8. Managing Users and Security

Abstract: Controlling database access and resource limits is an important aspect of the DBA's function. You use profiles to manage the database and system resources and to manage database passwords and password verification. You control database and data access using privileges, and you create roles to manage the privileges. This lesson covers creating users and assigning proper resources and privileges. It also discusses the auditing capabilities of the database and using Globalization Support.

Contents

Objectives:

- Manage passwords using profiles
- Administer profiles
- Control use of resources using profiles
- Obtain information about profiles, password management, and resources
- Create new database users
- Alter and drop existing database users
- Monitor information about existing users
- Identify system and object privileges
- Grant and revoke privileges
- Identify auditing capabilities
- Create and modify roles
- Control availability of roles
- Remove roles
- Use predefined roles
- Display role information from the data dictionary

1. Profiles

You use profiles to control the database and system resource usage. Oracle provides a set of predefined resource parameters that you can use to monitor and control database resource usage. You can define limits for each resource by using a database profile. You also use profiles for password management. You can create profiles for different user communities and then assign a profile to each user. When you create the database, Oracle creates a profile named DEFAULT, and if you do not specify a profile for the user, Oracle assigns the user the DEFAULT profile.

1.1. Managing Resources

Oracle lets you control the following types of resource usage through profiles:

- Concurrent sessions per user
- Elapsed and idle time connected to the database
- CPU time used
- Private SQL and PL/SQL area used in the SGA (System Global Area)
- Logical reads performed
- Amount of private SGA space used in Shared Server configurations

Resource limits are enforced only if you have set the parameter RESOURCE_LIMIT to TRUE. Even if you have defined profiles and assigned profiles to users, Oracle enforces them only when this parameter is set to TRUE. You can set this parameter in the initialization parameter file so that every time the database starts, the resource usage is controlled for each user using the assigned profile. You enable or disable resource limits using the ALTER SYSTEM command. The default value of RESOURCE_LIMIT is FALSE.

The limits for each resource are specified as an integer; you can set no limit for a given resource by specifying UNLIMITED, or you can use the value specified in the DEFAULT profile by specifying DEFAULT. The DEFAULT profile initially has the value UNLIMITED for all resources. After you create the database, you can modify the DEFAULT profile.

Most resource limits are set at the session level; a session is created when a user connects to the database. You can control certain limits at the statement level (but not at the transaction level). If a user exceeds a resource limit, Oracle aborts the current operation, rolls back the changes made by the statement, and returns an error. The user has the option of committing or rolling back the transaction, because the statements issued earlier in the transaction are not aborted. No other operation is permitted when a session-level limit is reached. The user can disconnect, in which case the transaction is committed. You use the following parameters to control resources:

SESSIONS_PER_USER Limits the number of concurrent user sessions. No more sessions from the current user are allowed when this threshold is reached.

CPU_PER_SESSION Limits the amount of CPU time a session can use. The CPU time is specified in hundredths of a second.

CPU_PER_CALL Limits the amount of CPU time a single SQL statement can use. The CPU time is specified in hundredths of a second. This parameter is useful for controlling runaway queries, but you should be careful to specify this limit for batch programs.

LOGICAL_READS_PER_SESSION Limits the number of data blocks read in a session, including the blocks read from memory and from physical reads.

LOGICAL_READS_PER_CALL Limits the number of data blocks read by a single SQL statement, including the blocks read from memory and from physical reads.

PRIVATE_SGA Limits the amount of space allocated in the SGA for private areas, per session. Private areas for SQL and PL/SQL statements are created in the SGA in the multithreaded architecture. You can specify K to indicate the size in KB or M to indicate the size in MB. If you specify neither K or M, the size is in bytes. This limit does not apply to dedicated server architecture connections.

CONNECT_TIME Specifies the maximum number of minutes a session can stay connected to the database (total elapsed time, not CPU time). When the threshold is reached, the user is automatically disconnected from the database; any pending transaction is rolled back.

IDLE_TIME Specifies the maximum number of minutes a session can be continuously idle, that is, without any activity for a continuous period of time. When the threshold is reached, the user is disconnected from the database; any pending transaction is rolled back.

COMPOSITE_LIMIT A weighted sum of four resource limits: CPU_PER_SESSION, LOGICAL_READS_PER_SESSION, CONNECT_TIME, and PRIVATE_SGA. You can define a cost for the system resources (the resource cost on one database may be different from another, based on the number of transactions, CPU, memory, and so on) known as the composite limit. The upcoming section "Managing Profiles" discusses setting the resource cost.

1.2. Managing Password

You also use profiles to manage passwords. You can set the following by using profiles:

Account locking Number of failed login attempts, and the number of days the password will be locked.

Password expiration How often passwords should be changed, whether passwords can be reused, and the grace period after which the user is warned that the password change is due.

Password complexity Use of a customized function to verify the password complexity—for example, the password should not be the same as the user ID, cannot include commonly used words, and so on.

You can use the following parameters in profiles to manage passwords:

FAILED_LOGIN_ATTEMPTS Specifies the maximum number of consecutive invalid login attempts (providing an incorrect password) allowed before the user account is locked.

PASSWORD_LOCK_TIME Specifies the number of days the user account will remain locked after the user has made FAILED_LOGIN_ATTEMPTS number of consecutive failed login attempts.

PASSWORD_LIFE_TIME Specifies the number of days a user can use one password. If the user does not change the password within the number of days specified, all connection requests return an error. The DBA then has to reset the password.

PASSWORD_GRACE_TIME Specifies the number of days the user will get a warning before the password expires. This is a reminder for the user to change the password.

PASSWORD_REUSE_TIME Specifies the number of days a password cannot be used again after changing it.

PASSWORD_REUSE_MAX Specifies the number of password changes required before a password can be reused. You cannot specify a value for both PASSWORD_REUSE_TIME and PASSWORD_REUSE_MAX; one should always be set to UNLIMITED, because you can enable only one type of password history method.

PASSWORD_VERIFY_FUNCTION Specifies the function you want to use to verify the complexity of the new password. Oracle provides a default script, which you can modify.

TIP: You can specify minutes or hours as a fraction or expression in parameters that require days as a value. One hour can be represented as 0.042 days or 1/24, and one minute can be specified as 0.000694 days, 1/24/60, or 1/1440.

1.3. Managing Profiles

You can create many profiles in the database that specify both resource management parameters and password management parameters. However, you can assign a user only one profile at any given time. To create a profile, you use the CREATE PROFILE command. You need to provide a name for the profile and then specify the parameter names and their values separated by space(s).

As an example, let's create a profile to manage passwords and resources for the accounting department users. The users are required to change their password every 60 days, and they cannot reuse a password for 90 days. They are allowed to make a typo in the password only six consecutive times while connecting to the database; if the login fails a seventh time, their account is locked forever (until the DBA or security department unlocks the account).

The accounting department users are allowed a maximum of six database connections; they can stay connected to the database for 24 hours, but an inactivity of 2 hours will terminate their session. To prevent users from performing runaway queries, in this example we will set the maximum number of blocks they can read per SQL statement to 1 million.

```
SQL> CREATE PROFILE ACCOUNTING_USER LIMIT
  2  SESSIONS_PER_USER 6
  3  CONNECT_TIME 1440
  4  IDLE_TIME 120
  5  LOGICAL_READS_PER_CALL 1000000
  6  PASSWORD_LIFE_TIME 60
  7  PASSWORD_REUSE_TIME 90
  8  PASSWORD_REUSE_MAX UNLIMITED
  9  FAILED_LOGIN_ATTEMPTS 6
 10  PASSWORD_LOCK_TIME UNLIMITED
Profile created.
```

In the example, parameters such as PASSWORD_GRACE_TIME, CPU_PER_SESSION, and PRIVATE_SGA are not used. They will have a value of DEFAULT, which means the value will be taken from the DEFAULT profile.

The DBA or security officer can unlock a locked user account by using the ALTER USER command. The following example shows the unlocking of SCOTT's account.

```
SQL> ALTER USER SCOTT ACCOUNT UNLOCK;

User altered.
```

Composite Limit

The composite limit specifies the total resource cost for a session. You can define a weight for each resource based on the available resources. The following resources are considered for calculating the composite limit:

- CPU_PER_SESSION
- LOGICAL_READS_PER_SESSION
- CONNECT_TIME
- PRIVATE_SGA

The costs associated with each of these resources are set at the database level by using the ALTER RESOURCE COST command. By default, the resources have a cost of 0, which means they should not be considered for a composite limit (they are inexpensive). A higher cost means that the resource is expensive. If you do not specify any of these resources in ALTER RESOURCE COST, Oracle will keep the previous value. For example:

```
SQL> ALTER RESOURCE COST
  2  LOGICAL_READS_PER_SESSION 10
  3  CONNECT_TIME 2;

Resource cost altered.
```

Here CPU_PER_SESSION and PRIVATE_SGA will have a cost of 0 (if they have not been modified before).

You can define limits for each of the four parameters in the profile as well as set the composite limit. The limit that is reached first is the one that takes effect. The following statement adds a composite limit to the profile you created earlier.

```
SQL> ALTER PROFILE ACCOUNTING_USER LIMIT
  2  COMPOSITE_LIMIT 1500000;

Profile altered.
```

The cost for the composite limit is calculated as follows:

```
Cost = (10 × LOGICAL_READS_PER_SESSION) + (2 × CONNECT_TIME)
```

If the user has performed 100,000 block reads and was connected for two hours, the cost thus far would be (10 × 100,000) + (2 × 120) = 1,000,240.

The user will be restricted when this cost exceeds 1,500,000 or when the values for LOGICAL_READS_PER_SESSION or CONNECT_TIME set in the profile are reached.

