Oracle RMAN ohne Shell – Good Bye Crontab

Johannes Kraus, Herrmann & Lenz Services GmbH

Oracle-Datenbank-Sicherungen sind von enormer Wichtigkeit und im produktiven Betrieb nicht mehr wegzudenken. Mithilfe der erweiterten Möglichkeiten des Oracle Scheduler unter Verwendung von gespeicherten Credentials ist es nun möglich, Backups ohne Shell-Skripte und ohne Crontab aus der Datenbank heraus zu starten. Der Artikel zeigt die Vor- und Nachteile sowie eine Beispiel-Implementierung.

Das Erstellen einer Oracle-Datenbank-Sicherung ist aufgrund der immer schneller wachsenden Datenmengen und vor allem auch aus sicherheitstechnischen Gesichtspunkten enorm wichtig. Im Zuge dieser Entwicklung haben sich verschiedene Hochverfügbarkeitslösungen etabliert, die jedoch alle nicht funktionieren würden, wenn nicht zum richtigen Zeitpunkt ein passendes und geeignetes Backup vorhanden wäre. Diese Hochverfügbarkeitslösungen sind jedoch kein Grund dafür, auf eine dauerhafte Erstellung neuer Backups zu verzichten.

In den meisten Fällen werden für eine Sicherung verschiedene plattformabhängige Skripte erstellt und mithilfe der unterschiedlichsten Jobverwaltungs-Tools wie Crontab gestartet. Dabei sind Passwörter oft im Klartext und somit lesbar in den Skripten abgespeichert. An dieser Stelle entsteht ein erhebliches Sicherheitsproblem, da Usernamen und Passwörter von allen Personen mit Zugriff auf den Server eingesehen werden können. Die erweiterten Möglichkeiten des Oracle Scheduler ab Version 12*c* erlauben es unter Verwendung des Package „DBMS_CREDENTIAL" nun, Backups ohne Shell-Skript und ohne Verwendung der Crontab aus der Datenbank heraus zu starten.

Installation und Konfiguration

Um ein Backup mithilfe des Oracle Scheduler starten zu können, sind zuerst ein paar Vorarbeiten erforderlich. In dem folgenden Beispiel werden alle Sicherungen über einen zu erstellenden Benutzer durchgeführt *(siehe Listing 1)*. Dieser Benutzer erhält ausschließlich die benötigten Rechte, um die Sicherung als externen Job zu starten.

Es gibt Gründe, warum das Ganze unter einem Extrabenutzer erfolgen muss: Wie bei vielen anderen Themen gilt auch hier die Regel, dass nach Möglichkeit keine Aktivitäten unter einem administrativen Benutzer laufen sollen. Zum einen ist eine strikte Kapselung der Aufgaben innerhalb der Datenbank erwünscht, zum anderen lässt sich so aus sicherheitstechnischen Hintergründen eine starke Rechte-Verwaltung etablieren. Benutzer sollen immer nur genau die Rechte erhalten, die sie für ihren Existenzgrund auch benötigen – nicht mehr und nicht weniger.

Unter dem neu erstellten Benutzer müssen nun die Zugangsdaten für den Oracle-Betriebssystem-User beziehungsweise den User angelegt werden, der die Rechte besitzt, die Anwendung „RMAN" aufzurufen. Sollte dieser Benutzer kein Passwort haben, muss dieses erstellt werden. Die Zugangsdaten können wie in dem folgenden Beispiel mithilfe des Package „DBMS_CREDENTIAL" angelegt werden *(siehe Listing 2)*.

Der „credential_name" ist dabei eine freie, jedoch eindeutige Referenz, über die man die Zugangsdaten durch den Scheduler anspricht. Eine Übersicht der bereits erstellten Zugangsdaten lässt sich mithilfe des SQL „select * from all_credentials;" darstellen. *Abbildung 1* zeigt das Ergebnis, wie hier im Oracle SQL Developer zu sehen.

