtlast des Systems. Dies muss je nach Applikation genau untersucht werden.

Um die gesammelten Daten sinnvoll verwenden zu können, kommt das nächste Feature zum Einsatz, die Automatic Data Optimization (ADO). Vereinfacht gesprochen, werden hier ADO-Regeln (Policies) auf die Heat Map gesetzt, die mit entsprechenden Aktionen verbunden werden können. Grundsätzlich gibt es zwei Möglichkeiten:

- Storage Tiering
- Compression Tiering

Storage Tiering ermöglicht die Verwaltung mehrerer Storage-Klassen in der Datenbank. Diese werden über verschiedene Tablespaces, die auf unterschiedlichen physikalischem Storage liegen, abgebildet. Mit „ALTER TABLE config_mgr ILM ADD POLICY TIER TO tbs_nas_storage;" lässt sich die Tiering Policy auf Tabellen- beziehungsweise auf Partitions-Ebene aktivieren.

Das Keyword „TIER TO" definiert die Storage-Policy. So wird die Tabelle „config_mgr" beziehungsweise das Segment „config_mgr" nach einer definierbaren Belegung des momentanen Tablespace verschoben. Es wird nur die Tabelle verschoben, die Indizes bleiben unberührt (siehe MOS Doc ID 2025280.1).

Im Hintergrund wird das betroffene Segment mit „MOVE" bewegt. Dadurch entsteht ein exklusiver Lock. Falls dieser Lock wegen blockierender Sessions nicht erreicht werden kann, wird die Policy nicht ausgeführt. Die Ausführung wird im nächsten Wartungsfenster beziehungsweise bei der nächsten manuellen Ausführung wiederholt.

Am häufigsten ist die Anwendung von Storage Tiering bei partitionierten Tabellen. Dabei erfolgt ein „MOVE" auf Partitionsebene. In der Version 12*c* wurde für Partitionen die Möglichkeit „MOVE ONLINE" eingeführt, die hier nicht genutzt wird. Leider ist kein Tiering nach Zugriffzeit der Segmente auf Basis der Heat-Map-Daten vorgesehen. Dies kann allerdings über sogenannte „Custom" Policies erfolgen. Die Syntax ist dazu um „ON FUNKTIONSNAME" zu erweitern. Jede Funktion, die als Rückgabewerte einen „BOOLEAN"-Datentyp hat, kann genutzt werden. Sobald die Policy einmal erfolgreich etabliert ist, wird sie automatisch deaktiviert. Für eine weitere Ausführung ist eine manuelle Reaktivierung mithilfe von „ENABLE POLICY" erforderlich.

Die zweite Möglichkeit der Policy-Definition ist das Compression Tiering. Dabei lassen sich Policies auf Zeilen-, Segment- oder Group-Ebene definieren. Diese führen eine Komprimierung auf Basis der Zeitstempel der letzten Zugriffe durch. Hier kann sowohl der Kompressions-Algorithmus als auch die Auswahl der zu bewertenden Heat-Map-Daten beeinflusst werden.

Dazu ein Syntax-Beispiel für Advanced Row/Low Securefile Compression auf Block-Ebene: „ALTER TABLE config_mgr ILM ADD POLICY ROW STORE COMPRESS ADVANCED SEGMENT AFTER 30 DAYS OF NO MODIFICATION;". Die Syntax sieht hier zwar „ROW" vor, im Hintergrund wird allerdings jeweils der ganze Block geprüft. Das bedeutet, dass der gesamte Block dem ausgewählten Kriterium entsprechen muss.

Stärkere Kompressions-Algorithmen setzen eine Oracle-Hardware voraus. „COMPRESS FOR QUERY" und „COMPRESS FOR ARCHIVE" aktivieren die Hybrid Columnar Compression sowie „MEDIUM"- beziehungsweise „HIGH"-SecureFile-Kompression. Jedes Kompressionsverfahren komprimiert die Indizes zusätzlich mit der Standard-Komprimierung. Für die Betrachtung der Zugriffszeit der Heat-Map-Daten stehen „NO ACCESS", „NO MODIFICATION", „CREATION" und „LOW ACCESS" zur Verfügung. Bei Compression Tiering auf Segment-Basis kann analog zum Storage Tiering die „ON"-Klausel mit eigener Funktion verwendet werden.

