Datenbank-Services – nicht nur im RAC-Umfeld sinnvoll

Thomas Lehmann, Robotron Datenbank-Software GmbH

Datenbank-Services gibt es schon seit der Datenbank-Version 8i. Dennoch wird diese Konfiguration noch viel zu selten genutzt. Meist verbinden viele DBAs das Thema „Services" mit Oracle Real Application Cluster, aber auch unter Single Instances ist der Einsatz von Services sinnvoll. Der Artikel geht auf die Konfiguration von Services ein und beschreibt beispielhaft, welche Anwendungsfälle man durch die Verwendung von Services abdecken kann.

Datenbank-Services stellen neben der Benutzung der SID eine weitere Möglichkeit dar, sich mit einer Datenbank zu verbinden. Jede Datenbank besitzt einen Default-Service, der sich aus der Kombination von „db_name" und „db_domain" zusammensetzt. Mit Einführung von Oracle Multitenant werden die einzelnen Datenbank-Container ebenfalls über den Service-Namen angesprochen. Dazu später mehr. Die nachfolgenden Beispiele basieren auf einer Single Instance unter Oracle 12*c* Release 1 ohne Grid Infrastructure.

Einrichtung und Benutzung von Datenbank-Services

Die Einrichtung und Administration von Services erfolgt mit dem Datenbank-Package „DBMS_SERVICE". Das Beispiel in *Listing 1* zeigt, wie man relativ einfach einen Service innerhalb der Datenbank anlegt. Um sich mit einem Datenbank-Service zu verbinden, muss dieser gestartet werden *(siehe Listing 2)*.

Mit „lsnrctl status" kann man prüfen, ob ein Service erreichbar ist. Damit Services beim nächsten Datenbank-Neustart automatisch gestartet werden, müssen

sie im Initialisierungsparameter „service_names“ angegeben sein. Welche Services in einer Datenbank angelegt sind, lässt sich aus der View „DBA_SERVICES“ abfragen.

Hat man seine Services erfolgreich angelegt und gestartet, kann man sich mit diesen verbinden. Je nach verwendeter Applikation beziehungsweise Umgebung kommen verschiedene Möglichkeiten in Betracht. *Listing 3* zeigt, wie der entsprechende Eintrag in der „TNSNAMES.ORA“ anzupassen ist, um sich mit dem Service zu verbinden, während *Listing 4* die Verbindung per „SQLPLUS“ und *Listing 5* die Implementierung per JDBC zeigt.

Monitoring und Workload Management mit Datenbank-Services

Mithilfe der Services kann man einzelne Anwendungen oder Anwendungsteile gruppieren. Ein Service stellt damit eine logische Gruppierung von Sessions dar, auf deren Grundlage ein Monitoring oder eine Priorisierung durchgeführt werden kann.

Performance-Werkzeuge wie der AWR-Report oder das Performance-Tab im Cloud Control beziehungsweise EM Express bieten Ansichten gruppiert nach Service-Namen an. Zudem besteht die Möglichkeit, über das Package „DBMS_MONITORING“ Statistiken oder Traces je Service/Modul zu aktivieren.

Mit dem Beispiel in *Listing 6* ist es möglich, SQL Trace für alle Sessions eines bestimmten Service zu aktivieren. Über die Parameter „module_name“ beziehungsweise „action_name“ lässt sich die Session, für die das Tracing aktiviert werden soll, weiter einschränken – sofern die Anwendung diese Werte setzt. Darüber hinaus ist es möglich, für einen bestimmten Zeitraum Performance-Werte von Sessions einzusammeln, die sich über den definierten Service verbinden *(siehe Listing 7)*. Die Ergebnisse sind aus der View „V$SERV_MOD_ACT_STATS“ ersichtlich.

Natürlich besteht auch die Möglichkeit, einzelne Services zu stoppen *(siehe Listing 8)* beziehungsweise alle Sessions, die an einem Service angemeldet sind, zu beenden *(siehe Listing 9)*. Wird ein Service beendet, bleiben alle bestehenden Sessions angemeldet; lediglich neue Sessions können sich nicht mehr zum Service verbinden.

