Module 14-3 – Roles

Objectives

- Create roles as a database object.
- Drop roles.
- Manage the allocation of privileges through roles.
- Familiarize with various dictionary views that provide information about roles.

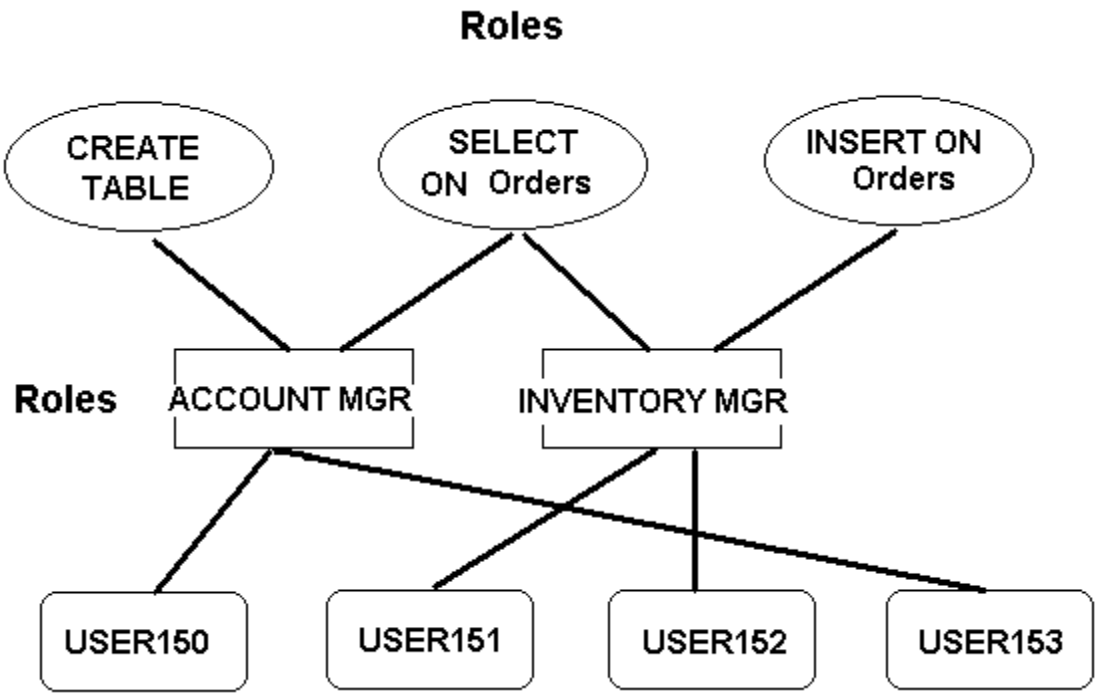
General

The **Role** database object is used to improve the management of various system objects, such as tables, indexes, and clusters by granting privileges to access these objects to roles. As you learned in earlier studies, there are two types of privileges, **System** and **Object**. Both types of privileges can be allocated to roles.

The concept of a role is a simple one – a role is created as a **container** for groups of privileges that are granted to system users who perform similar, typical tasks in a business.

Example: A system user fills the position of **Account_Manager**. This is a business role. The role is created as a database object and privileges are allocated to the role. In turn the role is allocated to all employees that work as account managers, and all account managers thereby inherit the privileges needed to perform their duties.

This figure shows privileges being allocated to roles, and the roles being allocated to two types of system users – **Account_Mgr** and **Inventory_Mgr**.



From the figure it should be obvious that if you add a new system user who works as an **Account_Manager**, then you can allocate almost all of the privileges this user will need by simply allocating the **role** named **ACCOUNT_MGR** to the system user.

Facts About Roles

- You may also grant a role to another role (except to itself).

- A role can include both system and object privileges. Roles have system and object privileges granted to them just the same way that these privileges are granted to system users.

- You can require a password to enable a role.

- A role name must be unique.

- Roles are not owned by anyone - are not in anyone's schema.