Über dieses benötigte Credential beziehungsweise die dort hinterlegten Zu-

```
CREATE USER backup IDENTIFIED BY strenggeheim
DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp;
GRANT CREATE SESSION TO backup;
GRANT ALTER SESSION TO backup;
GRANT CREATE JOB TO backup;
GRANT CREATE EXTERNAL JOB TO backup;
GRANT CREATE CREDENTIAL TO backup;
```

Listing 1

```
BEGIN
   DBMS_CREDENTIAL.create_credential(
       credential_name    => 'oracle_os_login',
       username           => 'oracle',
       password           => 'geheim'
   );
END;
/
```

Listing 2

	OWNER	CREDENTIAL_NAME	USERNAME	WINDOWS_DOMAIN	COMMENTS	ENABLED
1	BACKUP	ORACLE_OS_LOGIN	oracle	(null)	(null)	TRUE
2	BACKUP	SYSDBA_LOGIN	sys	(null)	SYSDBA LOGIN FÜR RMAN BACKUPS	TRUE

Abbildung 1: Ausgabe der View „all_credentials“

gangsdaten wird der Scheduler auf das Betriebssystem zugreifen und den RMAN-Prozess starten. Im nächsten Schritt wird das Backup eingerichtet *(siehe Listing 3)*.

Der oben angegebene anonyme Block dient der Erstellung des Scheduler Jobs. Zuerst wird in einer Variablen das RMAN-Skript deklariert. Je nach Konfiguration der Datenbank ergeben sich an dieser Stelle mehrere unterschiedliche Lösungen. Bei der hier verwendeten Test-Datenbank wurden die Parameter „db_recovery_file_dest“ und „db_recovery_file_dest_size“ gesetzt. Somit müssen in dem RMAN-Skript keine Pfad-Angaben über den Speicherort des Backups angegeben werden. Wer diese beiden Parameter nicht gesetzt hat, kann nach wie vor verschiedene Disk-Channels deklarieren oder mit den entsprechenden „Configure“-Befehlen des RMAN arbeiten.

```
DECLARE
   v_job_script VARCHAR2(32767);
BEGIN
   v_job_script := ,connect target /
          backup check logical database plus archivelog delete all input;
          DELETE NOPROMPT OBSOLETE;';

   DBMS_SCHEDULER.create_job(
       job_name           => 'RMAN_FULL',
       job_type           => 'BACKUP_SCRIPT',
       job_action         => v_job_script,
       credential_name    => 'ORACLE_OS_LOGIN',
       start_date         => trunc(sysdate)+18/24,
       repeat_interval    => 'FREQ=DAILY;INTERVAL=1;BYHOUR=18',
       enabled            => TRUE,
       comments           => 'Oracle RMAN Full Backup incl. Archive-Logs'
   );
END;
/
```

Listing 4

```
CONFIGURE RETENTION POLICY TO REDUNDANCY 3;
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO
'/ora00/oradata/DB12C/%F';
CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'/ora00/oradata/DB12C/snapcf_DB12C.f';
```

Listing 4

```
BEGIN
   DBMS_CREDENTIAL.create_credential(
       credential_name    => 'sysdba_login',
       username           => 'sys',
       password           => 'change_on_install',
       database_role      => 'SYSDBA',
       comments           => 'SYSDBA LOGIN FÜR RMAN BACKUPS'
   );
END;
/
```

Listing 5

Eine Alternative wäre die Erstellung und Speicherung der RMAN-Befehle als Skript im Betriebssystem. Dazu ist es notwendig, den gesamten Pfad inklusive des Namens des Backup-Skripts anzugeben. Bei RAC-Datenbanken hat dies den Vorteil, dass das Skript im ACFS abgelegt werden kann. Somit ist es an einer zentralen Stelle verfügbar, auf den alle Knoten Zugriff haben. Der Scheduler wird in diesem Fall als reines Jobverwaltungs-Tool verwendet, um das Backup zu starten. Durch die zentrale Ablage entfällt eine mehrfache Haltung und Wartung der Skripte. Selbstverständlich ist es auch im RAC-Umfeld möglich, die im Beispiel verwendete Methode zu benutzen.

