

SECURITY MANAGEMENT IN ORACLE8

Elizabeth Boss, Boss Consulting Services, Inc.

Abstract

This presentation will provide information for the new database administrator regarding user management and privilege management. New password management capabilities including password expiration and password verification functions will be discussed. Granting users and roles and object level privileges will be covered. Scripts will be provided that help the database administrator determine what privileges have been granted to users and roles.

Creating New Users

One of the first tasks that many new database administrators are assigned is creating new users. A new user can only be created by someone who has the CREATE USER system privilege. The CREATE USER command is used to identify new users to Oracle. The following example creates a new user called student1.

```
SQL> CREATE USER student1
      IDENTIFIED BY train1
      DEFAULT TABLESPACE user_data
      TEMPORARY TABLESPACE temporary_data
      QUOTA 50k ON user_data
      PROFILE end_user
      PASSWORD EXPIRE;
```

The *identified clause* specifies either a password or external authentication, which bypasses the password for Oracle tools.

The *default tablespace* designates the storage destination for user objects. If it is not specified, the SYSTEM tablespace is the default. This is not a good practice, as the data dictionary resides in the SYSTEM tablespace.

The *temporary tablespace* designates the temporary segment workplace. If it is not specified, the SYSTEM tablespace is the default. Temporary segments are allocated and de-allocated based upon SQL statements and will cause fragmentation of the SYSTEM tablespace.

The *quota* limits the number of bytes/blocks allocated for user objects. If not specified, the user cannot create objects. A quota of UNLIMITED removes any limitation on space usage. Each tablespace requires a quota, therefore, if the user will have privileges on multiple tablespaces, use multiple quota clauses.

The *profile* assigns a profile limiting database resources available to the user. The profile must exist prior to assigning it to a user. If it is not specified, the user is assigned to the DEFAULT profile, which has no resource limits.

A list of Oracle users can be obtained by querying the data dictionary table DBA_USERS.

```
SQL> SELECT username,
      Created
      FROM dba_users;
```

Information about tablespace quotas can be obtained from the data dictionary table DBA_TS_QUOTAS.

```
SQL> SELECT username,
      tablespace_name,
      max_bytes
      FROM dba_ts_quotas
      ORDER BY username;
```

Create Session Privilege

At this point, student1 will be unable to do anything unless the CREATE SESSION privilege is granted. The CREATE SESSION privilege allows a user to connect to the Oracle database (i.e., create a session). The following example grants the CREATE SESSION privilege to the user student1:

```
SQL> GRANT create session TO student1;
```

Now, student1 is at least able to log into Oracle and access anything listed in the data dictionary table ALL_OBJECTS. Additional access privileges for specific tables and system commands still need to be granted to the new user. Granting privileges will be discussed in a later section.

Altering Users

Any of the options associated with the CREATE USER statement can be modified using the ALTER USER command. The ALTER USER system privilege is required to issue an ALTER USER statement. The following example modifies the user student1:

```
SQL> ALTER USER student1
      IDENTIFIED BY train1
      DEFAULT TABLESPACE class_data
      TEMPORARY TABLESPACE temp;
```

Unless the password is being changed, the IDENTIFIED BY clause is not needed because the user already exists.

Changing a User Password

A user's password may be changed using one of three different methods. One method of changing a password is by issuing a GRANT command, which specifies a new password. The privilege to grant CREATE SESSION is required in the following example:

```
SQL> GRANT create session TO student1
      IDENTIFIED BY training;
```

The ALTER USER command may also be used to modify a users password. The ALTER ANY USER system privilege is required in the following example. A user may change their own password using the ALTER USER command.

```
SQL> ALTER USER student1
      IDENTIFIED BY training;
```

The PASSWORD command may also be used to change a users password. The PASSWORD command can be issued by any user to change their own password. It can also be called from any PL/SQL program unit or a precompiler program (like Pro*C). The database administrator can use the PASSWORD command to change any users password.

```
SQL> password student1
Changing password for STUDENT1
Old password: ****
New password: ****
Retype new password: ****
Password changed
```

Linking Username to Operating System Login

Linking the username to the operating system login eliminates the need to re-type the username and password for each Oracle session. This can be done when creating the Oracle user. The "autologin" name is the same as the operating system login with a prefix specified by the initialization parameter OS_AUTHENT_PREFIX. The default value is operating system dependent, but is usually OPS\$. When the user connects to the database, a password is not required.

```
SQL> CREATE USER ops$student1
      IDENTIFIED EXTERNALLY
      DEFAULT TABLESPACE user_data
      TEMPORARY TABLESPACE temporary_data;
```

Locking User Accounts

An Oracle user account can be locked manually by issuing the ALTER USER command with the ACCOUNT LOCK option. To lock an account you must have ALTER USER privileges.

```
SQL> ALTER USER student1 ACCOUNT LOCK;
```

Subsequent attempts by the user student1 to connect to the database will result in an error. The user student1 will not be able to connect to the database until the account is unlocked.

```
sqlplus student1/train1

Connect student1 ORA-28000: the account is locked.
```

