Managing Oracle Users

Each Oracle database has a list of valid database users. To access a database, a user must run a database application and connect to the database instance using a valid user name defined in the database. This section explains how to manage users for a database, and contains the following topics:

- Creating Users
- Altering Users
- Dropping Users

> **See Also:** *Oracle9i SQL Reference* for more information about SQL statements used for managing users

Creating Users

You create a database user with the CREATE USER statement.To create a user, you must have the CREATE USER system privilege. Because it is a powerful privilege, a DBA or security administrator is normally the only user who has the CREATE USER system privilege.

The following example creates a user and specifies that user's password, default tablespace, temporary tablespace where temporary segments are created, tablespace quotas, and profile.

```
CREATE USER jward
    IDENTIFIED BY aZ7bC2
    DEFAULT TABLESPACE data_ts
    QUOTA 100M ON test_ts
    QUOTA 500K ON data_ts
    TEMPORARY TABLESPACE temp_ts
    PROFILE clerk;
GRANT connect TO jward;
```

A newly created user cannot connect to the database until granted the CREATE SESSION system privilege. Usually, a newly created user is granted a role similar to the predefined roll CONNECT (used in this example) that specifies the CREATE SESSION and other basic privileges required to access a database.

This section refers to the above example as it discusses the following aspects of creating a user:

- Specifying a Name
- Setting a User's Authentication

- Assigning a Default Tablespace
- Assigning Tablespace Quotas
- Assigning a Temporary Tablespace
- Specifying a Profile
- Setting Default Roles

See Also: "Granting System Privileges and Roles" on page 25-11

Specifying a Name

Within each database a user name must be unique with respect to other user names and roles. A user and role cannot have the same name. Furthermore, each user has an associated schema. Within a schema, each schema object must have a unique name.

Setting a User's Authentication

In the previous CREATE USER statement, the new user is to be authenticated using the database. In this case, the connecting user must supply the correct password to the database to connect successfully.

Selecting and specifying the method of user authentication is discussed in "User Authentication Methods" on page 24-8.

Assigning a Default Tablespace

Each user should have a default tablespace. When a user creates a schema object and specifies no tablespace to contain it, Oracle stores the object in the user's default tablespace.

The default setting for every user's default tablespace is the SYSTEM tablespace. If a user does not create objects, and has no privileges to do so, this default setting is fine. However, if a user creates any type of object, you should specifically assign the user a default tablespace. Using a tablespace other than SYSTEM reduces contention between data dictionary objects and user objects for the same datafiles. In general, it is not advisable for user data to be stored in the SYSTEM tablespace.

You can set a user's default tablespace during user creation, and change it later with the ALTER USER statement. Changing the user's default tablespace affects only objects created after the setting is changed.

When you specify the user's default tablespace, also specify a quota on that tablespace.

In the previous CREATE USER statement, jward's default tablespace is data_ts, and his quota on that tablespace is 500K.

Assigning Tablespace Quotas

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

- Users with privileges to create certain types of objects can create those objects in the specified tablespace.
- Oracle 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, 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.

You can assign a user either individual quotas for a specific amount of disk space in each tablespace or an unlimited amount of disk space in all tablespaces. Specific quotas prevent a user's objects from consuming too much space in the database.

You can assign a user's tablespace quotas when you create the user, or add or change quotas later. If a new quota is less than the old one, then the following conditions hold true:

- If a user has already exceeded a new tablespace quota, the user's objects in the tablespace cannot be allocated more space until the combined space of these objects falls below the new quota.
- If a user has not exceeded a new tablespace quota, or if the space used by the user's objects in the tablespace falls under a new tablespace quota, the user's objects can be allocated space up to the new quota.

Revoking Users Ability to Create Objects in a Tablespace You can revoke a user's ability to create objects in a tablespace by changing the user's current quota to zero. After a quota of zero is assigned, the user's objects in the tablespace remain, but new objects cannot be created and existing objects cannot be allocated any new space.

UNLIMITED TABLESPACE System Privilege To permit a user to use an unlimited amount of any tablespace in the database, grant the user the UNLIMITED TABLESPACE system privilege. This overrides all explicit tablespace quotas for the user. If you later revoke the privilege, explicit quotas again take effect. You can grant this privilege only to users, not to roles.

Before granting the UNLIMITED TABLESPACE system privilege, consider the consequences of doing so.

Advantage:

- You can grant a user unlimited access to all tablespaces of a database with one statement.

Disadvantages:

- The privilege overrides all explicit tablespace quotas for the user.
- You cannot selectively revoke tablespace access from a user with the UNLIMITED TABLESPACE privilege. You can grant access selectively only after revoking the privilege.

Assigning a Temporary Tablespace

Each user also should be assigned a temporary tablespace. When a user executes a SQL statement that requires a temporary segment, Oracle stores the segment in the user's temporary tablespace. These temporary segments are created by the system when doing sorts or joins and are owned by SYS, which has resource privileges in all tablespaces.

In the previous CREATE USER statement, jward's temporary tablespace is temp_ts, a tablespace created explicitly to contain only temporary segments. Such a tablespace is created using the CREATE TEMPORARY TABLESPACE statement.

If a user's temporary tablespace is not explicitly set, the user is assigned the default temporary tablespace that was specified at database creation, or by an ALTER DATABASE statement at a later time. If there is no default temporary tablespace, the default is the SYSTEM tablespace. It is not advisable for user data to be stored in the SYSTEM tablespace. Also, assigning a tablespace to be used specifically as a temporary tablespace eliminates file contention among temporary segments and other types of segments.

Note: If your SYSTEM tablespace is a locally managed tablespace, then users must by assigned a specific default (locally managed) temporary tablespace and not be allowed to default to using the SYSTEM tablespace. This is because temporary objects cannot be placed in permanent locally managed tablespaces.

You can set a user's temporary tablespace at user creation, and change it later using the ALTER USER statement. Do not set a quota for temporary tablespaces.

See Also:

- "Temporary Tablespaces" on page 11-12
- "Creating a Default Temporary Tablespace" on page 2-24

Specifying a Profile

You also specify a profile when you create a user. A profile is a set of limits on database resources and password access to the database. If no profile is specified, the user is assigned a default profile.

See Also:

- "Managing Resources with Profiles" on page 24-18
- "Password Management Policy" on page 23-12

Setting Default Roles

You cannot set a user's default roles in the CREATE USER statement. When you first create a user, the user's default role setting is ALL, which causes all roles subsequently granted to the user to be default roles. Use the ALTER USER statement to change the user's default roles.

See Also: "Specifying Default Roles" on page 25-21

Altering Users

Users can change their own passwords. However, to change any other option of a user's security domain, you must have the ALTER USER system privilege. Security administrators are normally the only users that have this system privilege, as it allows a modification of *any* user's security domain. This privilege includes the ability to set tablespace quotas for a user on any tablespace in the database, even if the user performing the modification does not have a quota for a specified tablespace.

You can alter a user's security settings with the ALTER USER statement. Changing a user's security settings affects the user's future sessions, not current sessions.

The following statement alters the security settings for user avyrros:

```
ALTER USER avyrros
    IDENTIFIED EXTERNALLY
    DEFAULT TABLESPACE data_ts
    TEMPORARY TABLESPACE temp_ts
    QUOTA 100M ON data_ts
```

```
    QUOTA 0 ON test_ts
    PROFILE clerk;
```

The ALTER USER statement here changes avyrros's security settings as follows:

- Authentication is changed to use avyrros's operating system account.
- avyrros's default and temporary tablespaces are explicitly set.
- avyrros is given a 100M quota for the data_ts tablespace.
- avyrros's quota on the test_ts is revoked.
- avyrros is assigned the clerk profile.

Changing a User's Authentication Mechanism

Most non-DBA users can still change their own passwords with the ALTER USER statement, as follows:

```
ALTER USER andy
   IDENTIFIED BY swordfish;
```

No special privileges (other than those to connect to the database) are required for a user to change passwords. Users should be encouraged to change their passwords frequently.

Users must have the ALTER USER privilege to switch between methods of authentication. Usually, only an administrator has this privilege.

> **See Also:** "User Authentication Methods" on page 24-8 for information about the authentication methods that are available for Oracle users

Changing a User's Default Roles

A default role is one that is automatically enabled for a user when the user creates a session. You can assign a user zero or more default roles.

> **See Also:** Chapter 25, "Managing User Privileges and Roles" for information about changing users' default roles

Dropping Users

When a user is dropped, the user and associated schema are removed from the data dictionary and all schema objects contained in the user's schema, if any, are immediately dropped.

> **Note:** If a user's schema and associated objects must remain but the user must be denied access to the database, revoke the CREATE SESSION privilege from the user.

A user that is currently connected to a database cannot be dropped. To drop a connected user, you must first terminate the user's sessions using the SQL statement ALTER SYSTEM with the KILL SESSION clause.

You can drop a user from a database using the DROP USER statement. To drop a user and all the user's schema objects (if any), you must have the DROP USER system privilege. Because the DROP USER system privilege is so powerful, a security administrator is typically the only type of user that has this privilege.

If the user's schema contains any schema objects, use the CASCADE option to drop the user and all associated objects and foreign keys that depend on the tables of the user successfully. If you do not specify CASCADE and the user's schema contains objects, an error message is returned and the user is not dropped. Before dropping a user whose schema contains objects, thoroughly investigate which objects the user's schema contains and the implications of dropping them. Pay attention to any unknown cascading effects. For example, if you intend to drop a user who owns a table, check whether any views or procedures depend on that particular table.

The following statement drops user jones and all associated objects and foreign keys that depend on the tables owned by jones.

```
DROP USER jones CASCADE;
```

> **See Also:** "Terminating Sessions" on page 5-21 for more information about terminating sessions

User Authentication Methods

Oracle provides several means for users to be authenticated before they are allowed to create a database session:

1. You can define users such that the database performs both identification and authentication of users. This is called **database authentication**.
2. You can define users such that authentication is performed by the operating system or network service. This is called **external authentication**.
3. You can define users such that they are authenticated **globally** by SSL (Secure Sockets Layer). These users are called **global users**. For global users, an

enterprise directory can be used to authorize their access to the database through **global roles**.

4. You can specify users who are allowed to connect through a middle-tier server. The middle-tier server authenticates and assumes the identity of the user and is allowed to enable specific roles for the user. This is called **proxy authentication** and authorization.

These means of authentication are discussed in the following sections:

- Database Authentication
- External Authentication
- Global Authentication and Authorization
- Proxy Authentication and Authorization

Database Authentication

If you choose database authentication for a user, administration of the user account including authentication of that user is performed entirely by Oracle. To have Oracle authenticate a user, specify a password for the user when you create or alter the user. 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 your database uses a multibyte character set.