- If a role has its privileges modified, then the privileges of the system users granted the role are also modified.

- There are no cascading revokes with roles.

- Using roles reduces how many Grants are stored in a database data dictionary.

- There is a limited set of privileges that cannot be granted to a role, but most privileges can be granted to roles.

Role Benefits

- **Easier privilege management:** Use roles to simplify privilege management. Rather than granting the same set of privileges to several users, you can grant the privileges to a role, and then grant that role to each user.

- **Dynamic privilege management:** If the privileges associated with a role are modified, all the users who are granted the role acquire the modified privileges automatically and immediately.

- **Selective availability of privileges:** Roles can be enabled and disabled to turn privileges on and off temporarily. Enabling a role can also be used to verify that a user has been granted that role.

- **Can be granted through the operating system:** Operating system commands or utilities can be used to assign roles to users in the database.

Predefined Roles

Numerous predefined roles are created as part of a database. These are listed and described in the following table.

The first 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.

ROLE	Script to Create Role	DESCRIPTION

CONNECT	SQL.BSQ	Includes system privileges: ALTER SESSION (This role has been deprecated and has only been retained with the ALTER SESSION privilege for compatibility with previous Oracle versions)
RESOURCE	SQL.BSQ	Includes system privileges: CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
DBA	SQL.BSQ	Gives all system privileges to the grantee WITH ADMIN OPTION.
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.
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_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.

Note: **HS (Heterogeneous Services)** – Heterogeneous Services (HS) is an integrated component within the Oracle Database server and the enabling technology for the current suite of Oracle Transparent Gateway products. HS provides the common architecture and administration mechanisms for Oracle Database gateway products and other heterogeneous access facilities. Also, it provides upwardly compatible functionality for users of most of the earlier Oracle Transparent Gateway releases. The transparent gateway agent facilitates communication between Oracle Database and non-Oracle Database systems and uses the Heterogeneous Services component in the Oracle Database server.

RESOURCE role – when granted to a system user, the system user automatically has the **UNLIMITED TABLESPACE** privilege.

- We grant this role to students that need to design with the Internet Developer Suite that includes Oracle Designer, Reports, Forms and other rapid application development software.

- Normally the **RESOURCE** role would not be granted to organizational members who are not information technology professionals.

You should design your own roles to provide data security.

Commands for Creating, Altering, and Dropping Roles

Creating Roles

Sample commands to create roles are shown here. You must have the **CREATE ROLE** system privilege.

```
CREATE ROLE Account_Mgr;
```

```
CREATE ROLE Inventory_Mgr
    IDENTIFIED BY <password>;
```

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 several ways.

- **The database using a password** – 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.

- **An application using a specified package** -- 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;
```

- **Externally by the operating system, network, or other 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;
```

- **Globally 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;
```

Altering Roles

Use the **ALTER ROLE** command as is shown in these examples.

```
ALTER ROLE Account_Mgr IDENTIFIED BY <password>;
```

```
ALTER ROLE Inventory_Mgr NOT IDENTIFIED;
```

Granting Roles

General facts about roles:

- Grant system privileges and roles to users and to other roles.
- To grant a privilege to a role, you must be granted a system privilege with the **ADMIN OPTION** or have the **GRANT ANY PRIVILEGE** system privilege.
- To grant a role, you must have been granted the role yourself with the **ADMIN OPTION** or have the **GRANT ANY ROLE** system privilege.
- You cannot grant a role that is **IDENTIFIED GLOBALLY** as global roles are controlled entirely by the enterprise directory service.

Use the **GRANT** command to grant a role to a system user or to another role, as is shown in these examples.