Password Verification Function

You can create a function to verify the complexity of the passwords and assign the function name to the PASSWORD_VERIFY_FUNCTION parameter in the profile. When a password is changed, Oracle checks to see whether the supplied password satisfies the conditions specified in this function. Oracle provides a default verification function, known as VERIFY_FUNCTION, which is in the rdbms/admin directory of your Oracle software installation; the script is named utlpwdmg.sql. The password verification function should be owned by SYS and should have the following characteristics.

```
FUNCTION SYS.<function_name>
( <userid_variable> IN VARCHAR2 (30),
<password_variable> IN VARCHAR2 (30),
<old_password_variable> IN VARCHAR2 (30) )
RETURN BOOLEAN
```

Oracle's default password verification function checks that the password conforms to the following:

- Is not the same as the username
- Has a minimum length
- Is not too simple; a list of words is checked
- Contains at least one letter, one digit, and one punctuation mark
- Differs from the previous password by at least three letters

If the new password satisfies all the conditions, the function returns a Boolean result of TRUE, and the user's password is changed.

Altering Profiles

Using the ALTER PROFILE command changes profile values. You can change any parameter in the profile using this command. The changes take effect the next time the user connects to the database. For example, to add a password verification function and set a composite limit to the profile you created in the previous example, use the following:

```
SQL> ALTER PROFILE ACCOUNTING_USER LIMIT
  2  PASSWORD_VERIFY_FUNCTION VERIFY_FUNCTION
  3  COMPOSITE_LIMIT 1500
```

```
Profile altered.
```

Dropping Profiles

To drop a profile, you use the DROP PROFILE command. If any user is assigned the profile you want to drop, Oracle returns an error. You can drop such profiles by specifying CASCADE, in which case the users who have that profile will be assigned the DEFAULT profile.

```
SQL> DROP PROFILE ACCOUNTING_USER CASCADE;

Profile dropped.
```

Assigning Profiles

To assign profiles to users, you use the CREATE USER or ALTER USER command. These commands are discussed later. This example assigns the ACCOUNTING_USER profile to an existing user named SCOTT:

```
SQL> ALTER USER SCOTT
  2  PROFILE ACCOUNTING_USER;

User altered.
```

1.4. Querying Profile Information

You can query profile information from the DBA_PROFILES view. The following example shows information about the profile created previously. The RESOURCE_TYPE column indicates whether the parameter is KERNEL (resource) or PASSWORD.

```
SQL> SELECT RESOURCE_NAME, LIMIT
  2  FROM DBA_PROFILES
  3  WHERE PROFILE = 'ACCOUNTING_USER'
  4  AND RESOURCE_TYPE = 'KERNEL';

RESOURCE_NAME                    LIMIT
-------------------------------- ----------
COMPOSITE_LIMIT                  1500
SESSIONS_PER_USER                6
CPU_PER_SESSION                  DEFAULT
```

```
CPU_PER_CALL                    DEFAULT
LOGICAL_READS_PER_SESSION       DEFAULT
LOGICAL_READS_PER_CALL          10000000
IDLE_TIME                       120
CONNECT_TIME                    UNLIMITED
PRIVATE_SGA                     DEFAULT
9 rows selected.
```

The view USER_RESOURCE_LIMITS shows the limit defined for the current user for resource, and the view USER_PASSWORD_LIMITS shows the limit defined for the password.

```
SQL> SELECT * FROM USER_PASSWORD_LIMITS;
RESOURCE_NAME                   LIMIT
------------------------------- ---------------
FAILED_LOGIN_ATTEMPTS           6
PASSWORD_LIFE_TIME              60
PASSWORD_REUSE_TIME             90
PASSWORD_REUSE_MAX              UNLIMITED
PASSWORD_VERIFY_FUNCTION        VERIFY_FUNCTION
PASSWORD_LOCK_TIME              UNLIMITED
PASSWORD_GRACE_TIME             UNLIMITED
7 rows selected.
```

You can query the system resource cost from the RESOURCE_COST view.

```
SQL> SELECT * FROM RESOURCE_COST;
RESOURCE_NAME                   UNIT_COST
------------------------------- ----------
CPU_PER_SESSION                 10
LOGICAL_READS_PER_SESSION       0
CONNECT_TIME                    2
PRIVATE_SGA                     0
```

2. Users

Access to the Oracle database is provided using database accounts known as usernames (users). If the user owns database objects, the account is known as a *schema*, which is a logical grouping of all the objects owned by the user. Persons requiring access to the database should have a valid username created in the database. The following properties are associated with a database user account:

Authentication method Each user must be authenticated to connect to the database by using a password, through the operating system, or via the Enterprise Directory Service. Operating system authentication is discussed in the "Privileges and Roles" section.

Default and temporary tablespaces The default tablespace specifies a tablespace for the user to create objects if another tablespace is not explicitly specified. The user needs a quota assigned in the tablespace to create objects, even if the tablespace is the user's default. You use the temporary tablespace to create the temporary segments; the user need not have any quota assigned in this tablespace.

Space quota The user needs a *space quota* assigned in each tablespace in which they want to create the objects. By default, a newly created user does not have any space quota allocated on any tablespace to create schema objects. For the user to create schema objects such as tables or materialized views, you must allocate a space quota in tablespaces.

Profile The user can have a profile to specify the resource limits and password settings. If you don't specify a profile when you create the user, the DEFAULT profile is assigned.

NOTE: When you create the database, the SYS and SYSTEM users are created. SYS is the schema that owns the data dictionary.

2.1. Managing Users

To create users in the database, you use the CREATE USER command. Specify the authentication method when you create the user. A common authentication method is using the database; the username is assigned a password, which is stored encrypted in the database. Oracle verifies the password when establishing a connection to the database.

As an example, let's create a user JOHN with the various clauses available in the CREATE USER command.

```
SQL> CREATE USER JOHN
  2  IDENTIFIED BY "B1S2!"
```

```
  3  DEFAULT TABLESPACE USERS
  4  TEMPORARY TABLESPACE TEMP
  5  QUOTA UNLIMITED ON USERS
  6  QUOTA 1M ON INDX
  7  PROFILE ACCOUNTING_USER
  8  PASSWORD EXPIRE
  9  ACCOUNT UNLOCK
User created.
```

The IDENTIFIED BY clause specifies that the user will be authenticated using the database. To authenticate the user using the operating system, specify IDENTIFIED EXTERNALLY. The password specified is not case sensitive.

If you do not specify the DEFAULT TABLESPACE clause, the SYSTEM tablespace is assigned as the default tablespace. When you create the database, you can specify a default temporary tablespace (DEFAULT TEMPORARY TABLESPACE clause of the CREATE DATABASE statement). You can also define it later using the ALTER DATABASE statement. If such default temporary tablespace is specified and you do not specify the TEMPORARY TABLESPACE clause in the CREATE USER command, the database's default temporary tablespace is assigned to the user. If the database does not have a default temporary tablespace defined, SYSTEM tablespace is assigned as the temporary tablespace by default. You cannot specify the undo tablespace as the default or temporary tablespace. The following query shows whether a default temporary tablespace is defined for the database.

```
SQL> SELECT * FROM DATABASE_PROPERTIES
  2  WHERE PROPERTY_NAME = 'DEFAULT_TEMP_TABLESPACE';
PROPERTY_NAME                  PROPERTY_VALUE
------------------------------ ------------------------
DESCRIPTION
--------------------------------------------------
DEFAULT_TEMP_TABLESPACE    TEMP
Name of default temporary tablespace
SQL>
```

Although the default and temporary tablespaces are specified, JOHN does not initially have any space quota on the USERS tablespace (or any tablespace).

You allocate quotas on the USERS and INDX tablespaces through the QUOTA clause. You can specify the QUOTA clause any number of times with the appropriate tablespace name and space limit. The space limit is specified in bytes, but can be followed by K or M to

indicate KB or MB. To create extents, the user should have a sufficient space quota in the tablespace. UNLIMITED specifies that the quota on the tablespace is not limited.

The PROFILE clause specifies the profile to be assigned. PASSWORD EXPIRE specifies that the user will be prompted (if using SQL*Plus, otherwise the DBA should change the password) for a new password at the first login. ACCOUNT UNLOCK is the default; you can specify ACCOUNT LOCK to initially lock the account.

The user JOHN can connect to the database only if he has the CREATE SESSION privilege. Granting privileges and roles is discussed later, in the section "Privileges and Roles." The CREATE SESSION privilege is granted to user JOHN by specifying the following:

```
SQL> GRANT CREATE SESSION TO JOHN;

Grant succeeded.
```

NOTE: To create extents, a user with the UNLIMITED TABLESPACE system privilege does not need any space quota in any tablespace.

Modifying User Accounts

You can modify all the characteristics you specified when creating a user by using the ALTER USER command. You can also assign or modify the default roles assigned to the user (discussed later on). Changing the default tablespace of a user affects only the objects created in the future.

The following example changes the default tablespace of JOHN and assigns a new password.

```
ALTER USER JOHN
IDENTIFIED BY SHADOW2#
DEFAULT TABLESPACE APPLICATION_DATA;
```

You can lock or unlock a user's account as follows:

```
ALTER USER <username> ACCOUNT [LOCK/UNLOCK]
```

You can also expire the user's password:

```
ALTER USER <username> PASSWORD EXPIRE
```

Users must change the password the next time they log in, or you must change the password. If the password is expired, SQL*Plus prompts for a new password at login time.

In the following example, setting the quota to 0 revokes the tablespace quota assigned. The objects created by the user in the tablespace remain there, but no new extents can be allocated in that tablespace.

```
ALTER USER JOHN QUOTA 0 ON USERS;
```

NOTE: Users can change their password by using the ALTER USER command; they do not

need the ALTER USER privilege to do so. They can also change their password using the PASSWORD command in SQL*Plus.