Note: Oracle Corporation recommends that you encode user names and passwords in ASCII or EBCDIC characters only, depending on your platform. This will maintain compatibility for supporting future changes to your database character set.

If user names or passwords are created based on characters that will have size expansion when migrating to a new target character set, then users can experience login difficulties due to authentication failures after the migration. This is because the encrypted user names and passwords stored in the data dictionary do not get updated during the migration to the new database character set.

For example, assuming the current database character set is WE8MSWIN1252 and the target database character set is UTF8, the user name `scött` (o with an umlaut) will change from 5 bytes to 6 bytes in UTF8. the user `scött` will no longer be able to log in.

If user names and passwords are not based on ASCII or EBCDIC characters, then the affected user names and passwords will need to be reset upon the migration to a new character set.

To enhance security when using database authentication, Oracle recommends the use of password management, including account locking, password aging and expiration, password history, and password complexity verification.

> **See Also:** "Password Management Policy" on page 23-12

Creating a User Who is Authenticated by the Database

The following statement creates a user who is identified and authenticated by Oracle. User `scott` must specify the password `tiger` whenever connecting to Oracle.

```
CREATE USER scott IDENTIFIED BY tiger;
```

> **See Also:** *Oracle9i SQL Reference* for more information about valid passwords, and how to specify the `IDENTIFIED BY` clause in the `CREATE USER` and `ALTER USER` statements

Advantages of Database Authentication

Following are advantages of database authentication:

- 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.

External Authentication

When you choose external authentication for a user, the user account is maintained by Oracle, but password administration and user authentication is performed by an external service. This external service can be the operating system or a network service, such as Oracle Net.

With external authentication, your database relies on the underlying operating system or network authentication service to restrict access to database accounts. A database password is not used for this type of login. If your operating system or network service permits, you can have it authenticate users. If you do so, set the initialization parameter `OS_AUTHENT_PREFIX`, and use this prefix in Oracle user names. The `OS_AUTHENT_PREFIX` parameter defines a prefix that Oracle adds to the beginning of every user's operating system account name. Oracle compares the prefixed user name with the Oracle user names in the database when a user attempts to connect.

For example, assume that `OS_AUTHENT_PREFIX` is set as follows:

```
OS_AUTHENT_PREFIX=OPS$
```

> **Note:** The text of the `OS_AUTHENT_PREFIX` initialization parameter is case sensitive on some operating systems. See your operating system specific Oracle documentation for more information about this initialization parameter.

If a user with an operating system account named `tsmith` is to connect to an Oracle database and be authenticated by the operating system, Oracle checks that there is a corresponding database user `OPS$tsmith` and, if so, allows the user to connect. All references to a user authenticated by the operating system must include the prefix, as seen in `OPS$tsmith`.

The default value of this parameter is `OPS$` for backward compatibility with previous versions of Oracle. However, you might prefer to set the prefix value to some other string or a null string (an empty set of double quotes: ""). Using a null

string eliminates the addition of any prefix to operating system account names, so that Oracle user names exactly match operating system user names.

After you set OS_AUTHENT_PREFIX, it should remain the same for the life of a database. If you change the prefix, any database user name that includes the old prefix cannot be used to establish a connection, unless you alter the user name to have it use password authentication.

Creating a User Who is Authenticated Externally

The following statement creates a user who is identified by Oracle and authenticated by the operating system or a network service. This example assumes that OS_AUTHENT_PREFIX = "".

```
CREATE USER scott IDENTIFIED EXTERNALLY;
```

Using CREATE USER ... IDENTIFIED EXTERNALLY, you create database accounts that must be authenticated by the operating system or network service. Oracle relies on this external login authentication to ensure that a specific operating system user has access to a specific database user.

> **See Also:** *Oracle Advanced Security Administrator's Guide* for more information about external authentication

Operating System Authentication

By default, Oracle only allows operating system authenticated logins over secure connections. Therefore, if you want the operating system to authenticate a user, by default that user cannot connect to the database over Oracle Net. This means the user cannot connect using a shared server configuration, since this connection uses Oracle Net. This default restriction prevents a remote user from impersonating another operating system user over a network connection.

If you are not concerned about remote users impersonating another operating system user over a network connection, and you want to use operating system user authentication with network clients, set the initialization parameter REMOTE_OS_AUTHENT (default is FALSE) to TRUE in the database's initialization parameter file. Setting the initialization parameter REMOTE_OS_AUTHENT to TRUE allows the RDBMS to accept the client operating system user name received over a nonsecure connection and use it for account access. The change take effect the next time you start the instance and mount the database.

Generally, user authentication through the host operating system offers the following benefits:

- Users can connect to Oracle faster and more conveniently without specifying a separate database user name or password.
- User entries in the database and operating system audit trails correspond.

Network Authentication

Network authentication is performed using Oracle Advanced Security, which can be configured to use a third party service such as Kerberos. If you are using Oracle Advanced Security as your only external authentication service, the setting of the parameter `REMOTE_OS_AUTHENT` is irrelevant, since Oracle Advanced Security only allows secure connections.

Advantages of External Authentication

Following are advantages of external authentication:

- More choices of authentication mechanism are available, such as smart cards, fingerprints, Kerberos, or the operating system.
- Many network authentication services, such as Kerberos and DCE, support single sign-on. This means that users have fewer passwords to remember.
- If you are already using some external mechanism for authentication, such as one of those listed above, there may be less administrative overhead to use that mechanism with the database as well.

Global Authentication and Authorization

Oracle Advanced Security enables you to centralize management of user-related information, including authorizations, in an LDAP-based directory service. Users can be identified in the database as global users, meaning that they are authenticated by SSL and that the management of these users is done outside of the database by the centralized directory service. Global roles are defined in a database and are known only to that database, but authorizations for such roles is done by the directory service.

Note: You can also have users authenticated by SSL, whose authorizations are not managed in a directory; that is, they have local database roles only. See the *Oracle Advanced Security Administrator's Guide* for details.

This centralized management enables the creation of **enterprise users** and **enterprise roles**. Enterprise users are defined and managed in the directory. They have unique identities across the enterprise, and can be assigned enterprise roles that determine their access privileges across multiple databases. An enterprise role consists of one or more global roles, and might be thought of as a container for global roles.

Creating a User Who is Authorized by a Directory Service

You have a couple of options as to how you specify users who are authorized by a directory service.

Creating a Global User The following statement illustrates the creation of a global user, who is authenticated by SSL and authorized by the enterprise directory service:

```
CREATE USER scott
    IDENTIFIED GLOBALLY AS 'CN=scott,OU=division1,O=oracle,C=US';
```

The string provided in the AS clause provides an identifier (**distinguished name,** or **DN**) meaningful to the enterprise directory.

In this case, `scott` is truly a global user. But, the disadvantage here is that user `scott` must then be created in every database that he must access, plus the directory.

Creating a Schema-Independent User Creating schema-independent users allows multiple enterprise users to access a shared schema in the database. A schema-independent user is:

- Authenticated by SSL or passwords
- *Not* created in the database with a CREATE USER statement of any type
- A user whose privileges are managed in a directory
- A user who connects to a shared schema

The process of creating a schema-independent user is as follows:

1. Create a shared schema in the database as follows.

   ```
   CREATE USER appschema INDENTIFIED GLOBALLY AS '';
   ```

2. In the directory, you now create multiple enterprise users, and a mapping object.

The mapping object tells the database how you want to map users' DNs to the shared schema. You can either do a full DN mapping (one directory entry for each unique DN), or you can map, for example, every user containing the following DN components to the appschema:

```
OU=division,O=Oracle,C=US
```

See the *Oracle Internet Directory Administrator's Guide* for an explanation of these mappings.

Most users do not need their own schemas, and implementing schema-independent users divorces users from databases. You create multiple users who share the same schema in a database, and as enterprise users, they can access shared schemas in other databases as well.

Advantages of Global Authentication and Global Authorization

Some of the advantages of global user authentication and authorization are the following:

- Provides strong authentication using SSL or Windows NT native authentication
- Enables centralized management of users and privileges across the enterprise
- Is easy to administer—for every user you do not have to create a schema in every database in the enterprise
- Facilitates single sign-on—users only need to sign on once to access multiple databases and services. Further, users using passwords can have a single password to access databases accepting password authenticated enterprise users.
- Because it provides password based access, previously defined password authenticated database users can be migrated to the directory (using the User Migration Utility) to be centrally administered. This makes global authentication and authorization available for prior Oracle release clients that are still supported.
- CURRENT_USER database links connect as a global user. A local user can connect as a global user in the context of a stored procedure—without storing the global user's password in a link definition.

See Also: The following books contain additional information about global authentication and authorization, and enterprise users and roles:

- *Oracle Advanced Security Administrator's Guide*
- *Oracle Internet Directory Administrator's Guide*

Proxy Authentication and Authorization

It is possible to design a middle-tier server to proxy clients in a secure fashion.

Oracle provides three forms of proxy authentication:

- The middle-tier server authenticates itself with the database server and a client, in this case an application user or another application, authenticates itself with the middle-tier server. Client identities can be maintained all the way through to the database.
- The client, in this case a database user, is not authenticated by the middle-tier server. The clients identity and database password are passed through the middle-tier server to the database server for authentication.
- The client, in this case a global user, is authenticated by the middle-tier server, and passes one of the following through the middle tier for retrieving the client's user name.
  - Distinguished name (DN)
  - Certificate

In all cases, the middle-tier server must be authorized to act on behalf of the client by the administrator.

To authorize a middle-tier server to proxy a client use the `GRANT CONNECT THROUGH` clause of the `ALTER USER` statement. You can also specify roles that the middle tier is permitted to activate when connecting as the client.

Operations done on behalf of a client by a middle-tier server can be audited.

The `PROXY_USERS` data dictionary view can be queried to see which users are currently authorized to connect through a middle tier.

Use the `REVOKE CONNECT THROUGH` clause of `ALTER USER` to disallow a proxy connection.

See Also:

- *Oracle Call Interface Programmer's Guide* and *Oracle9i Application Developer's Guide - Fundamentals* for details about designing a middle-tier server to proxy users
- *Oracle9i SQL Reference* for a description and syntax of the proxy clause for `ALTER USER`
- "Auditing in a Multi-Tier Environment" on page 26-13 for details of auditing operations done on behalf of a user by a middle tier

Authorizing a Middle Tier to Proxy and Authenticate a User

The following statement authorizes the middle-tier server `appserve` to connect as user `bill`. It uses the `WITH ROLE` clause to specify that `appserve` activate all roles associated with `bill`, except `payroll`.

```
ALTER USER bill
    GRANT CONNECT THROUGH appserve
    WITH ROLE ALL EXCEPT payroll;
```

To revoke the middle-tier server's (`appserve`) authorization to connect as user `bill`, the following statement is used:

```
ALTER USER bill REVOKE CONNECT THROUGH appserve;
```

Authorizing a Middle Tier to Proxy a User Authenticated by Other Means

Use the `AUTHENTICATED USING` clause of the `ALTER USER ... GRANT CONNECT THROUGH` statement to authorize a user to be proxied, but not authenticated, by a middle tier. Currently, `PASSWORD` is the only means supported.