```
GRANT Account_Mgr TO User150;
```

```
GRANT Inventory_Mgr TO Account_Mgr, User151;
```

```
GRANT Inventory_Mgr TO User152 WITH ADMIN OPTION;

GRANT Access_MyBank_Acct TO PUBLIC;
```

The **WITH ADMIN OPTION** provides the grantee expanded capabilities:

- Can grant or revoke the system privilege or role to or from any user or other database role.
- Can further grant the system privilege or role with ADMIN OPTION.
- Can alter or drop the role.
- **CANNOT** revoke a role from theirself.

When you create a role, the role is automatically granted to you with the ADMIN OPTION.

Granting with **ADMIN OPTION** is rarely done except to allocate privileges to security administrators, not to other administrators or system users.

Creating a New User with the GRANT Command

If you grant a role to a user name and the user does not exist, then a new user/password combination is created.

Example: This example creates a new user dbock with the specified password.

```
GRANT CONNECT TO Dbock IDENTIFIED BY
Secret_Pa$$w0rd;
```

Granting Object Privileges

To GRANT object privileges to a role or user, you must:

- Own the object specified, or
- Have the **GRANT ANY OBJECT PRIVILEGE** system privilege (to grant/revoke privileges on behalf of the object owner), or
- Have been granted an object privilege by the owner with the **WITH GRANT OPTION** clause.

You **cannot** grant system privileges and roles with object privileges in the same GRANT statement.

Example: This grants **SELECT**, **INSERT**, and **DELETE** privileges for all columns of the **EMPLOYEE** table to two user accounts.

```
GRANT SELECT, INSERT, DELETE ON Employee TO User350,
User349;
```

Example: This grants all object privileges on the **SUPERVISOR** view to a user by use of the **ALL** keyword.

```
GRANT ALL ON Supervisor TO User350;
```

Example: This specifies the **WITH GRANT OPTION** to enable **User350** to grant the object privileges to other users and roles.

- The grantee can grant object privileges to other users and roles in the database.
- The grantee can create views on the table.
- The grantee can grant corresponding privileges on the views to other users and roles.
- The grantee **CANNOT** use the **WITH GRANT OPTION** when granting object privileges to a role.

```
GRANT SELECT, INSERT, DELETE ON Employee TO User350
WITH GRANT OPTION;
```

Granting Column Privileges

Use this approach to control privileges on individual table columns.

- Before granting an **INSERT** privilege for a column, determine if any columns have **NOT NULL** constraints.

- Granting an **INSERT** privilege on a column where other columns are specified **NOT NULL** prevents inserting any table rows.

Example: This grants the **INSERT** and **UPDATE** privileges on the **Employee_ID**, **Last_Name**, and **First_Name** columns of the **Employee** table.

```
GRANT INSERT, UPDATE (Employee_Id, Last_Name,
First_Name) ON Employee TO User350, User349;
```

Default Roles

Oracle enables all privileges granted to a user and through user default roles when a user logs on.

The **ALTER USER** statement enables a DBA to specify the roles to be enabled when a system user connects to the database without requiring the user to specify the roles' passwords. These roles must have already been granted to the user with the GRANT statement.

System users can be assigned **default roles** as shown in these examples.

```
ALTER USER User152
    DEFAULT ROLE Account_Mgr;

ALTER USER User152, User151
    DEFAULT ROLE Account_Mgr, Inventory_Mgr;

ALTER USER User150
    DEFAULT ROLE ALL EXCEPT Account_Mgr;

ALTER USER User153 DEFAULT ROLE NONE;
```

Using the **ALTER USER** command to limit the default role causes privileges assigned to the user by other roles to be temporarily removed.

The last example limits **User153** only to privileges granted directly to the user, with no privileges being allowed through roles.

You can also enable/disable roles through the **SET ROLE** command.

You **cannot** set a user's default roles with the **CREATE USER** statement.

The number of default roles a user can have is specified with the **MAX_ENABLED_ROLES** parameter.

The SET ROLE Statement

This statement **enables/disables** roles for a session. You must have been granted any roles you name in a **SET ROLE** statement.

Example: This enables the role **Inventory_Mgr** that you have been granted by specifying the password.

```
SET ROLE Inventory_Mgr IDENTIFIED BY Pa$$w0rd;
```

Example: This disables all roles.

