Database Resource Manager – Stand der Dinge

Ulrike Schwinn
Oracle Deutschland B.V. & Co.KG
München

Schlüsselworte

Resource Plan, Resource Direktive, Resource Consumer Group, Instance Caging, SQL Monitoring, IORM, PDB Resource Plan, CDB Resource Plan, Monitoring

Einleitung
Ressourcen in der Datenbank mit dem Database Resource Manager zu verwalten, ist schon seit der Version 8*i* mit der Enterprise Edition der Datenbank möglich. Der Resource Manager kann helfen, Datenbank Ressourcen Engpässe zu lösen, die beispielsweise mit hoher CPU Last in Zusammenhang stehen. Zu Beginn war die Resource Manager Funktion der Datenbank noch wenig bekannt und fand nur spärlichen Einsatz. Gründe dafür lagen sicherlich in der eingeschränkten Verwendbarkeit, die ausschließlich über PL/SQL Packages und für dedizierte Datenbankuser-Sessions möglich war.

Spätestens mit der Oracle Database Version 10*g* und der Verfügbarkeit über die graphische Oberfläche des Enterprise Managers und die Erweiterung der Nutzung über Services, Module, Programme usw. sind diese Gründe aufgehoben. Damit können zum Beispiel CPU-Ressourcen für verschiedene Applikationen priorisiert werden, langlaufende Operationen unterbunden oder die Anzahl der Sessions begrenzt werden, um nur einige Beispiele zu nennen.

Mittlerweile verwenden immer mehr Kunden den Resource Manager und haben über die Resource Manager Funktionen eine zusätzliche wichtige Ebene der Administration kennen und schätzen gelernt. Nach einer Einarbeitungsphase, die durch die Verwendung von vorhandenen Skripten verkürzt werden kann und einer Testphase des Setups können durch den Einsatz eines entsprechenden Ressource Management Plans drängende Fragen der Administration gelöst werden. Nur noch ein einfaches Monitoring ist von Zeit zu Zeit erforderlich. Auch intern findet das Ressourcen-Management über den Database Resource Manager schon seit Oracle Database 10*g* seine Anwendung. Beispiele dafür sind die "Automated Maintenance Tasks" in 11*g* oder die Aufgabe "Manage Optimizer Statistics" in 10*g*.

Generelles
Um einen schnellen Einstieg in das Thema zu gewährleisten, werden im folgenden Abschnitt ein paar grundsätzliche Begriffe wie Resource Consumer Group, Resource Consumer Mapping und Resource Plan erläutert. Wer schon vertraut damit ist, kann zum nächsten Abschnitt übergehen.

Im ersten Schritt werden Benutzergruppen nach unterschiedlichen Ressource-Anforderungsprofilen erzeugt; sie werden als **Resource Consumer Group** bezeichnet. Resource Consumer Groups sind also Datenbankuser-Sessions, die als eine Einheit behandelt werden. Defaultmäßig existieren schon einige Oracle vordefinierte Resource Consumer Groups, wie zum Beispiel die sogenannte SYS_GROUP eine Gruppe, die aus den Usern SYS und SYSTEM besteht; OTHER_GROUPS hingegen besteht aus allen anderen Usern, die keine Zuweisung zu einer Gruppe erhalten haben. Möchte man über die bestehenden Consumer Groups hinaus, eigene Gruppen definieren, kann man folgendes Kommando verwenden.

```
execute dbms_resource_manager.create_consumer_group(
                        consumer_group => 'HPRIO_GR',
                        comment        => 'Sessions for HPRIO_GR');
```

Datenbankuser-Sessions können automatisch über Consumer Group **Mapping Rules** einer gewissen Consumer Group zugeordnet werden. Die Mapping Rules enthalten Regeln bzw. Eigenschaften, nach denen die einzelnen Sessions den Gruppen initial zugeordnet werden. Ändern sich die Eigenschaften einer Session oder tritt eine Überschreitung einzelner Ressourcengrenzen auf, kann dies zu einem automatischen Wechsel in eine andere Gruppe führen. Darüber hinaus kann ein Gruppenwechsel natürlich auch manuell angestossen werden. Die folgende Liste zeigt die wichtigsten Mapping Rules, die verwendet werden können:

- Oracle Datenbank User (dbms_resource_manager.oracle_user)
- TNS Servicename oder Oracle definierte Services wie SYS$USERS (dbms_resource_manager.service_name)
- MODULE und ACTION (siehe DBMS_APPLICATION_INFO) (dbms_resource_manager.module_name etc.)
- Client OS User (dbms_resource_manager.client_os_user)
- Client Program (z.B. SQL Developer) (dbms_resource_manager.client_program)
- Client Maschine (dbms_resource_manager.client_machine)

Damit es zu keinen Überlagerungen der einzelnen Regeln kommt, werden diese in einem separaten Schritt priorisiert. Die folgende Prozedur bildet die Sessions des Service S_SERVICE1 auf die Consumer Group HP_GROUP ab.

```
execute dbms_resource_manager.set_consumer_group_mapping(
         attribute      => dbms_resource_manager.service_name,
         value          => 'S_SERVICE1',
         consumer_group => 'HP_GROUP');
```

Für den Parameter `attribute` können die in den Mapping Rules angegebenen Konstanten wie `dbms_resource_manager.oracle_user`, `dbms_resource_manager.service_name` usw. verwendet werden. Möchte man beispielsweise erreichen, dass der APPDBA User der SYS_GROUP Gruppe angehört, ist folgendes Kommando erforderlich.

```
execute dbms_resource_manager.set_consumer_group_mapping(
    attribute      => dbms_resource_manager.oracle_user,
    value          => 'APPDBA',
    consumer_group => 'SYS_GROUP');
```

Damit ein User oder Rolle in eine Consumer Group wechseln kann, ist ein zusätzlicher Prozeduraufruf erforderlich. Mit der Verwendung von PUBLIC wird in folgendem Beispiel allen Usern erlaubt in die Gruppe HPRIO_GR zu wechseln.

```
execute dbms_resource_manager_privs.grant_switch_consumer_group(
          grantee_name   => 'public',
          consumer_group => 'HPRIO_GR',
          grant_option   => FALSE);
```

Nun kommen wir zum Kern des Ressourcen Managements – dem **Resource Manager Plan**: Ressource Pläne enthalten die Regeln (auch **Direktiven** genannt), nach denen die Ressourcen an die Resource Consumer Groups aufgeteilt werden. Ressource Pläne sind dynamisch und können mit einem einfachen Befehl aktiviert bzw. deaktiviert werden. Es gilt allerdings immer nur ein Plan zu einer Zeit. Ressource Pläne können aus einem einzigen Plan – also einer Ebene (auch Single Level Plan) – oder auch aus mehreren Subplänen – also mehreren Ebenen (auch Multi Level Plan) – bestehen. Wie bei den Consumer Groups gibt es auch hier vordefinierte Ressource Pläne wie z.B. den

Plan DEFAULT_MAINTENANCE_PLAN, der für die oben schon erwähnten Maintenance Aufgaben eingesetzt wird.

Ein Ausschnitt aus einem selbst definierten Ressource Plan (oder Subplan) könnte dann folgendermassen aussehen. Die Prozeduren `clear_`, `create_`, `validate_` und `submit_pending_area` verwalten dabei einen temporären Arbeitsbereich für die Dauer der Ressource Management Konfiguration. Die Änderungen sind solange nicht sichtbar/wirksam bis die Pending Area mit Submit abgeschlossen worden ist.

