module14-1-users.htm; updated June 18, 2012; Some figures shown in these notes are from Oracle document D11321GC11.

Module 14-1 – Create and Manage Oracle Users

Objectives

- Create Oracle user accounts.
- Alter Oracle user accounts.
- Familiarize with database/external authentication.
- Learn about Oracle site licensing.

Oracle User Accounts

User Account Creation

The **CREATE USER** command creates a system user as shown here.

```
CREATE USER Scott IDENTIFIED BY Tiger;
```

- The user **Scott** is a standard "**dummy**" user account found on many Oracle systems for the purposes of system testing – it needs to be disabled to remove a potential hacker access route.

- The **IDENTIFIED BY** clause specifies the user password.

- In order to create a user, a DBA must have the **CREATE USER system privilege**.

- Users also have a privilege domain – initially the user account has NO privileges – it is empty.

- In order for a user to connect to Oracle, you must grant the user the **CREATE SESSION system privilege**.

- Each username must be **unique** within a database. A username cannot be the same as the name of a role (roles are described in a later module).

Each user has a **schema** for the storage of objects within the database (see the figure below).

- Two users can name objects identically because the objects are referred to globally by using a combination of the username and object name.

- Example: **User350.Employee** – each user account can have a table named Employee because each table is stored within the user's schema.

Database Schema

- A schema is a named collection of objects.
- A user is created, and a corresponding schema is created.
- A user can be associated only with one schema.
- Username and schema are often used interchangeably.

A complete example of the **CREATE USER** command:

```
CREATE USER Scott
    IDENTIFIED BY New_Pa$$w0rd
    DEFAULT TABLESPACE Users
    TEMPORARY TABLESPACE Temp
    QUOTA 10M ON Users
    QUOTA 5M ON Data01
    PROFILE Accountant

    ACCOUNT UNLOCK

    PASSWORD EXPIRE;
```

Scott has two tablespaces identified, one for **DEFAULT** storage of objects and one for **TEMPORARY** objects.

Scott has a quota set on 2 tablespaces. More details about tablespace allocation are given later in these notes.

Scott has the resource limitations allocated by the **PROFILE** named accountant. The account is unlocked (the default – alternatively the account could be created initially with the **LOCK** specification).

The **PASSWORD EXPIRE** clause requires **Scott** to change the password prior to connecting to the database. After the password is set, when the user logs on using SQLPlus or any other software product that connects to the database, the user receives the following message at logon, and is prompted to enter a new password:

```
ERROR:

ORA-28001: the account has expired

Changing password for SCOTT

Old password:

New password:

Retype new password:

Password changed
```

Database Authentication

Database authentication involves the use of a standard user account and password. Oracle performs the authentication.

- System users can change their password at any time.
- Passwords are stored in an **encrypted** format.
- Each password must be made up of single-byte characters, even if the database uses a multi-byte character set.
- Advantages:
    - User accounts and all authentication are controlled by the database. There is no reliance on anything outside of the database.
    - Oracle provides strong password management features to enhance security when using database authentication.
    - It is easier to administer when there are small user communities.

Oracle recommends using password management that includes password aging/expiration, account locking, password history, and password complexity verification.

External Authentication

External Authentication requires the creation of user accounts that are maintained by Oracle. Passwords are administered by an external service such as the **operating system** or a **network service** (Oracle Networks – **Network authentication** through the network is covered in the course *Oracle Database Administration Fundamentals II*). This option is generally useful when a user logs on directly to the machine where the Oracle server is running.

- A database password is not used for this type of login.

- In order for the operating system to authenticate users, a DBA sets the **init.ora** parameter **OS_AUTHENT_PREFIX** to some set value – the default value is **OPS$** in order to provide for backward compatibility to earlier versions of Oracle.

- This prefix is used at the operating system level when the user's account username.