The following statement illustrates this form of authentication:

```
ALTER USER mary
    GRANT CONNECT THROUGH midtier
    AUTHENTICATED USING PASSWORD;
```

In the above statement, middle-tier server `midtier` is authorized to connect as `mary`, and `midtier` must also pass `mary`'s password to the database server for authorization.

Authorizing a Middle Tier to Proxy a User Identified by a Distinguished Name

In this case, the following statement authorizes the middle-tier server `WebDB` to present the distinguished name for global user `jeff` to the database server. The distinguished name is used to retrieve the user name. User `jeff` has been authenticated by the middle-tier server `WebDB`.

```
ALTER USER jeff
     GRANT CONNECT THROUGH WebDB
     AUTHENTICATED USING DISTINGUISHED NAME;
```

Optionally, the middle-tier server can be authorized to present an entire certificate (containing the distinguished name). This is illustrated in the following statement:

```
ALTER USER jeff
     GRANT CONNECT THROUGH WebDB
     AUTHENTICATED USING CERTIFICATE;
```

Passing the entire certificate costs time in authentication. However, some applications use other information contained in the certificate.

Managing Resources with Profiles

A profile is a named set of resource limits. A user's profile limits database usage and instance resources as defined in the profile. You can assign a profile to each user, and a default profile to all users who do not have specific profiles. For profiles to take effect, resource limits must be turned on for the database as a whole.

This section describes aspects of profile management, and contains the following topics:

- Enabling and Disabling Resource Limits
- Creating Profiles
- Assigning Profiles
- Altering Profiles
- Using Composite Limits
- Dropping Profiles

> **See Also:** *Oracle9i SQL Reference*.for more information about the SQL statements used for managing profiles

Enabling and Disabling Resource Limits

A profile can be created, assigned to users, altered, and dropped at any time by any authorized database user, but the resource limits set for a profile are enforced only when you enable resource limitation for the associated database. Resource limitation enforcement can be enabled or disabled by two different methods, as described in the next two sections.

To alter the enforcement of resource limitation while the database remains open, you must have the ALTER SYSTEM system privilege.

Enabling and Disabling Resource Limits Before Startup

If a database can be temporarily shut down, resource limitation can be enabled or disabled by the RESOURCE_LIMIT initialization parameter in the database's initialization parameter file. Valid values for the parameter are TRUE (enables enforcement) and FALSE. By default, this parameter's value is set to FALSE. Once the initialization parameter file has been edited, the database instance must be restarted to take effect. Every time an instance is started, the new parameter value enables or disables the enforcement of resource limitation.

Enabling and Disabling Resource Limits While the Database is Open

If a database cannot be temporarily shut down or the resource limitation feature must be altered temporarily, you can enable or disable the enforcement of resource limitation using the SQL statement ALTER SYSTEM. After an instance is started, an ALTER SYSTEM statement overrides the value set by the RESOURCE_LIMIT initialization parameter. For example, the following statement enables the enforcement of resource limitation for a database:

```
ALTER SYSTEM
   SET RESOURCE_LIMIT = TRUE;
```

Note: This does not apply to password parameters.

An ALTER SYSTEM statement does not permanently determine the enforcement of resource limitation. If the database is shut down and restarted, the enforcement of resource limits is determined by the value set for the RESOURCE_LIMIT parameter.

Creating Profiles

To create a profile, you must have the CREATE PROFILE system privilege. You can create profiles using the SQL statement CREATE PROFILE. At the same time, you can explicitly set particular resource limits.

The following statement creates the profile clerk:

```
CREATE PROFILE clerk LIMIT
    SESSIONS_PER_USER 2
    CPU_PER_SESSION unlimited
    CPU_PER_CALL 6000
    LOGICAL_READS_PER_SESSION unlimited
    LOGICAL_READS_PER_CALL 100
    IDLE_TIME 30
    CONNECT_TIME 480;
```

All unspecified resource limits for a new profile take the limit set by a DEFAULT profile.

Each database has a DEFAULT profile, and its limits are used in two cases:

- If a user is not explicitly assigned a profile, then the user conforms to *all* the limits of the DEFAULT profile.
- All unspecified limits of any profile use the corresponding limit of the DEFAULT profile.

Initially, all limits of the DEFAULT profile are set to UNLIMITED. However, to prevent unlimited resource consumption by users of the DEFAULT profile, the security administrator should change the default limits using the ALTER PROFILE statement:

```
ALTER PROFILE default LIMIT
   ...;
```

Any user with the ALTER PROFILE system privilege can adjust the limits in the DEFAULT profile. The DEFAULT profile cannot be dropped.

Assigning Profiles

After a profile has been created, you can assign it to database users. Each user can be assigned only one profile at any given time. If a profile is assigned to a user who already has a profile, the new profile assignment overrides the previously assigned profile. Profile assignments do not affect current sessions. Profiles can be assigned only to users and not to roles or other profiles.

Profiles can be assigned to users with the CREATE USER and ALTER USER statements.

See Also:

- "Creating Users" on page 24-2
- "Altering Users" on page 24-6

Altering Profiles

You can alter the resource limit settings of any profile using the SQL statement ALTER PROFILE. To alter a profile, you must have the ALTER PROFILE system privilege.

Any adjusted profile limit overrides the previous setting for that profile limit. By adjusting a limit with a value of DEFAULT, the resource limit reverts to the default limit set for the database. All profiles not adjusted when altering a profile retain the previous settings. Any changes to a profile do not affect current sessions. New profile settings are used only for sessions created after a profile is modified.

The following statement alters the clerk profile:

```
ALTER PROFILE clerk LIMIT
    CPU_PER_CALL default
    LOGICAL_READS_PER_SESSION 20000;
```

Using Composite Limits

In addition to being able to use the CREATE or ALTER PROFILE statements to assign resource limits to specific resources, you can limit the total resource cost for a session by using composite limits. A composite limit is expressed as a weighted sum, measured in **service units**, of certain resources.

You can set a profile's composite limit using the COMPOSITE_LIMIT clause of a CREATE PROFILE or ALTER PROFILE statement. The following CREATE PROFILE statement specifies the COMPOSITE_LIMIT clause:

```
CREATE PROFILE clerk LIMIT
    COMPOSITE_LIMIT 20000
    SESSIONS_PER_USER 2
    CPU_PER_CALL 1000;
```

Notice that both explicit resource limits and a composite limit can exist concurrently for a profile. The limit that is reached first stops the activity in a session. Composite limits allow additional flexibility when limiting the use of system resources.

Determining the Value of the Composite Limit

The correct composite limit depends on the total amount of resource used by an average profile user. As with each specific resource limit, historical information should be gathered to determine the normal range of composite resource usage for a typical profile user.

> **See Also:** *Oracle9i SQL Reference* for information on how to calculate the composite limit

Setting Resource Costs

Each Oracle database server environment has its own characteristics. Some system resources can be more valuable in one environment than another. Oracle enables you to assign the following resources a weight, which then affects their contribution to a total resource cost:

- CPU_PER_SESSION
- LOGICAL_READS_PER_SESSION
- CONNECT_TIME
- PRIVATE_SGA.

If you do not assign a weight to a resource, its weight defaults to 0, and the use of the resource does not contribute to the total resource cost.

Oracle calculates the total resource cost by first multiplying the amount of each resource used in the session by the resource's weight, and then summing the products for all four resources. For any session, this cost is limited by the value of the COMPOSITE_LIMIT parameter in the user's profile. Both the products and the total cost are expressed in units called service units.

To set weights for resources, use the ALTER RESOURCE COST statement.You must have the ALTER RESOURCE system privilege.The following example assigns weights to the CPU_PER_SESSION and LOGICAL_READS_PER_SESSION resources.

```
ALTER RESOURCE COST
    CPU_PER_SESSION 1
    LOGICAL_READS_PER_SESSION 50;
```

The weights establish this cost formula for a session:

```
cost = (1 * CPU_PER_SESSION) + (50 * LOGICAL_READS_PER_SESSION)
```

where the values of CPU_PER_SESSION and LOGICAL_READS_PER_SESSION are either values in the DEFAULT profile or in the profile of the user of the session.

Because the above statement assigns no weight to the resources CONNECT_TIME and PRIVATE_SGA, these resources do not appear in the formula.

See Also:

- *Oracle9i SQL Reference*
- Your operating system specific Oracle documentation

The above sources provide additional information and recommendations on setting resource costs

Dropping Profiles

To drop a profile, you must have the DROP PROFILE system privilege. You can drop a profile using the SQL statement DROP PROFILE. To successfully drop a profile currently assigned to a user, use the CASCADE option.

The following statement drops the profile clerk, even though it is assigned to a user:

```
DROP PROFILE clerk CASCADE;
```

Any user currently assigned to a profile that is dropped is automatically assigned to the DEFAULT profile. The DEFAULT profile cannot be dropped. When a profile is dropped, the drop does not affect currently active sessions. Only sessions created after a profile is dropped abide by any modified profile assignments.

Viewing Information About Database Users and Profiles

The following data dictionary views contain information about database users and profiles:

View	Description
DBA_USERS ALL_USERS USER_USERS	DBA view describes all users of the database. ALL view lists users visible to the current user, but does not describe them. USER view describes only the current user.
DBA_TS_QUOTAS USER_TS_QUOTAS	Describes tablespace quotas for users.

View	Description
USER_PASSWORD_LIMITS	Describes the password profile parameters that are assigned to the user.
USER_RESOURCE_LIMITS	Displays the resource limits for the current user.
DBA_PROFILES	Displays all profiles and their limits.
RESOURCE_COST	Lists the cost for each resource.
V$SESSION	Lists session information for each current session. Includes user name.
V$SESSTAT	Lists user session statistics.
V$STATNAME	Displays decoded statistic names for the statistics shown in the V$SESSTAT view.
PROXY_USERS	Describes users who can assume the identity of other users.