Hinweis: Die Verwendung eines temporären Arbeitsbereichs ist bei der Erzeugung von Consumer Groups oder Ressource Plänen obligatorisch! Aus Gründen der Übersichtlichkeit wurde diese Information in den übrigen Beispielen weggelassen.

```
begin
  dbms_resource_manager.clear_pending_area();
  dbms_resource_manager.create_pending_area();
  dbms_resource_manager.create_plan(
                plan    => 'HIGH_PRIO_PLAN',
                comment => 'Plan/Subplan for HighPrio');
 dbms_resource_manager.create_plan_directive(
                plan                     => 'HIGH_PRIO_PLAN',
                group_or_subplan         => 'HPRIO_GR',
                comment                  => 'HighPrio',
                MGMT_P1                  => 70,
                parallel_degree_limit_p1 => 16,
                max_utilization_limit    => 75);
  dbms_resource_manager.create_plan_directive(
                plan                     => 'HIGH_PRIO_PLAN'
                group_or_subplan         => 'HLOW_GROUP',
                comment                  => 'HighPrio with LowLoad Group',
                MGMT_P1                  => 30,
                parallel_degree_limit_p1 => 2,
                max_utilization_limit    => 35);
  …
  dbms_resource_manager.validate_pending_area();
  dbms_resource_manager.submit_pending_area();
end;
/
```

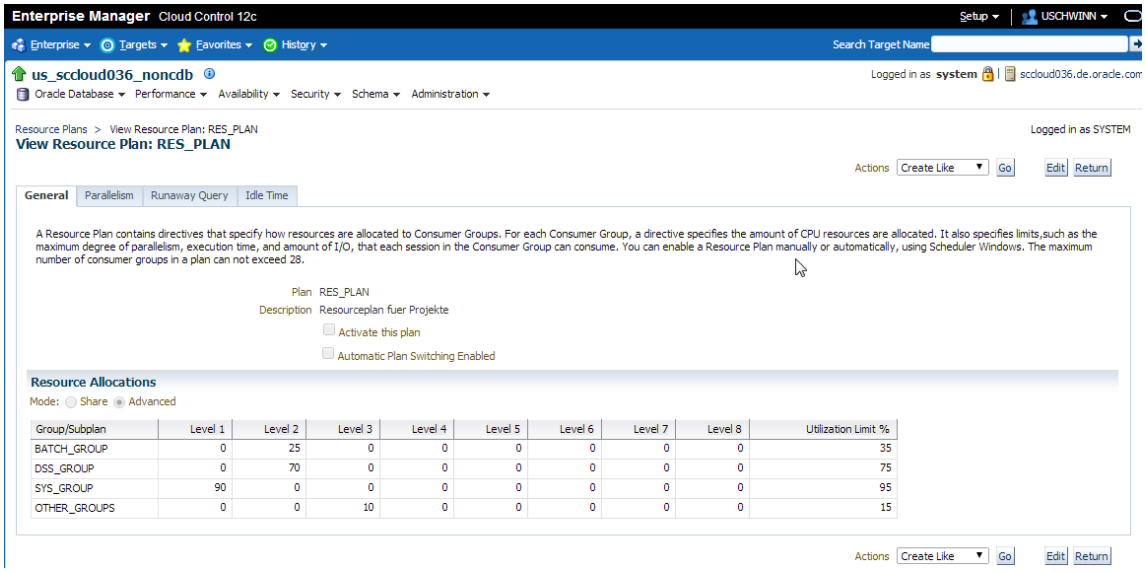
Der Parameter `mgmt_p1` gibt den Prozentsatz an CPU vor, der von der Consumer Group alloziert werden kann. Das Maximum an CPU liegt natürlich bei 100 Prozent. Falls die CPU nicht oder nicht vollständig von einer Consumer Group verwendet werden kann, wird dieser Anteil an die anderen Gruppen weitergegeben. Seit 11*g* Release 2 gibt es zusätzlich den sehr nützlichen Parameter `max_utilization_limit`. Hiermit kann man die Verwendung an CPU nach oben begrenzen. In vielen Ressource Plänen ist dieser Parameter mittlerweile ein sehr wichtiger Bestandteil.

Hinweise: Verwenden Sie `mgmt_p1` und nicht den „obsolete“ Parameter `cpu_p1`. Beachten Sie auch, dass der Parameter `max_utilization_limit` in Oracle Database 12*c* in `utilization_limit` umbenannt wurde.

