

Tipps & Tricks: Juni 2003

Bereich:	PL/SQL	Erstellung:	01/2005 HA
Versionsinfo:	9.2, 10.2, 11.1	Letzte Überarbeitung:	06/2009 EF

 [Als PDF Downloaden!](#)

Das Rollenkonzept in PL/SQL

Ist es Ihnen auch schon passiert, dass Sie in PL/SQL die Fehlermeldung erhalten haben "Table or View does not exist", obwohl Sie genau den gleichen Befehl alleinstehend problemlos absetzen können? Das liegt im Normalfall daran, dass Sie die entsprechenden Berechtigungen über eine Rolle erhalten haben.

Rollen wurden geschaffen, um die Verwaltung von Berechtigungen zu vereinfachen. Innerhalb von PL/SQL gibt es allerdings einige Fallstricke bezüglich Rollen. Was ist zu beachten?

Anonyme Blöcke

Innerhalb von Anonymen Blöcken sind Rollen generell wirksam. Hier verfügen Sie also über all ihre Rechte, egal, ob Sie sie direkt oder über eine Rolle erhalten haben. Dies gilt sowohl für Entwickler, die auf Tabellen oder Views zugreifen wollen, als auch für Enduser, die nur eine Prozedur aufrufen wollen, an der sie das EXECUTE-Recht haben. (Auch der EXEC-Befehl in SQL*Plus wird intern umgesetzt in einen Anonymen Block.)

Prozeduren / Funktionen / Packages

Sobald Sie als Entwickler ein in der Datenbank gespeichertes Programm schreiben wollen - Funktionen und Packages seien im nachfolgenden bei "Prozedur" eingeschlossen - , dann müssen Sie alle erforderlichen Rechte DIREKT erhalten haben, nicht über Rollen. Dabei macht es keinen Unterschied, ob Sie die Prozedur mit definer oder [invoker rights](#) schreiben. Dies gilt sowohl für Objekt-Berechtigungen auf Tabellen oder Views als auch für EXECUTE-Berechtigungen auf fremde Prozeduren.

Wie sieht es aus mit Berechtigungen bei der Ausführung einer Prozedur?

- Wurde sie mit Definer-Rechten erstellt, so braucht man nur das EXECUTE-Recht darauf, um sie aufzurufen
- Wurde sie mit Invoker-Rechten erstellt, so braucht man auch alle innerhalb der Prozedur nötigen Berechtigungen.
- Sind (in 11g) verwendete Ressourcen über ACLs geschützt (z.B. bei Verwendung von UTL_SMTP oder UTL_MAIL), so muss der Aufrufende das entsprechende Privileg ebenso haben.
- Für den Aufruf aus SQL heraus (bzw. im Anonymen Block) reicht es auch hier wieder, wenn man die Rechte über eine Rolle erhalten hat. Wollen Sie sie aber innerhalb einer eigenen Prozedur aufrufen, so brauchen Sie die entsprechenden Rechte in der Regel wiederum DIREKT. Dies gilt für die Datenbank-Versionen 8i, 10g und 11g. Nur in Version 9i reicht es aus, wenn Sie die Rechte über eine Rolle erhalten haben.

Beispiel:

```
CREATE PROCEDURE ha.dummy
AUTHID CURRENT_USER AS
  CURSOR c IS SELECT ename FROM scott.emp WHERE deptno = 10;
BEGIN
```

```
FOR r IN c LOOP
    DBMS_OUTPUT.PUT_LINE(r.ename);
END LOOP;
END;
/

GRANT EXECUTE on ha.dummy TO mp;

CREATE PROCEDURE mp.dummy
AS
BEGIN
    ha.dummy;
END;
/
```

HA braucht in diesem Fall direkt das SELECT-Recht auf scott.emp. mp braucht ebenfalls das SELECT-Recht auf scott.emp, allerdings ist es abhängig von seiner Datenbank-Version, ob er das Recht direkt braucht (8i, 10g, 11g), oder ob eine Rolle ausreicht (9i).

Secure Application Roles

Ab Version 9i können Rollen angelegt werden, die nur über eine Prozedur (wiederum incl. Package) aktiviert werden können.

Die Idee dahinter: alle zur Ausführung einer Applikation nötigen Berechtigungen werden an diese Rolle vergeben. Innerhalb der Applikation wird geprüft, wer sie wie aufgerufen hat (User, IP-Adresse, Zugang über Proxy...) und dementsprechend wird diese Rolle aktiviert oder nicht.

Voraussetzungen:

- Anlegen der Rolle mit IDENTIFIED USING-Klausel
- Die Prozedur zur Identifizierung (und ggf. alle sie aufrufenden Prozeduren) muss mit Invoker-Rechten arbeiten.
- Innerhalb der Prozedur wird diese Rolle aktiviert mit DBMS_SESSION.SET_ROLE.
- Enterprise Edition

Eine solche Rolle kann weder innerhalb von SQL (mit SET ROLE) noch innerhalb einer anderen Prozedur aktiviert werden, aber wenn Sie sie nicht wieder deaktivieren vor Beendigung der Applikation, so bleibt sie für den Rest der Session aktiv!

Beispiel:

```
CREATE ROLE APP_ROLE IDENTIFIED USING scott.dummy_proc;

GRANT SELECT ON scott.emp TO app_role;

CREATE Procedure scott.dummy_proc
AUTHID CURRENT_USER IS
    CURSOR c IS SELECT ename FROM scott.emp WHERE deptno = 10;
BEGIN
    IF check_function THEN
        DBMS_SESSION.SET_ROLE('APP_ROLE');
    END IF;

    IF DBMS_SESSION.IS_ROLE_ENABLED('APP_ROLE') THEN
        FOR r IN c LOOP
```

```
        DBMS_OUTPUT.PUT_LINE(r.ename);  
    END LOOP;  
END IF;  
END;  
  
GRANT EXECUTE ON dummy_proc TO mp;
```

Nun erhält mp keinerlei Rechte mehr an scott.emp. Innerhalb einer für mp zugänglichen Applikation wird dummy_proc aufgerufen. Über check_function werden etwaige Überprüfungen durchgeführt; sind diese erfolgreich, so wird die benötigte Rolle aktiviert.

Beachten Sie, dass APP_ROLE dem User mp NICHT (über GRANT) zugewiesen wird!

Nach dem Aufruf von dummy_proc ist APP_ROLE in der DD-View SESSION_ROLES sichtbar, vorher nicht.