Database Resource Manager – Stand der Dinge

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

Schon seit der Version 8*i* ist es möglich, Ressourcen in der Datenbank mit der Enterprise Edition und dem Database Resource Manager zu verwalten. Damit lassen sich Datenbank-Ressourcen-Engpässe 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. Ein Grund dafür war die eingeschränkte Verwendbarkeit, die ausschließlich über PL/SQL-Packages und für dedizierte Datenbankuser-Sessions möglich war. Spätestens seit der Datenbank-Version 10*g* mit der Verfügbarkeit über die grafische Oberfläche des Enterprise Manager sowie der Erweiterung der Nutzung über Services, Module, Programme etc. ist dieser Grund passé. So können zum Beispiel CPU-Ressourcen für verschiedene Applikationen priorisiert, lang laufende Operationen unterbunden oder die Anzahl der Sessions begrenzt werden, um nur einige Beispiele zu nennen. Darüber hinaus wird mit der Oracle Database 12*c* und der neuen Multitenant-Architektur der Einsatz des Resource Manager immer wichtiger.

Mittlerweile verwenden immer mehr Anwender den Resource Manager und haben damit eine zusätzliche wichtige Ebene der Administration kennen und schätzen gelernt. Nach einer Einarbeitungsphase, die sich durch vorhandene Skripte verkürzen lässt, sowie einer Testphase des Setups können durch den Einsatz eines entsprechenden Resource-Management-Plans drängende Fragen der Administration gelöst werden. Danach ist nur noch ein einfaches Monitoring von Zeit zu Zeit erforderlich. Auch intern findet der Database Resource Manager schon seit der Version 10*g* seine Anwendung, zum Beispiel in den sogenannten „Automated Maintenance Tasks".

Um einen schnellen Einstieg in das Thema zu gewährleisten, kommen nachfolgend ein paar grundsätzliche Begriffe wie „Resource Consumer Group", „Resource Consumer Mapping" und „Resource Plan" zur Sprache. Im ersten Schritt werden Benutzergruppen nach unterschiedlichen Ressourcen-Anforderungsprofilen erzeugt; sie werden als „Resource Consumer Group" bezeichnet. Es sind also Datenbankuser-Sessions, die als eine Einheit behandelt werden. Defaultmäßig existieren schon einige vordefinierte Resource Consumer Groups wie zum Beispiel die sogenannte „SYS_GROUP", 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 das Kommando aus *Listing 1* verwenden.

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

Listing 1

Datenbankuser-Sessions können automatisch einer gewissen Consumer Group zugeordnet werden. Diese „Mapping Rules“ enthalten Regeln beziehungsweise Eigenschaften, nach denen die einzelnen Sessions den Gruppen initial zugeordnet sind. 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 lässt sich ein Gruppenwechsel natürlich auch manuell anstoßen. Die wichtigsten Mapping Rules sind:

- 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 (dbms_resource_manager.module_name etc.)
- Client OS User (dbms_resource_manager.client_os_user)
- Client Program (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 *(siehe Listing 2)* bildet nun die Sessions des Service „S_SERVICE1“ auf die Consumer Group „HP_GROUP“ ab.

Für den Parameter „attribute“ können die in den Mapping Rules angegebenen Konstanten wie „dbms_resource_manager.oracle_user“ etc. verwendet werden. Möchte man beispielsweise erreichen, dass der „APPDBA“-User der „SYS_GROUP“-Gruppe angehört, ist das in *Listing 3* gezeigte Kommando erforderlich.

Damit ein User oder eine Rolle in eine Consumer Group wechseln kann, ist ein zusätzlicher Prozedur-Aufruf erforderlich. Mit der Verwendung von „PUBLIC“ wird beispielsweise allen Usern erlaubt, in die Gruppe „HPRIO_GR“ zu wechseln *(siehe Listing 4)*.