Der Parameter `parallel_degree_limit_p1` gibt die Grenze für den genutzten "degree of parallelism" (DOP) einer Operationen an. Da nicht mehr als zwei Operationen simultan parallelisiert werden können, errechnet sich die Gesamtanzahl der Parallel Execution Servers pro Statement aus der

doppelten Anzahl des DOPs. Der Defaultwert ist übrigens NULL, was unbegrenzt bedeutet. Soll nur seriell gearbeitet werden, muss explizit der Wert auf 0 gesetzt werden. Darüber hinaus gibt es noch weitere Möglichkeiten, Direktiven aufzusetzen. Ein detailliertes Parallel Statement Queuing beispielsweise ist seit 11*g* Release 2 über den Resource Manager möglich. Weitere Informationen über mögliche Direktiven finden Sie im PL/SQL Packages and Types Reference Guide.

Möchte man einen Multi Level Plan anlegen, verwendet man für die Ressourcen Direktiven der zusätzlichen Ebenen das Suffix `p2` bis `p8` – also `mgmt_p2` für CPU Ressource auf Ebene 2. Ein Multi Level Plan würde im Enterprise Manager Cloud Control folgendermaßen dargestellt werden.



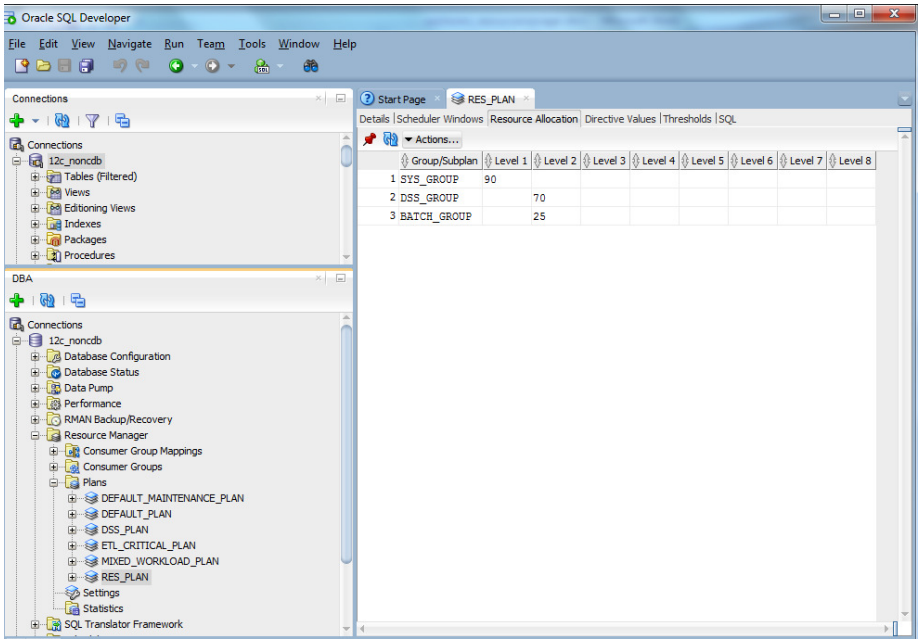
Wie kann der Plan nun aktiviert werden? Hier gibt es zwei Möglichkeiten - entweder manuell über den ALTER SYSTEM Befehl oder aber regelmäßig und automatisch über die sogenannten Scheduler Windows, die vom Oracle Scheduler Job beispielsweise für die Maintenance Jobs verwendet werden. Folgendes Beispiel zeigt die Verwendung des Ressource Plans PB_RES_PLAN im „Monday Window“.

```
execute dbms_scheduler.set_attribute(
           name       => '"SYS"."MONDAY_WINDOW"',
           attribute  => 'RESOURCE_PLAN',
           value      => 'PB_RES_PLAN');
```

Monitoring und Einsatz von Werkzeugen

Zusätzlich zu den verwendeten Prozeduren und Funktionen gibt es auch graphische Implementierungen zur Konfiguration und Verwaltung der Ressourcen im Resource Manager. Bis einschließlich Oracle Database 11*g* stehen dabei die beiden Werkzeuge Enterprise Manager Cloud Control und Database Control mit den entsprechenden Menüpunkten zur Verfügung. In **Cloud Control 12*c*** findet man die Resource Manager Funktionalität unter dem Menüpunkt „Administration => Resource Manager“. Die Resource Manager Funktionalität ist dabei um die Multitenant Architektur mit Container Database (auch CDB) und Pluggable Database (auch PDB) erweitert worden (mehr dazu im Abschnitt „Neu mit Oracle Database 12c“). Database Control hingegen ist seit Oracle Database 12*c* nicht mehr verfügbar.