Damit die Ausführung der Policy erfolgt, sind gesammelte Statistiken für die involvierten Objekte notwendig. Der Umfang einer Group ist derzeit nicht dokumentiert. Die Policy auf Segment-Ebene führt im Wartungsfenster „MOVE COMPRESS FOR OLTP" durch. Dafür muss genug freier Platz

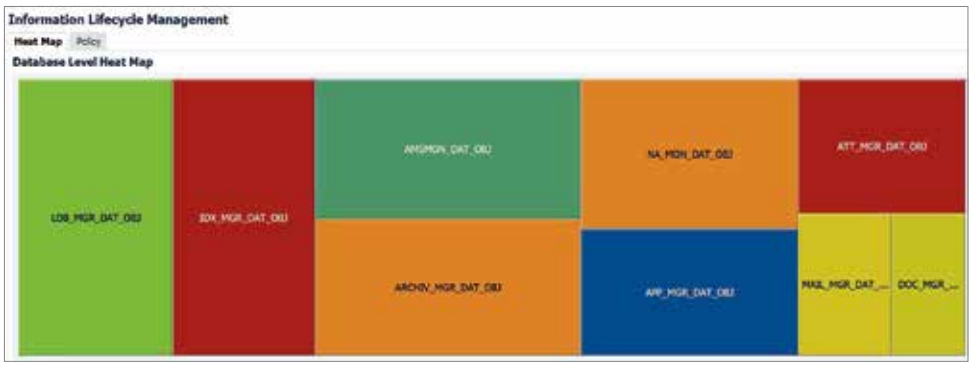

Abbildung 1: Heat Map im Enterprise Manager

im Tablespace vorhanden sein. Außerdem erfolgt wieder ein exklusiver Lock.

Bei der zeilenbasierten Variante werden aus der Heat Map die zu komprimierenden ROWIDs generiert. Anschließend wird mit „SELECT ROWNUM FROM config_mgr FOR COMPRESS 2" über die Session-Statistik „HSC OLTP Space Saving" das Einspar-Potenzial ermittelt. Leider ist nicht erkennbar, auf welche Art die einzelnen Rows komprimiert werden.

Diese Policies bieten den großen Vorteil, dass die jeweilige Applikation nicht angepasst werden muss. Storage und Compression Tiering kann transparent für die Applikation und gezielt auf dafür geeigneten Objekten erfolgen. Somit lassen sich die Auswirkungen auf die Performance deutlich verringern und die Komprimierung wirkungsvoller einsetzen. Row-basierte Policies erfolgen alle fünfzehn Minuten, Segment-basierte im Wartungsfenster der Datenbank. Alternativ steht mit „DBMS_ILM_ADMIN" ein Package zur Administrator-gesteuerten Ausführung der Policies zur Verfügung. Damit lassen sich auch weitere Parameter konfigurieren. Aufeinander aufbauende Policies sind problemlos möglich. Die Ausführung der Policies kann über die entsprechenden „DBA/USER/ALL"- Views *(siehe Listing 3)* überwacht werden.

Einschränkungen von Heat Map und ADO

Für die in diesem Artikel beschriebenen Features ist die Enterprise Edition und zusätzlich die „Advanced Compression"-Option erforderlich. Leider ist die Heat Map und damit auch ADO nur in Non-CDB-Datenbanken nutzbar. Darüber hinaus gibt es Einschränkungen für bestimme Datentypen. Die ADO-Policies funktionieren nur in eine Richtung. Demzufolge gibt es keine Dekomprimierung oder ein Zurückschieben auf den ursprünglichen Tablespace.

