

Rdb Release 7.1 & Role Based Security

A feature of Oracle Rdb

By Ian Smith
Oracle Rdb Relational Technology Group
Oracle Corporation

The examples in this article use SQL language from Oracle Rdb V7.1 and later versions.

Rdb Release 7.1 and Role Based Security

This is the third part of a series on Rdb security. The first article discussed *Role Based Security* for the current OpenVMS implementation of roles and users. In this article I will describe the new role and user model adopted for the newest release of Rdb.

This new model is based on the SQL:1999 SQL database language standard, which in turn was influenced by the Oracle RDBMS database system.

The following new security features were added to Oracle Rdb V7.1

- A new SECURITY CHECKING clause on CREATE and ALTER DATABASE
- A CREATE USER statement to define a user of the database
- A CREATE ROLE statement to define a role that can be granted to users of the database
- A CREATE PROFILE statement to define attributes for a class of users.
- New forms of the GRANT and REVOKE statements that allows a role to be granted to and revoke from users and roles.
- New built-in functions that return the user id values: UID, CURRENT_UID, SESSION_UID and SYSTEM_UID. These are companion functions for the existing USER, CURRENT_USER, SESSION_USER and SYSTEM_USER functions.
- New SHOW USER, SHOW PROFILE and SHOW ROLE statements in interactive SQL to display attributes and relationships between users and roles.

Security Checking is External

When you convert a database to V7.1 using either RMU/CONVERT, RMU/RESTORE (which performs an implicit RMU/CONVERT) or SQL IMPORT then the database is implicitly defined with SECURITY CHECKIN IS EXTERNAL. This informs Rdb that you want the operating system (i.e. OpenVMS) to provide the values for user and role values stored in the access control lists (ACL).

This setting represents the default behavior as used by all prior versions of Rdb and customers should see no change in behavior from previous releases.

Recent versions of the OpenVMS have supported a mechanism known as the PERSONA services which permit an application (such as Rdb) to impersonate other users. Rdb now uses these services when you include the PERSONA IS ENABLED clause. The default, as in prior releases, is to impersonate only the UIC of the user and not their full rights identifier list.

This new feature allows role-based security for remote users on TCP/IP network transport and for SQL/Services users.

Note: the *Rdb New and Changed Features Manual* incorrectly stated that PERSONA IS ENABLED was not supported with SQL/Services. Please ignore this statement; in fact this feature was designed for use by SQL/Services.

Security Checking is Internal

The major problem with the older model is that databases are not portable to different VMS systems because the binary values of the rights identifiers might be different. The AUTHORIZATION tools ADD/IDENTIFIER command will use the next available binary value for new rights identifiers, and on different systems the same rights identifier name may have a different binary value. It is also possible that the system managers assigned the next free UIC to a user on that system. In these cases SMITH might be represented as UIC [100,34] on one system but UIC [100,23] on the other.

The ACL is made up of entries (ACE) that include the binary user id and privilege mask. These differences between systems mean that the binary values stored in the ACL for each object may not match the correct (or any) user of the other system.

In prior releases the database could be IMPORTed using the NOACL clause so that only the default ACL remained for each object, or RMU/EXTRACT could be used with the /ITEM=REVOKE_ENTRY to allow these entries to be removed (usually before the EXPORT takes place). The output from RMU/EXTRACT/ITEM=PROTECTION can be later used to replace the access control information for the new database.

All Rdb V7.1 databases include two new system tables called RDB\$PROFILES and RDB\$GRANTED_PROFILES. These tables are used to hold values define using CREATE USER, CREATE PROFILE and CREATE ROLE. The RDB\$GRANTED_PROFILES table is used to store values assigned using the new GRANT statement for users and roles.

CREATE USER

This command can be used to define each user in the database. While it is possible for an anonymous user to attach to the database and perform queries, they will always be considered as only matching PUBLIC access and will never be permitted to perform DDL (data definition language) commands. The reason is that the default ACL requires a registered USER and since none is available for an anonymous user we prevent any attempts to perform DDL.

Rdb relies on the operating system to store and validate passwords therefore the user referenced must exist within the operating system register. Rdb validates the name during the CREATE command and stores the name as it is known to the operating system. The clause IDENTIFIED EXTERNALLY is required syntax and informs Rdb that this name is registered with the operating system.

Here are some users for our example.

```
SQL> create user JANET identified externally;  
SQL> create user JUAN identified externally;  
SQL> create user JACK identified externally;
```

Each of these statements will cause a unique user id to be stored by Rdb using the assigned UIC value.

To make this task easier, Rdb will automatically add users (when SECURITY CHECKING IS INTERNAL) when an unknown user is detected in the GRANT statement. So by simply granting access to a user an implicit CREATE USER statement is executed. An informational message will be reported in interactive SQL as shown here.

```
SQL> grant all on table * to FRANCIS;
%RDB-W-META_WARN, metadata successfully updated with the reported
warning
-RDMS-W-PRFCREATED, some users or roles were created
```

In addition the creator of the database is also implicitly added as a user.

```
SQL> show user
Users in database with filename db$:userswithroles
    FRANCIS
    JACK
    JANET
    JUAN
    SAM
```

CREATE ROLE

As we discussed in an earlier article roles closely resemble rights identifiers on OpenVMS. Therefore, when you create a role as IDENTIFIED EXTERNALLY Rdb requests the rights id value from VMS. If you use the NOT IDENTIFIED clause then Rdb will generate a distinct value. Care is taken to use distinct ranges of values so that OpenVMS selected values should never be the same as those used by Rdb.

Note: Rdb creates a special sequence in the database called RDB\$ROLE_PROFILE_ID_SEQ from which new values are fetched.

If a database contains a NOT IDENTIFIED role then it cannot be converted to a SECURITY CHECKING IS EXTERNAL database. The database administrator is required to use DROP ROLE ... CASCADE to remove those entries from each ACL and the RDB\$PROFILES table.

Rdb will implicitly create roles if you grant unknown roles to users.

```
SQL> grant EMPLOYEE to dora, sam;
%RDB-W-META_WARN, metadata successfully updated with the reported
warning
-RDMS-W-PRFCREATED, some users or roles were created
SQL> show role EMPLOYEE
EMPLOYEE
  Identified externally
  No roles have been granted to this role
  This role is granted to:
    DORA
    SAM
```

CREATE PROFILE

This feature is closely related to the CREATE USER command and allows some session characteristics to be assigned from a single source. For instance, all users performing business queries will be described using the **Business Query Users** profile. A profile specifies the default transaction and allowed transaction modes.

The default transaction is activated from applications using the START DEFAULT TRANSACTION or DECLARE DEFAULT TRANSACTION statements. The TRANSACTION MODES defined for the PROFILE are limited to those defined by the CREATE or ALTER DATABASE statement. For example, if the database administrator has denied the BATCH UPDATE transaction mode then it will not be available to the user even if specified by the CREATE PROFILE statement.

Note: In future releases more attributes will be added to the CREATE PROFILE command such as maximum rows returned from a query, and other query limits.

This example shows the profile being created and then assigned to a user:

```
SQL> create profile business_query_user
cont> comment is 'Limit types of transactions'
cont> default transaction read only wait 10
cont> transaction modes (read only, shared read);
SQL> show profile business_query_user
BUSINESS_QUERY_USER
Comment:      Limit types of transactions
Transaction modes (read only, shared read)
Default transaction read only wait 10
```

Note: Rdb creates a special sequence in the database called RDB\$QUOTA_PROFILE_ID_SEQ from which new values are fetched to uniquely identify this profile.

New GRANT and REVOKE syntax

There are now two formats for the GRANT and REVOKE statements in SQL. Those that grant privileges to users for specific objects, and those that grant roles to users or roles.

This second syntax is very simple and allows one or more roles to be granted to one or more roles or users. In our example we will create a hierarchy of roles; a manager is implicitly a supervisor and an employee; a supervisor is also an employee. Finally we grant the manager role to JANET.

```
SQL> grant employee to manager, supervisor;
SQL> grant supervisor to manager;
SQL> grant manager to JANET;
```

The SHOW USER statement will display the accumulated roles now available to JANET

```
SQL> show user JANET
JANET
Identified externally
Account is unlocked
Granted roles:
    MANAGER
```

Conversely the SHOW ROLE statement will show how this role is used.

```
SQL> show role MANAGER
MANAGER
  Identified externally
  Granted roles:
    EMPLOYEE
    SUPERVISOR
  This role is granted to:
    JANET
```

In addition a new REVOKE ALL ROLES clause has been added so that roles can be quickly revoked, and do not need to be enumerated. For instance, when JUAN is promoted and is no longer responsible for operations in this database a single REVOKE statement is enough to remove him from the database.

```
SQL> revoke all roles from JUAN;
SQL> show user JUAN
JUAN
  Identified externally
  Account is unlocked
  No roles have been granted to this user
```

What is a UID?