Ein weiteres graphisches Werkzeug, das zur Konfiguration im Ressource Management Umfeld nützlich sein kann, ist der **SQL Developer**. Dieser verfügt schon seit der Version 3 (aktuell in der Version 4.0.2) über einen „Administration Navigator“ Bereich, der unter anderem auch die Konfiguration und Administration des Ressourcen Managements unterstützt.



Ein exaktes und umfangreiches Monitoring lässt sich immer noch am Schnellsten über SQL Skripte durchführen. Es gibt wie bei jedem technischen Feature auch im Resource Manager Bereich eine Reihe von speziellen Data Dictionary Views, womit die Konfiguration im Einzelnen überprüft werden kann. Zusätzlich liefern spezielle Metrikdaten aus V$Views Hinweise aktuelle Performancedaten. Zum Beispiel kann es vorteilhaft sein, zu wissen, ob eine Instance "CPU-bound" ist - d.h. ob die Instance mehr CPU benötigt als zur Verfügung steht. Hinweise auf dieses Verhalten liefert das Event "resmgr:cpu quantum" im AWR oder Informationen aus der View `v$rsrcmgrmetric_history`, sogar die aktuelle CPU Utilization pro Consumer Group liefern kann. Das Ergebnis bei voller Last vergleicht man dann mit der Verwendung im Resource Manager (z.B. mit mgmt_p1 etc). Genaue Beispiele und Skripte finden sich in den White Paper „Using Oracle Database Resource Manager“ oder in der DBA Community.

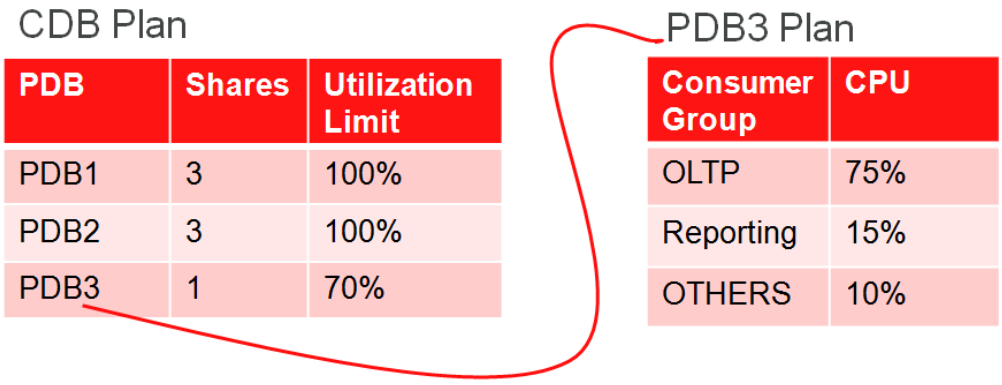
Neu mit Oracle Database 12c – Multitenant Architektur

Wie vorhin kurz angesprochen, ist die Resource Manager Technologie in Oracle Database 12*c* um die Konzepte der Multitenant Architektur erweitert worden. Da nicht alle PDBs gleich sind, kann der Ressourcen Verbrauch auch unterschiedlich sein. Um auch hier den wichtigen PDBs ein gewisses Minimum an Ressourcen zu garantieren, ist der Resource Manager in der Lage, den Ressourcen Verbrauch pro PDB zu verwalten. Dabei gibt es PDB und CDB Ressource Pläne. Ein **CDB Resource Plan** enthält Direktiven für alle PDBs**. PDB Resource Pläne** hingegen enthalten Direktiven für die Ressourcen Verteilung der Consumer Groups innerhalb einer PDB. Ein CDB Resource Plan wird immer innerhalb des ROOT Containers konfiguriert. Dabei wird das Konzept der Shares (Anteile) benutzt. Je höher der Share Wert ist, um so höher sind die daraus resultierenden Ressourcen. Mögliche Ressourcen sind dabei CPU (mit `shares` und `utilization_limit`) und Parallel Execution

