Stored Procedure 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 V7.0 and later versions.

Stored Procedure Security

In the last issue I discussed role-based security in Oracle Rdb. It seems to have been a popular topic so I will continue by examining some of the other uses of security identifiers in Rdb.

Definers and Invokers Rights

One of the most powerful security features in Oracle Rdb is that of definers rights stored routines. Stated simply it means that a procedure or function runs with the privileges and access rights of its definer, and not that of the executing user.

This allows database programmers to call procedures or functions that can read and update the database on behalf of unprivileged users. i.e. perform tasks that the current invoker of the procedure is unable to perform themselves. This secure packaging allows the procedure to enforce formal processes such as auditing, and data filtering.

Consider this example, a user who attaches to the database cannot read anything from the EMPLOYEES table. However, if they are granted access to the stored procedure SHOW_EMPLOYEE then it can be used to display the current employee details for the current invoker.

When the CALL statement executes a definers rights stored procedure, the access rights of the definer of the stored module temporarily replace the current security environment. The definer of the module is specified by the AUTHORIZATION clause in CREATE MODULE statement.

```
create module PUBLIC_ROUTINES
    language SQL
    authorization INFO_SERVICES

    procedure SHOW_EMPLOYEE;
    begin
    ...procedure body...
    end;
end module;
```

The AUTHORIZATION need not be a single user, as in this example; the AUTHORIZATION is an OpenVMS rights identifier (or role). Each table, view, sequence and routine referenced by the procedure SHOW_EMPLOYEE must have appropriate access granted to this role for the CREATE MODULE to suceed.

If the AUTHORIZATION is omitted then the invoker of the stored procedure must have the required access to the tables, views, sequences and routines used by the stored procedure. Otherwise they will be denied access when the CALL is executed.

Procedures and functions are defined in the context of a stored module and access is granted at the module level. In this example each database user requires EXECUTE access on the containing module before they can execute the SHOW_EMPLOYEE store procedure.

```
SQL> grant EXECUTE on module PUBLIC_ROUTINES to PUBLIC;
```

Definers rights procedures were introduced in Oracle Rdb V6.0, and underwent a major change in Oracle Rdb7. In the current release stored procedures can call other stored procedures and functions. This means that the security profiles may change many times during execution as new definers rights routines are executed on behalf of other definers.

Which user are you?

How would the procedure SHOW_EMPLOYEE use your identity to return specific information? This is done using one of the following SQL built-in functions: SYSTEM_USER, SESSION_USER, and CURRENT_USER. The built-in function USER is a synonym for CURRENT_USER.

If you selected these values in an interactive SQL session they would all be the same, but there are environments in which they can all return different values.

The SYSTEM_USER is the user that created the database attachment. In the case of an SQL/Services transaction reusable server, the database is pre-attached, and SYSTEM_USER can be used to determine the name of this attaching process. As the server is reused and each new session is established the SESSION_USER is changed to reflect the new session owner.

If a definers rights routine is active then CURRENT_USER will represent the AUTHORIZATION of the current active definer.

In our example the procedure SHOW_EMPLOYEE would use the value of SESSION_USER to locate the row for that employee. Using CURRENT_USER in this case would not be very useful because as a definers rights stored procedure CURRENT_USER will always return the same value; namely the AUTHORIZATION for the module.

The AUTHORIZATION clause also adds another performance advantage to stored routines, since all object access is for a single user then this checking can be done once per access when the module is first loaded. Without AUTHORIZATION the checking is done for each user using the server, such as an ODBC, JDBC or SQL/Services service.

Technical Tips: Default Access Control

Question: When a new table is created in the database Rdb always denies access to PUBLIC. In our environment we want PUBLIC to have SELECT access to each table. Is there a way to have Rdb do this automatically?

Answer: Yes this is possible. The first releases of Rdb always defaulted the access control list (ACL) of a new table or view to ALL access to the creator and ALL data access to PUBLIC (i.e. SELECT, DELETE, INSERT and UPDATE).

Rdb had many requests to change this default ACL and in Rdb V4.0 we changed the default to be all access for the table creator and no access for PUBLIC. To allow for upward compatibility, so some environments could retain open access to tables, we added special support for DEFAULT access control entry (ACE).

If you grant access on the database to the special user DEFAULT then those access rights are inherited by all new tables and views for PUBLIC. This change takes affect on the next attach to the database as shown in the example session below.

```
SQL> create table web1 (a integer);
SQL> show protection on table web1;
Protection on Table WEB1
    (IDENTIFIER=[RDB,SMITHI],ACCESS=SELECT+INSERT+UPDATE+
      DELETE+SHOW+CREATE+ALTER+DROP+DBCTRL+REFERENCES)
    (IDENTIFIER=[*,*],ACCESS=NONE)
SQL> grant select on database alias rdb$dbhandle to default;
SQL> create table web2 (a integer);
SQL> show protection on table web2;
Protection on Table WEB2
   (IDENTIFIER=[RDB,SMITHI],ACCESS=SELECT+INSERT+UPDATE+
      DELETE+SHOW+CREATE+ALTER+DROP+DBCTRL+REFERENCES)
    (IDENTIFIER=[*,*],ACCESS=NONE)
SQL> commit;
SQL> disconnect all;
SQL> create table web3 (a integer);
SQL> show protection on table web3;
Protection on Table WEB3
    (IDENTIFIER=[RDB,SMITHI],ACCESS=SELECT+INSERT+UPDATE+
      DELETE+SHOW+CREATE+ALTER+DROP+DBCTRL+REFERENCES)
    (IDENTIFIER=[*,*],ACCESS=SELECT)
```

This functionality requires that the OpenVMS AUTHORIZATION file (UAF) include a definition for the DEFAULT user. It is possible that this identifier is not present on your system, as it is often removed for security reasons.

Oracle Rdb
Stored Procedure Security, Revision #1
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