The following sections present some example of using these views, and assume a database in which the following statements have been executed:

```
CREATE PROFILE clerk LIMIT
    SESSIONS_PER_USER 1
    IDLE_TIME 30
    CONNECT_TIME 600;

CREATE USER jfee
    IDENTIFIED BY wildcat
    DEFAULT TABLESPACE users
    TEMPORARY TABLESPACE temp_ts
    QUOTA 500K ON users
    PROFILE clerk;

CREATE USER dcranney
    IDENTIFIED BY bedrock
    DEFAULT TABLESPACE users
    TEMPORARY TABLESPACE temp_ts
    QUOTA unlimited ON users;

CREATE USER userscott
     IDENTIFIED BY scott1;
```

See Also: *Oracle9i SQL Reference* for complete descriptions of the above data dictionary and dynamic performance views

Listing All Users and Associated Information

The following query lists users and their associated information as defined in the database:

```
SELECT USERNAME, PROFILE, ACCOUNT_STATUS FROM DBA_USERS;

USERNAME        PROFILE         ACCOUNT_STATUS
--------------- --------------- ---------------
SYS             DEFAULT         OPEN
SYSTEM          DEFAULT         OPEN
USERSCOTT       DEFAULT         OPEN
JFEE            CLERK           OPEN
DCRANNEY        DEFAULT         OPEN
```

All passwords are encrypted to preserve security. If a user queries the PASSWORD column, that user is not be able to determine another user's password.

Listing All Tablespace Quotas

The following query lists all tablespace quotas specifically assigned to each user:

```
SELECT * FROM DBA_TS_QUOTAS;

TABLESPACE    USERNAME    BYTES     MAX_BYTES    BLOCKS    MAX_BLOCKS
----------    ---------  --------   ----------   -------   ----------
USERS         JFEE              0       512000         0          250
USERS         DCRANNEY          0           -1         0           -1
```

When specific quotas are assigned, the exact number is indicated in the MAX_BYTES column. Note that this number is always a multiple of the database block size, so if you specify a tablespace quota that is not a multiple of the database block size, it is rounded up accordingly. Unlimited quotas are indicated by "-1".

Listing All Profiles and Assigned Limits

The following query lists all profiles in the database and associated settings for each limit in each profile:

```
SELECT * FROM DBA_PROFILES
   ORDER BY PROFILE;

PROFILE             RESOURCE_NAME              RESOURCE   LIMIT
-----------------   ---------------            ---------- --------------
CLERK               COMPOSITE_LIMIT            KERNEL     DEFAULT
```

```
CLERK                FAILED_LOGIN_ATTEMPTS      PASSWORD   DEFAULT
CLERK                PASSWORD_LIFE_TIME         PASSWORD   DEFAULT
CLERK                PASSWORD_REUSE_TIME        PASSWORD   DEFAULT
CLERK                PASSWORD_REUSE_MAX         PASSWORD   DEFAULT
CLERK                PASSWORD_VERIFY_FUNCTION   PASSWORD   DEFAULT
CLERK                PASSWORD_LOCK_TIME         PASSWORD   DEFAULT
CLERK                PASSWORD_GRACE_TIME        PASSWORD   DEFAULT
CLERK                PRIVATE_SGA                KERNEL     DEFAULT
CLERK                CONNECT_TIME               KERNEL     600
CLERK                IDLE_TIME                  KERNEL     30
CLERK                LOGICAL_READS_PER_CALL     KERNEL     DEFAULT
CLERK                LOGICAL_READS_PER_SESSION  KERNEL     DEFAULT
CLERK                CPU_PER_CALL               KERNEL     DEFAULT
CLERK                CPU_PER_SESSION            KERNEL     DEFAULT
CLERK                SESSIONS_PER_USER          KERNEL     1
DEFAULT              COMPOSITE_LIMIT            KERNEL     UNLIMITED
DEFAULT              PRIVATE_SGA                KERNEL     UNLIMITED
DEFAULT              SESSIONS_PER_USER          KERNEL     UNLIMITED
DEFAULT              CPU_PER_CALL               KERNEL     UNLIMITED
DEFAULT              LOGICAL_READS_PER_CALL     KERNEL     UNLIMITED
DEFAULT              CONNECT_TIME               KERNEL     UNLIMITED
DEFAULT              IDLE_TIME                  KERNEL     UNLIMITED
DEFAULT              LOGICAL_READS_PER_SESSION  KERNEL     UNLIMITED
DEFAULT              CPU_PER_SESSION            KERNEL     UNLIMITED
DEFAULT              FAILED_LOGIN_ATTEMPTS      PASSWORD   UNLIMITED
DEFAULT              PASSWORD_LIFE_TIME         PASSWORD   UNLIMITED
DEFAULT              PASSWORD_REUSE_MAX         PASSWORD   UNLIMITED
DEFAULT              PASSWORD_LOCK_TIME         PASSWORD   UNLIMITED
DEFAULT              PASSWORD_GRACE_TIME        PASSWORD   UNLIMITED
DEFAULT              PASSWORD_VERIFY_FUNCTION   PASSWORD   UNLIMITED
DEFAULT              PASSWORD_REUSE_TIME        PASSWORD   UNLIMITED
32 rows selected.
```

Viewing Memory Use for Each User Session

The following query lists all current sessions, showing the Oracle user and current UGA (user global area) memory use for each session:

```
SELECT USERNAME, VALUE || 'bytes' "Current UGA memory"
   FROM V$SESSION sess, V$SESSTAT stat, V$STATNAME name
WHERE sess.SID = stat.SID
   AND stat.STATISTIC# = name.STATISTIC#
   AND name.NAME = 'session uga memory';
```

```
USERNAME                       Current UGA memory
------------------------------ ------------------------------------------
                               18636bytes
                               17464bytes
                               19180bytes
                               18364bytes
                               39384bytes
                               35292bytes
                               17696bytes
                               15868bytes
USERSCOTT                      42244bytes
SYS                            98196bytes
SYSTEM                         30648bytes

11 rows selected.
```

To see the maximum UGA memory ever allocated to each session since the instance started, replace `'session uga memory'` in the query above with `'session uga memory max'`.

25

Managing User Privileges and Roles

This chapter explains how to use privileges and roles to control access to schema objects and to control the ability to execute system operations. The following topics are discussed:

- Understanding User Privileges and Roles
- Managing User Roles
- Granting User Privileges and Roles
- Revoking User Privileges and Roles
- Granting to and Revoking from the User Group PUBLIC
- When Do Grants and Revokes Take Effect?
- Granting Roles Using the Operating System or Network
- Viewing Privilege and Role Information

See Also:

- Chapter 24, "Managing Users and Resources" for information about controlling access to a database
- Chapter 23, "Establishing Security Policies" for suggested general database security policies

Understanding User Privileges and Roles

A user **privilege** is a right to execute a particular type of SQL statement, or a right to access another user's object. The types of privileges are defined by Oracle.

Roles, on the other hand, are created by users (usually administrators) and are used to group together privileges or other roles. They are a means of facilitating the granting of multiple privileges or roles to users.

This section describes Oracle user privileges, and contains the following topics:

- System Privileges
- Object Privileges
- User Roles

> **See Also:** *Oracle9i Database Concepts* for additional information about privileges and roles

System Privileges

There are over 100 distinct system privileges. Each system privilege allows a user to perform a particular database operation or class of database operations.

Caution: System privileges can be very powerful, and should be granted only when necessary to roles and trusted users of the database.

> **See Also:** *Oracle9i SQL Reference.* for the complete list of system privileges and their descriptions

Restricting System Privileges

Because system privileges are so powerful, Oracle recommends that you configure your database to prevent regular (non-DBA) users exercising ANY system privileges (such as `UPDATE ANY TABLE`) on the data dictionary. In order to secure the data dictionary, ensure that the `O7_DICTIONARY_ACCESSIBILITY` initialization parameter is set to FALSE. This feature is called the dictionary protection mechanism.

> **Note:** The O7_DICTIONARY_ACCESSIBILITY initialization parameter controls restrictions on system privileges when you upgrade from Oracle7 to Oracle8*i* and higher releases. If the parameter is set to TRUE, access to objects in the SYS schema is allowed (Oracle7 behavior). If this parameter is set to FALSE, system privileges that allow access to objects in "any schema" do not allow access to objects in SYS schema. The default for O7_DICTIONARY_ACCESSIBILITY is FALSE.
>
> When this parameter is not set to FALSE, the ANY privilege applies to the data dictionary, and a malicious user with ANY privilege could access or alter data dictionary tables.
>
> See the *Oracle9i Database Reference* for more information on the O7_DICTIONARY_ACCESSIBILITY initialization parameter and *Oracle9i Database Migration* to understand its usage.

If you enable dictionary protection (O7_DICTIONARY_ACCESSIBILITY is FALSE), access to objects in the SYS schema (dictionary objects) is restricted to users with the SYS schema. These users are SYS and those who connect as SYSDBA. System privileges providing access to objects in other schemas do *not* give other users access to objects in the SYS schema. For example, the SELECT ANY TABLE privilege allows users to access views and tables in other schemas, but does not enable them to select dictionary objects (base tables of dynamic performance views, views, packages, and synonyms). These users can, however, be granted explicit object privileges to access objects in the SYS schema.

Accessing Objects in the SYS Schema

Users with explicit object privileges or those who connect with administrative privileges (SYSDBA) can access objects in the SYS schema. Another means of allowing access to objects in the SYS schema is by granting users any of the following roles:

- SELECT_CATALOG_ROLE

  This role can be granted to users to allow SELECT privileges on all data dictionary views.

- EXECUTE_CATALOG_ROLE

This role can be granted to users to allow EXECUTE privileges for packages and procedures in the data dictionary.

- DELETE_CATALOG_ROLE

  This role can be granted to users to allow them to delete records from the system audit table (AUD$).

Additionally, the following system privilege can be granted to users who require access to tables created in the SYS schema:

- SELECT ANY DICTIONARY

  This system privilege allows query access to any object in the SYS schema, including tables created in that schema. It must be granted individually to each user requiring the privilege. It is not included in GRANT ALL PRIVILEGES, nor can it be granted through a role.

Caution: You should grant these roles and the SELECT ANY DICTIONARY system privilege with extreme care, since the integrity of your system can be compromised by their misuse.

Object Privileges

Each type of object has different privileges associated with it.

You can specify ALL [PRIVILEGES] to grant or revoke all available object privileges for an object. ALL is not a privilege; rather, it is a shortcut, or a way of granting or revoking all object privileges with one word in GRANT and REVOKE statements. Note that if all object privileges are granted using the ALL shortcut, individual privileges can still be revoked.

Likewise, all individually granted privileges can be revoked by specifying ALL. However, if you REVOKE ALL, and revoking causes integrity constraints to be deleted (because they depend on a REFERENCES privilege that you are revoking), you must include the CASCADE CONSTRAINTS option in the REVOKE statement.