Server (mit `parallel_server_limit`). Mit der neuen Prozedur `create_cdb_plan_directive` werden dann die CDB Resource Pläne im ROOT Container erzeugt.

```
PROCEDURE CREATE_CDB_PLAN_DIRECTIVE
Argument Name                  Type                    In/Out Default?
------------------------------ ----------------------- ------ --------
PLAN                           VARCHAR2                IN
PLUGGABLE_DATABASE             VARCHAR2                IN
COMMENT                        VARCHAR2                IN     DEFAULT
SHARES                         NUMBER                  IN     DEFAULT
UTILIZATION_LIMIT              NUMBER                  IN     DEFAULT
PARALLEL_SERVER_LIMIT          NUMBER                  IN     DEFAULT
```

PDB Resource Pläne enthalten Direktiven für die Ressourcen Verteilung der Consumer Groups innerhalb einer PDB; sie sind vergleichbar mit dem Setup eines NON CDB Resource Manager Plans und werden auch ebenso erzeugt. Allerdings existieren einige Einschränkungen. Ein PDB Resource Plan kann beispielsweise keine Subpläne enthalten und maximal acht Consumer Groups werden pro PDB unterstützt. Die Aktivierung erfolgt dabei in der jeweiligen PDB. Vorab muss ein CDB Resource Plan angelegt worden sein.



Weitere Funktionen
Außer den genannten CPU Ressourcen und Multitenant Funktionen in Oracle Database 12*c* gibt es noch weitere interessante Funktionen. Zum Beispiel gibt es die Möglichkeit, I/O Ressourcen zu verwalten – allerdings (im Moment) ausschließlich auf Exadata Storage Systemen. Um I/O Anfragen an die Exadata Storage Server zu regeln oder zu priorisieren, steht diese Funktion unter dem Namen **IO Resource Manager** (auch IORM) zur Verfügung. Das Konzept des Database Resource Manager Plans (hier INTRA Database Plan), wie hier im Text beschrieben, wird zusätzlich um weitere Funktionen wie den sogenannten INTER Database Plan erweitert. Dieser regelt die IO Ressourcen pro Datenbank und wird auf jedem Cellserver mit dem cellcli aktiviert oder deaktiviert. Auch hier gilt: Definierte Grenzen greifen wie bei den CPU Direktiven erst dann, wenn die Ressourcen unter Last geraten sind – im Fall des IO Resource Managers auf Cellserver Ebene. Eine genaue Beschreibung des Setups findet sich im Handbuch Oracle Exadata Storage Server Software User's Guide und in der My Oracle Support Note Master Note for Oracle Database Resource Manager (Doc ID 1339769.1).

Ein weiteres wichtiges Feature, das nur im Zusammenhang mit Resource Manager aktiviert werden kann, ist das sogenannte **Instance Caging**. Damit lässt sich seit Oracle 11*g* Release 2 auf allen Plattformen die maximale Anzahl der CPUs pro Instance angeben. Das Vorgehen dazu ist einfach: Mit ALTER SYSTEM wird pro Instance der Wert von CPU_COUNT festgelegt. Wichtig Voraussetzung

dabei ist, dass ein Resource Manager Plan aktiv sein muß. Falls man keine speziellen Anforderungen an einen eigenen Resource Plan hat, bietet sich der mitgelieferte Resource Plan DEFAULT_PLAN zur Verwendung an. Eine einfache Überprüfung des Setups sieht folgendermassen aus.

```
SELECT name, instance_caging FROM v$rsrc_plan WHERE is_top_plan = 'TRUE';
show parameter cpu_count
```