Der zweite Fall lässt sich über den Umweg einer Custom-Funktion mit entsprechender Heat-Map-Auswertung in der Storage-Tiering-Policy realisieren. Bei deren Einsatz ist vorab zu prüfen, ob im Ziel-Tablespace ausreichend Platz vorhanden ist – eine automatische Prüfung erfolgt nicht. Zudem ist zu beachten, dass die automatische Ausführung der ADO-Policies im Wartungsfenster der Datenbank nicht funktioniert, wenn „Supplemental Logging" aktiviert ist. Als Alternative bleibt die manuelle Ausführung. Nach zweimaliger fehlerhafter Ausführung werden die Policies automatisch deaktiviert.

Anwendungsfälle aus der Praxis

Im ersten Beispiel (Dokumentenverwaltung) legt die Applikation Dokumente und Metadaten in der Datenbank ab. Auf Kundenwunsch sollen alle Blöcke, die seit drei Monaten nicht geändert worden sind, automatisch komprimiert werden. Dafür kommen die Advanced Row Compression für die Heat-Tabellen und Low SecureFile Compression zum Einsatz. Um die Anforderung umzusetzen, wird nach Aktivierung der Heat Map auf allen Tabellen mit „ALTER TABLE document_mgr ILM ADD POLICY ROW STORE COMPRESS ADVANCED ROW AFTER 3 MONTH OF NO MODIFICATION;" eine Automatic-Data-Optimization-Policy erstellt. Diese Policy komprimiert nur die Blöcke, die dem geforderten Zeitfenster entsprechen, der Rest bleibt unberührt. Dadurch minimiert sich das Risiko von „Chained Rows" bei Updates auf „hot"-Daten deutlich.

Verschiedene Performance-Tests haben ergeben, dass der Overhead durch Aktivieren der Heat Map bei maximal zehn Prozent liegt. Aufgrund der durch Komprimierung deutlich kleineren Segmente konnte der Gesamt-Overhead aber sehr gering gehalten werden, um deutlich Speicherplatz zu sparen. Eine Monitoring-Lösung für die Überwachung der Policy-Ausführung ist unbedingt erforderlich *(siehe Listing 4)*.

Im zweiten Beispiel stehen im Unternehmen verschiedene Storage-Klassen zur Verfügung. Durch die interne Verrechnung der Storage-Belegung soll nach Anforderung der Datenbank-Abteilung, wo immer mög-

```
SELECT object_name
      ,track_time
      ,segment_write AS seg_w
      ,segment_read AS seg_r
      ,full_scan
      ,lookup_scan
  FROM gv$heat_map_segment;

OBJECT_NAME TRACK_TIME          SEG_W   SEG_R   FUL  LOO
----------- ------------------- -----   -----   ---  ---
CONF_OBJ_C1 09.05.2015 20:15:38 YES     NO      YES  NO
CONF_SERVIC 09.05.2015 20:15:38 YES     NO      NO   YES
```

Listing 1

```
SELECT owner
      ,segment_name
      ,block_id
      ,writetime
  FROM TABLE(
         dbms_heat_map.block_heat_map(
             owner=>'DOCMGR'
                    ,segment_name=>'CONFIG_MGR')
         );
```

Listing 2

```
SELECT policy_name
      ,policy_type
  FROM dba_ilmpolicies;

POLICY_NAME POLICY_TYPE
-----------  -------------
P443         DATA MOVEMENT
P445         DATA MOVEMENT
```

Listing 3

lich und sinnvoll, günstiger – wenn auch langsamer – Storage verwendet werden. Dabei sind verschiedene Varianten denkbar: zum einem eine Standard-Storage-Tiering-Policy, die Segmente nach Belegung der Tablespaces verschiebt, und zum anderen eine „ON FUNCTION"- Storage-Tiering-Policy, die anhand der Daten aus der Heat Map über die Storage-Klasse entscheidet.