> **See Also:** *Oracle9i SQL Reference.* for the complete list of object privileges

User Roles

A **role** groups several privileges and roles, so that they can be granted to and revoked from users simultaneously. A role must be enabled for a user before it can be used by the user.

Oracle provides some predefined roles to help in database administration. These roles, listed in Table 25–1, are automatically defined for Oracle databases when you run the standard scripts that are part of database creation. You can grant privileges and roles to, and revoke privileges and roles from, these predefined roles in the same way as you do with any role you define.

Table 25–1 Predefined Roles **(Page 1 of 2)**

Role Name	Created By (Script)	Description
CONNECT	SQL.BSQ	Includes the following system privileges: ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW
RESOURCE	SQL.BSQ	Includes the following system privileges: CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
DBA	SQL.BSQ	All system privileges WITH ADMIN OPTION
Note: The previous three roles are provided to maintain compatibility with previous versions of Oracle and may not be created automatically in future versions of Oracle. Oracle Corporation recommends that you design your own roles for database security, rather than relying on these roles.		
EXP_FULL_DATABASE	CATEXP.SQL	Provides the privileges required to perform full and incremental database exports. Includes: SELECT ANY TABLE, BACKUP ANY TABLE, EXECUTE ANY PROCEDURE, EXECUTE ANY TYPE, ADMINISTER RESOURCE MANAGER, and INSERT, DELETE, and UPDATE on the tables SYS.INCVID, SYS.INCFIL, and SYS.INCEXP. Also the following roles: EXECUTE_CATALOG_ROLE and SELECT_CATALOG_ROLE.
IMP_FULL_DATABASE	CATEXP.SQL	Provides the privileges required to perform full database imports. Includes an extensive list of system privileges (use view DBA_SYS_PRIVS to view privileges) and the following roles: EXECUTE_CATALOG_ROLE and SELECT_CATALOG_ROLE.

Table 25–1 Predefined Roles **(Page 2 of 2)**

Role Name	Created By (Script)	Description
DELETE_CATALOG_ROLE	SQL.BSQ	Provides DELETE privilege on the system audit table (AUD$)
EXECUTE_CATALOG_ROLE	SQL.BSQ	Provides EXECUTE privilege on objects in the data dictionary. Also, HS_ADMIN_ROLE.
SELECT_CATALOG_ROLE	SQL.BSQ	Provides SELECT privilege on objects in the data dictionary. Also, HS_ADMIN_ROLE.
RECOVERY_CATALOG_OWNER	CATALOG.SQL	Provides privileges for owner of the recovery catalog. Includes: CREATE SESSION, ALTER SESSION, CREATE SYNONYM, CREATE VIEW, CREATE DATABASE LINK, CREATE TABLE, CREATE CLUSTER, CREATE SEQUENCE, CREATE TRIGGER, and CREATE PROCEDURE
HS_ADMIN_ROLE	CATHS.SQL	Used to protect access to the HS (Heterogeneous Services) data dictionary tables (grants SELECT) and packages (grants EXECUTE). It is granted to SELECT_CATALOG_ROLE and EXECUTE_CATALOG_ROLE such that users with generic data dictionary access also can access the HS data dictionary.
AQ_USER_ROLE	CATQUEUE.SQL	Obsoleted, but kept mainly for release 8.0 compatibility. Provides execute privilege on DBMS_AQ and DBMS_AQIN.
AQ_ADMINISTRATOR_ROLE	CATQUEUE.SQL	Provides privileges to administer Advance Queuing. Includes ENQUEUE ANY QUEUE, DEQUEUE ANY QUEUE, and MANAGE ANY QUEUE, SELECT privileges on AQ tables and EXECUTE privileges on AQ packages.
SNMPAGENT	CATSNMP.SQL	This role is used by Enterprise Manager/Intelligent Agent. Includes ANALYZE ANY and grants SELECT on various views.

If you install other options or products, other predefined roles may be created.

Managing User Roles

This section describes aspects of managing roles, and contains the following topics:

- Creating a Role
- Specifying the Type of Role Authorization

- Dropping Roles

Creating a Role

You can create a role using the CREATE ROLE statement, but you must have the CREATE ROLE system privilege to do so. Typically, only security administrators have this system privilege.

> **Note:** Immediately after creation, a role has no privileges associated with it. To associate privileges with a new role, you must grant privileges or other roles to the new role.

You must give each role you create a unique name among existing usernames and role names of the database. Roles are not contained in the schema of any user. In a database that uses a multibyte character set, Oracle recommends that each role name contain at least one single-byte character. If a role name contains only multibyte characters, the encrypted role name / password combination is considerably less secure.

The following statement creates the clerk role, which is authorized by the database using the password bicentennial:

```
CREATE ROLE clerk IDENTIFIED BY bicentennial;
```

The IDENTIFIED BY clause specifies how the user must be authorized before the role can be enabled for use by a specific user to which it has been granted. If this clause is not specified, or NOT IDENTIFIED is specified, then no authorization is required when the role is enabled. Roles can be specified to be authorized by:

- The database using a password
- An application using a specified package
- Externally by the operating system, network, or other external source
- Globally by an enterprise directory service

These authorizations are discussed in following sections.

Later, you can set or change the authorization method for a role using the ALTER ROLE statement. The following statement alters the clerk role to specify that the user must have been authorized by an external source before enabling the role:

```
ALTER ROLE clerk IDENTIFIED EXTERNALLY;
```

To alter the authorization method for a role, you must have the `ALTER ANY ROLE` system privilege or have been granted the role with the `ADMIN OPTION`.

> **See Also:** *Oracle9i SQL Reference* for syntax, restrictions, and authorization information about the SQL statements used to manage roles and privileges

Specifying the Type of Role Authorization

The methods of authorizing roles are presented in this section. A role must be enabled for you to use it.

> **See Also:** "When Do Grants and Revokes Take Effect?" on page 25-20 for a discussion about enabling roles

Role Authorization by the Database

The use of a role authorized by the database can be protected by an associated password. If you are granted a role protected by a password, you can enable or disable the role by supplying the proper password for the role in a `SET ROLE` statement. However, if the role is made a default role and enabled at connect time, the user is not required to enter a password.

The following statement creates a role `manager`. When it is enabled, the password `morework` must be supplied.

```
CREATE ROLE manager IDENTIFIED BY morework;
```

> **Note:** In a database that uses a multibyte character set, passwords for roles must include only singlebyte characters. Multibyte characters are not accepted in passwords. See the *Oracle9i SQL Reference* for information about specifying valid passwords.

Role Authorization by an Application

The `INDENTIFIED USING` *`package_name`* clause lets you create an application role, which is a role that can be enabled only by applications using an authorized package. Application developers do not need to secure a role by embedding passwords inside applications. Instead, they can create an application role and specify which PL/SQL package is authorized to enable the role.

The following example indicates that the role admin_role is an application role and the role can only be enabled by any module defined inside the PL/SQL package hr.admin.

```
CREATE ROLE admin_role IDENTIFIED USING hr.admin;
```

When enabling the user's default roles at login as specified in the user's profile, no checking is performed for application roles.

Role Authorization by an External Source

The following statement creates a role named accts_rec and requires that the user be authorized by an external source before it can be enabled:

```
CREATE ROLE accts_rec IDENTIFIED EXTERNALLY;
```

Role Authorization by the Operating System Role authentication through the operating system is useful only when the operating system is able to dynamically link operating system privileges with applications. When a user starts an application, the operating system grants an operating system privilege to the user. The granted operating system privilege corresponds to the role associated with the application. At this point, the application can enable the application role. When the application is terminated, the previously granted operating system privilege is revoked from the user's operating system account.

If a role is authorized by the operating system, you must configure information for each user at the operating system level. This operation is operating system dependent.

If roles are granted by the operating system, you do not need to have the operating system authorize them also; this is redundant.

> **See Also:** "Granting Roles Using the Operating System or Network" on page 25-22 for more information about roles granted by the operating system

Role Authorization and Network Clients If users connect to the database over Oracle Net, by default their roles cannot be authenticated by the operating system. This includes connections through a shared server configuration, as this connection requires Oracle Net. This restriction is the default because a remote user could impersonate another operating system user over a network connection.

If you are not concerned with this security risk and want to use operating system role authentication for network clients, set the initialization parameter REMOTE_OS_

ROLES in the database's initialization parameter file to TRUE. The change will take effect the next time you start the instance and mount the database. The parameter is FALSE by default.

Role Authorization by an Enterprise Directory Service

A role can be defined as a global role, whereby a (global) user can only be authorized to use the role by an enterprise directory service. You define the global role locally in the database by granting privileges and roles to it, but you cannot grant the global role itself to any user or other role in the database. When a global user attempts to connect to the database, the enterprise directory is queried to obtain any global roles associated with the user.

The following statement creates a global role:

```
CREATE ROLE supervisor IDENTIFIED GLOBALLY;
```

Global roles are one component of enterprise user management. A global role only applies to one database, but it can be granted to an enterprise role defined in the enterprise directory. An enterprise role is a directory structure which contains global roles on multiple databases, and which can be granted to enterprise users.

A general discussion of global authentication and authorization of users, and its role in enterprise user management, was presented earlier in "Global Authentication and Authorization" on page 24-13.

> **See Also:** *Oracle Advanced Security Administrator's Guide* and *Oracle Internet Directory Administrator's Guide* for information about enterprise user management and how to implement it

Dropping Roles

In some cases, it may be appropriate to drop a role from the database. The security domains of all users and roles granted a dropped role are immediately changed to reflect the absence of the dropped role's privileges. All indirectly granted roles of the dropped role are also removed from affected security domains. Dropping a role automatically removes the role from all users' default role lists.

Because the creation of objects is not dependent on the privileges received through a role, tables and other objects are not dropped when a role is dropped.

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

The following statement drops the role CLERK:

```
DROP ROLE clerk;
```

Granting User Privileges and Roles

This section describes the granting of privileges and roles, and contains the following topics:

- Granting System Privileges and Roles
- Granting Object Privileges
- Granting Privileges on Columns

It is also possible to grant roles to a user connected through a middle tier or proxy. This is discussed in "Proxy Authentication and Authorization" on page 24-16.

Granting System Privileges and Roles

You can grant system privileges and roles to other users and roles using the GRANT statement. The following privileges are required:

- To grant a system privilege, you must have been granted the system privilege with the ADMIN OPTION or have been granted the GRANT ANY PRIVILEGE system privilege.
- To grant a role, you must have been granted the role with the ADMIN OPTION or have been granted the GRANT ANY ROLE system privilege.

> **Note:** You cannot grant a roll that is IDENTIFIED GLOBALLY to anything. The granting (and revoking) of global roles is controlled entirely by the enterprise directory service.