Dropping Users

You can drop a user from the database by using the DROP USER command. If the user (schema) owns objects, Oracle returns an error. If you specify the CASCADE keyword, Oracle drops all the objects owned by the user and then drops the user. If other schema objects, such as procedures, packages, or views, refer to the objects in the user's schema, they become invalid. When you drop objects, space is freed up immediately in the relevant tablespaces.

The following example shows how to drop the user JOHN, with all the owned objects.

```
DROP USER JOHN CASCADE;
```

WARNING: You cannot drop a user who is currently connected to the database.

2.2. Authenticating Users

In this section, we will discuss two widely used user-authenticating methods:

- Authentication by the database
- Authorization by the operating system

When you use database authentication, you define a password for the user (the user can change the password), and Oracle stores the password in the database (encrypted). When users connect to the database, Oracle compares the password supplied by the user with the password in the database.

By default, the password supplied by the user is not encrypted when sent over the network. To encrypt the user's password, you must set the ORA_ENCRYPT_LOGIN environment variable to TRUE on the client machine.

Similarly, when using database links, the password sent across the network is not encrypted. To encrypt such connections, you must set the DBLINK_ENCRYPT_LOGIN initialization parameter to TRUE. Passwords are encrypted using the data encryption standard (DES) algorithm.

When you use authorization by the operating system, Oracle verifies the operating system login account and connects to the database—users need not specify a username and password. Oracle does not store the passwords of such operating-system authenticated users, but they must have a username in the database. The initialization parameter OS_AUTHENT_PREFIX determines the prefix used for operating system authorization. By default, the value is OPS$. For example, if your operating system login name is ALEX, the database username should be

OPS$ALEX. When Alex specifies CONNECT / or does not specify a username to connect to the database, Oracle tries to connect Alex to the OPS$ALEX account. You can set the OS_AUTHENT_PREFIX parameter to a null string “”; this will not add any prefix. To create an operating-system authenticated user, use the following:

```
SQL> CREATE USER OPS$ALEX IDENTIFIED EXTERNALLY;
```

To connect to a remote database using operating system authorization, set the REMOTE_OS_AUTHENT parameter to TRUE. You must be careful in using this parameter, because connections can be made from any computer.

For example, if you have an operating system account named ORACLE and a database account OPS$ORACLE, you can connect to the database from the machine where the database resides. If you set REMOTE_OS_AUTHENT to TRUE, you can log in to any server with the ORACLE operating system account and connect to the database over the network. If a user creates an operating system ID named ORACLE and is on the network, the user can connect to the database using operating system authorization.

2.3. Complying with Oracle Licensing Terms

The DBA is responsible for ensuring that the organization complies with the Oracle licensing agreement. In lesson “Creating a Database and Data Dictionary” we discussed the parameters that can be set in the initialization file to enforce license agreements. They are as follows:

LICENSE_MAX_SESSIONS Maximum number of concurrent user sessions. When this limit is reached, only users with the RESTRICTED SESSION privilege are allowed to connect. The default is 0—unlimited. Set this parameter if your license is based on concurrent database usage.

LICENSE_SESSIONS_WARNING A warning limit on the number of concurrent user sessions. The default value is 0—unlimited. A warning message is written in the alert file when the limit is reached.

LICENSE_MAX_USERS Maximum number of users that can be created in the database. The default is 0—unlimited. Set this parameter if your license is based on the total number of database users.

You can change the value of these parameters dynamically by using the ALTER SYSTEM command. For example:

```
ALTER SYSTEM
        SET LICENSE_MAX_SESSIONS = 256
        LICENSE_SESSIONS_WARNING = 200;
```

The high-water mark (HWM) column of the V$LICENSE view shows the maximum number of concurrent sessions created since instance start-up and the limits set. A value of 0 indicates that no limit is set.

```
SQL> SELECT * FROM V$LICENSE;
```

```
SESSIONS_MAX    SESSIONS_WARNING    SESSIONS_CURRENT
      SESSIONS_HIGHWATER         USERS_MAX
------------          ----------------          ----------------
      ------------------                    ---------
256                   200                         105
      115                                   0
```

You can obtain the total number of database users from the DBA_USERS view:

```
SELECT COUNT(*) FROM DBA_USERS;
```

2.4. Querying User Information

You can query user information from the data dictionary views DBA_USERS and USER_USERS. USER_USERS shows only one row: information about the current user. You can obtain the user account status, password expiration date, account locked date (if locked), encrypted password, default and temporary tablespaces, profile name, and creation date from this view.

Oracle creates a numeric ID and assigns it to the user when the user is created. SYS has an ID of 0.

```
SQL> SELECT USERNAME, DEFAULT_TABLESPACE,
  2  TEMPORARY_TABLESPACE, PROFILE,
  3  ACCOUNT_STATUS, EXPIRY_DATE
  4  FROM DBA_USERS
  5  WHERE USERNAME = 'JOHN';

USERNAME  DEFAULT_TABLESPACE    TEMPORARY_TABLESPACE
      PROFILE                  ACCOUNT_ST       EXPIRY_DA
--------- ------------------ -------------------- --------------- ---------- ---------
JOHN          USERS                      TEMP
      ACCOUNTING_USER       OPEN               22-OCT-00
```

The view ALL_USERS shows the username and creation date.

```
SQL> SELECT * FROM ALL_USERS
  2  WHERE USERNAME LIKE 'SYS%';

USERNAME  USER_ID     CREATED
----------     ----------      ---------
SYS           0             13-JUL-00
```

```
SYSTEM      5              13-JUL-00
```

The views DBA_TS_QUOTAS and USER_TS_QUOTAS list the tablespace quota assigned to the user. A value of −1 indicates an unlimited quota.

```
SQL> SELECT TABLESPACE_NAME, BYTES, MAX_BYTES, BLOCKS,
  2  MAX_BLOCKS
  3  FROM DBA_TS_QUOTAS
  4  WHERE USERNAME = 'JOHN';

TABLESPACE    BYTES     MAX_BYTES      BLOCKS      MAX_BLOCKS
----------    ----------  ----------     ----------  ----------
INDX          0         1048576        0           128
USERS         0         -1             0           -1
```

The V$SESSION view shows the users currently connected to the database, and V$SESSTAT shows the session statistics. You can find the description for the statistic codes from V$SESSTAT in V$STATNAME.

```
SQL> SELECT USERNAME, OSUSER, MACHINE, PROGRAM
  2  FROM V$SESSION
  3  WHERE USERNAME = 'JOHN';

USERNAME    OSUSER      MACHINE     PROGRAM
-------------  ----------  -------------  -----------------
JOHN        KJOHN       USA.CO.AU   SQLPLUSW.EXE


SQL> SELECT A.NAME, B.VALUE
  2  FROM V$STATNAME A, V$SESSTAT B, V$SESSION C
  3  WHERE A.STATISTIC# = B.STATISTIC#
  4  AND B.SID = C.SID
  5  AND C.USERNAME = 'JOHN'
  6  AND A.NAME LIKE '%session%';

NAME                                     VALUE
---------------------------------------  ----------
session logical reads                    729
session stored procedure space           0
CPU used by this session                 12
```

```
session connect time                    0
session uga memory                      98368
session uga memory max                  159804
session pga memory                      296416
session pga memory max                  296416
session cursor cache hits               0
session cursor cache count              0
```

The current username connected to the database is available in the system variable USER. Using SQL*Plus, you can run SHOW USER to get the username.

```
SQL> SHOW USER
USER is "JOHN"
SQL>
```

REAL WORLD SCENARIO

Copying User Accounts from One Database to Another

The password provided with the IDENTIFIED BY clause is encrypted and stored in the database. You can also provide the encrypted password using the IDENTIFIED BY VALUES clause as the user's password. Enclose such entries in single quotes. This method is useful for copying user accounts from one database to another, without compromising their password. The PASSWORD column in the DBA_USERS view provides the encrypted password.

As an example, let's create a user called BENJAMIN with the password NOICE.

```
CREATE USER benjamin IDENTIFIED BY noice
DEFAULT TABLESPACE users;
```

You can query the encrypted password from the data dictionary.

```
SELECT username, password
FROM dba_users
WHERE username = 'BENJAMIN';
USERNAME                       PASSWORD
------------------------------ ----------------------
BENJAMIN                       C2851F759114DCAC
```

You can drop the user from the database and create the user again by supplying the encrypted password. If you execute the same SQL statement in any Oracle database, the password assigned to user BENJAMIN will be NOICE.

```
DROP USER benjamin;

CREATE USER benjamin IDENTIFIED BY
VALUES 'C2851F759114DCAC'
DEFAULT TABLESPACE users;
```

Let's now grant the CREATE SESSION privilege to Benjamin and verify that his password is really NOICE.

```
SQL> GRANT CREATE SESSION TO benjamin;

Grant succeeded.

SQL>

SQL> CONNECT benjamin/noice

Connected.

SQL>
```

Now let's generate a script from the data dictionary to create all user accounts with their existing passwords. You can change the CREATE USER to ALTER USER for synchronizing passwords between databases (for example, your test and production databases).

```
SET ECHO OFF FEEDBACK OFF PAGES 0 LINES 200 TRIMS ON

SPOOL createusers.sql

SELECT 'CREATE USER '|| username || ' IDENTIFIED BY VALUES '''' ||
        password || ''';'
FROM dba_users;

SPOOL OFF

SET FEEDBACK ON PAGES 24 LINES 80
```

The script generated will be saved in file createusers.sql, whose content will be similar to the following.

```
CREATE USER OE IDENTIFIED BY VALUES 'D1A2DFC623FDA40A';

CREATE USER SH IDENTIFIED BY VALUES '54B253CBBAAA8C48';

CREATE USER BENJAMIN IDENTIFIED BY VALUES 'C2851F759114DCAC';
```

3. Managing Privileges

In the Oracle database, *privileges* control access to the data and restrict the actions users can perform. Through proper privileges, users can create, drop, or modify objects in their own schema or in another user's schema. Privileges also determine the data to which a user should have access. You can grant privileges to a user by means of two methods:

- You can assign privileges directly to the user.
- You can assign privileges to a role, and then assign the role to the user.

