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

Wie sieht es mit den Top-Rollen im System aus? Welche Rolle wurde am meisten vergeben? Wenn man das SQL Statement etwas abwandelt, wird auch diese Frage schnell beantwortet *(siehe Listing 3)*.

Wenn hier die DBA-Rolle unter den Top 10 ist, besteht ein Problem. Wenn ein Applikations-Administrator sagt, er brauche eine Rolle unbedingt, bekommt er diese meist auch. Ob es sinnvoll und vor allem notwendig ist, wird selten überprüft. In den meisten Fällen ist dies jedoch nicht notwendig und sicherheitstechnisch eine Katastrophe. Bei diesen Usern sollte wirklich im Einzelnen untersucht werden, ob die DBA-Rechte wirklich erforderlich sind *(siehe Listing 4)*. Warum sollte ein Benutzer, der eigentlich nur Select-Statements ausführt, das Recht haben, beliebige Tabellen zu löschen?

Vorsicht vor den „ANY"-Rechten

In der DBA-Rolle sind auch etliche „ANY"-Privilegien enthalten. Vor allem die „CREATE ANY ..."-Rechte sind für den Datenbank Betrieb nicht ungefährlich. Mit einigen Kniffen kann man mit dem Privileg „Create any directory" die komplette Datenbank und danach auch den Server übernehmen. Um diesem Problem vorzubeugen, sollte kontrolliert werden, ob es neben der DBA-Rolle noch andere Rollen gibt, die „ANY"-Privilegien weitergeben oder enthalten. Diese Rollen beziehungsweise die entsprechenden User lassen sich ebenfalls mit einfachen SQL-Statements ermitteln *(siehe Listing 5 und 6)*.

Jetzt sollte mit den Prozessverantwortlichen und Entwicklern über die gefundenen Rollen und Benutzer darüber gesprochen werden, ob diese Rechte in diesem Umfang wirklich erforderlich sind. Zweifelsfrei ist es immer eine gute Idee, die Applikation auf einem Testsystem mit einem User ohne die „Create Any"-Rechte zu installieren. Wenn dies gelingt, können die Rechte im Allgemeinen dauerhaft entzogen werden.

Rollen weitergeben

Eine weitere wichtige Frage, die man sich stellen sollte, ist, ob es Rollen gibt, die mit der Admin-Option vergeben sind. Wenn ja, sollte dies auch schleunigst geändert werden. Nur der Datenbank-Administrator sollte in der Lage sein, Rollen und Privilegien weitergeben zu können *(siehe Listing 7)*.

Benutzer und Rollen ohne Funktion

Ein weiteres Problem kann sich ergeben, wenn es Rollen gibt, die niemandem zugewiesen und die keine vorinstallierten System-Rollen sind. Hier kann es sein, dass diese Rollen von der Applikation dynamisch zugewiesen werden. Hierzu sollte man ebenfalls mit dem Applikationsverantwortlichen Rücksprache halten *(siehe Listing 8)*. Das Statement im *Listing 9* verschafft Klarheit darüber, ob es im System Rollen ohne zugewiesene Rechte oder andere Rollen gibt.

Gibt es Benutzer, die außer den Rollen „Connect" und „Ressource" keine Rechte

```
SELECT grantee, count(*) cnt_roles
FROM dba_role_privs
GROUP BY grantee
ORDER BY cnt_roles DESC;
```

Listing 2: Anzeige der User nach Anzahl der Rollen

```
SELECT granted_role, count(*) cnt_roles,
  round(count(*)*100/sum(count(*)) over (), 2) pct
FROM dba_role_privs
GROUP BY granted_role
ORDER BY cnt_roles DESC;
```

Listing 3: Anzeige der Rollen nach Anzahl der Zuweisungen

```
SELECT grantee, admin_option
FROM dba_role_privs
WHERE granted_role='DBA'
AND grantee NOT IN(
SELECT username
FROM dba_users
WHERE oracle_maintained = 'Y' );
```

Listing 4: Ermitteln der User mit DBA-Privileg

```
SELECT grantee, privilege, admin_option, common
FROM dba_sys_privs
WHERE privilege like '%ANY%';
```

Listing 5: Ermitteln der Benutzer und Rollen mit „Any"-Privilegien

```
SELECT grantee, privilege, admin_option, common
FROM dba_sys_privs
WHERE privilege like 'CREATE ANY%';
```

Listing 6: Ermitteln der Benutzer und Rollen mit „Create Any"-Privilegien

```
SELECT grantee, privilege, admin_option, common
FROM dba_sys_privs
WHERE admin_option ='YES'
 AND grantee NOT IN (
SELECT username
FROM dba_users
WHERE oracle_maintained='Y')
 AND grantee NOT IN (
SELECT role
FROM dba_roles
WHERE oracle_maintained='Y');
```