- You can also use a **NULL** string (a set of empty double quotes: "") for the prefix so that the Oracle username exactly matches the Operating System user name. This eliminates the need for any prefix.

```
#init.ora parameter
OS_AUTHENT_PREFIX=OPS$


#create user command
CREATE USER OPS$Scott
    IDENTIFIED EXTERNALLY
    DEFAULT TABLESPACE users
    TEMPORARY TABLESPACE temp
    QUOTA UNLIMITED ON Users;
```

```
CREATE USER O$shoyaib
    IDENTIFIED EXTERNALLY
   account unlock;
grant dba to shoyaib;
```

When **Scott** attempts to connect to the database, Oracle will check to see if there is a database user named **OPS$Scott** and allow or deny the user access as appropriate. Thus, to use SQLPlus to log on to the system,

the LINUX/UNIX user **Scott** enters the following command from the operating system:

```
$ sqlplus /
```

All references in commands that refer to a user that is authenticated by the operating system must include the defined prefix **OPS$**.

Oracle allows operating-system authentication only for secure connections – this is the default. This precludes use of Oracle Net or a shared server configuration and prevents a remote user from impersonating another operating system user over a network.

The **REMOTE_OS_AUTHENT** parameter can be set to force acceptance of a client operating system user name from a nonsecure connection.

- This is NOT a good security practice.
- Setting **REMOTE_OS_AUTHENT = FALSE** creates a more secure configuration based on server-based authentication of clients.

- Changes in the parameter take effect the next time the instance starts and the database is mounted.

Global Authentication

Central authentication can be accomplished through the use of Oracle Advanced Security software for a directory service.

Global users termed **Enterprise Users** are authenticated by SSL (secure socket layers) and the user accounts are managed outside of the database.

Global Roles are defined in a database and known only to that database and authorization for the roles is done through the directory service. The roles can be used to provide access privileges

Enterprise Roles can be created to provide access across multiple databases. They can consist of one or more global roles and are essentially containers for global roles.

Creating a Global User Example:

```
CREATE USER Scott

  IDENTIFIED GLOBALLY AS 'CN=Scott, OU=division1,
O=oracle, C=US';
```

- Scott is authenticated by SSL and authorized by the enterprise directory service.
- The **AS** clause provides a string identifier (distinguished name – DN) to the enterprise directory.
- **Disadvantage:** Scott must have a user account created in **every** database to be accessed as well as in the directory service.

Creating a Schema-Independent User Example:

Schema-independent user accounts allow more than one enterprise user to access a shared database schema. These users are:

- Authenticated by **SSL** or **passwords**.
- Not created in the database with a CREATE USER statement.
- Privileges are managed in a directory.
- Most users don't need their own schemas – this approach separates users from databases.

```
CREATE USER inventory_schema IDENTIFIED GLOBALLY AS '';
```

- In the directory create multiple enterprise users and a mapping object to tell the database how to map users DNs to the shared schema.

Proxy Authentication and Authorization

This approach to authentication and authorization uses a **middle-tier server** to proxy clients securely.

Three forms of proxy authentication:

- Middle-tier server authenticates itself with the database server and client – an application user or another application.
- Client (a database user) is not authenticated by the middle-tier server – instead the identity and database password are passed through the middle-tier server to the database server for authentication.
- Global users are authenticated by the middle-tier server and it passes either a Distinguished Name (DN) or Certificate through the middle-tier for retrieval of a client user name.
-

The middle-tier server proxies a client through the GRANT CONNECT THROUGH clause of the ALTER USER statement.

```
ALTER USER Scott GRANT CONNECT THROUGH Proxy_Server
```

```
WITH ROLE ALL EXCEPT Inventory;
```

- This grants authorization through the middle-tier server named **Proxy_Server**.
- The **WITH ROLE** clause specifies that **Proxy_Server** can active all roles for the user **Scott** except the role named **Inventory**.

Revoking the middle-tier's proxy server authorization:

```
ALTER USER Scott REVOKE CONNECT THROUGH Proxy_Server;
```

Default Tablespace

If one is not specified, the default tablespace for a user is the **SYSTEM** tablespace – not a good choice for a default tablespace. The standard practice to always set a default tablespace as was shown in the **CREATE USER** command.

```
CREATE USER ops$Scott
    IDENTIFIED EXTERNALLY
    DEFAULT TABLESPACE Users
    TEMPORARY TABLESPACE Temp
    QUOTA UNLIMITED ON Users;
```

Use the ALTER USER command to change a user's default tablespace.

```
ALTER USER ops$Scott
    DEFAULT TABLESPACE Data01
    QUOTA 5M on Data01;
```

Changing a default tablespace does not affect the storage location of any user schema objects that were created before the default tablespace modification.