A *role* is a named set of privileges, which eases the management of privileges. For example, if you have 10 users needing access to the data in the accounting tables, you can grant the privileges required to a role and grant the role to the 10 users. There are two types of privileges:

> **Object privileges** *Object privileges* are granted on schema objects that belong to a different schema. The privilege can be on data (to read, modify, delete, add, or reference), on a program (to execute), or on an object (to change the structure).
>
> **System privileges** *System privileges* provide the right to perform a specific action on any schema in the database. System privileges do not specify an object, but are granted at the database level. Certain system privileges are very powerful and should be granted only to trusted users. System privileges and object privileges can be granted to a role.

PUBLIC is a user group defined in the database; it is not a database user or a role. Every user in the database belongs to this group. Therefore, if you grant privileges to PUBLIC, they are available to all users of the database.

NOTE: A user and a role cannot have the same name.

3.1. Object Privileges

Object privileges are granted on a specific object. The owner of the object has all privileges on the object. The owner can grant privileges on that object to any other users of the database. The owner can also authorize another user in the database to grant privileges on the object to other users. For example, user JOHN owns a table named CUSTOMER and grants read and update privileges to JAMES. (To specify multiple privileges, separate them with a comma.)

```
SQL> GRANT SELECT, UPDATE ON CUSTOMER TO JAMES;
```

JAMES cannot insert into or delete from CUSTOMER; JAMES can only query and update rows in the CUSTOMER table. JAMES cannot grant the privilege to another user in the database, because JAMES is not authorized by JOHN to do so. If the privilege is granted with the WITH GRANT OPTION, JAMES can grant the privilege to others.

```
SQL> GRANT SELECT, UPDATE ON CUSTOMER
  2  TO JAMES WITH GRANT OPTION;
```

The INSERT, UPDATE, or REFERENCES privileges can be granted on columns also. For example:

```
SQL> GRANT INSERT (CUSTOMER_ID) ON CUSTOMER TO JAMES;
```

The following are the object privileges that can be granted to users of the database:

SELECT Grants read (query) access to the data in a table, view, sequence, or materialized view.

UPDATE Grants update (modify) access to the data in a table, column, view, or materialized view.

DELETE Grants delete (remove) access to the data in a table, view, or materialized view.

INSERT Grants insert (add) access to a table, column, view, or materialized view.

EXECUTE Grants execute (run) privilege on a PL/SQL stored object, such as a procedure, package, or function.

READ Grants read access on a directory.

INDEX Grants index creation privilege on a table.

REFERENCES Grants reference access to a table or columns to create foreign keys that can reference the table.

ALTER Grants access to modify the structure of a table or sequence.

ON COMMIT REFRESH Grants the privilege to create a refresh-on-commit materialized view on the specified table.

QUERY REWRITE Grants the privilege to create a materialized view for query rewrite using the specified table.

WRITE Allows the external table agent to write a log file or a bad file to the directory. This privilege is associated only with the external tables.

UNDER Grants the privilege to create a sub-view under a view.

The following are some points related to object privileges that you need to remember:

- Object privileges can be granted to a user, a role, or PUBLIC.
- If a view refers to tables or views from another user, you must have the privilege WITH GRANT OPTION on the underlying tables of the view to grant any privilege on the view to another user. For example, JOHN owns a view, which references a table from JAMES. To grant the SELECT privilege on the view to another user, JOHN should have received the SELECT privilege on the table WITH GRANT OPTION.
- Any object privilege received on a table provides the grantee the privilege to lock the table.
- The SELECT privilege cannot be specified on columns; to grant column-level SELECT privileges, create a view with the required columns and grant SELECT on the view.
- You can specify ALL or ALL PRIVILEGES to grant all available privileges on an object (for example, GRANT ALL ON CUSTOMER TO JAMES).

- Even if you have the DBA privilege, to grant privileges on objects owned by another user you must have been granted the appropriate privilege on the object WITH GRANT OPTION.
- Multiple privileges can be granted to multiple users and/or roles in one statement. For example, GRANT INSERT, UPDATE, SELECT ON CUSTOMER TO ADMIN_ROLE, JULIE, SCOTT;

3.2. System Privileges

System privileges are the privileges that enable the user to perform an action; they are not specified on any particular object. Like object privileges, system privileges also can be granted to a user, a role, or PUBLIC. There are many system privileges in Oracle; Table 1 summarizes the privileges used to manage objects in the database. The CREATE, ALTER, and DROP privileges provide the ability to create, modify, and drop the object specified in the user's schema.

When a privilege is specified with ANY, the user is authorized to perform the action on any schema in the database. Table 2 shows the types of privileges that are associated with certain types of objects. For example, the SELECT ANY TABLE privilege gives the user the ability to query all tables or views in the database, regardless of who owns them; the SELECT ANY SEQUENCE privilege gives the user the ability to select from all sequences in the database.

Table 1. System Privileges for Managing Objects

OBJECT TYPE	CREATE	CREATE ANY	ALTER	ALTER ANY	DROP	DROP ANY	EXECUTE ANY	QUERY REWRITE	GLOBAL QUERY REWRITE
Cluster	√	√		√		√			
Context		√				√			
Database link	√								
Public database link	√				√				
Dimension	√	√		√		√			
Directory		√				√			
Indextype	√	√		√		√			
Index		√		√		√		√	√
Library	√	√			√	√			
Materialized view	√	√		√		√			
Operator	√	√				√	√		
Outline		√		√		√			
Procedure	√	√		√		√	√		
Profile	√		√		√				
Role	√			√		√			
Rollback segment	√		√		√				
Sequence	√	√		√		√			
Snapshot	√	√		√		√			
Synonym	√	√				√			
Public synonym	√				√				
Table	√	√		√		√			
Tablespace	√		√		√				
Trigger	√	√		√		√			
Type	√	√		√		√	√		
User	√		√		√				
View	√		√			√			

Table 2 lists other system privileges that do not fall into the categories outlined in Table 1.

Table 2. Additional System Privileges

If you have	You'll be able to
SELECT ANY TABLE/ SEQUENCE/ OUTLINE	Select from table, sequence or create a clone private outline from a public outline.
SELECT ANY DICTIONARY	Query data dictionary schema objects owned by the SYS schema.
ALTER DATABASE	Change the database configuration by using the ALTER DATABASE command.
ALTER SYSTEM	Use the ALTER SYSTEM command.
AUDIT SYSTEM	Audit SQL statements.
CREATE SESSION	Connect to the database.
ALTER RESOURCE COST	Use the ALTER RESOURCE COST command to set up resource costs.
ALTER SESSION	Change session properties by using ALTER SESSION.
RESTRICTED SESSION	Connect to the database when the database is in restricted mode.
BACKUP ANY TABLE	Export tables that belong to other users.
DELETE ANY TABLE	Delete rows from tables or views owned by any user in the database.
INSERT ANY TABLE	Insert rows into tables or views owned by any user into the database.
LOCK ANY TABLE	Lock tables or views owned by any user in the database.
UPDATE ANY TABLE	Update rows in tables or views owned by any user in the database.
MANAGE TABLESPACE	Perform tablespace management operations such as taking them offline or online and beginning or ending backup.
UNLIMITED TABLESPACE	Create objects in any tablespace; space in the database is not restricted.
ADMINISTER	Create a trigger in the database (you still need the CREATE TRIGGER or CREATE ANY TRIGGER

DATABASE TRIGGER	privilege).
BECOME USER	Become another user while doing a full import.
ANALYZE ANY	Use the ANALYZE command on any table, index, or cluster in any schema in the database.
AUDIT ANY	Audit any object or schema in the database.
COMMENT ANY TABLE	Create comments on any tables in the database.
GRANT ANY PRIVILEGE	Grant any system privilege.
GRANT ANY ROLE	Grant any role.
ON COMMIT REFRESH	Create a refresh-on-commit materialized view on any table in the database.
UNDER ANY VIEW	Create sub-views under object views
SYSOPER	Start up and shut down the database; mount, open, or back up the database; use ARCHIVELOG and RECOVER commands; use the RESTRICTED SESSION privilege.
SYSDBA	Perform all SYSOPER actions plus create or alter a database.

Privileges with the ANY parameter are powerful and should be granted to responsible users. Privileges with the ANY parameter provide access to all such objects in the database, including SYS-owned dictionary objects. For example, if you give a user the ALTER ANY TABLE privilege, that user can use the privilege on a data dictionary table. To protect the dictionary, Oracle provides an initialization parameter, O7_DICTIONARY_ACCESSIBILITY, that controls access to the data dictionary. If this parameter is set to TRUE (the Oracle7 behavior, default is FALSE), a user with the ANY privilege can exercise that privilege on the SYS dictionary objects. It is not possible to access the dictionary with the ANY privilege if this parameter is set to FALSE. For example, a user with SELECT ANY TABLE can query the DBA_ views, but when O7_DICTIONARY_ACCESSIBILITY is set to FALSE, the user cannot query the dictionary views. You can, however, grant the user specific access to the dictionary views (via object privileges). When we discuss roles later in this lesson, you'll learn how to provide query access to the dictionary.

Here are some points to remember about system privileges:

- To connect to the database, you need the CREATE SESSION privilege.

- To truncate a table that belongs to another schema, you need the DROP ANY TABLE privilege.
- The CREATE ANY PROCEDURE (or EXECUTE ANY PROCEDURE) privilege allows the user to create, replace, or drop (or execute) procedures, packages, and functions; this includes Java classes.
- The CREATE TABLE privilege gives you the ability to create, alter, drop, and query tables in a schema.
- SELECT, INSERT, UPDATE, and DELETE are object privileges, but SELECT ANY, INSERT ANY, UPDATE ANY, and DELETE ANY are system privileges (in other words, they do not apply to a particular object).