Listing 7: Benutzer und Rollen mit Admin-Option, die nicht von Oracle bei der Installation erstellt wurden

```
SELECT role
FROM dba_roles
WHERE role NOT IN (
      SELECT granted_role
      FROM dba_role_privs)
  AND oracle_maintained != 'Y';
```

Listing 8: Ermitteln von Rollen, die keinem User und keiner Rolle zugewiesen sind

```
SELECT role
FROM dba_roles
WHERE oracle_maintained != 'Y'
  AND role NOT IN (
SELECT grantee
FROM dba_role_privs)
  AND role NOT IN(
SELECT grantee
FROM dba_tab_privs)
  AND role NOT IN (
SELECT grantee
FROM dba_sys_privs);
```

Listing 9: Ermitteln von Rollen ohne Privilegien

```
SELECT username
FROM dba_users
WHERE oracle_maintained != 'Y'
 AND username NOT IN (
SELECT grantee
FROM dba_sys_privs
WHERE privilege = 'CREATE SESSION')
 AND username NOT IN (
SELECT grantee
FROM dba_role_privs
WHERE granted_role IN (
SELECT grantee
FROM dba_sys_privs
WHERE privilege = 'CREATE SES-
SION'));
```

Listing 10: User ohne das Privileg „Create Session“, die auch keine Rolle mit dem Privileg „Create Session“ zugewiesen bekommen haben

haben? Diese Benutzer können entweder prozessbedingt angelegt worden sein oder aber sie werden nicht mehr benötigt. In jedem Fall sollte mit den Entwicklungs- und den Prozessverantwortlichen Rücksprache dazu gehalten werden, was mit diesen Benutzern passieren soll *(siehe Listing 10)*.

Rolle in Rolle?

Ein viel größeres Problem stellen die Rollen dar, deren Rechte komplett in anderen Rollen enthalten sind. Mit dem SQL Statement in *Listing 11* erhält man die Rollen, die näher zu untersuchen sind. Im Ergebnis werden auch einige Rollen sein, die die gleichen Privilegien haben; es kann jedoch sein, dass dies prozessbedingt notwendig ist.

Fazit

Denkbar ist noch folgendes Szenario: Wie man in *Tabelle 1* sieht, hat Rolle 1 die Update-, Insert-, Select- und Delete-Rechte komplett. Rolle 2 hingegen hat nur die Rechte „Update“, „Insert“ und „Delete“. Rolle 3 besitzt das Select-Recht.

In *Tabelle 2* hat Rolle 1 die Rechte von Rolle 2 und Rolle 3 durch Zuweisung erhalten. Prinzipiell sind die Rechte gleich. In der Praxis kann dies jedoch unverhoffte Resultate haben. Wenn Rolle 3 ein zusätzliches Privileg erhält, erhält Rolle 1 dies implizit auch. Durch ein solches Konstrukt können die Rollen zwar übersichtlicher gestaltet werden, es ist allerdings fraglich, ob die Rechtevergabe dadurch transparenter und einfacher zu administrieren ist.

Patentrezepte kann und wird es nicht geben. Es ist jedoch wichtig, dass gerade bei dem zentralen Thema „Rechte-Vergabe und Verwaltung“ die Entwickler, Administratoren und Prozessverantwortlichen Mitspracherecht haben und aktuelle sowie zukünftige rechtliche Vorgaben berücksichtigt werden.

Rolle	Privilegien			
Rolle 1	U	I	S	D
Rolle 2	U	I		D
Rolle 3			S	

Tabelle 1: Alt

Rolle	Privilegien			
Rolle 2	U	I		D
Rolle 3			S	
Rolle 1	Rolle 2		Rolle 3	

Tabelle 2: Neu

Angelina Weinschenck
angelina.weinschenck@mt-ag.com

```
WITH cnt AS (SELECT grantee, count(*) anz FROM dba_tab_privs GROUP BY grantee),
xx AS (SELECT a.grantee sub_role, b.grantee master_role, count(*) cnt_identic
          FROM dba_tab_privs a
          JOIN dba_tab_privs b
          ON a.grantee != b.grantee
          AND a.privilege = b.privilege
          AND a.table_name = b.table_name
          AND a.grantor = b.grantor
          AND a.owner = b.owner
          WHERE a.grantee IN (SELECT role FROM dba_roles WHERE oracle_maintained != 'Y')
          AND b.grantee IN (SELECT role FROM dba_roles WHERE oracle_maintained != 'Y')
          HAVING count(*) > 1
          GROUP BY a.grantee, b.grantee
          ORDER BY a.grantee, b.grantee)
   SELECT master_role, sub_role, b.anz master_cnt, a.anz sub_count
          FROM xx, cnt a, cnt b
      WHERE master_role = b.grantee
          AND sub_role = a.grantee
          AND cnt_identic = a.anz;
```

Listing 11: Rollen mit den gleichen Tab-Privilegien