You can assign each user a tablespace quota for any tablespace (**except a temporary tablespace**). Assigning a quota does the following things:

- Users with privileges to create certain types of objects can create those objects in the specified tablespace.
- Oracle Database limits the amount of space that can be allocated for storage of a user's objects within the specified tablespace to the amount of the quota.

By default, a user has no quota on any tablespace in the database.

- If the user has the privilege to create a schema object, then you must assign a quota to allow the user to create objects.
- Minimally, assign users a quota for the default tablespace, and additional quotas for other tablespaces in which they can create objects.

Temporary Tablespace

The default **Temporary Tablespace** for a user is also the **SYSTEM** tablespace.

- Allowing this situation to exist for system users will guarantee that user processing will cause contention with access to the data dictionary.

- Generally a DBA will create a **TEMP** tablespace that will be shared by all users for processing that requires sorting and joins.

Tablespace Quotas

Assigning a **quota** ensures that users with privileges to create objects can create those objects in the tablespace.

A quota also ensures the amount of space allocated for storage by an individual user is not exceeded. The default is **NO QUOTA** on any tablespace so a quota must be set or else the Oracle user account cannot be used to create any objects.

Assigning Other Tablespace Quotas: You can assign a quota on tablespaces other than the **DEFAULT** and **TEMPORARY** tablespaces for users.

- This enables the user to create objects in the other tablespaces.

- This is often done for senior systems analysts and programmers who are authorized to create objects in a **DATA** tablespace.

If you change a quota and the new quota is smaller than the old one, then the following rules apply:

- For users who have already exceeded the new quota, new objects cannot be created, and existing objects cannot be allocated more space until the combined space of the user's objects is within the new quota.

- For users who have not exceeded the new quota, user objects can be allocated additional space up to the new quota.

Granting the **UNLIMITED TABLESPACE** privilege to a user account overrides all quota settings for all tablespaces.

Revoking Tablespace Access

A DBA can revoke tablespace access by setting the user's quota to zero for the tablespace through use of the **ALTER USER** command. This example alters the user named **SCOTT** for the **USERS** tablespace.

```
ALTER USER Scott QUOTA 0 ON Users;
```

Existing objects for the user will remain within the tablespace, but cannot be allocated additional disk space.

Alter User Command

Users can use the **ALTER USER** command to change their own password.

To make any other use of the command, a user must have the **ALTER USER system privilege** - something the DBA should not give to individual users.

Changing a user's security setting with the **ALTER USER** command changes future sessions, not a current session to which the user may be connected.

Example **ALTER USER** command:

```
ALTER USER Scott
    IDENTIFIED by New_Pa$$w0rd
    DEFAULT TABLESPACE Data01
    TEMPORARY TABLESPACE Temp
    QUOTA 100M ON Data01
    QUOTA 0 ON Inventory_TBS
    PROFILE Almost_Unemployeed;
```

Drop User Command

The **DROP USER** command is used to drop a user. Examples:

```
DROP USER User105;
DROP USER Scott CASCADE;
```

- Dropping a user causes the user and the user schema to be immediately deleted from the database.
- If the user has created objects within their schema, it is necessary to use the **CASCADE** option in order to drop a user.
- If you fail to specify **CASCADE** when user objects exist, an error message is generated and the user is **not** dropped.
- In order for a DBA to drop a user, the DBA must have the **DROP USER system privilege**.

CAUTION: You need to exercise caution with the **CASCADE** option to ensure that you don't drop a user where views or procedures exist that depend upon tables that the user created. In those cases, dropping a user requires a lot of detailed investigation and careful deletion of objects.

If you want to deny access to the database, but do not want to drop the user and the user's objects, you should revoke the **CREATE SESSION** privilege for the user temporarily.