```
SET ROLE NONE;
```

Revoking Roles and Privileges

Roles, system privileges, and object privileges are revoked with the REVOKE command.

- Requires the **ADMIN OPTION** to revoke a system privilege or role.
- Users with **GRANT ANY ROLE** can also revoke any role.

- You cannot revoke the **ADMIN OPTION** for a role or system privilege – you must revoke the privilege or role and then grant it again without the **ADMIN OPTION**.

```
REVOKE Account_Mgr FROM User151;
```

```
REVOKE Account_Mgr FROM Inventory_Mgr;

REVOKE Access_MyBank_Acct FROM PUBLIC;
```

The second example revokes the role **Account_Mgr** from the role **Inventory_Mgr**. The third example revokes the role **Access_MyBank_Acct** from **PUBLIC**.

When revoking object privileges:

- To revoke an object privilege you must have previously granted the object privilege to the user or role, or you have the **GRANT ANY OBJECT PRIVILEGE** system privilege.

- You can only revoke object privileges you directly granted, not grants made by others to whom you granted the **GRANT OPTION** – but there is a cascading effect – object privilege grants propagated with the **GRANT OPTION** are revoked if the grantor's object privilege is revoked.

Example: You are the original grantor, this **REVOKE** will revoke the specified privileges from the users specified.

```
REVOKE SELECT, INSERT, DELETE ON Employee FROM User350,
Inventory_Mgr;
```

Example: You granted User350 the privilege to UPDATE the Birth_Date, Last_Name, and First_Name columns for the Employee table, but now want to revoke the UPDATE privilege on the Birth_Date column.

```
REVOKE UPDATE ON Employee FROM User350;

GRANT UPDATE (Last_Name, First_Name) ON Employee TO
User350;
```

You must first revoke the **UPDATE** privilege on all columns, then issue a **GRANT** to regrant the **UPDATE** privilege on the specified columns.

Cascading Revoke Effects

There are no cascading effects for revoking a system privilege related to a DDL operation.

Example:

- You as the **DBA** grant the **CREATE VIEW** system privilege to **User350 WITH ADMIN OPTION**.
- **User350** creates a view named **Employee_Supervisor**.

- **User350** grants the **CREATE VIEW** system privilege to **user349**.
- **User349** creates a view named **Special_Inventory**.
- You as the **DBA** revoke **CREATE VIEW** from **User350**.
- The **Employee_Supervisor** view continues to exist.
- **User349** still has the **CREATE VIEW** system privilege and the **Special_Inventory** view continues to exist.

Cascading revoke effects do occur for system privileges related to DML operations.

Example:

- You as the **DBA** grant the **UPDATE ANY TABLE** to **User350**.
- **User350** creates a procedure that updates the Employee table, but **User350** has not received specific privileges on the Employee table.
- You as the **DBA** revoke the **UPDATE ANY TABLE** privilege.
- The procedure will fail.

Dropping Roles

If you drop a role:

- Oracle revokes the role from all system users and roles.
- The role is removed from the data dictionary.
- The role is automatically removed from all user default role lists.
- There is NO impact on objects created such as tables because the creation of objects is not depending on privileges received through a role.

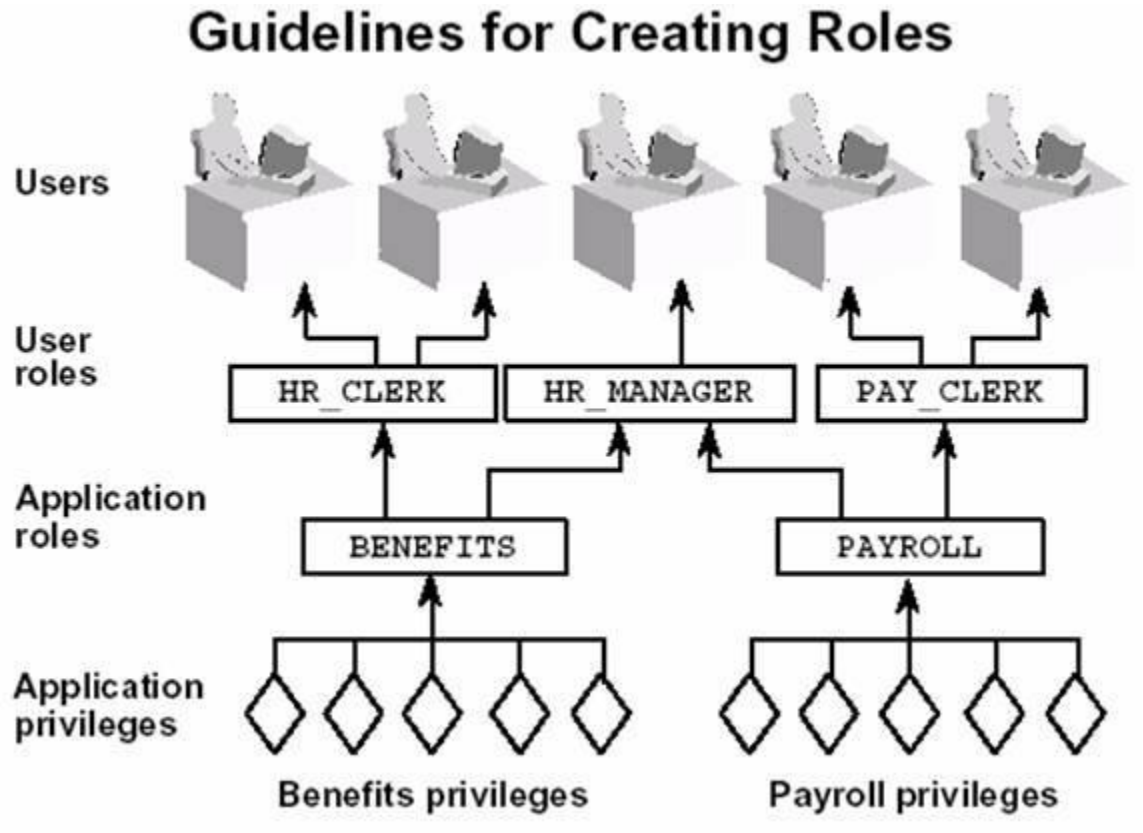
In order to drop a role, you must have been granted the role with the **ADMIN OPTION** or have the **DROP ANY ROLE** system privilege.