The UID function was added for Oracle RDBMS compatibility and it simply returns the user id for the current user. Since Rdb also includes CURRENT_USER, SESSION_USER and SYSTEM_USER there also exists companion functions that return the appropriate user id.

Applications that today store CURRENT_USER could store less data by using CURRENT_UID. A report could join the audit information with the RDB\$PROFILES table and translate the UID to a user name using the RDB\$PROFILE_NAME column.

Technical Tips: Frequently Asked Questions

Can I go back? If there are no roles defined using the NOT IDENTIFIED clause you simply use ALTER DATABASE ... SECURITY CHECKING IS EXTERNAL. Rdb does not automatically populate the USER and ROLE table but this can be easily achieved using the RDB_UNLOAD_PROFILE utility available as part of Rdb V7.1.

Can I use CREATE USER and CREATE ROLE in a SECURITY CHECKING IS EXTERNAL database? Yes, the system table RDB\$PROFILES is present in all Rdb V7.1 and later databases. These commands write values to the system tables that can be queried by applications. In particular the RDB_UNLOAD_PROFILE utility will read the database and generate a SQL script that can populate the RDB\$PROFILES table with user and role information. The user access checks are always enforced, so you can deny users access by using ALTER USER ... ACCOUNT LOCK.

Can I grant ROLES to ROLES? Yes, the user inherits the role hierarchy when they attach to the database. Using a hierarchy allows you to revoke just one branch and all dependent roles will be implicitly revoked. For instance, revoking MANAGER from JANET will implicitly remove the inherited SUPERVISOR and EMPLOYEE roles. In this example EMPLOYEE is inherited from SUPERVISOR.

How can I remove a role or user from the database? Rdb provides a CASCADE option for both DROP USER and DROP ROLE. This can be very useful tool for cleaning out the ACLs. Consider a user JAMES who has just retired. Even if you do not wish to use the user/role feature for day-to-day security it can be used to first create user JAMES and then a DROP USER JAMES CASCADE can be used to purge that user from all access control lists.

Do I need to create PUBLIC user? There is no need to create this user as it is implicit in every database. However, it can be created (the syntax IDENTIFIED EXTERNALLY is not required, nor allowed). The resulting definition can be combined with the ACCOUNT LOCK clause to prevent anonymous users querying the database.

```
SQL> attach 'filename db$:userandroles';
%SQL-F-ERRATTDEC, Error attaching to database db$:userandroles
-RDB-E-NO_PRIV, privilege denied by database facility
```

You also need to create public if you wish to explicitly grant roles to public.

```
SQL> grant CONTRACTOR to public;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-PRFNEXISTS, a user does not exist with the name "PUBLIC"
SQL> create user public;
SQL> grant CONTRACTOR to public;
```

Note: DROP USER PUBLIC CASCADE does not remove the PUBLIC entries from the access control lists because these entries are usually catchall entries that deny access.

How do I create the BATCH, INTERACTIVE and NETWORK roles for the special OpenVMS rights identifiers? These special identifiers are granted to the process by OpenVMS and cannot be created in the database. However, they can be granted when providing access as shown in this example.

```
SQL> grant select on table EMPLOYEES to JACK+INTERACTIVE;
```

This GRANT statement allows JACK to select from EMPLOYEES only if he is running from an interactive process.

ORACLE

Oracle Rdb
Rdb Release 7.1 Role Based Security
May 2002

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Oracle Corporation provides the software
that powers the internet.

Oracle is a registered trademark of Oracle Corporation. Various
product and service names referenced herein may be trademarks
of Oracle Corporation. All other product and service names
mentioned may be trademarks of their respective owners.

Copyright © 2002 Oracle Corporation
All rights reserved.