Im Bereich des Workload Management können Services ebenfalls eine zentrale Rolle spielen. Zum einen lässt sich damit eine Lastverteilung erreichen, zum anderen können per Service-Definitionen Ressourcen priorisiert werden.

```
begin
   DBMS_SERVICE.CREATE_SERVICE(
       service_name => 'APPL_PROD',
       network_name => 'APPL_PROD'
   ) ;
end;
```

Listing 1

```
begin
   DBMS_SERVICE.START_SERVICE('APPL_PROD') ;
end;
```

Listing 2

```
appl_prod=
   (DESCRIPTION=
       (ADDRESS_LIST=
           (ADDRESS=(PROTOCOL=TCP)(HOST=db_server)(PORT=1521)))
           (CONNECT_DATA=(SERVICE_NAME=appl_prod)
       )
   )
```

Listing 3

```
sqlplus username/pwd@db_server:1521/appl_prod
```

Listing 4

```
jdbc:oracle:thin:@//db_server:1521/appl_prod
```

Listing 5

```
Begin
   DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE(
       service_name => 'APPL_PROD',
       module_name => DBMS_MONITOR.ALL_MODULES,
       action_name => DBMS_MONITOR.ALL_ACTIONS,
       waits => TRUE,
       binds => TRUE);
end;
```

Listing 6

```
begin
   DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(
       service_name =>'APPL_PROD',
       module_name => 'WEBMODUL',
       action_name => DBMS_MONITOR.ALL_ACTIONS);
end;
```

Listing 7

Neben den Datenbank-Services ist auch der Oracle Database Resource Manager bereits jahrelang Bestandteil der Datenbank. Genau dieses Feature bietet sich an, um Services bei Ressourcen-Engpässen zu priorisieren. Folgende Datenbank-Ressourcen lassen sich damit zum Beispiel begrenzen: CPU-Nutzung, Anzahl von Sessions, UNDO-Nutzung, Session-IDL-Time, SQL-Execution-Time und I/O-Limits.

Um mit dem Resource Manager zu arbeiten, ist ein wenig Installationsaufwand erforderlich. Zuerst richtet man sogenannte „Resource Consumer Groups" ein. Es gibt bereits einige Gruppen wie „SYS_GROUP" oder „OTHERS_GROUP", aber wir wollen unsere Services mit unseren Gruppen verbinden. Aus diesem Grund sind individuelle Gruppen anzulegen *(siehe Listing 10)*.

Im zweiten Schritt erfolgt das sogenannte „Resource Group Mapping", also die Zuordnung von Sessions zu Resource Groups. Im konkreten Beispiel *(siehe Listing 11)* wird der Service-Name als Zuordnungskriterium gewählt. Sessions, die sich über den Datenbank-Service „SERVICE_JOBS" anmelden, gelangen in die Resource Group „GRUPPE_JOBS"; die Sessions über den Service „SERVICE_DIALOG" werden der Resource Group „GROUP_DIALOG" initial zugeordnet.

Im dritten Schritt wird der Resource Plan angelegt, also der Plan, der bestimmt, welche Resource Group wie viel Prozent einer Ressource bekommt, wenn diese knapp wird *(siehe Listing 12)*. In diesem Beispiel wird ein Resource Plan erstellt, in dem alle Sessions der „SYS_GROUP" 100 Prozent CPU-Last verbrauchen dürfen. Sessions für den Dialogbetrieb erhalten 80 Prozent und Scheduler-Jobs 20 Prozent der verfügbaren CPU-Ressourcen. Sind noch CPU-Ressourcen frei, können alle anderen Sessions bedient werden. Die Priorisierung erfolgt nur, wenn eine Überbeanspruchung der Ressourcen vorliegt.

Zu guter Letzt muss der Resource Plan noch aktiviert werden. Anschließend ordnen sich die Sessions der zugeordneten Resource Group zu. Bei Überbeanspruchung einer Ressource kann über Plan-Direktiven eine Verschiebung dieser Session in eine andere Ressourcen-Gruppe erfolgen. Mithilfe der Plan-Direktiven können sogar einzelne SQL-Statements unterbrochen beziehungsweise ganze Sessions bei einer Überbeanspruchung einer Ressource abgebrochen werden.

```
begin
DBMS_SERVICE.STOP_SERVICE(
      service_name => 'APPL_PROD');
end;
```

Listing 8

```
begin
DBMS_SERVICE.DISCONNECT_SESSION(
   service_name => 'APPL_PROD',
   disconnect_option => DBMS_SERVICE.IMMEDIATE);
end;
```

Listing 9

```
begin
   DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA();
   DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
   DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(
       consumer_group=>'GRUPPE_JOBS',
       comment=>'Ressourcengruppe für Jobs');
   DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(
       consumer_group=>'GRUPPE_DIALOG',
       comment=>'Ressourcengruppe für Dialogbetrieb');
   DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
end;
```

Listing 10

```
begin
   DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA();
   DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
   DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING(
       attribute => dbms_resource_manager.service_name,
       value => 'SERVICE_JOBS',
       consumer_group => 'GRUPPE_JOBS');
   DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING(
       attribute => dbms_resource_manager.service_name,
       value => 'SERVICE_DIALOG',
       consumer_group => 'GRUPPE_DIALOG');
   DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
end;
```

Listing 11

Datenbank-Services im HA-Umfeld

Im High-Availability-Umfeld (Data Guard / RAC) kommt man an Services nicht mehr vorbei. Services sollten in dieser Umgebung immer mit dem Kommandozeilenwerkzeug „srvctl" angelegt werden.