Ein weiteres interessantes Szenario ist der Einsatz des Data Guard oder Active Data Guard. Oft wird in solchen Umgebungen die Sicherung auf der Standby-Datenbank erstellt. Da sich diese jedoch im „Mount“-Modus befinden, steht der Oracle Scheduler an dieser Stelle nicht zur Verfügung. Er bietet jedoch die Möglichkeit, Jobs auf einem Remote Host zu starten. Eine Überlegung wäre also, ob die Primary-Datenbank und deren Scheduler als Trigger für eine Sicherungserstellung auf einem Remote-Host und somit auf einer Standby-Datenbank verwendet werden kann. Dieses Szenario konnte bisher leider nicht getestet werden, sodass bisher nur die Überlegung im Raum steht.

Die RMAN-Konfiguration kann, um auf das Beispiel zurückzukommen, für alle Bedürfnisse unterschiedlich konfiguriert sein. In der verwendeten Test-Datenbank sind folgende Einstellungen explizit gesetzt *(siehe Listing 4)*.

Darüber hinaus ist zu erwähnen, dass in diesem Fall kein RMAN Catalog zum Einsatz kam. Sofern sich dieser jedoch im Einsatz befindet, muss das Backup-Skript um den entsprechenden Zusatz ergänzt sein. An dieser Stelle müssen leider immer noch der

RMAN-Catalog-User und das dazugehörige Passwort im Klartext eingetragen werden.

Der Vorteil – beispielsweise im Vergleich zur Crontab – besteht darin, dass der Kreis der Personen, die den Usernamen und das Passwort einsehen können, minimiert ist. Dies ist ab sofort nur noch möglich, wenn ein mit entsprechenden Berechtigungen vorhandener Datenbank-User zur Verfügung steht.

Sollte das Backup nicht unter dem Benutzer oder einem Benutzer mit der gleichen Gruppenzugehörigkeit gestartet werden, unter dem auch die Datenbank läuft, ist ein „target /" nicht möglich. Es muss also einen Weg geben, sich an der Datenbank mit SYSDBA-Rechten anzumelden. Dazu wird ein weiteres Credential angelegt *(siehe Listing 5)*.

Mit dessen Hilfe ist es nun möglich, sich ohne sichtbare Login-Daten als „SYSDBA" mit der Datenbank zu verbinden. Ein weiterer Vorteil ist, dass der RMAN-Job nur noch die Backup-Befehle enthält. Damit das Backup jedoch erfolgreich erstellt werden kann, ist es nötig, den Job mit abweichenden Parametern einzurichten *(siehe Listing 6)*.

Im Vergleich zum ersten Job hat sich zum einen das RMAN-Skript verändert, zum anderen wurde mit „DBMS_SCHEDULER.set_attribute()" auf eine weitere Prozedur des Package „DBMS_SCHEDULER" zugegriffen. Diese sorgt dafür, dass der Job eine Verbindung zur Datenbank mithilfe eines hinterlegten Credential für einen bestimmten Job („RMAN_FULL") öffnet. In diesem Fall handelt es sich um das Credential „SYSDBA_LOGIN".

Bei der Job-Erstellung selbst werden mehrere Parameter übergeben. Ein alter Bekannter, jedoch mit neuen Möglichkeiten, ist der Übergabeparameter „JOB_TYPE". Mit 12*c* kamen für diesen drei neue Typen hinzu:

- EXTERNAL_SCRIPT
- SQL_SCRIPT
- BACKUP_SCRIPT

In unserem Fall ist der Parameter „BACKUP_SCRIPT" von Bedeutung. Er macht es überhaupt erst möglich, dass das Binary „RMAN" von der Datenbank über ein Credential ausgeführt werden kann. Bei einer weiteren näheren Betrachtung des RMAN-Skripts fällt auf, dass keine Angaben über ein mögliches Logging der RMAN-Ausgabe gemacht wurden. Der Grund liegt darin, dass seit 12*c* die Ausgaben der Programme wie RMAN in der Datenbank abgespeichert und über die View „ALL_SCHEDULER_JOB_RUN_DETAILS" abgerufen werden können. Dazu bekam die View vier neue Spalten:

- ERRORS
- OUTPUT
- BINARY_ERRORS
- BINARY_OUTPUT

```
DECLARE
   v_job_script VARCHAR2(32767);
BEGIN
   v_job_script := ,backup check logical database plus archivelog delete
all input;
                  DELETE NOPROMPT OBSOLETE;';

  DBMS_SCHEDULER.create_job(
      job_name           => 'RMAN_FULL',
      job_type           => 'BACKUP_SCRIPT',
      job_action         => v_job_script,
      credential_name    => 'ORACLE_OS_LOGIN',
      start_date         => trunc(sysdate)+18/24,
      repeat_interval    => 'FREQ=DAILY;INTERVAL=1;BYHOUR=18',
      enabled            => TRUE,
      comments           => 'Oracle RMAN Full Backup incl. Archive-Logs'
   );

   DBMS_SCHEDULER.set_attribute(
      name       => 'RMAN_FULL',
      attribute  => 'connect_credential_name',
      value      => 'SYSDBA_LOGIN'
   );
END;
/
```

Listing 6

```
CREATE OR REPLACE PROCEDURE output_blob(p_input BLOB) IS
   l_offset   BINARY_INTEGER := 1;
   l_length   BINARY_INTEGER;
   l_range    BINARY_INTEGER := 32767;
BEGIN
   l_length := LENGTH(p_input);

   WHILE l_offset + l_range <= l_length
   LOOP
       DBMS_OUTPUT.put_line(UTL_RAW.cast_to_varchar2(DBMS_LOB.SUBSTR(p_input, l_range, l_offset)));

       l_offset := l_offset + l_range;
   END LOOP;
   DBMS_OUTPUT.put_line(UTL_RAW.cast_to_varchar2(DBMS_LOB.SUBSTR(p_input, LENGTH(p_input) - l_offset + 1, l_
offset)));
END output_blob;
/
```

Listing 7

```
DECLARE
V_OUTPUT BLOB;
BEGIN
SELECT BINARY_OUTPUT into V_OUTPUT FROM ALL_SCHEDULER_JOB_RUN_DETAILS
WHERE LOG_ID=(SELECT MAX(LOG_ID) FROM ALL_SCHEDULER_JOB_RUN_DETAILS
WHERE JOB_NAME= 'RMAN_FULL');

   output_blob(V_OUTPUT);
END;
/
```

Listing 8

```
SELECT rs.status, rs.start_time, rs.end_time, ro.output
   FROM v$rman_status rs
   JOIN v$rman_output ro ON ro.rman_status_recid = rs.recid
 WHERE rs.session_recid = (select max(session_recid) from v$rman_status)
 order by ro.recid;
```

Listing 9

Die in diesem Fall relevanten Spalten sind „OUTPUT“ und „BINARY_OUTPUT“. Der Unterschied liegt darin, dass „OUTPUT“, eine VARCHAR2(4000)-Spalte ist und die „BINARY_OUTPUT“ vom Typ „BLOB“. Die Spalte „OUTPUT“ beinhaltet die ersten 4.000 Bytes der RMAN-Ausgabe, während „BINARY_OUTPUT“ die gesamte Ausgabe enthält. Leider lassen sich BLOB-Spalten nicht ohne Weiteres über ein einfaches SQL-Statement auslesen. Hier kann durch verschiedene Lösungen Abhilfe erfolgen. Da gerade bei RMAN-Backups der Output immer als Ganzes zu sehen ist, wurde eine Prozedur erstellt, die das BLOB in 32.767-Byte-Blöcken ausliest und über DBMS_OUTPUT ausgibt *(siehe Listing 7)*.

Soll die genannte Prozedur unter dem neuen Benutzer erstellt werden, benötigt dieser mit „GRANT CREATE PROCEDURE TO backup;“ ein weiteres Recht. Mithilfe des folgenden anonymen Blocks kann das BLOB-Feld und somit das Logging des RMAN ausgelesen werden *(siehe Listing 8)*.