Um die Ressourcen zu kontrollieren, kann auch hier die View `v$rsrcmgrmetric_history` genutzt werden. In folgendem Beispiel wird die durchschnittliche Anzahl von laufenden und wartenden Sessions angezeigt.

```
SELECT to_char(begin_time, 'HH24:MI')time, sum(avg_running_sessions)
avg_running_sessions, sum(avg_waiting_sessions) avg_waiting_sessions
FROM v$rsrcmgrmetric_history
GROUP BY begin_time ORDER BY begin_time;
```

Vor Oracle Database 12*c* war es schon möglich im Database Resource Manager eine Grenze (Threshold) für Queries anzugeben - man spricht hier auch von "Runaway Queries". Nach Erreichung dieser Grenze wurde eine Aktion durchgeführt. Dauert beispielsweise eine Query oder ein PL/SQL Aufruf mehr als 30 Sekunden (in CPU), dann kann man die Query beenden, einen Wechsel (Switch) zu einer anderen Consumer Group vollziehen oder sogar die ganze Session beenden. Wer hat diese Queries ausgeführt? Welcher Code (SQL oder PL/SQL) wurde verwendet? Und was für eine Aktion wurde durchgeführt? Eine Erweiterung in Oracle Database 12*c* im **SQL Monitoring** Umfeld liefert Antworten auf diese Fragen. Die neuen Spalten `rm_consumer_group, rm_last_action, rm_last_action_reason`, `rm_last_action_time` in `v$sql_monitor` liefern die entsprechenden Informationen. Im folgendem Beispiel wurde nach Erreichung einer Grenze der Wechsel von der GRUPPE_LOW_CPU zur OTHER_GROUPS initiiert.

```
SQL> SELECT username, elapsed_time, plsql_exec_time, sql_text, cpu_time,
     rm_last_action, rm_last_action_reason, rm_last_action_time,
     rm_consumer_group
     FROM v$sql_monitor WHERE username is not null;

USERNAME                       ELAPSED_TIME PLSQL_EXEC_TIME
------------------------------ ------------ ---------------
SQL_TEXT
--------------------------------------------------------------------------------
CPU_TIME   RM_LAST_ACTION
---------- ----------------------------------------------------
RM_LAST_ACTION_REASON          RM_LAST_A RM_CONSUMER_GROUP
------------------------------ --------- ------------------------------
SH                                  378358                0
select /*+ use_nl(c) parallel ordered*/ count(*) from sh.sales
s,sh.customers c where c.cust_id=s.cust_id and cust_first_name='Dina'
10998 SWITCH TO OTHER_GROUPS
SWITCH_CPU_TIME                19-FEB-14 OTHER_GROUPS
```

Will man dabei die Runaway Queries nur überwachen und keine zusätzlichen Aktion durchführen, gibt es die neue "LOG_ONLY" Aktion, die mit den Prozeduren `create_plan_direktive` oder `update_plan_direktive` eingestellt werden kann. Im folgenden Listing wird ein Beispiel die Einstellungen illustrieren.

```
BEGIN
dbms_resource_manager.clear_pending_area();
```

```
dbms_resource_manager.create_pending_area();
dbms_resource_manager.update_plan_directive(
                   plan             => 'TEST_RUNAWAY',
                   group_or_subplan =>'GRUPPE_HIGH_CPU',
                   new_switch_group => 'LOG_ONLY',
                   new_switch_time  => 200);

dbms_resource_manager.update_plan_directive(
                    plan             => 'TEST_RUNAWAY',
                    group_or_subplan => 'GRUPPE_LOW_CPU',
                    new_switch_group => 'OTHER_GROUPS',
                    new_switch_time  => 30);

dbms_resource_manager.update_plan_directive(
                   plan             => 'TEST_RUNAWAY',
                   group_or_subplan => 'OTHER_GROUPS',
                   new_switch_group => 'CANCEL_SQL',
                   new_switch_time  => 100);

dbms_resource_manager.submit_pending_area();
END;
/
```