You cannot drop a user who is connected to the database - you must first terminate the user's session with the **ALTER SYSTEM KILL SESSION** command.

Data Dictionary Tables for User Accounts

The only data dictionary table used by a DBA for user account information is **DBA_USERS**.

```
COLUMN username FORMAT A15;

COLUMN account_status FORMAT A20;

COLUMN default_tablespace FORMAT A19;

SELECT username, account_status, default_tablespace

FROM dba_users;
```

```
USERNAME        ACCOUNT_STATUS       DEFAULT_TABLESPACE

--------------- --------------------
-------------------

OUTLN           OPEN                 SYSTEM
```

```
USER350          OPEN                   USERS

DBOCK            OPEN                   DATA01

SYS              OPEN                   SYSTEM

SYSTEM           OPEN                   SYSTEM

USER349          EXPIRED                SYSTEM

SCOTT            EXPIRED                USERS

TSMSYS           EXPIRED & LOCKED       SYSTEM

DIP              EXPIRED & LOCKED       SYSTEM

DBSNMP           EXPIRED & LOCKED       SYSAUX

ORACLE_OCM       EXPIRED & LOCKED       SYSTEM


11 rows selected.
```

Site Licensing

One of the DBA's responsibilities is to ensure that the Oracle Server license agreement is maintained.

A DBA can track and limit session access for users concurrently accessing the database through use of the **LICENSE_MAX_SESSIONS**, **LICENSE_SESSIONS_WARNING**, and **LICENSE_MAX_USERS** parameters in the PFILE. If an organization's license is unlimited, these parameters may have their value set to 0.

If the limit for the number of authorized connections to an Oracle Instance session is met, Oracle will only allow users with the **RESTRICTED SESSION** privilege (usually DBAs) to connect to the database.

When the maximum limit is reached, Oracle writes a message in the **ALERT** file indicating the maximum connections was reached. A DBA can also set a **warning limit** on the number of concurrent sessions so that Oracle writes a message to the ALERT file indicating that the warning limit was reached.

When the maximum limit is reached, Oracle enforces the limit by restricting access to the database. Oracle also tracks the highest number of concurrent sessions for each instance. This is termed the "**high water mark**" and the information is written to the ALERT file.

Setting Concurrent Session and Warning Limits

Set the maximum number of concurrent sessions in the **init.ora** file with the command:

```
LICENSE_MAX_SESSIONS = 80
```

A DBA does not have to set the warning limit (**LICENSE_SESSIONS_WARNING**), but this parameter makes it easier to manage site licensing. Set the warning limit in the init.ora file with the command:

```
LICENSE_SESSIONS_WARNING = 70
```

The usage limits can be changed while the database is running with the **ALTER SYSTEM** command. This example alters the number of concurrent sessions and the warning limit:

```
ALTER SYSTEM
    SET LICENSE_MAX_SESSIONS = 100
        LICENSE_SESSIONS_WARNING = 90;
```

If the new value is lower than the number of users currently logged on, Oracle does not force any users off of the system, but enforces the new limit for new users who attempt to connect.

Limiting Named Users

If a site license is for **named users** as opposed to concurrent accesses, you can limit the number of named users by limiting the number of users that can be created in the database before an instance is started up. This command in the **init.ora** file sets the maximum number of users:

```
LICENSE_MAX_USERS = 100
```

Attempting to create users after the limit is reached generates an error and a message is written to the **ALERT** file. A DBA can change the maximum named users limit with the **ALTER SYSTEM** command as shown here:

```
ALTER SYSTEM SET LICENSE_MAX_USERS = 125;
```

To view the current session limits, query the **V$LICENSE** data dictionary view as shown in this SELECT statement.

```
SELECT sessions_max s_max,

    sessions_warning s_warning,
    sessions_current s_current,

    sessions_highwater s_high,
    users_max
FROM v$license;
```

```
S_MAX   S_WARNING   S_CURRENT   S_HIGH   USERS_MAX
-----   ---------   ---------   ------   ---------
100     80          65          82       50
```

End of Notes