Kern des Ressourcen-Managements ist der „Resource Manager Plan“. Resource-Pläne enthalten die Regeln (auch „Direktiven“ genannt), nach denen die Ressourcen an die Resource Consumer Groups aufgeteilt werden. Resource-Pläne sind dynamisch und können mit einem einfachen Befehl aktiviert beziehungsweise deaktiviert werden. Es gilt allerdings immer nur ein Plan zu einer Zeit. Resource-Pläne können aus einem einzigen Plan, also einer Ebene (auch „Single Level Plan“), oder auch aus mehreren Subplänen, das heißt aus mehreren Ebenen (auch „Multi Level Plan“) bestehen. Wie bei den Consumer Groups gibt es auch hier vordefinierte Resource-Pläne wie „DEFAULT_MAINTENANCE_PLAN“, der für die oben erwähnten Maintenance-Aufgaben zum Einsatz kommt.

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

Listing 2

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

Listing 3

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

Listing 4

Ein Code-Ausschnitt aus einem selbstdefinierten Resource-Plan (oder Subplan) könnte dann folgendermaßen aussehen *(siehe Listing 5)*. Die Prozeduren „clear_“, „create_“, „validate_“ und „submit_pending_area“ verwalten dabei einen temporären Arbeitsbereich für die Dauer der Resource-Management-Konfiguration. Die Änderungen sind so lange nicht sichtbar/wirksam, bis die Pending Area mit „submit_pending_area“ abgeschlossen worden ist. Hinweis: Die Verwendung eines temporären Arbeitsbereichs ist bei der Erzeugung von Consumer Groups oder Resource-Plänen obligatorisch. Aus Gründen der Übersichtlichkeit wurde diese Information in den übrigen Beispielen weggelassen.

Der Parameter „mgmt_p1“ gibt den Prozentsatz an CPU an, 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 der Version 11*g* R2 gibt es zusätzlich den sehr nützlichen Parameter „max_utilization_limit“. Damit kann man die Verwendung an CPU nach oben begrenzen. In vielen Resource-Plänen ist dieser Parameter mittlerweile ein wichtiger Bestandteil. Hinweise: Man sollte „mgmt_p1“ und nicht den veralteten Parameter „cpu_p1“ verwenden. Zu beachten ist auch, dass der Parameter „max_utilization_limit“ in der Version 12*c* in „utilization_limit“ umbenannt wurde.

Der Parameter „parallel_degree_limit_p1“ gibt die Grenze für den genutzten „degree of parallelism“ (DOP) einer Operation an. Da nicht mehr als zwei Operationen simultan parallelisiert werden können, errechnet sich die Gesamtanzahl der Parallel Execution Server pro Statement aus der doppelten Anzahl des DOP. Der Defaultwert ist übrigens „NULL“, was „unbegrenzt“ bedeutet. Soll nur seriell gearbeitet werden, muss explizit der Wert auf „0“ gesetzt sein. Darüber hinaus gibt es noch weitere Möglichkeiten, Direktiven aufzusetzen. Ein detailliertes Parallel Statement Queuing beispielsweise ist seit 11*g*

R2 über den Resource Manager möglich. Weitere Informationen über mögliche Direktiven finden sich im „PL/SQL Packages and Types Reference Guide“. Zusätzliche Ebenen lassen sich über den Suffix „p2“ bis „p8“ – also „mgmt_p2“ für die CPU-Ressource auf Ebene 2 steuern.

Es gibt zwei Möglichkeiten, den Plan zu aktivieren: entweder manuell über den „ALTER SYSTEM“-Befehl oder aber regelmäßig und automatisch über die sogenannten „Scheduler Windows“, die der Oracle-Scheduler-Job beispielsweise für die Maintenance-Jobs verwendet. Das Beispiel in *Listing 6* zeigt die Verwendung des Resource-Plans „PB_RES_PLAN“ im „Monday Window“.

Monitoring und Einsatz von Werkzeugen

Zusätzlich zu den verwendeten Prozeduren und Funktionen gibt es auch grafische Implementierungen zur Konfiguration und Verwaltung der Ressourcen. Bis einschließlich Oracle Database 11*g* stehen dabei die beiden Werkzeuge „Enterprise Manager Cloud Control“ und „Database Control“ mit den entsprechenden Menüpunkten zur Verfügung. Ab Oracle Database 12*c* findet man die Resource- Manager-Funktionalität in „Cloud Control 12*c*“ unter dem Menüpunkt „Administration => Resource Manager“.