The following statement grants the system privilege CREATE SESSION and the accts_pay role to the user jward:

```
GRANT CREATE SESSION, accts_pay TO jward;
```

Note: Object privileges *cannot* be granted along with system privileges and roles in the same GRANT statement.

Granting the ADMIN OPTION

A user or role that is granted a privilege or role specifying the WITH ADMIN OPTION clause has several expanded capabilities:

- The grantee can grant or revoke the system privilege or role to or from *any* user or other role in the database. Users cannot revoke a role from themselves.
- The grantee can further grant the system privilege or role with the ADMIN OPTION.
- The grantee of a role can alter or drop the role.

In the following statement, the security administrator grants the new_dba role to michael:

```
GRANT new_dba TO michael WITH ADMIN OPTION;
```

The user michael cannot only use all of the privileges implicit in the new_dba role, but can grant, revoke, or drop the new_dba role as deemed necessary. Because of these powerful capabilities, exercise caution when granting system privileges or roles with the ADMIN OPTION. Such privileges are usually reserved for a security administrator and rarely granted to other administrators or users of the system.

When a user creates a role, the role is automatically granted to the creator with the ADMIN OPTION

Creating a New User with the GRANT Statement

Oracle allows you to create a new user with the GRANT statement. If you specify a password using the IDENTIFIED BY clause, and the username/password does not exist in the database, a new user with that username and password is created. The following example creates ssmith as a new user while granting ssmith the CONNECT system privilege:

```
GRANT CONNECT TO ssmith IDENTIFIED BY p1q2r3;
```

See Also: "Creating Users" on page 24-2

Granting Object Privileges

You also use the GRANT statement to grant object privileges to roles and users. To grant an object privilege, you must fulfill one of the following conditions:

- You own the object specified.

- You possess the GRANT ANY OBJECT PRIVILEGE system privilege that enables you to grant and revoke privileges on behalf of the object owner.
- The WITH GRANT OPTION clause was specified when you were granted the object privilege by its owner.

> **Note:** System privileges and roles cannot be granted along with object privileges in the same GRANT statement.

The following statement grants the SELECT, INSERT, and DELETE object privileges for all columns of the emp table to the users jfee and tsmith:

```
GRANT SELECT, INSERT, DELETE ON emp TO jfee, tsmith;
```

To grant all object privileges on the salary view to the user jfee, use the ALL keyword, as shown in the following example:

```
GRANT ALL ON salary TO jfee;
```

Specifying the GRANT OPTION

Specify WITH GRANT OPTION to enable the grantee to grant the object privileges to other users and roles. The user whose schema contains an object is automatically granted all associated object privileges with the GRANT OPTION. This special privilege allows the grantee several expanded privileges:

- The grantee can grant the object privilege to any users in the database, with or without the GRANT OPTION, or to any role in the database.
- If both of the following are true, the grantee can create views on the table and grant the corresponding privileges on the views to any user or role in the database.
  - The grantee receives object privileges for the table with the GRANT OPTION.
  - The grantee has the CREATE VIEW or CREATE ANY VIEW system privilege.

The GRANT OPTION is not valid when granting an object privilege to a role. Oracle prevents the propagation of object privileges through roles so that grantees of a role cannot propagate object privileges received by means of roles.

Granting Object Privileges on Behalf of the Object Owner

The GRANT ANY OBJECT PRIVILEGE system privilege allows users to grant and revoke any object privilege on behalf of the object owner. This provides a convenient means for database and application administrators to grant access to objects in any schema without requiring that they connect to the schema. This eliminates the need to maintain login credentials for schema owners so that they can grant access to objects, and it reduces the number of connections required during configuration.

This system privilege is part of the Oracle supplied DBA role and is thus granted (with the ADMIN OPTION) to any user connecting AS SYSDBA (user SYS). As with other system privileges, the GRANT ANY OBJECT PRIVILEGE system privilege can only be granted by a user who possesses the ADMIN OPTION.

When you exercise the GRANT ANY OBJECT PRIVILEGE system privilege to grant an object privilege to a user, if you already possess the object privilege with the GRANT OPTION, then the grant is performed in the usual way. In this case, you become the grantor of the grant. If you do not possess the object privilege, then the object owner is shown as the grantor, even though you, with the GRANT ANY OBJECTPRIVILEGE system privilege, actually performed the grant.

> **Note:** The audit record generated by the GRANT statement will always show the real user who performed the grant.

For example, consider the following. User adams possesses the GRANT ANY OBJECT PRIVILEGE system privilege. He does not possess any other grant privileges. He issues the following statement:

```
GRANT SELECT ON hr.employees TO blake WITH GRANT OPTION;
```

If you examine the DBA_TAB_PRIVS view, you will see that hr is shown as being the grantor of the privilege:

```
SQL> SELECT GRANTEE, OWNER, GRANTOR, PRIVILEGE, GRANTABLE
  2>     FROM DBA_TAB_PRIVS
  3>     WHERE TABLE_NAME = 'EMPLOYEES' and OWNER = 'HR';

GRANTEE  OWNER GRANTOR PRIVILEGE    GRANTABLE
-------- ----- ------- -----------  ----------
BLAKE    HR    HR       SELECT      YES
```

Now assume that blake also has the GRANT ANY OBJECT PRIVILEGE system. He, issues the following statement:

```
GRANT SELECT ON hr.employees TO clark;
```

In this case, when you again query the DBA_TAB_PRIVS view, you see that blake is shown as being the grantor of the privilege:

```
GRANTEE  OWNER GRANTOR  PRIVILEGE    GRANTABLE
-------- ----- -------- --------     ----------
BLAKE    HR    HR       SELECT       YES
CLARK    HR    BLAKE    SELECT       NO
```

This is because blake already possesses the SELECT privilege on hr.employees with the GRANT OPTION.

> **See Also:** "Revoking Object Privileges on Behalf of the Object Owner" on page 25-17

Granting Privileges on Columns

You can grant INSERT, UPDATE, or REFERENCES privileges on individual columns in a table.

> **Caution:** Before granting a column-specific INSERT privilege, determine if the table contains any columns on which NOT NULL constraints are defined. Granting selective insert capability without including the NOT NULL columns prevents the user from inserting any rows into the table. To avoid this situation, make sure that each NOT NULL column is either insertable or has a non-NULL default value. Otherwise, the grantee will not be able to insert rows into the table and will receive an error.

The following statement grants INSERT privilege on the acct_no column of the accounts table to scott:

```
GRANT INSERT (acct_no) ON accounts TO scott;
```

In another example, object privilege for the ename and job columns of the emp table are granter to the users jfee and tsmith:

```
GRANT INSERT(ename, job) ON emp TO jfee, tsmith;
```

Revoking User Privileges and Roles

This section describes aspects of revoking user privileges and roles, and contains the following topics:

- Revoking System Privileges and Roles
- Revoking Object Privileges
- Cascading Effects of Revoking Privileges

Revoking System Privileges and Roles

You can revoke system privileges and roles using the SQL statement REVOKE.

Any user with the ADMIN OPTION for a system privilege or role can revoke the privilege or role from any other database user or role. The revoker does not have to be the user that originally granted the privilege or role. Users with GRANT ANY ROLE can revoke *any* role.

The following statement revokes the CREATE TABLE system privilege and the accts_rec role from tsmith:

```
REVOKE CREATE TABLE, accts_rec FROM tsmith;
```

> **Note:** The ADMIN OPTION for a system privilege or role cannot be selectively revoked. The privilege or role must be revoked and then the privilege or role re-granted without the ADMIN OPTION.

Revoking Object Privileges

The REVOKE statement is used to revoke object privileges. To revoke an object privilege, you must fulfill one of the following conditions:

- You previously granted the object privilege to the user or role.
- You possess the GRANT ANY OBJECT PRIVILEGE system privilege that enables you to grant and revoke privileges on behalf of the object owner.

You can only revoke the privileges that you, the grantor, directly authorized, not the grants made by other users to whom you granted the GRANT OPTION. However, there is a cascading effect. The object privilege grants propagated using the GRANT OPTION are revoked if a grantor's object privilege is revoked.

Assuming you are the original grantor, the following statement revokes the SELECT and INSERT privileges on the emp table from the users jfee and tsmith:

```
REVOKE SELECT, insert ON emp FROM jfee, tsmith;
```

The following statement revokes all object privileges for the dept table that you originally granted to the human_resource role

```
REVOKE ALL ON dept FROM human_resources;
```

> **Note:** The GRANT OPTION for an object privilege cannot be selectively revoked. The object privilege must be revoked and then re-granted without the GRANT OPTION. Users cannot revoke object privileges from themselves.

Revoking Object Privileges on Behalf of the Object Owner

The GRANT ANY OBJECT PRIVILEGE system privilege allows you to revoke any specified object privilege where the object owner is the grantor. This occurs when the object privilege is granted by the object owner, or on behalf of the owner by any user holding the GRANT ANY OBJECT PRIVILEGE system privilege.

In a situation where the object privilege has been granted by both the owner of the object and the user executing the REVOKE statement (who has both the specific object privilege and the GRANT ANY OBJECT PRIVILEGE system privilege), Oracle only revokes the object privilege granted by the user issuing the REVOKE. This can be illustrated by continuing the example started in "Granting Object Privileges on Behalf of the Object Owner" on page 25-14.

At this point, blake has granted the SELECT privilege on hr.employees to clark. Even though blake possesses the GRANT ANY OBJECT PRIVILEGE system privilege, he also holds the specific object privilege, thus this grant is attributed to him. Assume that hr also grants the SELECT privilege on hr.employees to clark. A query of the DBA_TAB_PRIVS view shows that the following grants are in effect for the hr.employees table:

```
GRANTEE  OWNER GRANTOR PRIVILEGE    GRANTABLE
-------- ----- ------- -----------  ----------
BLAKE    HR    HR      SELECT       YES
CLARK    HR    BLAKE   SELECT       NO
CLARK    HR    HR      SELECT       NO
```

User blake now issues the following REVOKE statement:

```
REVOKE  SELECT ON hr.employees FROM clark;
```

Only the object privilege for clark granted by blake is removed. The grant by the object owner, hr, remains.

```
GRANTEE  OWNER GRANTOR PRIVILEGE    GRANTABLE
-------- ----- ------- -----------  ----------
BLAKE    HR    HR      SELECT       YES
CLARK    HR    HR      SELECT       NO
```

If blake issues the REVOKE statement again, this time the effect will be to remove the object privilege granted by hr.