The account can be unlocked manually by issuing the ALTER USER command with the ACCOUNT UNLOCK option. To unlock an account you must have ALTER USER privileges.

```
SQL> ALTER USER student1 ACCOUNT UNLOCK;
```

To obtain a list of Oracle user accounts that have been locked, query the DBA_USERS data dictionary table.

```
SQL> SELECT username,
      account_status,
      lock_date
      FROM dba_users
      WHERE account_status = 'LOCKED';
```

User Profiles

A profile is used to place a limit on the resources that may be used by an Oracle user. The DEFAULT profile is provided by Oracle, and allows for unlimited resource usage. Additional profiles may be created which limit system resources that may be used by a specific type of user. The CREATE PROFILE command is used to create a new profile.

```
SQL> CREATE PROFILE end_user LIMIT
      idle_time 90
      connect_time 600
      sessions_per_user 1;
```

The profile name can be 30 characters or less and must start with an alphabetic character. Any resource not specified in the profile uses the value set in the DEFAULT profile (which by default, is unlimited). The profile is assigned to one or more users using the ALTER USER command.

Password Expiration

A profile can be used to manage passwords, their expiration frequency, reuse limits, and complexity. The following example creates a profile that causes a password to expire after 90 days allowing two days of grace time after password expiration. Failed login attempts is also limited to three, which will cause the account to be locked after the third unsuccessful login attempt.

```
SQL> CREATE PROFILE devel_profile LIMIT
      password_life_time 90
      password_grace_time 2
      failed_login_attempts 3;
```

Once the profile has been created it can be assigned to the appropriate users using the ALTER USER command.

```
SQL> ALTER USER student1
      PROFILE devel_profile;
```

Password reuse can be limited through the use of either the password_reuse_max or password_reuse_time options. These two parameters are mutually exclusive - if one is limited the other must be set to unlimited.

```
SQL> CREATE PROFILE user_profile LIMIT
      password_life_time 90
      password_grace_time 2
      password_reuse_time 180
      password_reuse_max unlimited
      failed_login_attempts 3;
```

Password Complexity

The complexity of an Oracle password can be enforced through the use of a function that is specified in the user profile. A predefined function is included with the Oracle database that does basic checking of passwords to ensure the following criteria are met:

- Passwords cannot be the same as username.
- Must be at least four characters in length.
- Must contain at least one number, one character, and one punctuation mark.
- Needs to differ from previous password by at least three letters.

The script file for the function is located in the ORACLE_HOME/rdbms/admin directory and is named utlpwdmg.sql. The script file needs to be executed when logged in as the user SYS. This script file creates a function named *verify_function* and assigns it to the password_verify_function resource of the DEFAULT profile. This causes all users who have not been explicitly assigned a profile to be limited by the restrictions set forth in the verify_function. It is possible to modify the script utlpwdmg.sql so that it alters a profile other than the DEFAULT profile. The DBA can make any changes necessary to this utlpwdmg.sql file that are required by the specific security regulations of their company.

Privileges

Privileges control access to a specific object or the ability to issue an SQL statement. They are rights granted explicitly to individual users or roles. Privileges are divided into two groups: system level privileges and object level privileges. Object level privileges grant rights to perform a particular action on a specified table, view, package, function, or procedure. System privileges provide the right to access or manipulate a specific type of object or to perform a specific action.

Roles

A role consists of a set of privileges that have been granted to a named object (the role). Privilege management is made easier through the use of roles. Groups of privileges are granted to the named role one time. The role can then be granted to multiple users.

To create a role, the system privilege CREATE ROLE is required. The CREATE ROLE command is used to create a new role. The name of the role is user-defined and must be unique within the entire database.

```
SQL> CREATE ROLE end_user;
```

Roles are considered database objects, and, therefore, are not owned by the user who created the role. After creating the role, it exists as a database object, but contains no privileges. Privileges are granted to a role just as they would be granted to a user, simply specifying the role name in place of the user name.

Dropping a Role

If a role is no longer necessary, it may be removed by issuing a DROP ROLE command. All privileges associated with the role are revoked from the users immediately.

```
SQL> DROP ROLE end_user;
```

Role Management

The Oracle database includes several predefined roles: CONNECT, RESOURCE, DBA, IMP_FULL_DATABASE, EXP_FULL_DATABASE. The following query can be used to determine the roles that have been created in the database.

```
SQL> COLUMN "Password?" FORMAT a10
SQL> SELECT role "Role Name",
           password_required "Password?"
FROM dba_roles
ORDER BY role;
```

Granting System Privileges

System privileges are assigned from within Oracle using the GRANT command. The following example grants the CREATE SYNONYM privilege to the role END_USER.

```
SQL> GRANT create synonym
      TO end_user;
```

System privileges can also be specified using the WITH ADMIN OPTION, which allows the grantee to grant the privilege to any other user in the system. The following example grants the CREATE ANY SYNONYM privilege with the admin option.

```
SQL> GRANT create any synonym
      TO end_user
      WITH ADMIN OPTION;
```