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

Ein exaktes und umfangreiches Monitoring lässt sich allerdings 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, mit denen die Konfiguration im Einzelnen überprüft werden kann. Zusätzlich liefern spezielle Metrikdaten aus „V$Views“ Hinweise auf aktuelle Performance-Daten. Zum Beispiel kann es vorteilhaft sein zu wissen, ob eine Instanz „CPU-bound“ ist – also ob sie mehr CPU benötigt, als zur Verfügung steht. Hinweise auf dieses Verhalten liefern das Event „resmgr:cpu quantum“ im AWR oder Informationen aus der View „v$rsrcmgrmetric_history“, die sogar die aktuelle „CPU Utilization“ pro Consumer Group liefern kann. Das Ergebnis bei voller Last vergleicht man dann mit der Verwendung im Resource Manager (mit „mgmt_p1“ etc.). Genaue Beispiele und Skripte finden sich im White Paper „Using Oracle Database Resource Manager“ oder in der DBA-Community.

Resource Manager und Multitenant-Architektur

Mit der Datenbank-Version 12*c* und dem neuen Konzept der Multitenant-Architektur wurde auch der Resource Manager erweitert. Da nicht alle Pluggable Databases (PDBs) gleich sind, kann auch der Ressourcen-Verbrauch unterschiedlich sein. Um auch hier den wichtigen PDBs ein gewisses Minimum an Ressourcen zu garantieren, ist der Resource Manager in der

```
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                 => 'Hlowgroup',
              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;
/
```

Listing 5

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

Listing 6

```
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
```

Listing 7

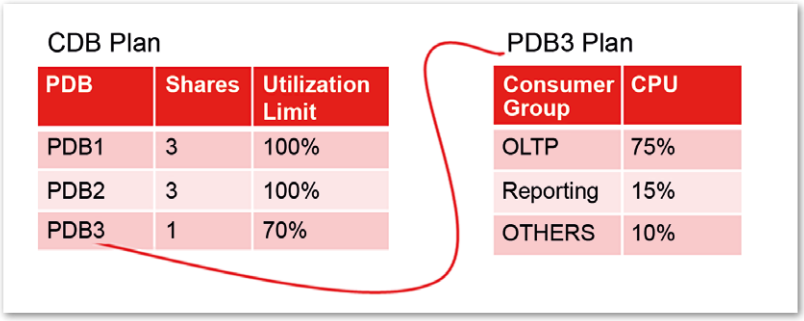

Abbildung 1: PDB- und CDB-Resource-Pläne

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

Listing 8

```
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;
```

Listing 9

```
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
```

Listing 10

Lage, den Ressourcen-Verbrauch pro PDB zu verwalten. Dabei gibt es PDB- und CDB-Resource-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 neue Konzept der Shares (Anteile) benutzt. Je höher der Share-Wert ist, desto höher sind die daraus resultierenden Ressourcen-Anteile. Mögliche Ressourcen sind dabei CPU („shares" und „utilization_limit") und Parallel Execution Server („parallel_server_limit"). Die neue Prozedur „create_cdb_plan_directive" erzeugt dann die CDB-Resource-Pläne im ROOT-Container *(siehe Listing 7)*.

PDB-Resource-Pläne 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 *(siehe Abbildung 1)*.

Weitere Funktionen

Außer den beschriebenen Features gibt es noch weitere interessante Funktionen im Zusammenhang mit dem Resource Manager. So besteht die Möglichkeit, I/O-Ressourcen detailliert auf Datenbank- und Cell-Ebene eines Exadata Storage Servers zu verwalten. Um solche I/O-Anfragen zu regeln oder zu priorisieren, steht diese Funktion unter dem Namen „IO Resource Manager" (auch IORM) zur Verfügung. Dabei können die Ressourcen nicht nur innerhalb einer Datenbank („INTRA Database Plan"), sondern auch zwischen den Datenbanken („INTER Database Plan") verwaltet werden. Eine genaue Beschreibung des Setups findet sich im Handbuch „Oracle Exadata Storage Server Software User's Guide" sowie 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 der Version 11*g* R2 auf allen Plattformen die maximale Anzahl der CPUs pro Instance einstellen. Das Vorgehen ist einfach: Mit „ALTER SYSTEM" wird pro Instanz der Wert von „CPU_COUNT" dynamisch festgelegt; dabei ist eine Aufteilung der vorhandenen CPUs auf Instanzen oder sogar ein Over-Subscribe der CPUs möglich. Hinweis: Änderungen an „CPU_COUNT" haben auch direkten Einfluss auf andere Technologien wie den Optimizer. Nun muss noch ein Resource-Manager-Plan aktiv sein. Falls man keine speziellen Anforderungen an einen eigenen Resource-Plan hat, bietet sich der mitgelieferte Resource-Plan „DEFAULT_PLAN" an. *Listing 8* zeigt eine einfache Überprüfung des Setups.

Um die Ressourcen zu kontrollieren, eignet sich die vorhin schon erwähnte View „v$rsrcmgrmetric_history". Im Beispiel *(siehe Listing 9)* wird die durchschnittliche Anzahl von laufenden und wartenden Sessions angezeigt.

Vor Oracle Database 12*c* war es schon möglich, im Database Resource Manager eine Grenze (Threshold) für lang laufende Queries anzugeben – man spricht hier auch von „Runaway Queries". Diese Grenze kann in der „create_plan"-Prozedur über Parameter wie „switch_io_megabytes", „switch_io_reqs", „switch_for_call", „switch_time", „switch_estimate", „max_est_exec_time" oder neu in 12*c* über „switch_io_logical" und „switch_elapsed_time" eingestellt werden. Nach Erreichen dieser Grenze erfolgt eine Aktion. Dauert beispielsweise eine Query oder ein PL/SQL-Aufruf länger 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 gibt Antworten auf diese Fragen. Die neuen Spalten „rm_consumer_group", „rm_last_action", „rm_last_action_reason" und „rm_last_action_time" in „v$sql_monitor" liefern die entsprechenden Informationen. Im folgenden Beispiel wird nach Erreichen einer Grenze der Wechsel von „GRUPPE_LOW_CPU" zu „OTHER_GROUPS" initiiert *(siehe Listing 10)*.

Will man dabei die Runaway Queries nur überwachen und keine zusätzliche Aktion durchführen, gibt es den neuen Wert „LOG_ONLY", der mit den Prozeduren „create_plan_direktive" oder „update_plan_direktive" eingestellt werden kann *(siehe Listing 11)*. Das Monitoring kann dann über eine Abfrage erfolgen *(siehe Listing 12)*. Der User „SCOTT" führte offensichtlich um 16:44 eine lang laufende Prozedur aus und wurde in „v$sql_monitor" aufgelistet.

```
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;
/
```

Listing 11

```
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
```

Listing 12

Fazit und abschließende Tipps

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. 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 stehen. Bevor man mit dem Skripting startet, sollte man zuerst in Ruhe überlegen, welchen Gruppen welche Ressourcen garantiert werden sollen. Man bildet dann die Gruppen am besten auf entsprechende TNS-Services ab; das ist eine gängige Praxis.

Der Plan sollte auf jeden Fall einfach gehalten werden. Zudem muss man dafür sorgen, dass der Administrator in der „SYS_GROUP" immer ausreichend Ressourcen zur Verfügung hat. Cloud Control 12*c* bietet gute Übersichten zur Anzeige von Resource-Plänen und Consumer Groups. Man kann für den eigenen Plan die vorhandenen Skripte (siehe „Weitere Informationen") nutzen, die sich leicht an 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" (beziehungsweise „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 Manager
  - Im Blog von Joel Kallman: joelkallman.blogspot.co.uk
- *Oracle White Paper*
  - Using Oracle Database Resource Manager (mit guten Monitoring-Skripten)
  - Effective Resource Management Using Oracle Database Resource Manager
- *Handbücher*
  - Oracle Database Administrator's Guide
  - PL/SQL Packages and Types Reference

Ulrike Schwinn
ulrike.schwinn@oracle.com

S3-Programm – Klappe, die erste

Yasmin Misch, DOAG Dienstleistungen GmbH

Fast sechzig Studierende von dreizehn unterschiedlichen Hochschulen, fünfzehn Sponsoren und neun Hochschul-Professoren: So lauten die Zahlen des ersten StudentSponsorShip-Programms (kurz: „S3"). Dieses fand erstmals im Rahmen der DOAG 2014 Konferenz + Ausstellung statt. „Ein voller Erfolg" und „Wir hoffen auf Wiederholung im nächsten Jahr" meinten die Teilnehmer. Kurzum: eine gelungene Premiere!

Montag, 14 Uhr: Alle Studierenden sind angereist. Gleich wird das Orga-Team sie einweisen. Raum- und Referentenbetreuung, Zutrittskontrolle oder die Ausgabe von Dolmetscher-Headsets sind nur einige der Tätigkeiten, die Studierende nun schon seit einigen Jahren auf der DOAG Konferenz + Ausstellung im Rahmen des Studenten-Programms übernehmen. Im Gegenzug erhalten die Studierenden mehr als freien Zutritt zu Vorträgen und Ausstellung, mehr als die Möglichkeit, kostenfrei am Schulungstag teilzunehmen, und mehr als einen aufregenden Community-Abend. Wer am S3 teilnimmt, bekommt zusätzlich auch noch Reise- und Übernachtungskosten erstattet.

Kontakte sind mehr wert als Geld

Der finanzielle Vorteil ist für die meisten Studierenden nicht der alleinige Anreiz für eine S3-Teilnahme. Der Kontakt zu den sponsernden Unternehmen und die Aussicht auf lukrative Job-Angebote seien Gold wert, berichten die Studierenden einheitlich. Pro finanzierten Sponsorship vermittelt die DOAG Gespräche mit Studierenden. Doch auch darüber hinaus suchen diese den Kontakt zu den Sponsoren. Auf dem Get-together am zweiten Tag der Konferenz kommen alle Teilnehmer des S3 zusammen. Die Sponsoren stellen sich vor. Wer es schafft, das Interesse der Studierenden zu wecken, ergattert zusätzliche Zeit mit dem Nachwuchs.

Auszeichnungen für Professor und Studierende

Mittwochabend: Ernennung zum DOAG-Botschafter des Jahres im Bereich „Technologie". Die Auszeichnung geht an Prof. Dr. Harm Knolle von der Hochschule Bonn-Rhein-Sieg. Das Komitee würdigt sein Engagement bei den studentischen Aktivitäten. Er hatte einmal mehr eine Exkursion zur DOAG Konferenz organisiert und war mit rund dreißig Studierenden angereist. Außerdem war er maßgeblich an der Entstehung des S3 beteiligt. An dieser Stelle nochmal „Herzlichen Glückwunsch!"

Auszeichnungen gibt es auch unter den Studierenden für ihre Teilnahme an einem Test. Zur Auswahl standen Online-Tests in den Bereichen „Java" oder „SQL". Alternativ konnten sie an einem eigens für die Konferenz kreierten Quiz teilnehmen. Die zwanzig Fragen stammten direkt von den Sponsoren. Der Erstplatzierte, Tobias Lindner, erhielt eine Mini-Drohne. Für den zweiten (Anja Schäfer) und den dritten Platz (Sebastian Mahlke) erhielten die Studenten jeweils eine Solarblume, mit der sie nun über USB den Akku ihres Handys laden können.

S3 wird ausgeweitet

Bisher war das S3 ein Privileg der DOAG Konferenz + Ausstellung. Nach der guten Resonanz hat die DOAG beschlossen, diese in Zukunft auch für die Java-Konferenz JavaLand anzubieten. Sie findet vom 24. bis 26. März 2015 im Phantasialand in Brühl statt. Dem bisherigen Feedback folgend, deutet auch hier alles auf einen gelungenen Start für das S3 im JavaLand hin.

Yasmin Misch
yasmin.misch@doag.org