3.3. Granting System Privileges

System privileges are also granted to a user, a role, or PUBLIC by using the GRANT command. The WITH ADMIN OPTION clause gives the grantee the privilege to grant the privilege to another user, role, or PUBLIC. For example, if JOHN needs to create a table under JAMES's schema, he needs the CREATE ANY TABLE privilege. This privilege not only allows JOHN to create a table under JAMES's schema, but also allows the creation of a table under any schema in the database.

```
SQL> GRANT CREATE ANY TABLE TO JOHN;
```

If John must be able to grant this privilege to others, he should be granted the privilege with the WITH ADMIN OPTION clause (or should have the GRANT ANY PRIVILEGE privilege).

```
SQL> GRANT CREATE ANY TABLE TO JOHN WITH ADMIN OPTION;
```

3.4. Revoking Privileges

You can revoke a user's object privileges and system privileges by using the REVOKE statement. You can revoke a privilege if you have granted it to the user or if you have been granted that privilege with the WITH ADMIN OPTION (for system privileges) or the WITH GRANT OPTION (for object privileges) clauses. Here are some examples of revoking privileges.

To revoke the UPDATE privilege granted to JAMES from JOHN on JOHN's CUSTOMER table, use the following:

```
SQL> REVOKE UPDATE ON CUSTOMER FROM JAMES;
```

To revoke the SELECT ANY TABLE and CREATE TRIGGER privileges granted to JULIE, use the following:

```
SQL> REVOKE SELECT ANY TABLE, CREATE TRIGGER
  2  FROM JULIE;
```

To revoke the REFERENCES privilege, specify the CASCADE CONSTRAINTS clause, which will drop the referential integrity constraint created using the privilege. You must use this clause if any constraints exist.

```
SQL> REVOKE REFERENCES ON CUSTOMER
  2  FROM JAMES CASCADE CONSTRAINTS;
```

The following statement revokes all the privileges granted by JAMES on the STATE table to JULIE. JAMES executes this statement.

```
SQL> REVOKE ALL ON STATE FROM JULIE;
```

Keep the following in mind when revoking privileges:

- If multiple users (or administrators) have granted an object privilege to a user, revoking the privilege by one administrator will not prevent the user from performing the action, because the privileges granted by the other administrators are still valid.
- To revoke the WITH ADMIN OPTION or WITH GRANT OPTION, you must revoke the privilege and re-grant the privilege without the clause.
- You cannot selectively revoke column privileges; you must revoke the privileges from the table and grant them again with the appropriate columns.
- If a user has used their system privileges to create or modify an object, and subsequently the user's privilege is revoked, no change is made to the objects that the user has already created or modified. The user just can no longer create or modify the object.
- If a PL/SQL program or view is created based on an object privilege (or a DML system privilege such as SELECT ANY, UPDATE ANY, and so on), revoking the privilege will invalidate the object.
- If user A is granted a system privilege WITH ADMIN OPTION, and grants the privilege to user B, user B's privilege still remains when user A's privilege is revoked.
- If user A is granted an object privilege WITH GRANT OPTION, and grants the privilege to user B, user B's privilege is also automatically revoked when user A's privilege is revoked, and the objects that use the privileges under user A and user B are invalidated.

3.5. Querying Privilege Information

You can query privilege information from the data dictionary by using various views. Table 3 lists and describes the views that provide information related to privileges.

Table 3. Privilege Information

View Name	Description
ALL_TAB_PRIVS DBA_TAB_PRIVS USER_TAB_PRIVS	Lists the object privileges. ALL_TAB_PRIVS shows only the privileges granted to the user and to PUBLIC.
ALL_TAB_PRIVS_MADE USER_TAB_PRIVS_MADE	Lists the object grants made by the current user or grants made on the objects owned by the current user.
ALL_TAB_PRIVS_RECD USER_TAB_PRIVS_RECD	Lists the object grants received by the current user or PUBLIC.
ALL_COL_PRIVS DBA_COL_PRIVS USER_COL_PRIVS	Lists column privileges.
ALL_COL_PRIVS_MADE USER_COL_PRIVS_MADE	Lists column privileges made by the current user.
ALL_COL_PRIVS_RECD USER_COL_PRIVS_RECD	Lists column privileges received by the current user.
DBA_SYS_PRIVS USER_SYS_PRIVS	Lists system privilege information.
SESSION_PRIVS	Lists the system privileges available for the current session.

Here are some sample queries using the dictionary views to query privileges information.

To list information about privileges granted on the table CUSTOMER, use the following:

```
SQL> SELECT * FROM DBA_TAB_PRIVS
  2   WHERE TABLE_NAME = 'CUSTOMER';
GRANTEE                 OWNER TABLE_NAME GRANTOR PRIVILEGE  GRA
--------------- --------  ---------------- ------------  ------------- ------------
SCOTT                   JOHN   CUSTOMER   JAMES    SELECT      NO
JAMES                   JOHN   CUSTOMER   JOHN     SELECT      YES
JAMES                   JOHN   CUSTOMER   JOHN     UPDATE      YES
ACCOUNTS_MANAGE     JOHN   CUSTOMER   JOHN     SELECT      NO
```

To list system privileges granted to JOHN:

```
SQL> SELECT * FROM DBA_SYS_PRIVS
  2   WHERE GRANTEE = 'JOHN';
GRANTEE     PRIVILEGE                      ADM
--------------- --------------------       ---
JOHN        CREATE SESSION                 NO
```

```
JOHN            UNLIMITED TABLESPACE            NO
```

REAL WORLD SCENARIO

Fixing an Insufficient Privilege Error after Querying the Dictionary

User Benjamin is getting an error message when he tries to do the following update. As a DBA, you need to understand the roles and privileges and grant Benjamin the appropriate privilege.

```
SQL> UPDATE COUNTRIES
  2  SET COUNTRY_NAME = 'USA'
  3  WHERE COUNTRY_ID = 'US';
UPDATE COUNTRIES
       *
ERROR at line 1:
ORA-01031: insufficient privileges
SQL>
```

Since Benjamin is not qualifying the table name with a username, a synonym must exist. Let's find the synonym definition and the owner of the COUNTRIES table.

```
SQL> SELECT * FROM DBA_SYNONYMS
  2  WHERE SYNONYM_NAME = 'COUNTRIES';
OWNER SYNONYM_NAME TABLE_OWNER TABLE_NAME DB_LINK
--------- --------------- ------------ -------------- ----------------------------------
PUBLIC COUNTRIES HR COUNTRIES OE COUNTRIES HR COUNTRIES
SQL>
```

Two synonyms are created; one is a private synonym owned by OE and the other is a public synonym. Benjamin must be using the public synonym. Both synonyms refer to the COUNTRIES table owned by the HR schema. Let's query and find out which privilege on the HR.COUNTRIES table is assigned to Benjamin.

```
SQL> SELECT PRIVILEGE
  2  FROM DBA_TAB_PRIVS
  3  WHERE GRANTEE = 'BENJAMIN'
  4  AND TABLE_NAME = 'COUNTRIES'
  5  AND OWNER = 'HR';
```

```
no rows selected

SQL>
```

No privileges are assigned on table HR.COUNTRIES to Benjamin. Let's query and find out whether the privileges on this table are granted to any role.

```
SQL> SELECT GRANTEE, PRIVILEGE
  2  FROM DBA_TAB_PRIVS
  3  WHERE TABLE_NAME = 'COUNTRIES'
  4  AND OWNER = 'HR'
  5  AND GRANTEE IN (SELECT ROLE FROM DBA_ROLES);

GRANTEE                        PRIVILEGE
------------------------------ ----------------------
HR_SELECT                      SELECT
HR_UPDATE                      DELETE
HR_UPDATE                      INSERT
HR_UPDATE                      SELECT
HR_UPDATE                      UPDATE

SQL>
```

The HR_SELECT role has query privileges, and the HR_UPDATE role has all privileges on this table. Query to find out which roles are assigned to Benjamin.

```
SQL> SELECT * FROM DBA_ROLE_PRIVS
  2  WHERE GRANTEE = 'BENJAMIN';

GRANTEE         GRANTED_ROLE       ADM    DEF
--------------- ------------------ ---    ---
BENJAMIN        CONNECT            NO     YES
BENJAMIN        HR_SELECT          NO     YES

SQL>
```

Benjamin has two roles assigned, CONNECT and HR_SELECT. To fix Benjamin's problem, you can either grant him the HR_UDPATE role or grant update privileges on the HR.COUNTRIES table.

```
GRANT HR_UPDATE TO BENJAMIN;
```

Or

```
CONNECT HR/HR
```

```
GRANT UPDATE ON COUNTRIES TO BENJAMIN;
```

4. Managing Roles

A *role* is a named group of privileges that you can use to ease the administration of privileges. For example, if your accounting department has 30 users and all need similar access to the tables in the accounts receivable application, you can create a role and grant the appropriate system and object privileges to the role. You can grant the role to each user of the accounting department, instead of granting each object and system privilege to individual users.