```
DROP ROLE Account_Mgr;
```

Guidelines for Creating Roles

Role names are usually an application task or job title because a role has to include the privileges needed to perform a task or work in a specific job. The figure shown here uses both application tasks and

job titles for role names.



Use the following steps to create, assign, and grant users roles:

1. Create a role for each application task. The name of the application role corresponds to a task in the application, such as **PAYROLL**.

2. Assign the privileges necessary to perform the task to the application role.

3. Create a role for each type of user. The name of the user role corresponds to a job title, such as **PAY_CLERK.**

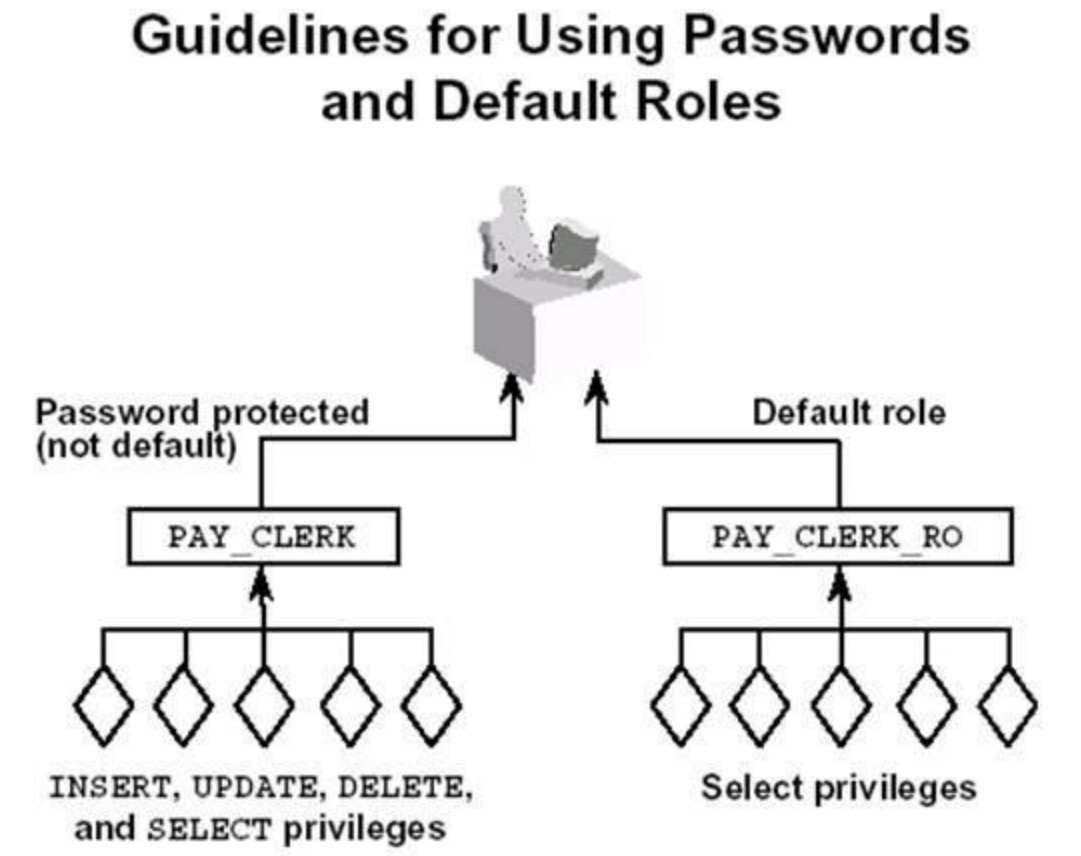
4. Grant application roles to user's roles.

5. Grant user's roles to users.

If a modification to the application requires that new privileges are needed to perform the payroll task, then the DBA only needs to assign the new privileges to the application role, **PAYROLL**. All of the users that are currently performing this task will receive the new privileges.

Guidelines for Using Passwords and Default Roles

Passwords provide an additional level of security when enabling a role. For example, the application might require a user to enter a password when enabling the **PAY_CLERK** role, because this role can be used to issue checks. Passwords allow a role to be enabled only through an application. This technique is shown in the example in the figure.



The DBA has granted the user two roles, **PAY_CLERK** and **PAY_CLERK_RO**.

- The **PAY_CLERK** role has been granted all of the privileges that are necessary to perform the payroll clerk function.
- The **PAY_CLERK_RO** (**RO** for **read only**) role has been granted only **SELECT** privileges on the tables required to perform the payroll clerk function.
- The user can log in to SQL*Plus to perform queries, but cannot modify any of the data, because the **PAY_CLERK** is not a default role, and the user does not know the password for **PAY_CLERK**.

- When the user logs in to the payroll application, it enables the **PAY_CLERK** by providing the password. It is coded in the program; the user is not prompted for it.

Role Data Dictionary Views

The following views provide information about roles that are useful for managing a database.

- DBA_ROLES - Listing of all roles in the database.
- DBA_ROLE_PRIVS - Listing of roles granted to system users and to other roles.
- ROLE_ROLE_PRIVS - Roles granted to roles.
- DBA_SYS_PRIVS - System privileges granted to users and roles.
- ROLE_SYS_PRIVS - System privileges granted to roles.
- ROLE_TAB_PRIVS - Table privileges granted to roles.
- SESSION_ROLES - Roles the user has enabled.

END OF NOTES