Granting Object Level Privileges

As with the system commands, the GRANT command is used to assign an object level privilege. Object level privileges can be granted by the owner of the object or by someone else granted the privilege through the WITH GRANT OPTION.

```
SQL> GRANT select, insert, update
      ON airline
      TO end_user;
```

The user whose schema contains the object automatically has the WITH GRANT OPTION for that object. The WITH GRANT OPTION cannot be granted to a role; it can only be granted to an individual user.

```
SQL> GRANT select, insert
      ON flight_schedule
      TO student1
      WITH GRANT OPTION;
```

Role Grants

Once a role has been granted privileges, the role can be granted to individual users. Roles may also be granted to other roles, creating a hierarchy of privileges.

```
SQL> GRANT end_user TO student1;
```

The user student1 now has all access privileges associated with the role end_user. In this example, the privileges include the system privilege CREATE ANY SYNONYM and the object level privileges SELECT, INSERT, and UPDATE on the airline table.

Privileges Assigned to Roles

The following script file displays all roles and the privileges that have been assigned to the roles:

```
BREAK ON "Role Name" SKIP 2
COLUMN "Admin?" FORMAT a6

SELECT role "Role Name",
       privilege "Privilege",
       admin_option "Admin?"
FROM dba_roles,
     Db SYS PRIVS
WHERE role = grantee
ORDER BY role;
```

Roles Granted to Users

To obtain a list of what roles have been granted to which users, the following query can be used:

```
BREAK ON "User Name" SKIP 2
COLUMN "Admin?" FORMAT a6
COLUMN "Default?" FORMAT a8

SELECT grantee "ser Name",
       granted_role "Role Name",
       admin_option "Admin?",
       default_role "Default?"
FROM dba_role_privs
ORDER BY grantee;
```

Roles Granted to Roles

The following script shows all the roles and privileges granted to those roles as well as roles granted to other roles.

```
COLUMN "Privilege" FORMAT a24 word_wrapped
COLUMN "Granted Role" FORMAT a20
COLUMN "Role" FORMAT a20
BREAK ON "Role" SKIP 2

SELECT grantee "Role",
       privilege "Privilege",
       ' ' "Granted Role"
FROM dba_sys_privs,
     dba_roles
WHERE role = grantee
UNION
SELECT grantee "Role",
       ' ' "Privilege",
       granted_role,
FROM dba_role_privs,
     dba_roles
WHERE grantee = role
ORDER BY 1,2,3;
```

Privilege Errors

The Oracle error ORA-00942 is a common error: ORA-00942: table or view does not exist. The first action the database administrator should take is to check object level privileges against the table or view.

```
SELECT owner||'.'||table_name "Object",
       ' ' "Column",
       privilege "Privilege",
       grantee "Grantee",
FROM dba_tab_privs
UNION
SELECT owner||'.'||table_name,
       column_name,
       privilege,
       grantee
FROM dba_col_privs;
```

If appropriate privileges have been granted to the user, verify that either a synonym for the table name has been created, or that the user is specifying owner.table_name in the SQL statement. To verify synonyms, query the DBA_SYNONYMS or USER_SYNONYMS data dictionary table depending upon whether the synonym is public or private.

```
SELECT synonym_name "Synonym Name",
       table_owner "Table Owner",
       table_name "Table Name"
FROM user_synonyms
ORDER BY synonym_name;
```

Conclusion

New Oracle users can be created using the CREATE USER command by any user that has the CREATE USER system privilege. When a user account is created a default tablespace can be specified that will be used as a storage location when one is not specified. User accounts can be locked and unlocked by the administrator using the ALTER USER ACCOUNT LOCK or UNLOCK command. User profiles allow the administrator to set resource limits on groups of users based on their needs. Profiles can also be used for password management and include the ability for passwords to expire, require a certain degree of complexity, and limit reuse of passwords.

Object privileges manage access to named database objects and can be granted by the owner of the object to other users. System level privileges control the right to issue a command or set of commands. System privileges are typically managed by the database administrator. Roles are used to group together system and object level privileges based on the needs of a specific set of users. The CREATE ROLE command is used to create a role, which is just a named "container" that will be granted a series of privileges. Roles, system, and object level privileges can be granted to users. Using roles greatly reduces the amount of work performed by the administrator.

About the Author

Elizabeth Boss is president of Boss Consulting Services, Inc. She has more than 15 years of experience in application development and database administration as a database administrator, consultant, instructor, and curriculum developer. As a senior consultant/instructor, Elizabeth has worked directly with customers in all phases and aspects of the design, development, and administration of Oracle systems. She is a frequent presenter at international and local user groups, and, in 1997, was listed among the top 25 speakers at the IOUG-A Conference.

Elizabeth can be reached at:

Boss Consulting Services, Inc.

710 Kings Deer Point

Monument, CO 80132

Phone: (877) 489-7745 Fax: (719)481-5820

E-mail: eboss@boss-consulting-inc.com

Web: www.boss-consulting-inc.com