Using the CREATE ROLE command creates the role. No user owns the role; it is owned by the database. When a role is created, no privileges are associated with it. You must grant the appropriate privileges to the role. For example, to create a role named ACCTS_RECV and grant certain privileges to the role, use the following:

```
CREATE ROLE ACCTS_RECV;

GRANT SELECT ON GENERAL_LEDGER TO ACCTS_RECV;

GRANT INSERT, UPDATE ON JOURNAL_ENTRY TO ACCTS_RECV;
```

Similar to users, roles can also be authenticated. The default is NOT IDENTIFIED, which means no authorization is required to enable or disable the role. The following authorization methods are available:

Database Using a password associated with the role, the database authorizes the role. Whenever such roles are enabled, the user is prompted for a password if the role is not one of the default roles for the user. In the following example, a role ACCOUNTS_MANAGER is created with a password.

```
SQL> CREATE ROLE ACCOUNTS_MANAGER IDENTIFIED BY ACCMGR;
```

Operating system The role is authorized by the operating system. This is useful when the operating system can associate its privileges with the application privileges, and information about each user is configured in operating system files. To enable operating system role authorization, set the parameter OS_ROLES to TRUE. The following example creates a role, authorized by the operating system.

```
SQL> CREATE ROLE APPLICATION_USER IDENTIFIED EXTERNALLY;
```

You can change the role's password or authentication method by using the ALTER ROLE command. You cannot rename a role. For example:

```
SQL> ALTER ROLE ACCOUNTS_MANAGER IDENTIFIED BY MANAGER;
```

To drop a role, use the DROP ROLE command. Oracle will let you drop a role even if it is granted to users or other roles. When you drop a role, it is immediately removed from the users' role lists.

```
SQL> DROP ROLE ACCOUNTS_MANAGER;
```

4.1. Using Predefined Roles

When you create the database, Oracle creates six predefined roles. These roles are defined in the sql.bsq script, which is executed when you run the CREATE DATABASE command. The following roles are predefined:

CONNECT Privilege to connect to the database, to create a cluster, a database link, a sequence, a synonym, a table, and a view, and to alter a session.

RESOURCE Privilege to create a cluster, a table, and a sequence, and to create programmatic objects such as procedures, functions, packages, indextypes, types, triggers, and operators.

DBA All system privileges with the ADMIN option, so the system privileges can be granted to other users of the database or to roles.

SELECT_CATALOG_ROLE Ability to query the dictionary views and tables.

EXECUTE_CATALOG_ROLE Privilege to execute the dictionary packages (SYS-owned packages).

DELETE_CATALOG_ROLE Ability to drop or re-create the dictionary packages.

Also, when you run the catproc.sql script as part of the database creation, the script executes catexp.sql, which creates two more roles:

EXP_FULL_DATABASE Ability to make full and incremental exports of the database using the Export utility.

IMP_FULL_DATABASE Ability to perform full database imports using the Import utility. This is a very powerful role.

4.2. Removing Roles

You can remove roles from the database using the DROP ROLE statement. When you drop a role, all privileges that users had through the role are lost. If they used the role to create objects in the database or to manipulate data, those objects and changes remain in the database. To drop a role named HR_UPDATE, use the following statement:

```
DROP ROLE HR_UPDATE;
```

To drop a role, you must have been granted the role with the ADMIN OPTION, or you must have the DROP ANY ROLE system privilege.

4.3. Enabling and Disabling Roles

If a role is not the default role for a user, it is not enabled when the user connects to the database. You use the ALTER USER command to set the default roles for a user. You can use the DEFAULT ROLE clause with the ALTER USER command in four ways, as illustrated in the following examples.

To specify the named roles CONNECT and ACCOUNTS_MANAGER as default roles, use the following:

```
ALTER USER JOHN DEFAULT ROLE
CONNECT, ACCOUNTS_MANAGER;
```

To specify all roles granted to the user as the default, use the following:

```
ALTER USER JOHN DEFAULT ROLE ALL;
```

To specify all roles except certain roles as the default, use the following:

```
ALTER USER JOHN DEFAULT ROLE ALL
EXCEPT RESOURCE, ACCOUNTS_ADMIN;
```

To specify no roles as the default, use the following:

```
ALTER USER JOHN DEFAULT ROLE NONE;
```

You can specify only roles granted to the user as default roles. The DEFAULT ROLE clause is not available in the CREATE USER command. Default roles are enabled when the user connects to the database and do not require a password.

You enable or disable roles using the SET ROLE command. You specify the maximum number of roles that can be enabled in the initialization parameter MAX_ENABLED_ROLES (the default is 20). You can enable or disable only roles granted to the user. If a role is defined with a password, you must supply the password when you enable the role. For example:

```
SET ROLE ACCOUNTS_ADMIN IDENTIFIED BY MANAGER;
```

To enable all roles, specify the following:

```
SET ROLE ALL;
```

To enable all roles, except the roles specified, use the following:

```
SET ROLE ALL EXCEPT RESOURCE, ACCOUNTS_USER;
```

To disable all roles, including the default roles, use the following:

```
SET ROLE NONE;
```

REAL WORLD SCENARIO How Can You Establish a Security Policy Using Default Roles?
You have an application that was developed in house, and you want to control the application's security through roles. Your users are smart and use tools such as Microsoft Access and SQL*Plus to access the database. You do not want any updates to the database

outside the application, but at the same time you do not want to limit the users' ability to use these other tools. The application tracks data changes, so you want all users to use their own ID to connect to the application and make changes. The application uses the database privileges of the user to make changes.

Solution:

You need at least two roles defined in the database—one to query the application tables and another to update data. The role that updates data needs to be password protected. Let's create two roles.

```
CREATE ROLE APP_QUERY;

CREATE ROLE APP_UPDATE IDENTIFIED BY FNDMYPSWD;
```

Now, grant SELECT privileges on tables to the APP_QUERY role, and grant INSERT, UPDATE, and DELETE privileges on tables to the APP_UPDATE role.

Grant the necessary roles to users.

```
GRANT APP_QUERY, APP_UPDATE TO CHRIS;
```

Next, change the users default role to only APP_QUERY.

```
ALTER USER CHRIS DEFAULT ROLE APP_QUERY;
```

If you have other roles that you want to establish as the default for the user, you can use the following:

```
ALTER USER CHRIS DEFAULT ROLE ALL EXCEPT APP_UPDATE;
```

You can query the DBA_ROLE_PRIVS view to see default and non-default roles for the user. The DEFAULT_ROLE column will display NO for non-default roles. In the application, you must set the appropriate role that gives the privilege to manipulate the data.

```
SET ROLE APP_UPDATE IDENTIFIED BY FNDMYPSWD;
```

If you need to set more than one role, you can specify all the required roles in the SET command.

```
SET ROLE APP_UPDATE IDENTIFIED BY FNDMYPSWD, APP_ADMIN;
```

You can develop methods to encrypt and decrypt passwords, instead of hard coding them in the application.

5. Querying Role Information

The data dictionary view DBA_ROLES lists the roles defined in the database. The column PASSWORD specifies the authorization method.

```
SQL> SELECT * FROM DBA_ROLES;

ROLE                            PASSWORD
```

```
------------------------------  --------
CONNECT                         NO
RESOURCE                        NO
DBA                             NO
SELECT_CATALOG_ROLE             NO
EXECUTE_CATALOG_ROLE            NO
DELETE_CATALOG_ROLE             NO
EXP_FULL_DATABASE               NO
IMP_FULL_DATABASE               NO
APPLICATION_USER                EXTERNAL
ACCOUNTS_MANAGER                YES
```

The view SESSION_ROLES lists the roles that are enabled in the current session.

```
SQL> SELECT * FROM SESSION_ROLES;
ROLE
------------------------------
CONNECT
DBA
SELECT_CATALOG_ROLE
HS_ADMIN_ROLE
EXECUTE_CATALOG_ROLE
DELETE_CATALOG_ROLE
EXP_FULL_DATABASE
IMP_FULL_DATABASE
JAVA_ADMIN
```

The view DBA_ROLE_PRIVS (or USER_ROLE_PRIVS) lists all the roles granted to users and roles.

```
SQL> SELECT * FROM DBA_ROLE_PRIVS
  2   WHERE GRANTEE = 'JOHN';
GRANTEE     GRANTED_ROLE          ADM   DEF
----------  --------------------  ---   ---
JOHN        ACCOUNTS_MANAGER      YES   NO
```

```
JOHN            RESOURCE                       NO     YES
```

The view ROLE_ROLE_PRIVS lists the roles granted to the roles, ROLE_SYS_PRIVS lists the system privileges granted to roles, and ROLE_TAB_PRIVS shows information on the object privileges granted to roles.

```
SQL> SELECT * FROM ROLE_ROLE_PRIVS
  2  WHERE ROLE = 'DBA';

ROLE  GRANTED_ROLE                   ADM
------- ------------------------------ ---
DBA   DELETE_CATALOG_ROLE            YES
DBA   EXECUTE_CATALOG_ROLE           YES
DBA   EXP_FULL_DATABASE              NO
DBA   IMP_FULL_DATABASE              NO
DBA   JAVA_ADMIN                     NO
DBA   SELECT_CATALOG_ROLE            YES

SQL> SELECT * FROM ROLE_SYS_PRIVS
  2  WHERE ROLE = 'CONNECT';

ROLE           PRIVILEGE                ADM
-------------- ------------------------ ---
CONNECT        ALTER SESSION            NO
CONNECT        CREATE CLUSTER           NO
CONNECT        CREATE DATABASE LINK     NO
CONNECT        CREATE SEQUENCE          NO
CONNECT        CREATE SESSION           NO
CONNECT        CREATE SYNONYM           NO
CONNECT        CREATE TABLE             NO
CONNECT        CREATE VIEW              NO

SQL> SELECT * FROM ROLE_TAB_PRIVS
  2  WHERE TABLE_NAME = 'CUSTOMER';
```

```
ROLE          OWNER TABLE_NAME COLUMN_NAME    PRIVILEGE GRA
------------  ------ ---------- -----------  --------- ---
ACC_MANAGER JOHN     CUSTOMER                 SELECT    NO
```

6. Auditing the Database

Auditing is storing information about database activity. You can use auditing to monitor suspicious database activity and to collect statistics on database usage. When you create the database, Oracle creates the SYS.AUD$ table, known as the *audit trail*, which stores the audited records. To enable auditing, set the initialization parameter AUDIT_TRAIL to TRUE or DB. When this parameter is set to OS, Oracle writes the audited records to an operating system file instead of inserting them into the SYS.AUD$ table. You use the AUDIT command to specify the audit actions. Oracle has three types of auditing capabilities:

Statement auditing Audits SQL statements. (Example: AUDIT SELECT BY SCOTT audits all SELECT statements performed by SCOTT.)

Privilege auditing Audits privileges. (Example: AUDIT CREATE TRIGGER audits all users who exercise their CREATE TRIGGER privilege.)

Object auditing Audits the use of a specific object. (Example: AUDIT SELECT ON JOHN.CUSTOMER monitors the SELECT statements performed on the CUSTOMER table.)

You can restrict the auditing scope by specifying the user list in the BY clause. You can use the WHENEVER SUCCESSFUL clause to specify that only successful statements are to be audited. The WHENEVER NOT SUCCESSFUL clause limits auditing to failed statements. You can also specify BY SESSION or BY ACCESS; BY SESSION as the default. BY SESSION specifies that one audit record is inserted for one session, regardless of the number of times the statement is executed. BY ACCESS specifies that one audit record is inserted each time the statement is executed. Following are some examples of auditing.

To audit the connection and disconnection to the database, use the following:

```
AUDIT SESSION;
```

To audit only successful logins, use the following:

```
AUDIT SESSION WHENEVER SUCCESSFUL;
```

To audit only failed logins, use the following:

```
AUDIT SESSION WHENEVER NOT SUCCESSFUL;
```

To audit successful logins of specific users, use the following:

```
AUDIT SESSION BY JOHN, ALEX WHENEVER SUCCESSFUL;
```

To audit the successful updates and deletes on the CUSTOMER table, use the following:

```
AUDIT UPDATE, DELETE ON JOHN.CUSTOMER
```

```
BY ACCESS WHENEVER SUCCESSFUL;
```

To turn off auditing, use the NOAUDIT command. You can specify all options available in the AUDIT statement to turn off auditing except BY SESSION and BY ACCESS. When you turn off auditing, Oracle turns off the action, regardless of its BY SESSION or BY ACCESS specification. To turn off the object auditing enabled on the CUSTOMER table, use the following:

```
NOAUDIT UPDATE, DELETE ON JOHN.CUSTOMER;
```

TIP: Oracle always monitors certain database activities and writes them to operating system files, even if auditing is disabled. Oracle writes audit records when the instance starts up and shuts down and when a user connects to the database with administrator privileges.

7. Using Globalization Support

You use *Globalization Support*, known as *National Language Support* (NLS) in previous releases of Oracle, to store and retrieve data in a native language and format.

Oracle supports a wide variety of languages and character sets. Globalization support lets you communicate with end users in their native language using their familiar date formats, number formats, and sorting sequence. Oracle uses Unicode (a worldwide encoding standard for computer usage) to support the languages.

You define the database character set when you create the database using the CHARACTER SET clause of the CREATE DATABASE command. The character set stores data in the CHAR, VARCHAR2, CLOB, and LONG columns, and stores table names, column names, and so on in the dictionary, stores PL/SQL variables in memory, and so on. If you do not specify a character set when you create the database, Oracle uses the US7ASCII character set. US7ASCII is a seven-bit ASCII character set that uses a single byte to store a character, and it can represent 128 characters (27).

Other widely used single-byte character sets are WE8ISO8859P1 (the Western European eight-bit ISO [International Organization for Standardization] standard 8859 Part I) and UTF8. These character sets use eight bits to represent a character and can represent 256 characters (28). Oracle also supports multibyte character encoding. Multibyte encoding is used to represent languages such as Japanese, Chinese, Hindi, and so on. Multibyte encoding schemes can be fixed-width encoding schemes or variable-width encoding schemes. In a variable-width encoding scheme, certain characters are represented using one byte, and two or more bytes represent other characters.

The options to change the database character set after you create the database are limited. You can change the database character set only if the new character set is a superset of the current character set; that is, all the characters represented in the current character set must be available in the new character set. WE8ISO8859P1 and UTF8 are a superset of US7ASCII. To change the database character set, use the following:

```
ALTER DATABASE CHARACTER SET WE8ISO8859P1;
```

> *WARNING*: You must be careful when changing the database character set; be sure to back up of the database before the change. You cannot roll back this action, which may result in loss of data or data corruption.

Oracle lets you choose an additional character set for the database that enhances the character-processing capabilities. You specify the second character set when you create the database using the NATIONAL CHARACTER SET clause. If you do not specify NATIONAL CHARACTER SET, Oracle uses the Unicode character set AF16UTF16. The national character set stores data in NCHAR, NVARCHAR2, and NCLOB data type columns.

The national character set can be either AF16UTF16 or UTF8. AF16UTF16 is the default. AF16UTF16 and UTF8 are Unicode character sets. You cannot specify AF16UTF16 as the database character set. When choosing a multibyte character set for your database, remember that, by default, the VARCHAR2, CHAR, NVARCHAR2, and NCHAR data types specify the maximum length in bytes, not in characters. You can change this default behavior by setting the NLS_LENGTH_SEMANTICS=CHAR or by providing the semantic information along with the column definition (as in VARCHAR2 (20 CHAR)). If the character set used is two bytes, VARCHAR2 (10) can hold a maximum of five characters.

The client machine can specify a character set different from the database character set by using local environment variables. The database character set should be a superset to the client character set. Oracle converts the character set automatically, but there is some overhead associated with this conversion.

Certain character sets can support multiple languages. For example, the character set WE8ISO8859P1 can support all western European languages such as English, Finnish, Italian, Swedish, Danish, French, German, Spanish, and so on.

7.1. The Unicode Character Set

Unicode is a universal character encoding scheme that allows you to store information from any major language using a single character set. Unicode provides a unique code value for every character, regardless of the platform, program, or language. Unicode has both 16-bit and 8-bit encoding. UTF-16 is the 16-bit encoding of Unicode. It is a fixed-width multibyte encoding in which the character codes 0x00 through 0x7F have the same meaning as they do in ASCII. One Unicode character is 2 bytes in this encoding.

Characters from both European and Asian scripts are represented in 2 bytes. AF16UTF16 is UTF-16 encoded character set. UTF-8 is the 8-bit encoding of Unicode. It is a variable-width multibyte encoding in which the character codes 0x00 through 0x7F have the same meaning as they do in ASCII. One Unicode character can be 1 byte, 2 bytes, or 3 bytes in this encoding. Characters from the European scripts are represented in either 1 or 2 bytes, and

characters from most Asian scripts are represented in 3 bytes. AL32UTF8, UTF8, and UTFE are UTF-8 encoded character sets. You can specify a UTF-8 encoded character set as a database character set; such a database is known as a *Unicode database*.

7.2. Using NLS Parameters

Oracle provides several NLS parameters to customize the database and client workstations to suit the native format. These parameters have a default value based on the database and national character set chosen to create the database. Specifying the NLS parameters can change the default values. You can customize the NLS parameters in the following ways:

- By specifying the initialization file, which will be used at the instance startup (Example: NLS_DATE_FORMAT = "YYYY-MM-DD")
- By setting the parameter as an environment variable (Example on Unix: csh: setenv NLS_DATE_FORMAT YYYY-MM-DD or using the MS-Windows registry)
- By setting the parameter in the Oracle session using the ALTER SESSION command (Example: ALTER SESSION SET NLS_DATE_FORMAT = "YYYY-MM-DD")
- By using certain SQL functions (Example: TO_CHAR (SYSDATE, 'YYYY-MM-DD', 'NLS_DATE_LANGUAGE = AMERICAN'))

The parameter specified in SQL functions has the highest priority; the next highest is the parameter specified using ALTER SESSION, then the environment variable, then the initialization parameters, and finally the database default parameters have the lowest priority. You cannot change certain parameters using ALTER SESSION, and you cannot specify certain parameters as environment variables. Parameter specification areas are discussed in the following sections.

NLS_LENGTH_SEMANTICS Specified at the session level (using ALTER SESSION) or as an initialization parameter. Defines the character length semantics as byte or character. The default is BYTE. NLS_LENGTH_SEMANTICS does not apply to tables in SYS and SYSTEM; they are always in BYTE semantic.

NLS_LANG Specified only as an environment variable. NLS_LANG has three parts: the language, the territory, and the character set. None of the parts are mandatory. The format to specify NLS_LANG is <LANGUAGE>_<TERRITORY>.<CHARACTER SET>. The language specifies the language to be used for displaying Oracle error messages, day names, month names, and so on. The territory specifies the default date format, numeric formats, and monetary formats. The character set specifies the character set to be used by the client machine—for example, AMERICAN_AMERICA.WE8ISO8859P1, in which AMERICAN is the language, AMERICA is the territory, and WE8ISO8859P1 is the character set.

NLS_LANGUAGE Specified at the session level or as an initialization parameter. Sets the language to be used. The session value overrides the NLS_LANG setting. The default values for the NLS_DATE_LANGUAGE and NLS_SORT parameters are derived from NLS_LANGUAGE.

NLS_TERRITORY Specified at the session level or as an initialization parameter. Sets the territory. The session value overrides the NLS_LANG setting. The default values for parameters such as NLS_CURRENCY, NLS_ISO_CURRENCY, NLS_DATE_FORMAT, and NLS_NUMERIC_CHARACTERS are derived from NLS_TERRITORY.

NLS_DATE_FORMAT Specified at the session level, as an environment variable, or as an initialization parameter. Sets a default format for date displays.

NLS_DATE_LANGUAGE Specified at the session level, as an environment variable, or as an initialization parameter. Sets a language explicitly for day and month names in date values.

NLS_TIMESTAMP_FORMAT Specified at the session level, as an environment variable, or as an initialization parameter. This parameter defines the default timestamp format to use with the TO_CHAR and TO_TIMESTAMP functions.