Wem der Schritt über das BLOB zu aufwändig ist, kann auch einen anderen Weg beschreiten. Sowohl der Status des Backups als auch die Ausgabe des RMAN stehen in folgenden VIEWS:

- V$RMAN_STATUS
- V$RMAN_OUTPUT

Mit einem SQL-Statement kann sowohl der Status des Backups als auch die gesamte Ausgabe des RMAN für ein bestimmtes Backup eingesehen werden. Wichtig dabei ist, dass der User mindestens die SELECT-Rechte auf die beiden genannten Views besitzt *(siehe Listing 9)*.

Wenn die RMAN-Log-Informationen also bereits in der Datenbank vorhanden sind, wieso braucht es dann diese beiden neuen OUTPUT-Felder überhaupt? Seit 12*c* können verschiedene Skripte aus der Datenbank heraus gestartet werden. Und eben diese hätten keine Ausgabe in der Datenbank, wenn die neuen Spalten nicht hinzugefügt worden wären.

Vor- und Nachteile

Die Umstellung des Backups auf den Oracle Scheduler bringt einige Vor- sowie auch Nachteile mit sich. Die Vorteile sind:

- Sicherheit (Usernamen und Passwörter sind nicht mehr für jeden sichtbar)
- Login-Daten innerhalb der Datenbank (bei Credentials)
- Verbesserte Monitoring-Möglichkeiten gegenüber der Crontab
- Zentraler Zugang aller Informationen über die Datenbank
- Zugriff auf alle vorhandenen Möglichkeiten des Oracle Scheduler
- Zugriff nur durch Personal mit Datenbank-Zugang (Rechte-Management)
- Plattform-Unabhängigkeit

Als Nachteil lässt sich aufführen:

- Initiale Einrichtung dauert etwas länger als bei Crontab

Fazit

Oracle hat in der Vergangenheit seine Job-Verwaltung kontinuierlich und konsequent ausgebaut. Der erste große Schritt war die Umstellung von „DBMS_JOBS“ auf „DBMS_SCHEDULER“ und brachte viele Neuerungen und Möglichkeiten mit sich. Mit den nun erweiterten Möglichkeiten des „DBMS_SCHEDULER“ wurde ein neuer Meilenstein geschaffen, der neue Möglichkeiten und Optionen mit sich bringt.

Datenbank-Administratoren können jetzt viele Skript-Arten wie SQL- und Shell-Skripte, die von der Datenbank zur Verfügung gestellt werden oder für die Datenbank gedacht sind, aus der Datenbank heraus starten. Dies sorgt für eine bessere Übersicht und führt zu einer Zentralisierung der Aufgabenbereiche.

Zusätzlich kann mit dem bereits vorhandenen Oracle-Rechtekonzept für mehr Sicherheit gesorgt beziehungsweise Zugriffsbeschränkungen durchgesetzt werden. Zudem vereinfacht der neue Scheduler die Überwachung der ausgeführten Skripte durch eingesetzte Monitoring-Tools wie beispielsweise das Monitoring Module von Herrmann & Lenz.

Ein weiterer Vorteil ist die schnellere Einsehbarkeit des Job-Status und die damit einhergehende verbesserte Reaktion auf auftretende Fehler. Der Scheduler ist inzwischen ein mächtiges Werkzeug geworden, das nun nur noch verstärkt eingesetzt werden sollte.

Weiterführende Links

[1] DBMS_SCHEDULER: https://docs.oracle.com/database/121/ARPLS/d_sched.htm

[2] View ALL_SCHEDULER_JOB_RUN_DETAILS: https://docs.oracle.com/cloud/latest/db121/REFRN/refrn20386.htm#REFRN20386

[3] View ALL_CREDENTIALS: https://docs.oracle.com/database/121/ARPLS/d_credential.htm

[4] HL-Monitoring: http://www.hl-solutions.de/produkte/monitoring-module

Johannes Kraus
johannes.kraus@hl-services.de