Damit können verschiedene Grenzen für unterschiedliche Consumer Groups definiert und überwacht werden.

```
SQL> SELECT username, elapsed_time, plsql_exec_time, sql_text, cpu_time,
     rm_last_action, rm_last_action_reason, rm_last_action_time,
     sql_exe_start, rm_consumer_group
     FROM v$sql_monitor WHERE username is not null;

USERNAME                       ELAPSED_TIME PLSQL_EXEC_TIME
------------------------------ ------------ ---------------
SQL_TEXT
--------------------------------------------------------------------------------
CPU_TIME   RM_LAST_ACTION
---------- ------------------------------------------------
RM_LAST_ACTION_REASON          RM_LAST_ACTION_T SQL_EXEC_START
------------------------------ ---------------- -----------------
RM_CONSUMER_GROUP
------------------------------
SCOTT                             140668120       140658478
BEGIN last1.sortiere(32767); END;
116168340
                                                  19.02.2014 16:44
GRUPPE_HIGH_CPU
```

Der User SCOTT führte offensichtlich um 16:44 eine langlaufende Prozedur aus und wurde in `v$sql_monitor` aufgelistet.

Fazit
Der Database Resource Manager bietet eine Vielzahl an Möglichkeiten, Ressourcen zu verwalten und zu überwachen. Weitere Funktionen werden in den nächsten Database Releases sicherlich folgen.

Zusammenfassend noch ein paar Tipps zur Verwendung: Wie bei allen Features gilt auch hier, bevor man den Resource Plan aktiviert, sollte man Tests durchführen. Dabei ist wichtig zu wissen, dass die

Direktiven erst dann greifen können, wenn die Ressourcen unter Last geraten und an die definierten Grenzen stoßen. Bevor sie mit dem Skripting starten, überlegen Sie zuerst in Ruhe, welchen Gruppen welche Ressourcen garantiert werden sollen. Bilden Sie dann die Gruppen am besten auf entsprechende TNS Services ab; das ist ein gängige Praxis.

Halten Sie den Plan auf jeden Fall einfach. Sorgen Sie auch dafür, dass Sie als Administrator in der SYS_GROUP immer ausreichend Ressourcen zur Verfügung haben. Lernen Sie auch aus den vordefinierten Plänen und Consumer Groups. Cloud Control 12*c* bietet gute Übersichten zur Anzeige von Ressource Plänen und Consumer Groups. Nutzen Sie für den eigenen Plan die vorhandene Skripte (siehe Abschnitt „Weitere Informationen"), die sich leicht auf eigene Anforderungen anpassen lassen. Ein abschließendes genaues Ressourcen Monitoring läßt sich mit entsprechenden Skripten durchführen. Wer sich mit Cloud Control ein wenig besser auskennt, kann natürlich auch zusätzlich eigene „user defined metrics (bzw. „metric extensions") im Cloud Control hinterlegen, um ganz automatisch Notifikationen bei bestimmten Ereignissen zu erhalten.

Weitere Informationen

- **My Oracle Support**
  Master Note for Oracle Database Resource Manager (Doc ID 1339769.1)
- **Deutschsprachige Tipps der DBA Community**
  http://blogs.oracle.com/dbacommunity_deutsch
- **Skripte**
  - Im DBA Community Blog: Erfahrungsbericht zur Nutzung des Database Resource Managers
  - Im Blog von Joel Kallman: joelkallman.blogspot.co.uk
- **Oracle White Paper**
  - Using Oracle Database Resource Manager (mit guten Monitoring Skripte)
  - Effective Resource Management Using Oracle Database Resource Manager
- **Handbücher**
  - Oracle Database Administrator's Guide
  - PL/SQL Packages and Types Reference

Kontaktadresse

Ulrike Schwinn
Oracle BU DB – Business Unit Database

ORACLE Deutschland B.V. & Co. KG
Riesstr 25, 80992 München
Telefon: +49 89 1430 1865
E-Mail Ulrike.Schwinn@oracle.com
Internet: http://blogs.oracle.com/dbacommunity_deutsch