In einer einfachen Data-Guard-Umgebung, bestehend aus einer Primary- und einer Standby-Datenbank, möchte man der Applikation eine Möglichkeit bereitstellen, sich immer mit der aktiven Datenbank zu verbinden. Arbeitet man an dieser Stelle mit SIDs, muss man vorher wissen, welche Datenbank aktiv ist. Das ergibt wenig Sinn. Viel besser ist es, wenn man sich mit einem Service verbindet, der nur auf der aktiven Datenbank läuft. Erfolgt ein Switchover der Datenbank-Rollen, wird auch der Service umgehängt.

Das ist seit der Datenbank-Version 11 über sogenannte „Role Based Database Services“ möglich. Als definierbare Rollen kommen folgende Optionen infrage: „PRIMARY“, „PHYSICAL_STANDBY“, „LOGICAL_STANDBY“ und „SNAPSHOT_STANDBY“. Je nachdem, welche Rolle die Datenbank-Instanz einnimmt, werden die zugeordneten Services automatisch gestartet. Im Beispiel in *Listing 13* wird auf der Datenbank-Instanz der Service „APPL_PROD“ angelegt, der nur auf der PRIMARY-Seite läuft.

Diese Konfiguration führt man auf allen Knoten der Data-Guard-Umgebung aus, damit beim Switch- beziehungsweise Failover der Service auf den anderen Knoten ebenfalls startet, sobald die Instanz die definierte Rolle einnimmt. In einer Active-Data-Guard-Umgebung kann es unter Umständen gewünscht sein, dass Services auf der Standby-Seite laufen. Zum Beispiel kann man die Standby-Datenbank im „Read Only“-Modus starten und diese Instanz beispielsweise für ein Reporting nutzen.

```
begin
   DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA();
   DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
   DBMS_RESOURCE_MANAGER.CREATE_PLAN(
       plan => 'MYPLAN',
       comment => 'Resource Management for Jobs');
   DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(
       plan => 'MYPLAN',
       group_or_subplan => 'SYS_GROUP',
       comment => 'SYS Group',
       mgmt_p1 => 100 );
   DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(
       plan => 'MYPLAN',
       group_or_subplan => 'GRUPPE_DIALOG',
       comment => 'Dialogbetrieb',
       mgmt_p2 => 80 );
   DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(
       plan => 'MYPLAN',
       group_or_subplan => 'GRUPPE_JOBS',
       comment => 'Scheduler Jobs',
       mgmt_p2 => 20 );
   DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(
       plan => 'MYPLAN',
       group_or_subplan => 'OTHER_GROUPS',
       comment => 'Others',
       mgmt_p3 => 100 );
   DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
end;
```

Listing 12

```
srvctl add service -db APPLDB1 -service APPL_PROD –role PRIMARY
```

Listing 13

```
begin
   DBMS_SERVICE.CREATE_SERVICE(
       service_name => 'APPL_PROD',
       network_name => 'APPL_PROD',
       edition => 'APPL_PROD_ED1'
   ) ;
end;
```

Listing 14

Im Oracle-RAC-Umfeld werden Datenbank-Services in erster Linie zur Lastverteilung genutzt. Dabei können Services an ganz bestimmte RAC-Knoten („preferred“) gebunden sein. Sind diese Knoten aus irgendwelchen Gründen nicht verfügbar, so kann man Fallback-Knoten („available“) definieren. Sind Services an mehrere Knoten gebunden, kann man über Load Balancing die Verteilung der Sessions zwischen diesen RAC-Knoten bestimmen. Dafür stehen mit den Parametern „rlbgoal“ (Runtime Load Balancing Goal) beziehungsweise „clbgoal“ (Connection Load Balancing Goal) Steuerungsmöglichkeiten zur Verfügung. Darüber hinaus kann man an der Service-Konfiguration weitere Einstellungen vornehmen, zum Beispiel für Transparent Application Failover (TAF), Application Continuity oder Fast Application Notifications (FAN).

Datenbank Services unter Oracle Multitenant

Mit Einführung der Container-Datenbank unter 12*c* Release 1 ist es unumgänglich, Services zu benutzen. Über die SID kann man jetzt nur noch die CDB-Instanz ansprechen, alle darin enthaltenen PDBs müssen extern über den Service-Namen verbunden sein. Auch hier empfiehlt es sich, nicht den Standard-Service zu nutzen, sondern eigene Services anzulegen.

Datenbank-Services und Datenbank-Jobs

Wer das Package „DBMS_SCHEDULER“ zur Jobsteuerung nutzt, kann auch hier Services sinnvoll nutzen. Für die genannten Monitoring-Zwecke lässt sich der Default-Service aller Jobs ändern. Darüber hinaus kann man Jobs über Job-Klassen zusammenfassen und jeder davon einen eigenen Service mitgeben. Das bewirkt zwei Dinge: Zum einen kann man, wie bereits angesprochen, über Services Ressourcen-Gruppen anlegen. Damit ist eine Priorisierung der Jobs möglich. Zum anderen lassen sich im RAC-Umfeld Jobs auf einzelne RAC-Knoten verteilen (Lastverteilung) oder gezielt zusammenfassen (Global Cache Tuning). Jobs, die über Job-Klassen einen Service zugeordnet bekommen, laufen im RAC-Umfeld nur auf dem Knoten, auf dem der Service gestartet ist.

Datenbank-Services für Entwickler

Datenbank-Services können für Entwickler ebenfalls eine große Rolle spielen. Wird im Entwicklungsprozess bereits mit dem Edition-Feature gearbeitet, liegen also mehrere Versionen (Editionen) eines Datenbank-Objekts vor, so können Services angelegt werden, bei denen nach dem Login eine bestimmte Edition aktiv ist. Das Beispiel aus *Listing 14* zeigt, wie man einen Service für eine definierte Edition anlegt.

Mit dieser Konfiguration ist es möglich, über die reine Konnektivität zu verschiedenen Services unterschiedliche Software-Stände auf ein und derselben Datenbank zu testen.

Fazit

Auch wenn die meisten DBAs Datenbank-Services ins Umfeld von Oracle Real Application Cluster einordnen, ist die Einrichtung auf Single-Instanzen durchaus sinnvoll. Der Mehrwert von Services liegt vor allem in den Bereichen „Monitoring", „Tracing" beziehungsweise „Ressourcen-Zuteilung". Im Umfeld von HA-Umgebungen ist die Benutzung von Services ein Muss, ebenso bei Oracle-Multitenant-Umgebungen.

Man sollte immer eigene Services einrichten und benutzen und nicht auf den Default-Service zurückgreifen, da dieser nicht alle Funktionen unterstützt. Eine Einrichtung von Services ist relativ einfach, der Mehrwert sehr hoch. Es gibt daher keinen Grund zu zögern.

Thomas Lehmann
thomas.lehmann@robotron.de

Datenbanken von der Rolle

Angelina Weinschenck, MT AG

Vor allem in größeren Applikationen findet sich oft ein Wildwuchs bei der Rechtevergabe. Oft werden Rechte nur mal eben zum Testen vergeben, weil irgendetwas nicht funktioniert – ohne diese jemals wieder zurückzunehmen.

Es kann insbesondere problematisch sein, weil ein User zur Laufzeit nicht mehr als 254 aktive Rollen haben kann. Das bedeutet im Umkehrschluss leider nicht, dass einem User nur 254 Rollen zugewiesen werden können *(siehe Listing 1)*. Es funktioniert, lediglich im „alert-log" erscheint die Fehlermeldung, dass nicht alle Rollen aktiviert sind: „Maximum of xxx enabled roles exceeded for user MANY_ROLES. Not loading all the roles".

In einem solchen Fall steht man vor einem Problem. Es kann sein, dass die Applikation funktioniert, der User aber nicht alle Rollen aktiviert hat. Das ist doch ein klares Zeichen dafür, dass zu viele Rollen in der Datenbank vergeben sind. Auf der anderen Seite kann es natürlich sein, dass ein Teil der Fehlermeldungen, die aus der Applikation kommen, durch fehlende Rechte verursacht sind. Was tun in so einem Fall?

Einen Überblick verschaffen

Erst einmal sollte ein Überblick über die User erstellt werden, auf die das Problem zutrifft. Man sollte sich darüber im Klaren sein, dass dieses Problem nur gemeinsam mit allen Verantwortlichen zu lösen ist. Einfach Rollen beziehungsweise Rechte zu verschlanken, ist nicht zielführend und führt eher zu noch mehr Problemen bis hin zum Produktionsausfall. Das Statement in *Listing 2* liefert eine Liste der User, sortiert nach Anzahl der Rollen. Wenn User mit mehr als 254 beziehungsweise 148 (XE) zugewiesenen Rollen auftauchen, ist das der erste Anhaltspunkt.

```
CREATE USER many_roles IDENTIFIED BY oracle;
GRANT create session TO many_roles;
SET PAGES 1000;
spool /tmp/egal.sql
SELECT 'create role r'||level||';' FROM dual CONNECT BY level < 512;
SELECT 'grant r'||level||' to many_roles;' FROM dual CONNECT BY level <
512;
spool off;
@/tmp/egal.sql
CONNECT many_roles/oracle
```

Listing 1: Erzeugen von Rollen und Zuweisung an einen Benutzer