> **See Also:** "Granting Object Privileges on Behalf of the Object Owner" on page 25-14

Revoking Column-Selective Object Privileges

Although users can grant column-selective INSERT, UPDATE, and REFERENCES privileges for tables and views, they cannot selectively revoke column specific privileges with a similar REVOKE statement. Instead, the grantor must first revoke the object privilege for all columns of a table or view, and then selectively re-grant the column-specific privileges that should remain.

For example, assume that role human_resources has been granted the UPDATE privilege on the deptno and dname columns of the table dept. To revoke the UPDATE privilege on just the deptno column, issue the following two statements:

```
REVOKE UPDATE ON dept FROM human_resources;
GRANT UPDATE (dname) ON dept TO human_resources;
```

The REVOKE statement revokes UPDATE privilege on all columns of the dept table from the role human_resources. The GRANT statement re-grants UPDATE privilege on the dname column to the role human_resources.

Revoking the REFERENCES Object Privilege

If the grantee of the REFERENCES object privilege has used the privilege to create a foreign key constraint (that currently exists), the grantor can revoke the privilege only by specifying the CASCADE CONSTRAINTS option in the REVOKE statement:

```
REVOKE REFERENCES ON dept FROM jward CASCADE CONSTRAINTS;
```

Any foreign key constraints currently defined that use the revoked REFERENCES privilege are dropped when the CASCADE CONSTRAINTS clause is specified.

Cascading Effects of Revoking Privileges

Depending on the type of privilege, there may be cascading effects when a privilege is revoked.

System Privileges

There are no cascading effects when revoking a system privilege related to DDL operations, regardless of whether the privilege was granted with or without the ADMIN OPTION. For example, assume the following:

1. The security administrator grants the CREATE TABLE system privilege to jfee with the ADMIN OPTION.
2. User jfee creates a table.
3. User jfee grants the CREATE TABLE system privilege to tsmith.
4. User tsmith creates a table.
5. The security administrator revokes the CREATE TABLE system privilege from jfee.
6. User jfee's table continues to exist. tsmith still has the table and the CREATE TABLE system privilege.

Cascading effects can be observed when revoking a system privilege related to a DML operation. If SELECT ANY TABLE is revoked from a user, then all procedures contained in the users schema relying on this privilege will fail until the privilege is reauthorized.

Object Privileges

Revoking an object privilege can have cascading effects that should be investigated before issuing a REVOKE statement.

- Object definitions that depend on a DML object privilege can be affected if the DML object privilege is revoked. For example, assume the procedure body of the test procedure includes a SQL statement that queries data from the emp table. If the SELECT privilege on the emp table is revoked from the owner of the test procedure, the procedure can no longer be executed successfully.
- When a REFERENCES privilege for a table is revoked from a user, any foreign key integrity constraints defined by the user that require the dropped REFERENCES privilege are automatically dropped. For example, assume that the user jward is granted the REFERENCES privilege for the deptno column of the dept table and creates a foreign key on the deptno column in the emp table

that references the deptno column. If the references privilege on the deptno column of the dept table is revoked, the foreign key constraint on the deptno column of the emp table is dropped in the same operation.

- The object privilege grants propagated using the GRANT OPTION are revoked if a grantor's object privilege is revoked. For example, assume that user1 is granted the SELECT object privilege with the GRANT OPTION, and grants the SELECT privilege on emp to user2. Subsequently, the SELECT privilege is revoked from user1. This REVOKE is cascaded to user2 as well. Any objects that depended on user1's and user2's revoked SELECT privilege can also be affected, as described in previous bullet items.

Object definitions that require the ALTER and INDEX DDL object privileges are not affected if the ALTER or INDEX object privilege is revoked. For example, if the INDEX privilege is revoked from a user that created an index on someone else's table, the index continues to exist after the privilege is revoked.

Granting to and Revoking from the User Group PUBLIC

Privileges and roles can also be granted to and revoked from the user group PUBLIC. Because PUBLIC is accessible to every database user, all privileges and roles granted to PUBLIC are accessible to every database user.

Security administrators and database users should grant a privilege or role to PUBLIC only if every database user requires the privilege or role. This recommendation reinforces the general rule that at any given time, each database user should only have the privileges required to accomplish the group's current tasks successfully.

Revoking a privilege from PUBLIC can cause significant cascading effects. If any privilege related to a DML operation is revoked from PUBLIC (for example, SELECT ANY TABLE, UPDATE ON emp), all procedures in the database, including functions and packages, must be *reauthorized* before they can be used again. Therefore, exercise caution when granting and revoking DML-related privileges to PUBLIC.

> **See Also:** "Managing Object Dependencies" on page 21-23 for more information about object dependencies

When Do Grants and Revokes Take Effect?

Depending on what is granted or revoked, a grant or revoke takes effect at different times:

- All grants / revokes of system and object privileges to anything (users, roles, and PUBLIC) are immediately observed.
- All grants / revokes of roles to anything (users, other roles, PUBLIC) are only observed when a current user session issues a SET ROLE statement to re-enable the role after the grant / revoke, or when a new user session is created after the grant / revoke.

You can see which roles are currently enabled by examining the SESSION_ROLES data dictionary view.

The SET ROLE Statement

During the session, the user or an application can use the SET ROLE statement any number of times to change the roles currently enabled for the session. You must already have been granted the roles that you name in the SET ROLE statement. The number of roles that can be concurrently enabled is limited by the initialization parameter MAX_ENABLED_ROLES.

This example enables the role clerk, which you have already been granted, and specifies the password.

```
SET ROLE clerk IDENTIFIED BY bicentennial;
```

You can disable all roles with the following statement:

```
SET ROLE NONE;
```

Specifying Default Roles

When a user logs on, Oracle enables all privileges granted explicitly to the user and all privileges in the user's default roles.

A user's list of default roles can be set and altered using the ALTER USER statement. The ALTER USER statement allows you to specify roles that are to be enabled when a user connects to the database, without requiring the user to specify the roles' passwords. The user must have already been directly granted the roles with a GRANT statement. You cannot specify as a default role any role managed by an external service including a directory service (external roles or global roles).

The following example establishes default roles for user jane:

```
ALTER USER jane DEFAULT ROLE payclerk, pettycash;
```

You cannot set a user's default roles in the CREATE USER statement. When you first create a user, the user's default role setting is ALL, which causes all roles

subsequently granted to the user to be default roles. Use the ALTER USER statement to limit the user's default roles.

> **Caution:** When you create a role (other than a user role), it is granted to you implicitly and added as a default role. You receive an error at login if you have more than MAX_ENABLED_ROLES. You can avoid this error by altering the user's default roles to be less than MAX_ENABLED_ROLES. Thus, you should change the DEFAULT ROLE settings of SYS and SYSTEM before creating user roles.

Restricting the Number of Roles that a User Can Enable

A user can enable as many roles as specified by the initialization parameter MAX_ENABLED_ROLES. All indirectly granted roles enabled as a result of enabling a primary role are included in this count. The database administrator can alter this limitation by modifying the value for this parameter. Higher values permit each user session to have more concurrently enabled roles. However, the larger the value for this parameter, the more memory space is required on behalf of each user session; this is because the PGA size is affected for each user session, and requires four bytes for each role. Determine the highest number of roles that will be concurrently enabled by any one user and use this value for the MAX_ENABLED_ROLES parameter.

Granting Roles Using the Operating System or Network

This section describes aspects of granting roles through your operating system or network, and contains the following topics:

- Using Operating System Role Identification
- Using Operating System Role Management
- Granting and Revoking Roles When OS_ROLES=TRUE
- Enabling and Disabling Roles When OS_ROLES=TRUE
- Using Network Connections with Operating System Role Management

Instead of a security administrator explicitly granting and revoking database roles to and from users using GRANT and REVOKE statements, the operating system that operates Oracle can grant roles to users at connect time. Roles can be administered using the operating system and passed to Oracle when a user creates a session. As part of this mechanism, each user's default roles and the roles granted to a user

with the ADMIN OPTION can be identified. Even if the operating system is used to authorize users for roles, all roles must be created in the database and privileges assigned to the role with GRANT statements.

Roles can also be granted through a network service.

The advantage of using the operating system to identify a user's database roles is that privilege management for an Oracle database can be externalized. The security facilities offered by the operating system control a user's privileges. This option may offer advantages of centralizing security for a number of system activities, such as the following situation:

- MVS Oracle administrators want RACF groups to identify a database user's roles.
- UNIX Oracle administrators want UNIX groups to identify a database user's roles.
- VMS Oracle administrators want to use rights identifiers to identify a database user's roles.

The main disadvantage of using the operating system to identify a user's database roles is that privilege management can only be performed at the role level. Individual privileges cannot be granted using the operating system, but can still be granted inside the database using GRANT statements.

A secondary disadvantage of using this feature is that by default users cannot connect to the database through the shared server, or any other network connection, if the operating system is managing roles. However, you can change this default; see "Using Network Connections with Operating System Role Management" on page 25-25.

Note: The features described in this section are available only on some operating systems. See your operating system specific Oracle documentation to determine if you can use these features.

Using Operating System Role Identification

To operate a database so that it uses the operating system to identify each user's database roles when a session is created, set the initialization parameter OS_ROLES to TRUE (and restart the instance, if it is currently running). When a user attempts to create a session with the database, Oracle initializes the user's security domain using the database roles identified by the operating system.

To identify database roles for a user, each Oracle user's operating system account must have operating system identifiers (these may be called groups, rights identifiers, or other similar names) that indicate which database roles are to be available for the user. Role specification can also indicate which roles are the default roles of a user and which roles are available with the ADMIN OPTION. No matter which operating system is used, the role specification at the operating system level follows the format:

ora_*ID_ROLE*[_[d][a]]

where:

- ID has a definition that varies on different operating systems. For example, on VMS, ID is the instance identifier of the database; on MVS, it is the machine type; on UNIX, it is the system ID.

  Note: ID is case sensitive to match your ORACLE_SID. ROLE is not case sensitive.

- ROLE is the name of the database role.
- d is an optional character that indicates this role is to be a default role of the database user.
- a is an optional character that indicates this role is to be granted to the user with the ADMIN OPTION. This allows the user to grant the role to other roles only. Roles cannot be granted to users if the operating system is used to manage roles.

  Note: If either the d or a characters are specified, they must be preceded by an underscore.

For example, an operating system account might have the following roles identified in its profile:

```
ora_PAYROLL_ROLE1
ora_PAYROLL_ROLE2_a
ora_PAYROLL_ROLE3_d
ora_PAYROLL_ROLE4_da
```

When the corresponding user connects to the payroll instance of Oracle, role3 and role4 are defaults, while role2 and role4 are available with the ADMIN OPTION.