NLS_TIMESTAMP_TZ_FORMAT Specified at the session level, as an environment variable, or as an initialization parameter. This parameter defines the default timestamp with time zone format to use with the TO_CHAR and TO_TIMESTAMP_TZ functions

NLS_CALENDAR Specified at the session level, as an environment variable, or as an initialization parameter. Sets the calendar Oracle uses.

NLS_NUMERIC_CHARACTERS Specified at the session level, as an environment variable, or as an initialization parameter. Specifies the decimal character and group separator (for example, in 234,224.99, the comma is the group separator and the period is the decimal character).

NLS_CURRENCY Specified at the session level, as an environment variable, or as an initialization parameter. Specifies a currency symbol.

NLS_ISO_CURRENCY Specified at the session level, as an environment variable, or as an initialization parameter. Specifies the ISO currency symbol. For example, when the NLS_ISO_CURRENCY value is AMERICA, the currency symbol for U.S. dollars is $, and the ISO currency symbol is USD.

NLS_DUAL_CURRENCY Specified at the session level, as an environment variable, or as an initialization parameter. Specifies an alternate currency symbol. Introduced to support the Euro.

NLS_SORT Specified at the session level, as an environment variable, or as an initialization parameter. Specifies the language to use for sorting.

You can specify any valid language. The ORDER BY clause in a SQL statement uses this value for the sort mechanism. For example:

```
ALTER SESSION SET NLS_SORT = GERMAN;

SELECT * FROM CUSTOMERS

ORDER BY NAME;
```

In this example, the NAME column will be sorted using the German linguistic sort mechanism. You can also explicitly set the sort language by using the NLSSORT function, rather than altering the session parameter. The following example demonstrates this method:

```
SELECT * FROM CUSTOMERS
```

```
ORDER BY NLSSORT(NAME, "NLS_SORT= GERMAN");
```

7.3. Using NLS to Change Application Behavior

Globalization Support enables applications to use local symbols and semantics regardless of where the database resides. The various NLS parameters define locale specific to the country and language. Oracle9i supports many languages. The user of the application need not know any other culture or language or conventions to use the application. You can set the following NLS parameters so that everything is in local format for the user.

Language You can store, retrieve, and manipulate data in local native languages. Oracle9i supports all major languages and subsets of languages using the Unicode character set. Use the NLS_LANG and the NLS_LANGUAGE parameters.

Geographical location You can set the geographic-specific information, such as currency symbol, date formats, and numeric conventions, local to the user. Use NLS_TERRITORY.

Date and time formats You can set date and time formats specific to the location or convenience. Use NLS_DATE_FORMAT, NLS_DATE_LANGUAGE, NLS_TIMESTAMP_FORMAT, and NLS_TIMESTAMP_TZ_FORMAT.

Currency and numeric formats You can display the currency symbol local to the application. Some countries use the comma (,) as a decimal separator, and some countries use the period (.) as a decimal separator.

You can specify such settings using NLS_CURRENCY, NLS_DUAL_CURRENCY, NLS_ISO_CURRENCY, and NLS_NUMERIC_CHARACTERS.

Calendar and Sorting You can set local calendars and specify linguistic sorting using the NLS parameters NLS_CALENDAR and NLS_SORT.

7.4. Obtaining NLS Data Dictionary Information

You can obtain NLS data dictionary information from the data dictionary using the following views:

NLS_DATABASE_PARAMETERS Shows the parameters defined for the database (the database default values)

NLS_INSTANCE_PARAMETERS Shows the parameters specified in the initialization parameter file

NLS_SESSION_PARAMETERS Shows the parameters that are in effect in the current session

V$NLS_VALID_VALUES Shows the allowed values for the language, territory, and character set definitions

The following examples show NLS information from the data dictionary views and examples of changing session NLS values.

```
SQL> SELECT * FROM NLS_DATABASE_PARAMETERS;
PARAMETER                      VALUE
------------------------       ------------------------------
NLS_LANGUAGE                   AMERICAN
NLS_TERRITORY                  AMERICA
NLS_CURRENCY                   $
NLS_ISO_CURRENCY               AMERICA
NLS_NUMERIC_CHARACTERS         .,
NLS_CHARACTERSET               UTF8
NLS_CALENDAR                   GREGORIAN
NLS_DATE_FORMAT                DD-MON-YY
NLS_DATE_LANGUAGE              AMERICAN
NLS_SORT                       BINARY
NLS_TIME_FORMAT                HH.MI.SSXFF AM
NLS_TIMESTAMP_FORMAT           DD-MON-YY HH.MI.SSXFF AM
NLS_TIME_TZ_FORMAT             HH.MI.SSXFF AM TZH:TZM
NLS_TIMESTAMP_TZ_FORMAT DD-MON-YY HH.MI.SSXFF AM TZH:T
NLS_DUAL_CURRENCY              $
NLS_COMP                       BINARY
NLS_NCHAR_CHARACTERSET         US7ASCII
NLS_RDBMS_VERSION              9.0.1.1.1
NLS_LENGTH_SEMANTICS           BYTE
NLS_NCHAR_CONV_EXCP            FALSE
20 rows selected.

SQL> ALTER SESSION SET NLS_DATE_FORMAT =
'DD-MM-YYYY HH24:MI:SS';
Session altered.
SQL> ALTER SESSION SET NLS_DATE_LANGUAGE = 'GERMAN';
```

```
Session altered.
SQL> SELECT TO_CHAR(SYSDATE, 'Day, Month'),
                                        SYSDATE FROM DUAL;
TO_CHAR(SYSDATE,'DAY, SYSDATE
-------------------- -------------------
Dienstag ,      August          29-08-2000  16:24:14
SQL> ALTER SESSION SET NLS_CALENDAR = "Persian";
Session altered.
SQL> SELECT SYSDATE FROM DUAL;
SYSDATE
-----------------
09 Shahruoar 1379
SQL> SELECT * FROM NLS_SESSION_PARAMETERS;
PARAMETER                      VALUE
------------------------ ------------------------------
NLS_LANGUAGE                   AMERICAN
NLS_TERRITORY                  AMERICA
NLS_CURRENCY                   $
NLS_ISO_CURRENCY               AMERICA
NLS_NUMERIC_CHARACTERS         .,
NLS_CHARACTERSET               UTF8
NLS_CALENDAR                   GREGORIAN
NLS_DATE_FORMAT                DD-MON-RR
NLS_DATE_LANGUAGE              AMERICAN
NLS_SORT                       BINARY
NLS_TIME_FORMAT HH.MI.SSXFF    AM
NLS_TIMESTAMP_FORMAT           DD-MON-RR HH.MI.SSXFF AM
NLS_TIME_TZ_FORMAT             HH.MI.SSXFF AM TZR
NLS_TIMESTAMP_TZ_FORMAT        DD-MON-RR HH.MI.SSXFF AM TZR
NLS_DUAL_CURRENCY              $
```

```
NLS_COMP                         BINARY
NLS_LENGTH_SEMANTICS             BYTE
NLS_NCHAR_CONV_EXCP              FALSE
NLS_NCHAR_CHARACTERSET           AL16UTF16
NLS_RDBMS_VERSION                9.0.1.1.1
20 rows selected
SQL>
```

8. Summary

This lesson discussed the security aspects of the Oracle database: profiles, privileges, and roles. You use profiles to control the database and system resource usage. You also use profiles to manage passwords. You can create various profiles for different user communities and assign a profile to each user. When you create the database, Oracle creates a profile named DEFAULT, which is assigned to the users when you do not specify a profile for the user. Profiles can monitor the resource use by session or on a per-call basis. Resource limits are enforced only when the parameter RESOURCE_LIMIT is set to TRUE.

Using profiles, you can lock an account, manage password expiration and reuse, and verify password complexity. When an account is locked, the DBA must unlock it in order for the user to connect to the database. You cannot drop profiles when they are assigned to users; you can drop such profiles only when you use the CASCADE keyword.

You create users in the database to use the database. Every user needs the CREATE SESSION privilege to be able to connect to the database. Users are assigned a default and a temporary tablespace. When the user does not specify a tablespace when creating a table, the table is created in the default tablespace. The temporary tablespace is used for creating sort segments. If you do not specify default or temporary tablespaces, Oracle assigns SYSTEM as the user's default and temporary tablespace. Users are granted a space quota on the tablespace. When they exceed this quota, no more extents can be allocated to the objects. If a user has the UNLIMITED TABLESPACE system privilege, there are no space quota restrictions.

Before connecting a user to the database, Oracle authenticates the username. The authentication method can be via the database, whereby the user specifies a password, or it can be via the operating system. The operating system authentication method uses the operating system login information to connect to the database; such users are created with the IDENTIFIED EXTERNALLY clause. You cannot drop a user who is connected to the database. You must terminate the user session before dropping the user.

To control the actions performed by users on the database, you use privileges.

A role is a named set of privileges, which makes managing privileges easy. There are two types of privileges: object and system. Object privileges specify allowed actions on a specific object (the owner of the object has to grant the privilege to other users or authorize other users to grant the privileges on the object); system privileges specify the allowed actions on the database. To manage privileges, you use the GRANT and REVOKE commands.

Any privilege granted to PUBLIC is available to all users in the database. You can monitor database actions or statements using the AUDIT command. You can audit statements, privilege usage, or object usage, and you can restrict auditing to specific users, successful statements, or failed statements. You can also limit the number of audit records generated by specifying auditing by session (one record per session) or by access (one record per DDL, DML, and so on).

Oracle can store and retrieve data in a native language and format using the Globalization Support feature. The character set used determines the default language and the conventions used. You specify the character set when you create the database. You can specify only the Unicode character set as the national character set for the database. Oracle provides several parameters that can determine the characteristics and conventions of data displayed to the user. You can specify these parameters for a session, for the instance, or in certain SQL functions. If none are specified, the database defaults are used.

References

[1] Oracle9i DBA Fundamentals I.