Aufgrund der detaillierten Auswertungsmöglichkeit wird die zweite Variante ausgewählt. Das erstellte Konzept sieht vor, dass alle Daten, auf die seit vier Wochen nicht zugegriffen wurde, im günstigeren Storage gespeichert werden können. Dazu definiert man zunächst High-End- und Low-End-Tablespaces. Zur Auswertung der Heat-Map-Daten wird eine Funktion erstellt *(siehe Listing 5)*. Sie wird mit „ALTER TABLE concept ILM ADD POLICY TIER TO low_end_tbs ON not_accessed;" in die Applikations-Tabellen per Policy eingebunden.

```
SELECT task_id
      ,job_name
      ,job_state
  FROM dba_ilmresults;

TASK_ID JOB_NAME         JOB_STATE
-------   -------------  ------------------------
854       ILMJOB2760     COMPLETED SUCCESSFULLY
853       ILMJOB2752     COMPLETED SUCCESSFULLY
```

Listing 4

```
CREATE FUNCTION not_accessed(objn IN NUMBER)
RETURN BOOLEAN
IS
  days NUMBER:=0;
BEGIN
  SELECT sysdate - segment_read_time
      INTO days
      FROM dba_heat_map_segment
      WHERE object_name = (SELECT object_name FROM user_objects WHERE
object_id = objn);

  IF (days > 28)
  THEN
      RETURN true;
  ELSE
      RETURN false;
  END IF;
END;
/
```

Listing 5

```
SELECT e.object_owner
      ,e.object_name
      ,r.job_state
      ,h.full_scan
  FROM dba_ilmresults r
      ,dba_ilmevaluationdetails e
      ,dba_heat_map_segment h
      ,dba_segments s
 WHERE r.task_id = e.task_id
  AND e.object_owner = h.owner
  AND e.object_name = h.object_name
  AND e.object_owner = s.owner
  AND e.object_name = s.segment_name
  AND r.job_state = ,COMPLETED SUCCESSFULLY'
  AND s.tablespace_name = ,LOW_END_TBS';
```

Listing 6

Um tatsächliche Platzeinsparungen zu ermöglichen, laufen regelmäßig Jobs, die „RESIZE DATAFILE" auf dem High-End-Storage durchführen. Zusätzlich ist auf der Low-End-Seite zu beachten, dass genug freier Speicherplatz in den Tablespaces zur Verfügung steht. Ein weiterer wichtiger Punkt des Konzepts ist das Monitoring der Heat-Map-Views in Bezug auf alle bereits verschobenen Objekte. Da die Storage-Tiering-Policy kein erneutes Verschieben der Daten ermöglicht, muss manuell geprüft werden, ob die Segmente wieder „hot" sind. Diese Überwachung erfolgt mithilfe der kumulierten Ergebnisse aus den Heat-Map- und ILM-Views *(siehe Listing 6)*.

Fazit

Heat Map und Automatic Data Optimization erleichtern den Einsatz der Advanced-Compression-Features in der Oracle-Datenbank deutlich. Die Aktivierung der Heat Map gibt ein klares Bild über die Nutzungshäufigkeit der einzelnen Objekte in verschiedenen Detailstufen. Entsprechende Automatic-Data-Optimization-Policies nutzen die gesammelten Daten automatisiert zu konkreten Aktionen. Der Datenbank-Administrator hat damit die Möglichkeit, die Auswirkungen auf die Zugriffszeiten gering zu halten und gleichzeitig, wo es sinnvoll ist, Storage-Einsparungen zu erreichen.

Die Verwendung von Storage Tiering kann verschiedene Storage-Klassen am Datenbankserver einfach verwalten. Für den Einsatz der Automatic Data Optimization ist anforderungsspezifisch ein Designkonzept erforderlich. Dabei ist das Monitoring unbedingt zu berücksichtigen. Zu bedenken ist allerdings, dass die neuen Features Heat Map und ADO für die Multitenant-Architektur noch nicht freigegeben sind.

Florian Feicht
florian.feicht@trivadis.com