Using Operating System Role Management

When you use operating system managed roles, it is important to note that database roles are being granted to an operating system user. Any database user to which the OS user is able to connect will have the authorized database roles enabled. For this reason, you should consider defining all Oracle users as IDENTIFIED EXTERNALLY if you are using OS_ROLES = TRUE, so that the database accounts are tied to the OS account that was granted privileges.

Granting and Revoking Roles When OS_ROLES=TRUE

If OS_ROLES is set to TRUE, the operating system completely manages the grants and revokes of roles *to users*. Any previous grants of roles to users using GRANT statements do not apply; however, they are still listed in the data dictionary. Only the role grants made at the operating system level to users apply. Users can still grant privileges to roles and users.

Note: If the operating system grants a role to a user with the ADMIN OPTION, the user can grant the role only to other roles.

Enabling and Disabling Roles When OS_ROLES=TRUE

If OS_ROLES is set to TRUE, any role granted by the operating system can be dynamically enabled using the SET ROLE statement. This still applies, even if the role was defined to require a password or operating system authorization. However, any role not identified in a user's operating system account cannot be specified in a SET ROLE statement, even if a role has been granted using a GRANT statement when OS_ROLES = FALSE. (If you specify such a role, Oracle ignores it.)

When OS_ROLES = TRUE, a user can enable as many roles as specified by the initialization parameter MAX_ENABLED_ROLES.

Using Network Connections with Operating System Role Management

If you choose to have the operating system to manage roles, by default users cannot connect to the database through the shared server. This restriction is the default

because a remote user could impersonate another operating system user over a non-secure connection.

If you are not concerned with this security risk and want to use operating system role management with the shared server, or any other network connection, set the initialization parameter REMOTE_OS_ROLES in the database's initialization parameter file to TRUE. The change will take effect the next time you start the instance and mount the database. The default setting of this parameter is FALSE.

Viewing Privilege and Role Information

To access information about grants of privileges and roles, you can query the following data dictionary views:

View	Description
DBA_COL_PRIVS ALL_COL_PRIVS USER_COL_PRIVS	DBA view describes all column object grants in the database. ALL view describes all column object grants for which the current user or PUBLIC is the object owner, grantor, or grantee. USER view describes column object grants for which the current user is the object owner, grantor, or grantee.
ALL_COL_PRIVS_MADE USER_COL_PRIVS_MADE	ALL view lists column object grants for which the current user is object owner or grantor. USER view describes column object grants for which the current user is the grantor.
ALL_COL_PRIVS_RECD USER_COL_PRIVS_RECD	ALL view describes column object grants for which the current user or PUBLIC is the grantee. USER view describes column object grants for which the current user is the grantee.
DBA_TAB_PRIVS ALL_TAB_PRIVS USER_TAB_PRIVS	DBA view lists all grants on all objects in the database. ALL view lists the grants on objects where the user or PUBLIC is the grantee. USER view lists grants on all objects where the current user is the grantee.
ALL_TAB_PRIVS_MADE USER_TAB_PRIVS_MADE	ALL view lists the all object grants made by the current user or made on the objects owned by the current user. USER view lists grants on all objects owned by the current user.
ALL_TAB_PRIVS_RECD USER_TAB_PRIVS_RECD	ALL view lists object grants for which the user or PUBLIC is the grantee. USER view lists object grants for which the current user is the grantee.
DBA_ROLES	This view lists all roles that exist in the database.
DBA_ROLE_PRIVS USER_ROLE_PRIVS	DBA view lists roles granted to users and roles. USER view lists roles granted to the current user.

View	Description
DBA_SYS_PRIVS USER_SYS_PRIVS	DBA view lists system privileges granted to users and roles. USER view lists system privileges granted to the current user.
ROLE_ROLE_PRIVS	This view describes roles granted to other roles. Information is provided only about roles to which the user has access.
ROLE_SYS_PRIVS	This view contains information about system privileges granted to roles. Information is provided only about roles to which the user has access.
ROLE_TAB_PRIVS	This view contains information about object privileges granted to roles. Information is provided only about roles to which the user has access.
SESSION_PRIVS	This view lists the privileges that are currently enabled for the user.
SESSION_ROLES	This view lists the roles that are currently enabled to the user.

Some examples of using these views follow. For these examples, assume the following statements have been issued:

```
CREATE ROLE security_admin IDENTIFIED BY honcho;

GRANT CREATE PROFILE, ALTER PROFILE, DROP PROFILE,
    CREATE ROLE, DROP ANY ROLE, GRANT ANY ROLE, AUDIT ANY,
    AUDIT SYSTEM, CREATE USER, BECOME USER, ALTER USER, DROP USER
    TO security_admin WITH ADMIN OPTION;

GRANT SELECT, DELETE ON SYS.AUD$ TO security_admin;

GRANT security_admin, CREATE SESSION TO swilliams;

GRANT security_admin TO system_administrator;

GRANT CREATE SESSION TO jward;

GRANT SELECT, DELETE ON emp TO jward;

GRANT INSERT (ename, job) ON emp TO swilliams, jward;
```

See Also: *Oracle9i Database Reference* for a detailed description of these data dictionary views

Listing All System Privilege Grants

The following query returns all system privilege grants made to roles and users:

```
SELECT * FROM DBA_SYS_PRIVS;

GRANTEE             PRIVILEGE                                ADM
--------------      -------------------------------- ---
SECURITY_ADMIN      ALTER PROFILE                            YES
SECURITY_ADMIN      ALTER USER                               YES
SECURITY_ADMIN      AUDIT ANY                                YES
SECURITY_ADMIN      AUDIT SYSTEM                             YES
SECURITY_ADMIN      BECOME USER                              YES
SECURITY_ADMIN      CREATE PROFILE                           YES
SECURITY_ADMIN      CREATE ROLE                              YES
SECURITY_ADMIN      CREATE USER                              YES
SECURITY_ADMIN      DROP ANY ROLE                            YES
SECURITY_ADMIN      DROP PROFILE                             YES
SECURITY_ADMIN      DROP USER                                YES
SECURITY_ADMIN      GRANT ANY ROLE                           YES
SWILLIAMS           CREATE SESSION                           NO
JWARD               CREATE SESSION                           NO
```

Listing All Role Grants

The following query returns all the roles granted to users and other roles:

```
SELECT * FROM DBA_ROLE_PRIVS;

GRANTEE             GRANTED_ROLE                               ADM
------------------- ---------------------------------- ---
SWILLIAMS           SECURITY_ADMIN                             NO
```

Listing Object Privileges Granted to a User

The following query returns all object privileges (not including column-specific privileges) granted to the specified user:

```
SELECT TABLE_NAME, PRIVILEGE, GRANTABLE FROM DBA_TAB_PRIVS
    WHERE GRANTEE = 'JWARD';

TABLE_NAME   PRIVILEGE    GRANTABLE
-----------  ------------ ----------
EMP          SELECT       NO
EMP          DELETE       NO
```

To list all the column-specific privileges that have been granted, use the following query:

```
SELECT GRANTEE, TABLE_NAME, COLUMN_NAME, PRIVILEGE
    FROM DBA_COL_PRIVS;

GRANTEE      TABLE_NAME     COLUMN_NAME      PRIVILEGE
-----------  ------------   -------------    --------------
SWILLIAMS    EMP            ENAME            INSERT
SWILLIAMS    EMP            JOB              INSERT
JWARD        EMP            NAME             INSERT
JWARD        EMP            JOB              INSERT
```

Listing the Current Privilege Domain of Your Session

The following query lists all roles currently enabled for the issuer:

```
SELECT * FROM SESSION_ROLES;
```

If `swilliams` has enabled the `security_admin` role and issues this query, Oracle returns the following information:

```
ROLE
------------------------------
SECURITY_ADMIN
```

The following query lists all system privileges currently available in the issuer's security domain, both from explicit privilege grants and from enabled roles:

```
SELECT * FROM SESSION_PRIVS;
```

If `swilliams` has the `security_admin` role enabled and issues this query, Oracle returns the following results:

```
PRIVILEGE
----------------------------------------
AUDIT SYSTEM
CREATE SESSION
CREATE USER
BECOME USER
ALTER USER
DROP USER
CREATE ROLE
DROP ANY ROLE
GRANT ANY ROLE
AUDIT ANY
CREATE PROFILE
ALTER PROFILE
DROP PROFILE
```

If the security_admin role is disabled for swilliams, the first query would have returned no rows, while the second query would only return a row for the CREATE SESSION privilege grant.

Listing Roles of the Database

The DBA_ROLES data dictionary view can be used to list all roles of a database and the authentication used for each role. For example, the following query lists all the roles in the database:

```
SELECT * FROM DBA_ROLES;

ROLE                  PASSWORD
----------------      --------
CONNECT               NO
RESOURCE              NO
DBA                   NO
SECURITY_ADMIN        YES
```

Listing Information About the Privilege Domains of Roles

The ROLE_ROLE_PRIVS, ROLE_SYS_PRIVS, and ROLE_TAB_PRIVS data dictionary views contain information on the privilege domains of roles.

For example, the following query lists all the roles granted to the system_admin role:

```
SELECT GRANTED_ROLE, ADMIN_OPTION
   FROM ROLE_ROLE_PRIVS
   WHERE ROLE = 'SYSTEM_ADMIN';

GRANTED_ROLE              ADM
---------------           ----
SECURITY_ADMIN            NO
```

The following query lists all the system privileges granted to the security_admin role:

```
SELECT * FROM ROLE_SYS_PRIVS WHERE ROLE = 'SECURITY_ADMIN';

ROLE                    PRIVILEGE                      ADM
----------------------- -----------------------------  ---
SECURITY_ADMIN           ALTER PROFILE                 YES
SECURITY_ADMIN           ALTER USER                    YES
```

```
SECURITY_ADMIN          AUDIT ANY                    YES
SECURITY_ADMIN          AUDIT SYSTEM                 YES
SECURITY_ADMIN          BECOME USER                  YES
SECURITY_ADMIN          CREATE PROFILE               YES
SECURITY_ADMIN          CREATE ROLE                  YES
SECURITY_ADMIN          CREATE USER                  YES
SECURITY_ADMIN          DROP ANY ROLE                YES
SECURITY_ADMIN          DROP PROFILE                 YES
SECURITY_ADMIN          DROP USER                    YES
SECURITY_ADMIN          GRANT ANY ROLE               YES
```

The following query lists all the object privileges granted to the `security_admin` role:

```
SELECT TABLE_NAME, PRIVILEGE FROM ROLE_TAB_PRIVS
    WHERE ROLE = 'SECURITY_ADMIN';

TABLE_NAME                     PRIVILEGE
-----------------------------  ----------------
AUD$                           DELETE
AUD$                           SELECT
```

26

Auditing Database Use

This chapter describes how to use the Oracle database server's auditing facilities, and contains these topics: