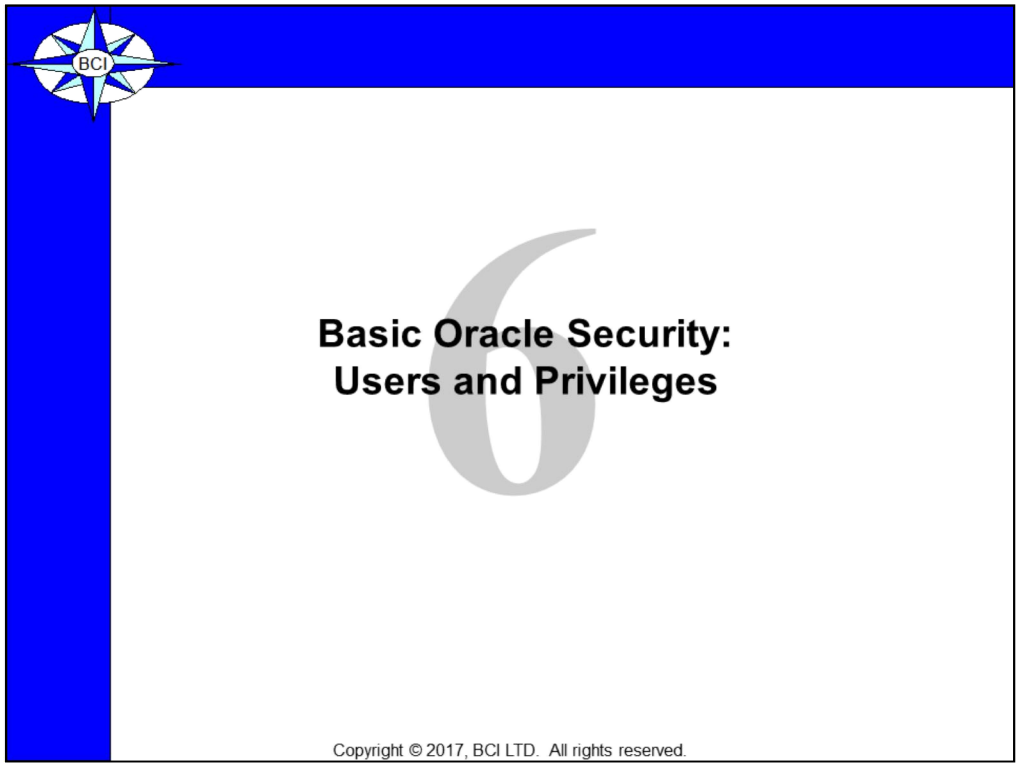
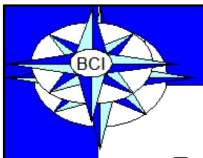

BCI
6
Basic Oracle Security:
Users and Privileges
Copyright © 2017, BCI LTD. All rights reserved.

Terminal Learning Objective

ACTION: Maintain Oracle Database User Accounts.

CONDITION: Given a student handout and Oracle DBA Handbook.

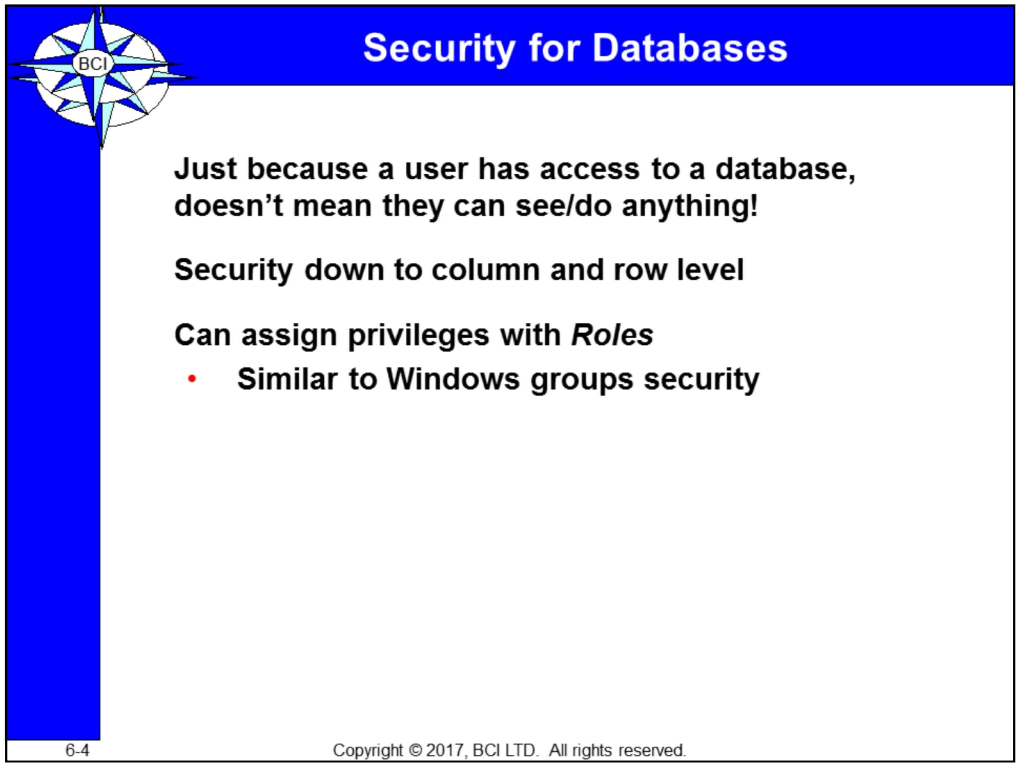
STANDARD: Students will successfully create users, roles and perform resource and password management.



Learning Objectives

By the end of this module, you will:

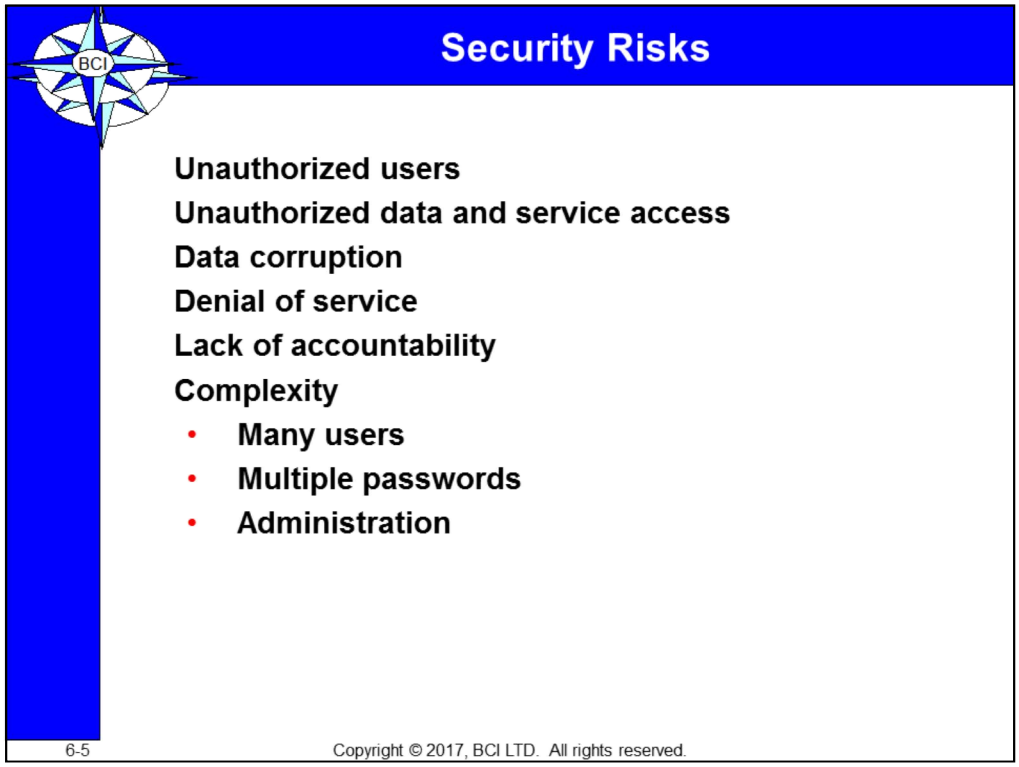
- **Understand Oracle Security**
- **Understand System Privileges**
- **Understand Object Privileges**
- **Create Users, Roles, and Profiles**



In our workplace security is paramount. Databases contain some very sensitive information that could be dangerous if placed into the wrong hands. In Oracle, security is managed by assigning privileges to a user based on their needs.

Just because a user has access to a database, doesn't mean they can see/do anything! Oracle provides security down to column and row level through use of Roles.

You should also refer to the Database Security Technical Implementation Guide (STIG) for further security requirements. This STIG can be downloaded at: http://iase.disa.mil/stigs



More and more frequently, data stored within a database management system (DBMS) has become a target of attack for malicious users. The effect of such an attack can result in identity and/or credit card theft, financial loss, loss of privacy, a breach of national security or any other type of corruption that can result from unauthorized access to sensitive data. As database products have evolved, more and more security options are becoming available. DBMSs have also joined the ranks of victims of malicious attacks and DBMS vendors have had to respond by issuing fixes for discovered vulnerabilities. The STIG presents the known security configuration items, vulnerabilities, and issues required to be addressed by DOD policy. In addition to this STIG, compliance validation tools and checklists are available to **.mil** and **.gov** customers to assist in the efforts to implement the required configuration.

It should be noted that Defense Information Systems Agency (DISA) Field Security Operations (FSO) Support for the STIGs, Checklists, and Tools is only available to DOD Customers.

Unauthorized users

You need to know your users. This is commonly done by requiring the user to enter a password.

Unauthorized data access

Anywhere the data is stored and transmitted; it is subject to unauthorized access.

Data Corruption

Privacy of communication is essential to ensure that data cannot be modified or viewed in transit. In a distributed environment there is the possibility of a third party tampering with data as it moves between sites.

In a data modification attack, an unauthorized party on the network intercepts data in transit, changes parts of that data, and then sends the modified data to its original destination. An example of this is changing the dollar amount of a banking transaction from $100 to $10,000.

In a replay attack, an entire set of valid data is repeatedly interjected onto the network. An example would be to repeat a valid $100 bank account transfer transaction one thousand times.

Denial of service

Your computer resources can be overloaded by denial of service (DOS) attacks by creating multiple requests for a service, to which your systems cannot respond. A denial of service attack can occur by using authorized or unauthorized access.

Lack of accountability

If the system administrator is unable to track the activities of the users, the users cannot be held responsible for their actions. There must be some reliable way to monitor who is performing what operations on the data.

Complexity

The large number of users and applications makes the enforcement of security more complex.

Multiple user passwords

Many of the security measures are invisible to the user until they violate a security policy. However, the maintenance of the user's own passwords is visible to the user. When users must remember passwords for multiple systems, they may use techniques that compromise security.

Multitier systems

This problem of complexity issue becomes worse in multitier systems. Here, and in most packaged applications, the typical security model is that of "One Big Application User." The user connects to the application, and the application or application server logs on and provides complete access for everyone, with no auditing and unlimited privileges. This model places your data at risk, especially in the Internet, where your Web server or application server depends on a firewall for security. Firewalls can be easily broken.

Administration of large user communities

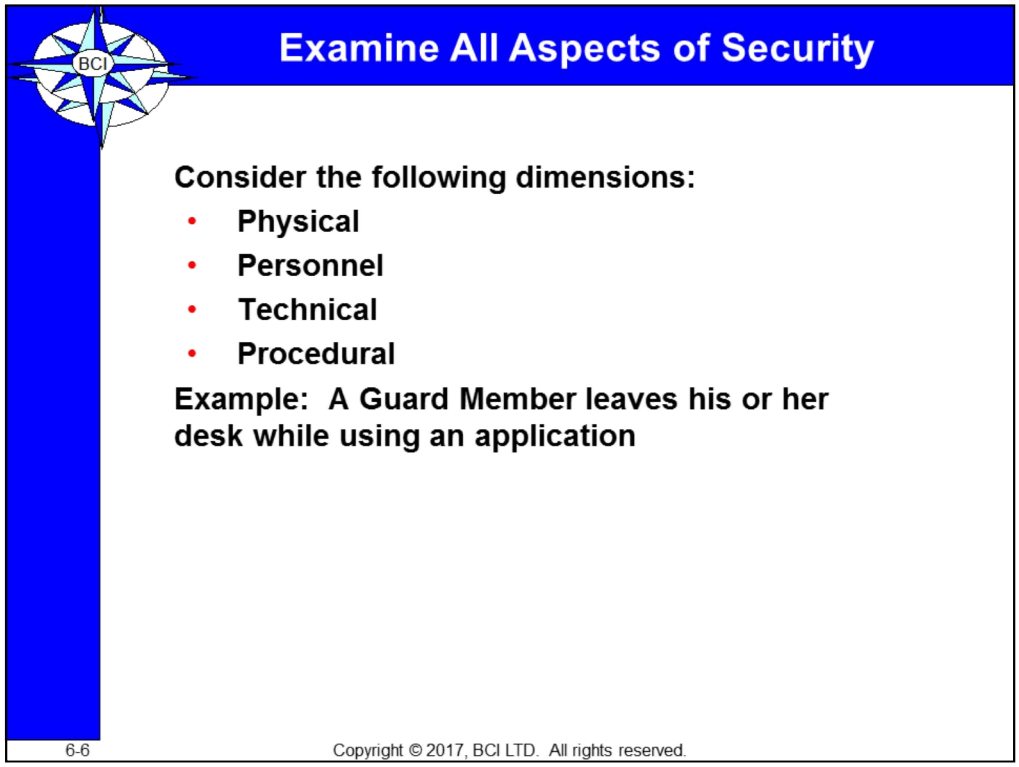
Systems must be scalable, as they often have to support hundreds of thousands of users. In such large-scale environments, the burden of managing user accounts and passwords make your system vulnerable to error and attack. To have reliable security you need to know who the user really is across all tiers of the application.

Administration of multiple systems

Administration of hundreds of thousands of users is difficult on a single system. This burden is compounded when security must be administered on multiple systems.

To meet the challenges of scalability in security administration, you should be able to centrally manage users and privileges across multiple applications and databases, using a directory based on industry standards. This can reduce system management costs and increase business efficiency.

Further, creating and building separate databases for multiple application subscribers is not a cost-efficient model for an application service provider. While technically possible, the separate database model would quickly become unmanageable. To be successful, a single application installation should be able to host multiple companies and be administered centrally.



To protect all the elements of complex computing systems, you must address security issues in the following dimensions:

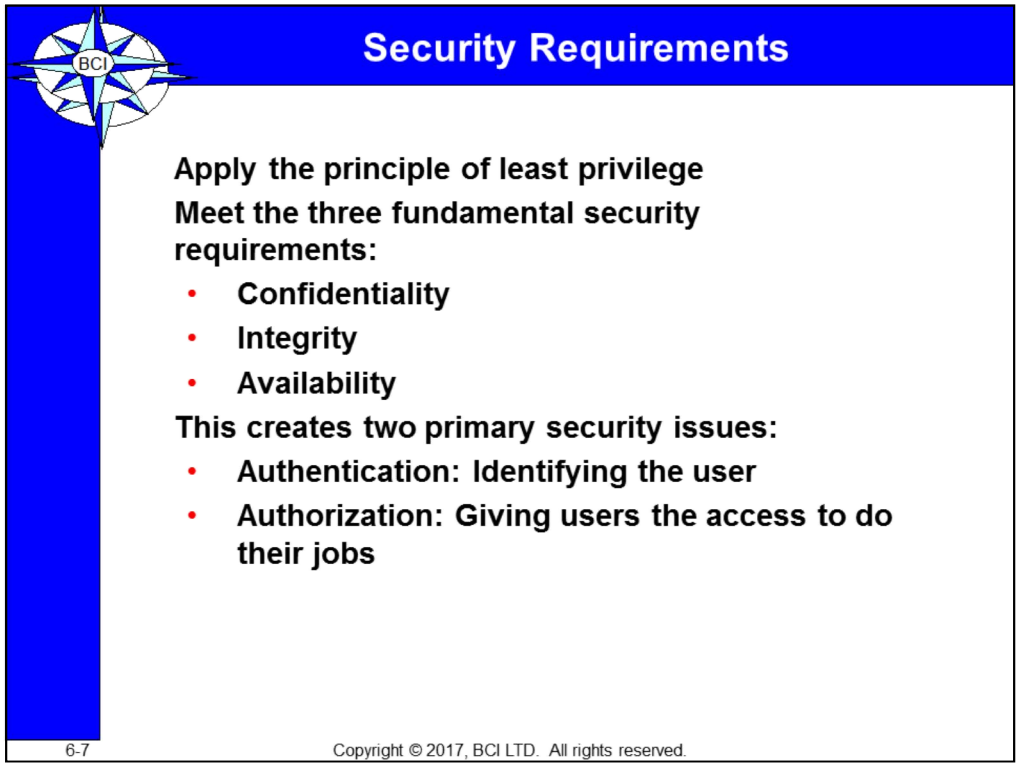
•**Physical**: Your computers must be physically inaccessible to unauthorized users. This means that you must keep them in a secure physical environment.

•**Personnel**: The people responsible for system administration and data security at your site must be reliable. You may need to perform background checks on DBAs before making hiring decisions.

•**Technical**: Storage, access, manipulation, and transmission of data must be safeguarded by technology that enforces your particular information control policies.

•**Procedural**: The procedures used in the operation of your system must assure reliable data. These procedures are developed after looking at the other three aspects of security. For example, one person might be responsible for database backups. Her only role is to make sure that the database is up and running. Another person might be responsible for generating application reports involving payroll or sales data. His role is to examine the data and verify its integrity. It may be wise to separate out the functional roles of the users in data management.

Think carefully about the specific security risks to your data, and make sure that the solutions you adopt actually address the problems. In some instances, a technical solution may be inappropriate.



All security requirements are derived from the basic requirement called the principle of least privilege. This principle states that you should give the users only those privileges that they need to do their job. To apply the principle of least privilege, you must be able to do the following:

•Authenticate users so that you know who the users are.

•Authorize users so they have access to the data and services that they need to perform their jobs.

Authentication

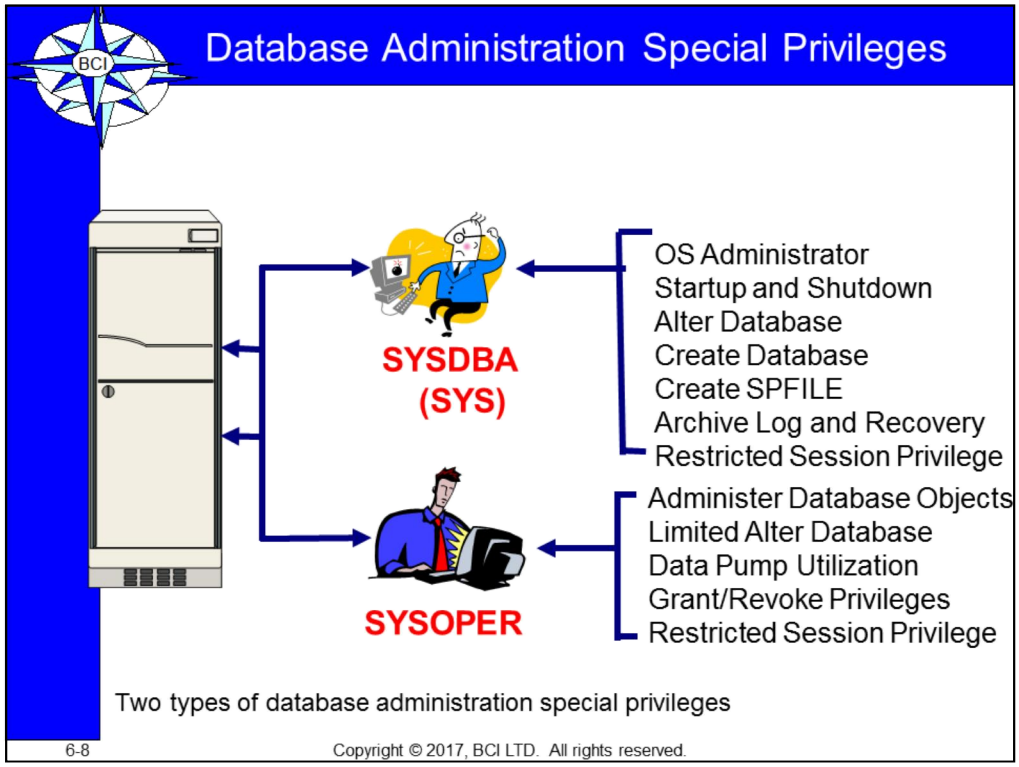
Make sure that you know the user who is accessing your systems. This is done by requiring the users to identify themselves when accessing the service. Depending on the required level of security, you can enforce stricter authentication techniques, such as certificates. Also, you may have public users that require no authentication.

It is important to educate your users about security. They are responsible for identifying themselves when accessing computer resources. Therefore, they must understand security-related procedures and know the consequences of violating

these procedures.

Authorization

Make sure that the users can only access those services and data that are required to perform their jobs. Authorization techniques used by different computer systems vary, but most allow the administrator to group users with common authorization. Also, because the user access to data may only occur when the user is logged onto an application, access to services (the application) is as important as access to data. For example, you might allow your human resources clerk to update the employee table through the human resources applications, but that does not mean that you will also give the clerk update access to the employee table through SQL*Plus.



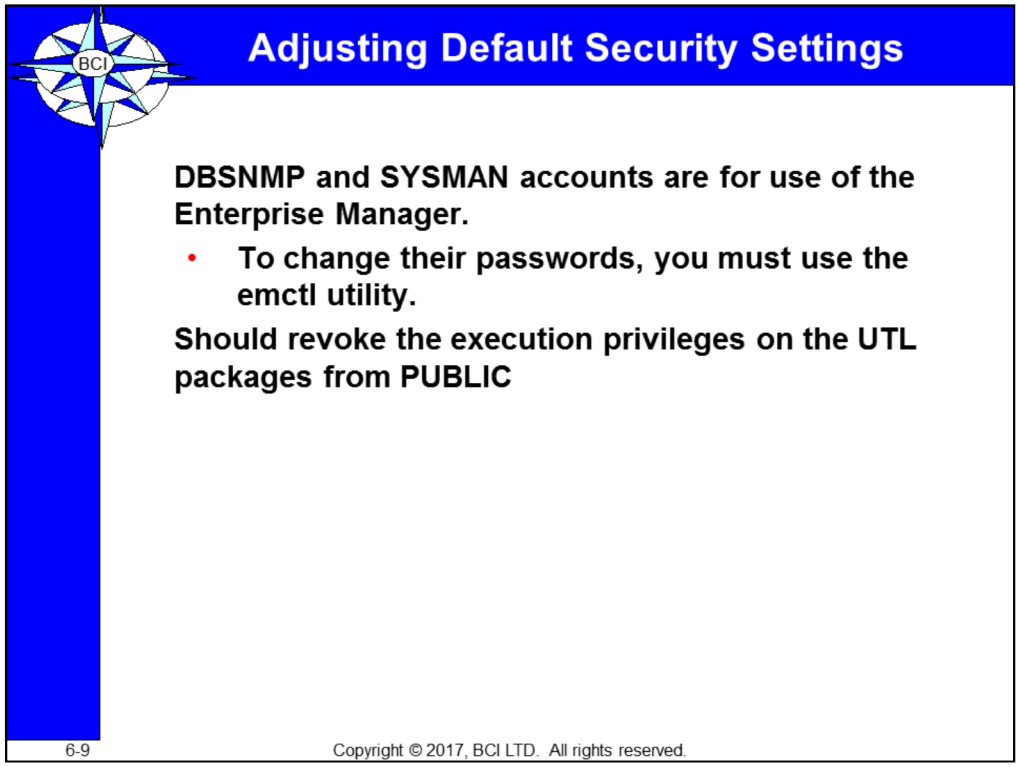
Oracle administrative connections (SYS as SYSDBA) will only be used to perform administrative functions available exclusively to an SUPER USER administrative connection. The Oracle administrative connection will not be used to perform everyday operations. Any use of an Oracle administrative connection will be documented in the alert log. By default, Oracle records administrative connections in the OS audit log.

Examples of appropriate operations requiring use of an administrative connection include installation, database creation, backup and recovery, database startup, and database shutdown. The use of administrative connections is not recommended for automated procedures or utilities

that perform automated functions for the DBA. The ability to authenticate to the database with an administrative connection will be restricted to authorized DBAs.

A password file will not be used unless remote database administration is justified and required. In such cases, its use will be authorized and documented by the IAO. If remote administration is required, the password file will be used in exclusive mode. Exclusive mode requires individual

account authentication and restricts assignment of database administrative privileges to accounts granted the SYSDBA privilege. Where remote administration is required, a password file will be used in exclusive mode.



The DBSNMP and SYSMAN accounts are for the use of Enterprise Manager, either Grid Control or Database Control. To change their passwords, you must use the emctl utility.

It should only be necessary to unlock a default account in exceptional circumstances. These accounts are used to store data and code required by certain options within the database, not for users to connect to. For example, the MDSYS schema stores the objects required by the Oracle Spatial option, which extends the capabilities of the database to manage geographical information. Users can make use of the spatial option without needing to connect to the schema directly. Even the demonstration schemas (HR, OE, and so on) are locked after you create them.

There is a pseudo-user called PUBLIC. Any privileges granted to PUBLIC have, in effect, been granted to every user; every account you create will have access to these privileges. By default, the public user has a large number of privileges. In particular, he has execute permission on a number of PL/SQL utility packages. You should always consider revoking execution privileges on the UTL packages, though remember that application software may assume that the privilege is there. Revoke the privilege as follows:

revoke execute on utl_file from public;

Some of the more dangerous packages are listed below. Consult the most current database STIG for other revocations.

•**UTL_FILE** This allows users to read and write any file and directory that is accessible to the operating system user under whose identity the Oracle processes are running. This includes all the database files and the ORACLE_HOME directory. This is particular dangerous on Windows systems and many Windows databases run with Administrator privileges.

•**UTL_TCP** This allows users to open TCP ports on the server machine for connections to any accessible address on the network. The interface provided in the package only allows connections to be initiated by the PL/SQL program; it does not allow the PL/SQL program to accept connections initiated outside the program. However, it does allow malicious users to use your database as the starting point for launching attacks on other systems, or for transmitting data to unauthorized recipients.

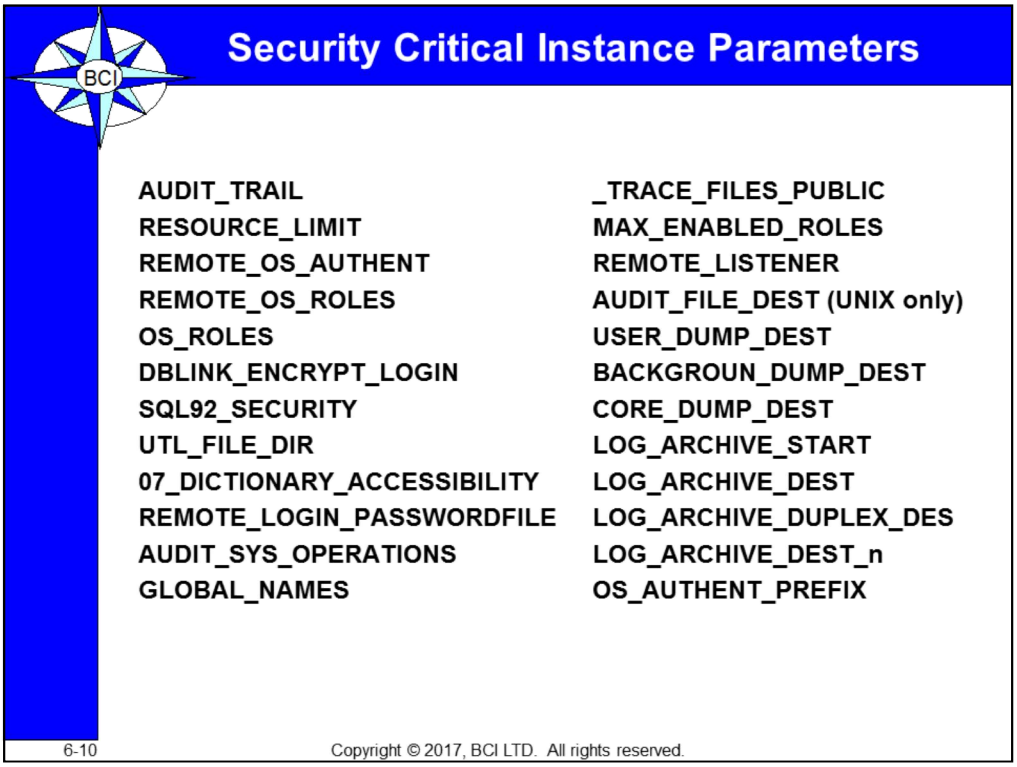
•**UTL_SMTP** Written using UTL_TCP calls, this package lets users send mail messages. It is restricted by the UTL_SMTP_SERVER instance parameter, which gives the address of the outgoing mail server, but even so, you probably do not want your database to be used for exchange of mail messages without your knowledge.

•**UTL_HTTP** This too is written with UTL_TCP calls. It allows users to send HTTP messages and receive responses, in effect, converting your database into a web browser.

In addition to the above recommendations, the Database STIG, V7R2, section B.5.3 states that the DBA will revoke object privileges assigned to PUBLIC on the following:

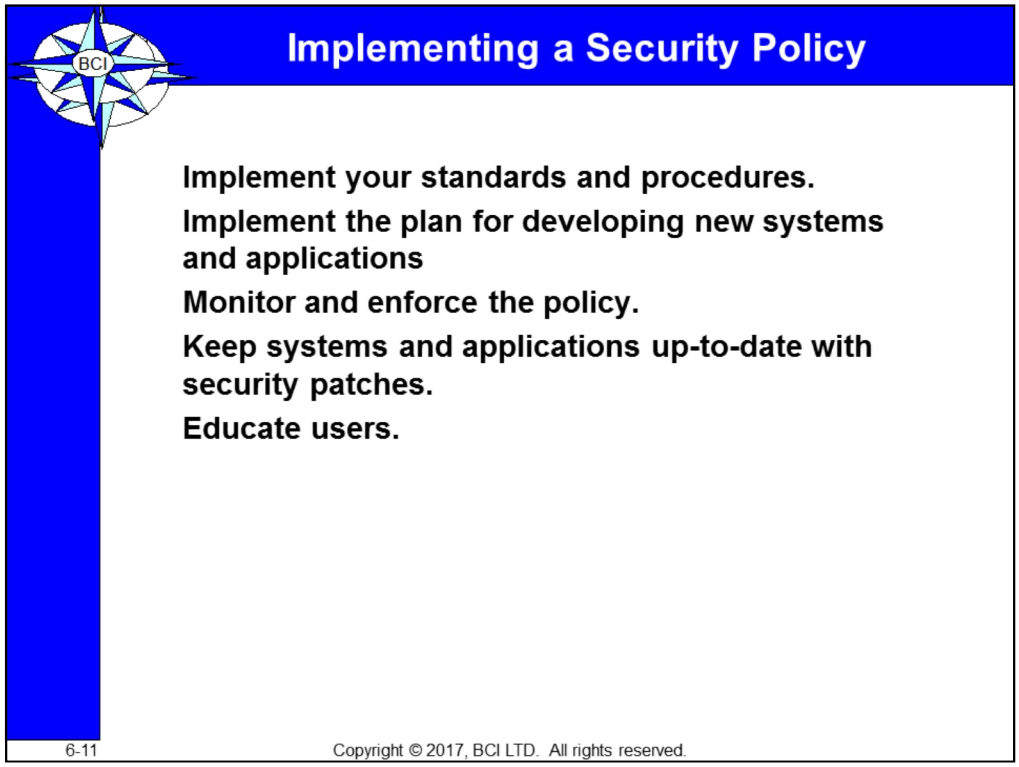
•DBMS_RANDOM
•DBMS_LOB
•DBMS_SQL
•DBMS_JOB
•DBMS_BACKUP_RESTORE
•DBMS_OBFUSCATION_TOOLKIT

Always remember that, by default, these packages are available to absolutely anyone who has a logon to your database and furthermore, that your database may have a number of well-known accounts with well-known passwords.



The following table discusses the Critical Instance Parameters that must be set in accordance with the Database STIG, V7R2, Section B.18:

<INSERT C:\Courseware\ITTC-040 Phase I Oracle 12c\Images\Mod5-6.doc>



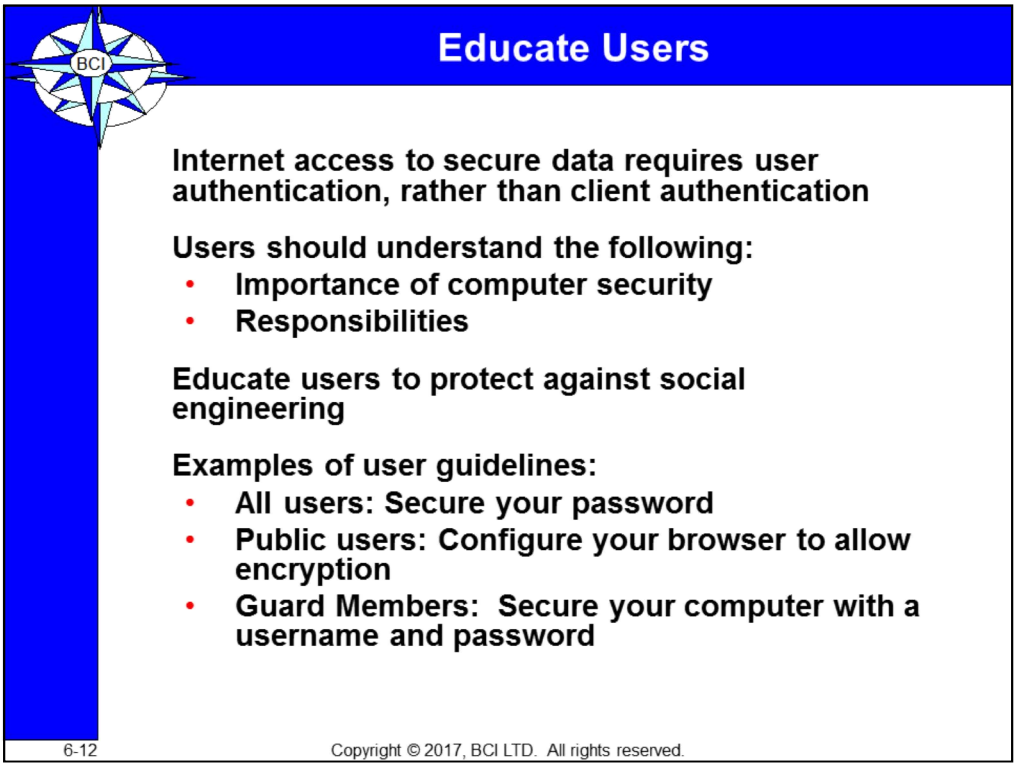
Using the most current STIG, the DBA should work closely with the SA to implement it to secure the data on a day-to-day basis. If you need to add systems or develop new applications to complete your security policy, immediately implement as much of the policy as you can without these systems. Then, as you complete the new systems and applications, revise your security policy to include them.

Monitoring and enforcing security

Be sure that all employees understand the significance of keeping your organization secure. The employees should understand the security issues associated with their function in the organization. Employees also need to be aware of how they might be disciplined if they breach the security policy.

Apply security patches

The SA and DBA need to search for and apply security patches on a periodic basis. As part of the security policy, include a schedule to search for and apply new security patches.



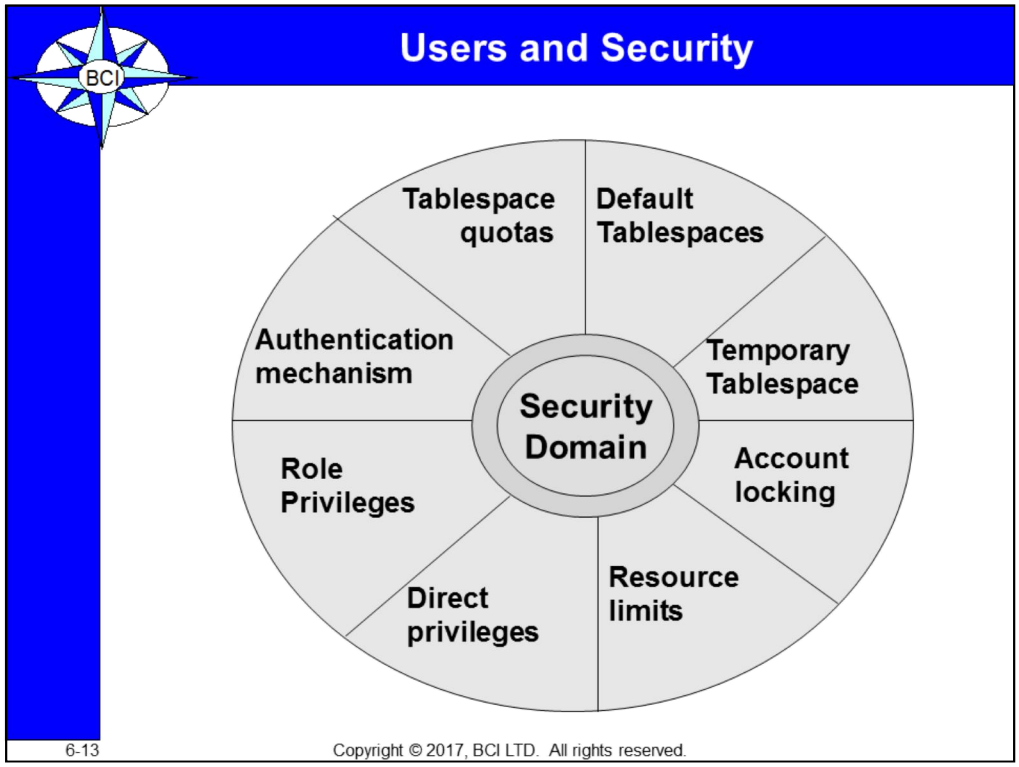
Make sure that the user understands the security-related responsibilities.

Company personnel guidelines should include employee responsibilities and consequences when the employee does not meet these responsibilities.

User education is more difficult in the Internet environment. In this environment, the customer may enter sensitive data, which is commonly protected with a username and password. To emphasize the importance of security in this environment, disclose your security requirements to the customer and have them accept these requirements.

Social Engineering

One technique for compromising security is social engineering. For example, the user gets a call from a hacker, who identifies himself as a support technician. The hacker asks the user to help him or her solve a security problem by giving the user's password. To keep the user from compromising his or her password, instruct the user that support will never ask them for their password.



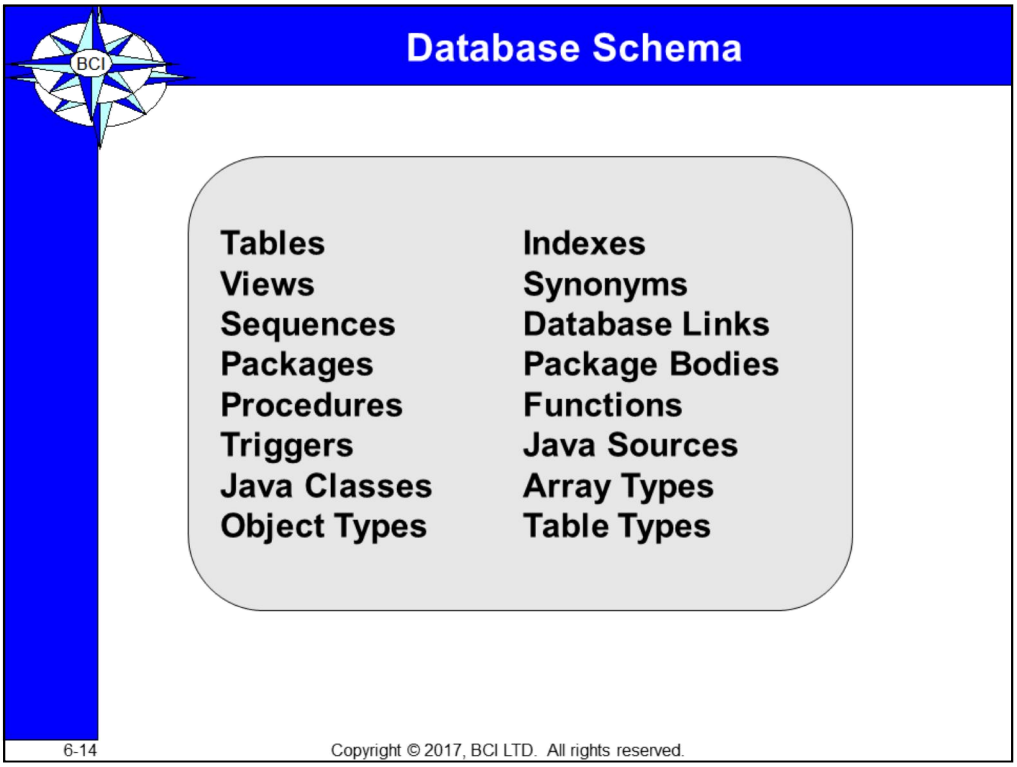
As a database administrator, you must ensure that individuals that need access to the database have the appropriate level of permission. This means creating users in the database to allow individuals to connect to the instance, as well as granting the appropriate system permissions to let users create and manage objects. Once a user creates an object, he can then grant others permissions to manipulate those objects; the DBA need not be involved in managing permissions to an individual user's objects. Finally, you want to ensure that no user can consume all database resources when issuing an errant SQL statement or by other means.

In order for anyone to be able to access the database, that person needs to be authenticated by Oracle as a valid database user. Applications can be configured to require each individual needing access to have a separate database account, or the application itself can connect to the database as a common user and handle application-level permissions internally. No matter which method is chosen, at least one database user will need to be created in the database to allow data manipulation.

Security for database users includes:

•Using Default Tablespaces
•Assigning Temporary Tablespaces
•Account Locking capabilities
•Resource Limits
•Direct Privileges
•Role Privileges
•Authentication Mechanism
•Tablespace quotas

We will look at each of these items in detail in this module.



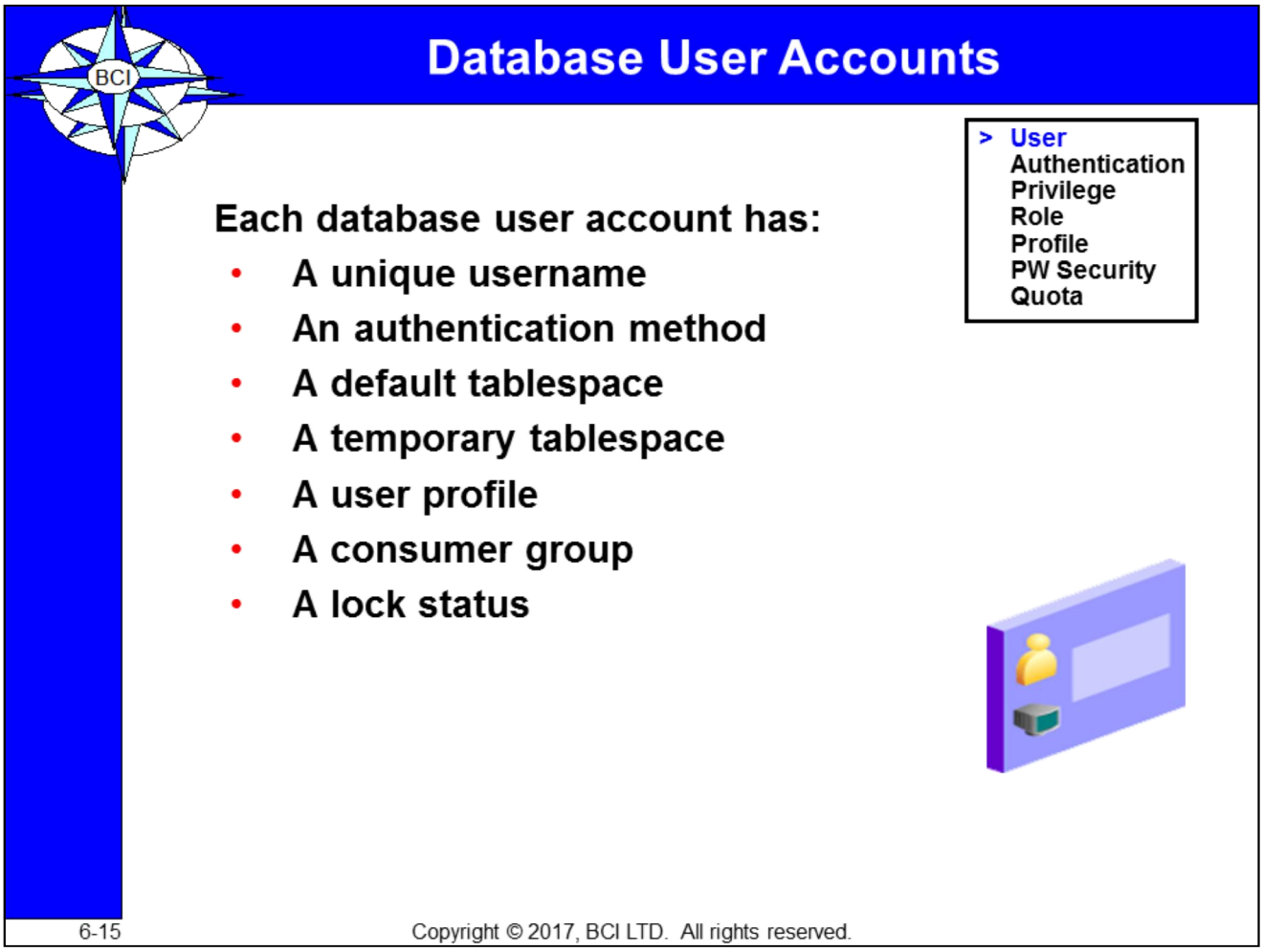
When you create a database, depending on what options were selected, a number of database users are created, but are locked by default. Two database users are always created and are always unlocked: SYS and SYSTEM. The SYS user owns the data dictionary and all of its associated objects. The user SYSTEM has access to all objects in the database. The distinction between a user owning objects and a user only having access to objects that are owned by another user is an important one in Oracle.

Any user that has been given the permission to create objects and does so is said to own a schema. The schema is a collection of all objects that are owned by a user. The schema has the same name as the user. For example, the database user called SIDPERS owns all the SIDPERS tables and is considered the schema owner of the “SIDPERS” schema. Anyone wanting to query the tables in SIDPERS schema can prefix the table name with the schema name and query the data as SIDPERS.PERS_PERSON_TBL (assuming they have the permissions to do so). If another user called Annie wants to query the SIDPERS tables she can do so if she has permissions. Annie does not have to be a schema owner like SIDPERS, just a database user. In fact, the majority of users in a database are not schema owners (i.e. they do not own database objects) but are simply users accessing other schemas.

Each schema may have any or all of the following:

•Tables
•Indexes

- Views
- Synonyms
- Sequences
- Database Links
- Packages
- Package Bodies
- Procedures
- Functions
- Triggers
- Java Sources
- Java Classes
- Array Types
- Object Types
- Table Types



Database User Accounts

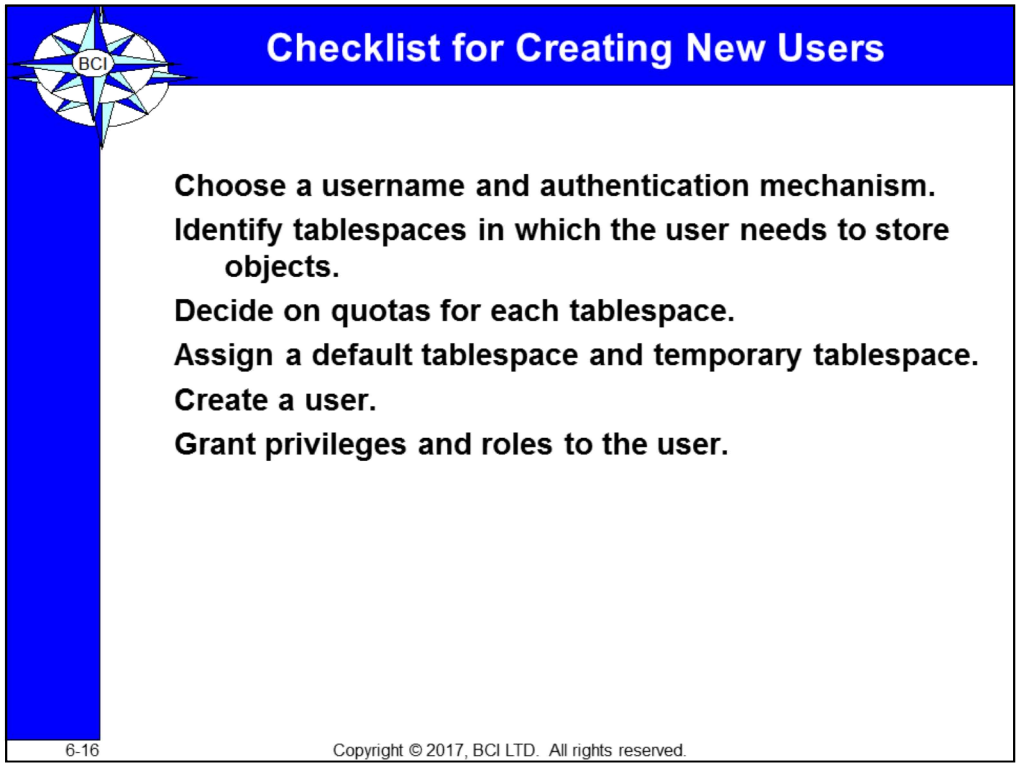
To access the database, a user must specify a valid database user account and successfully authenticate as required by that user account. Each database user has his or her own database account. This is Oracle's best practice recommendation to avoid potential security holes and provide meaningful data for certain audit activities. However, in rare cases, users share a common database account. In this case, operating system and applications must provide adequate security for the database. Each user account has:

A unique username: Usernames cannot exceed 30 bytes, cannot contain special characters, and must start with a letter.

An authentication method: The most common authentication method is a password, but Oracle Database 12c supports several other authentication methods, including biometric, certificate, and token authentication.

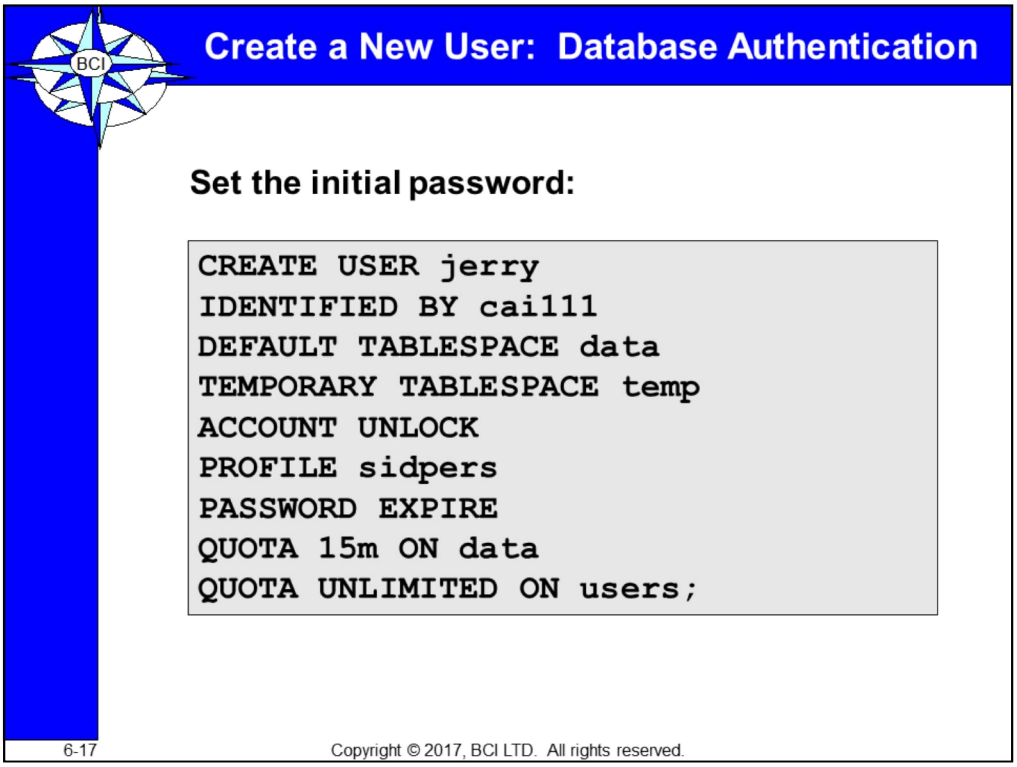
A default tablespace: This is a place where the user creates objects if he or she does not specify some other tablespace. Note that having a default tablespace does not imply that the user has the *privilege* of creating

objects in that tablespace, nor does the user have a *quota* of space within that tablespace in which to create objects. Both these are granted separately.



You should consider using the following checklist for creating users. This information will help you to create the user.

1. Choose a username and authentication mechanism.
2. Identify tablespaces in which the user needs to store objects.
    1. If necessary, create tablespaces specifically for the user.
3. Decide on quotas for each tablespace.
4. Assign a default tablespace and a temporary tablespace.
5. Create the user.
6. Grant privileges and roles to the users.
7. Assign a profile to the user if not using the default profile.

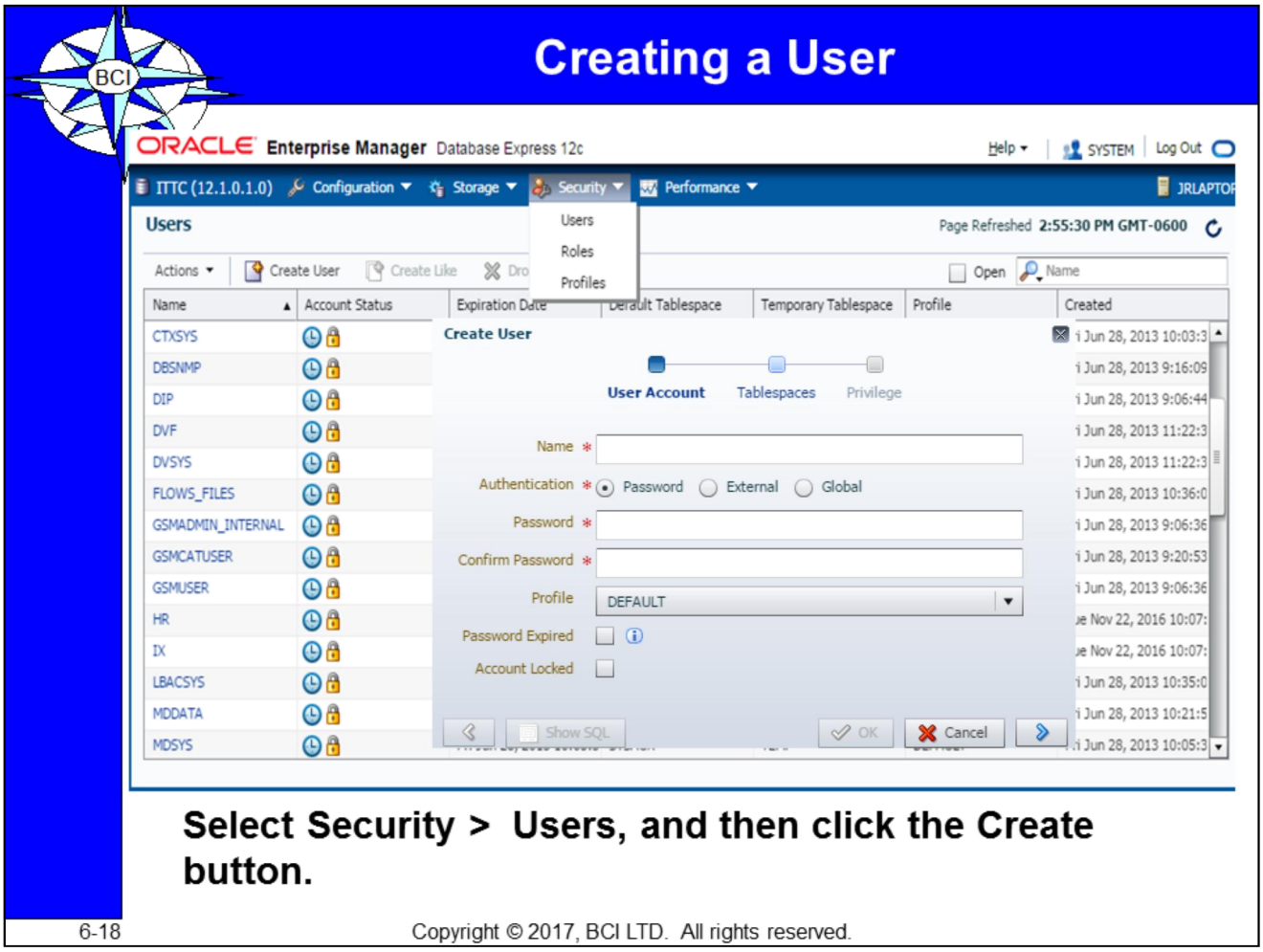


Gaining access to an Oracle database requires that you have a user account in the database, which is created for you by the DBA. The DBA is an Oracle user who has been granted all permissions in the database because he owns the database. The SYS in Oracle has that power and has also granted it to the SYSTEM user by granting SYSTEM the DBA role (we'll discuss roles shortly). SYS can also grant this authority to other Oracle users, so that not everyone needs to know the SYS or SYSTEM password. The password for the SYS user if manual database creation is used is by default "change_on_install", and for the SYSTEM, it is "manager". These are WELL KNOWN passwords. If they were not changed upon creation of the database, they must be changed as soon as possible.

The command that creates a user in Oracle is the CREATE USER command. The syntax for this command is listed here, with the table following the syntax providing additional information on the various parameters for the CREATE USER command:

```
CREATE USER username
IDENTIFIED [BY password | EXTERNALLY | GLOBALLY AS extname]
[DEFAULT TABLESPACE tablespacename]
[TEMPORARY TABLESPACE tablespacename]
[ACCOUNT LOCK | UNLOCK]
[PROFILE profilename | DEFAULT]
[PASSWORD EXPIRE]
[QUOTA num [K|M] | UNLIMITED ON tablespace]
[QUOTA num [K|M] ] UNLIMITED ON tablespace] …]
```

You can also use Oracle Enterprise Manager to create a user, as well as see the syntax to perform the action. Enterprise Manager also allows the creation of users with similar settings as an already-existing database user.



Creating a User

In Enterprise Manager, you can manage the list of database users, who are allowed to access the current database, by using the Users page. You can use this page to create, delete, and modify the settings of a user.

To create a database user, perform the following steps:

1. In Enterprise Manager Database Express, select > Security > Users.
2. Click the Create button.

Provide the required information. Mandatory items, such as Name, are marked with a star.

The following pages give you more information about authentication. Profiles are covered later in this lesson.

Assign a default tablespace and a temporary tablespace to each user. This allows you to control where their objects are created, if users do not specify a tablespace in the creation of an object.

If you do not choose a default tablespace, then the system-defined default

permanent tablespace is used. Similarly for the temporary tablespace: if you do not specify one, then the system-defined temporary tablespace is used.

Authenticating Users

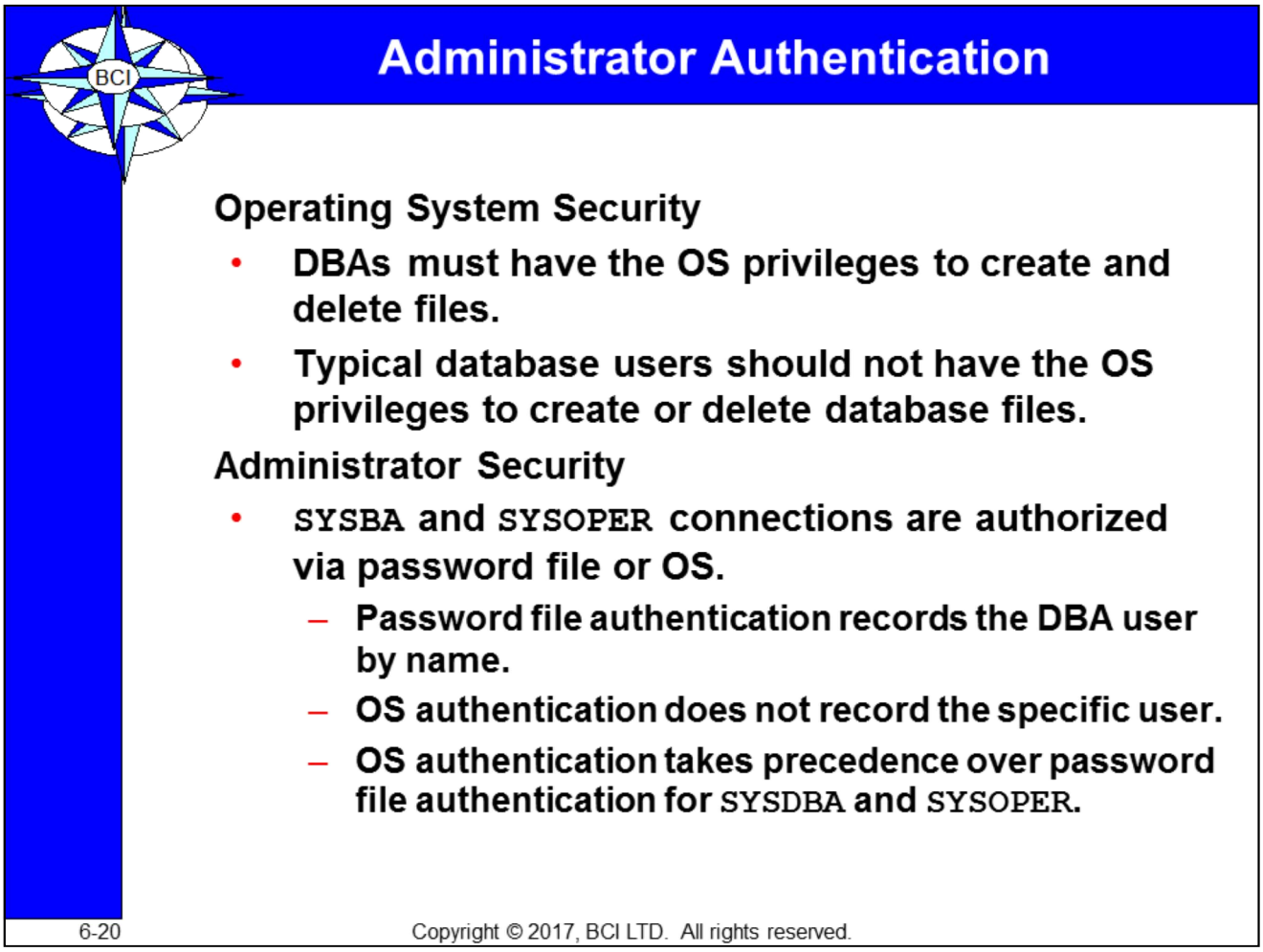
Authentication means verifying the identity of someone (a user, device, or other entity) who wants to use data, resources, or applications. Validating that identity establishes a trust relationship for further interactions. Authentication also enables accountability by making it possible to link access and actions to specific identities. After authentication, authorization processes can allow or limit the levels of access and action permitted to that entity.

When you create a user, you must decide on the authentication technique to use, which can be modified later.

Password: This is also referred to as authentication by the Oracle database. Create each user with an associated password that must be supplied when the user attempts to establish a connection. When setting up a password, you can expire the password immediately, which forces the user to change the password after first logging in. If you decide on expiring user passwords, make sure that users have the ability to change the password. Some applications do not have this functionality.

Passwords are always automatically and transparently encrypted during network

(client/server and server/server) connections, by using a modified Data Encryption Standard (DES) algorithm, before sending them across the network.



Administrator Authentication

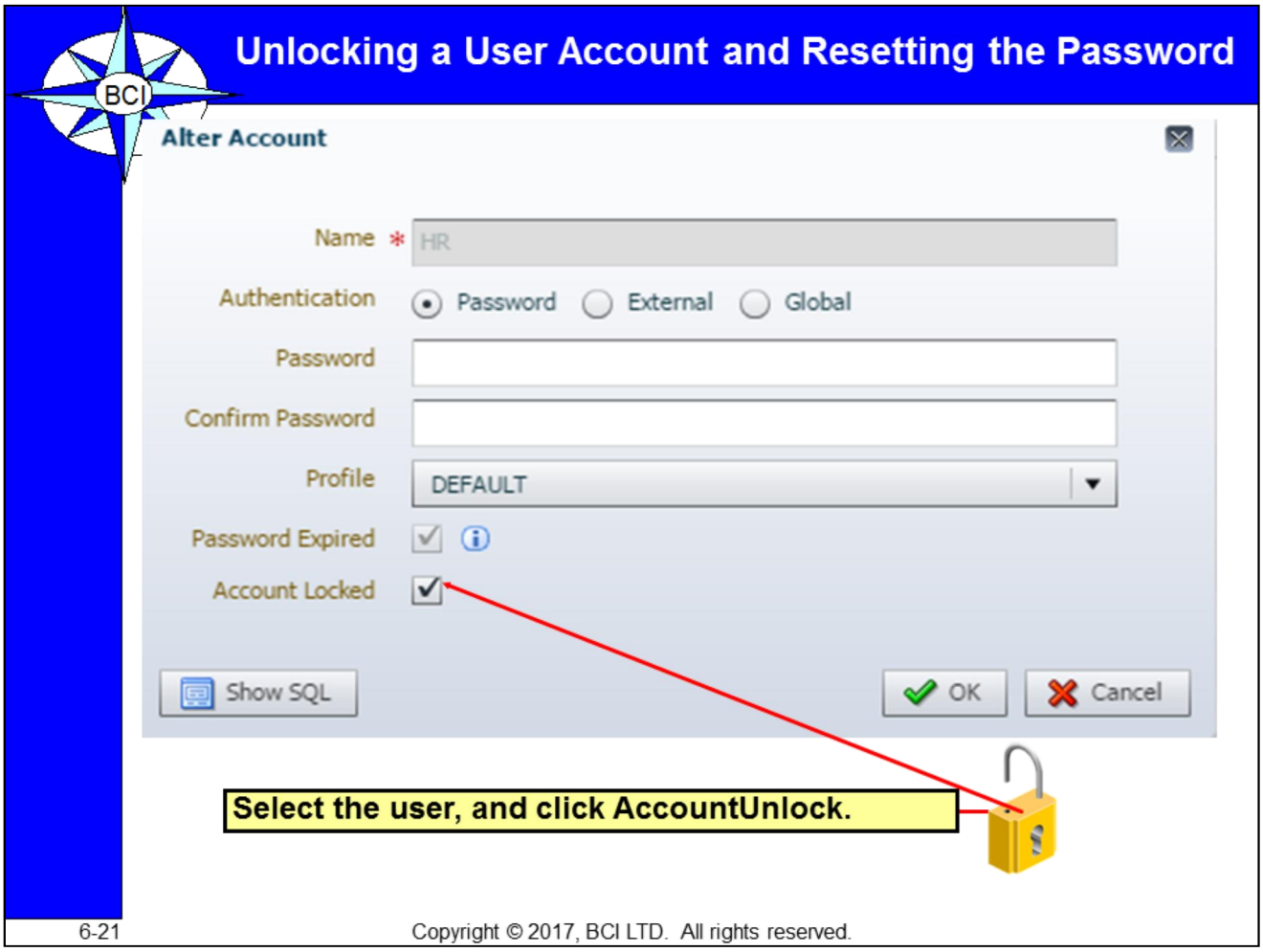
Operating System Security: In UNIX and Linux, by default, DBAs belong to the `install` OS group, which has the required privileges to create and delete database files.

Administrator Security: `SYSBA` and `SYSOPER` connections are authorized only after verification with the password file or with the operating system privileges and permissions. If operating system authentication is used, then the database does *not* use the supplied username and password. Operating system authentication is used if there is no password file, if the supplied username or password is not in that file, or if no username and password is supplied.

However, if authentication succeeds by means of the password file, then the connection is logged with the username. If authentication succeeds by means of the operating system, then it is a `CONNECT /` connection that does not record the specific user.

Note: OS authentication takes precedence over password file authentication. Specifically, if you are a member of the `OSDBA` or `OSOPER` group for the operating system, and you connect as `SYSDBA` or `SYSOPER`, you will be

connected with the associated administrative privileges regardless of the username and password that you specify.



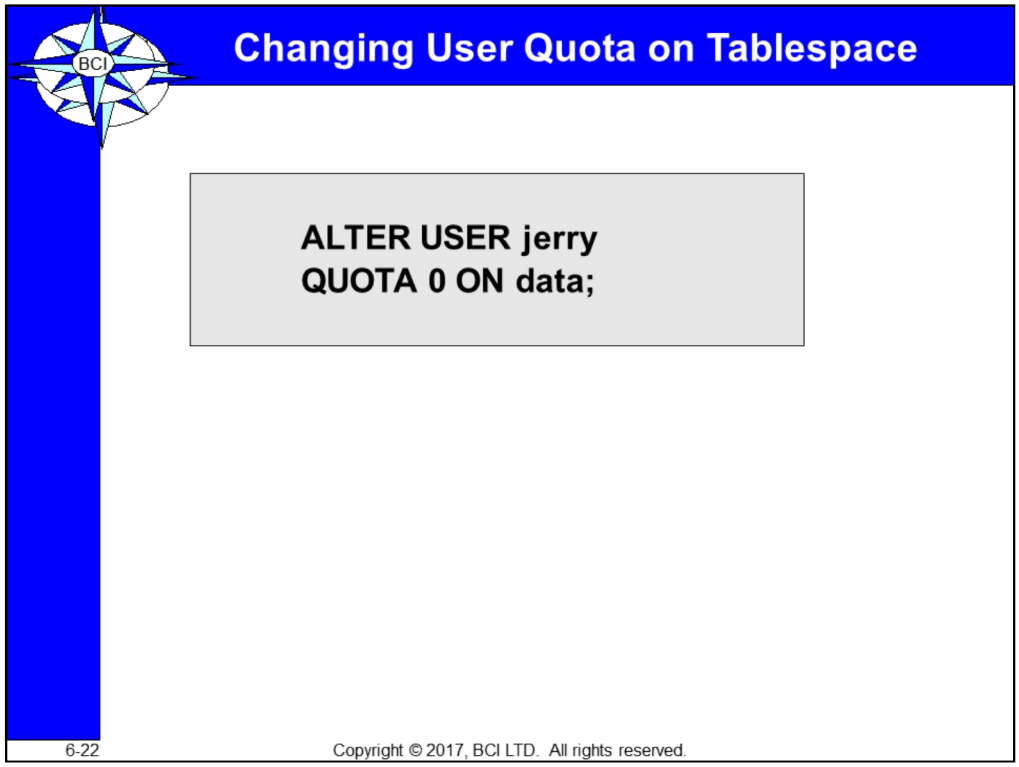
Unlocking a User Account and Resetting the Password

During installation and database creation, you can unlock and reset many of the Oracle-supplied database user accounts. If you have not chosen to unlock the user accounts at that time, you can unlock the users and reset the passwords by selecting the user on the Users page and clicking Account Unlock.

Alternatively, if you are on the Edit Users page, perform the following steps:

1. Enter the new password in the Enter Password and Confirm Password fields.
2. Select the Unlocked check box.
3. Click Apply to reset the password and unlock the user account.

4. Command line execute SQL> alter user scott account unlock;



While it is possible to have users never change anything for the duration of their existence in the database, this is probably not a good security practice, nor is it likely. Users may forget their password, or you may need to specify a different tablespace as the user's default tablespace, or grant new quotas or increased quotas for the user's objects, or lock out a user temporarily. The ALTER USER command can be used to accomplish these tasks. Its syntax is as follows:

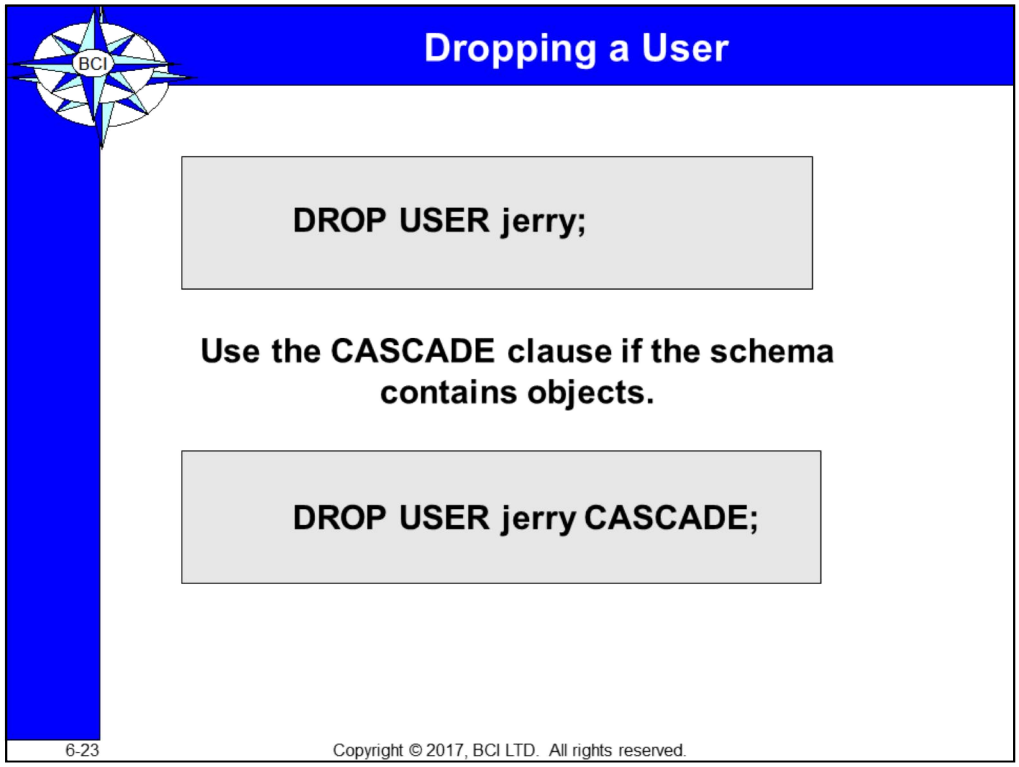
```
ALTER USER username
IDENTIFIED [BY password | EXTERNALLY | GLOBALLY AS extname]
[DEFAULT TABLESPACE tablespacename]
[TEMPORARY TABLESPACE tablespacename]
[ACCOUNT LOCK | UNLOCK]
[PROFILE profilename]
[PASSWORD EXPIRE]
[QUOTA num [K|M] | UNLIMITED ON tablespace
[QUOTA num [K|M] | UNLIMTED ON tablespace] … ]
```

All user properties with the exception of the username, can be modified using the ALTER USER command. The database user can also use this command to change their password:

```
ALTER USER Annie IDENTIFIED BY newpassword;
```

Note that the password itself is not masked when the command is issued. If you want the password masked, you can use the Edit User page in Enterprise Manager to change a user's password, though this is not a method the user himself could use. Enterprise Manager can also be used to perform any other action in modifying a user, such as assigning a quota.

If you change the user's QUOTA to 0 on a tablespace, that means that the user cannot create any more objects in that particular tablespace, although any previously created objects that the user created are still in the tablespace and available.



If a user owns objects in the database, his account cannot be dropped, because other users may depend on the objects in that schema. The syntax of the DROP USER command is as follows:

DROP USER username [CASCADE];

Oracle prevents you from dropping a user from the database whose schema contains objects. This is to ensure that objects created by one user and depended upon by another user's objects are not inadvertently removed from the database. Since the user and the schema are linked, dropping the user will also drop the schema. Oracle does not allow you to drop both the user and the schema, unless you specify the CASCADE option on the DROP USER command.

Using the CASCADE option will drop all objects, as well as any data contained in tables, that the user owns (i.e., that are in the user's schema). This can have drastic side effects in the database if not planned properly. It is always recommended that before dropping a user, you determine if the user owns any objects in the database, and if so, drop the objects after verifying that they are not depended upon by other users.

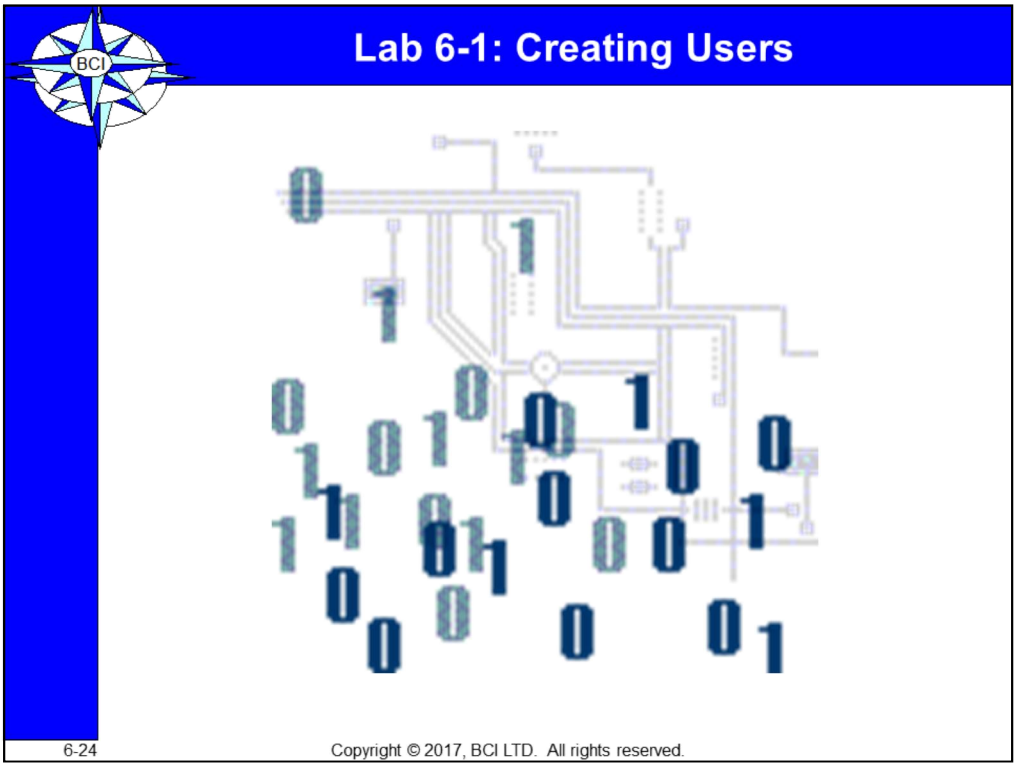
If the user that owns the objects has left your site, you can always revoke the CREATE SESSION privilege and lock their account and expire their password, leaving their privileges intact. Other users can still access the tables in the user's schema.

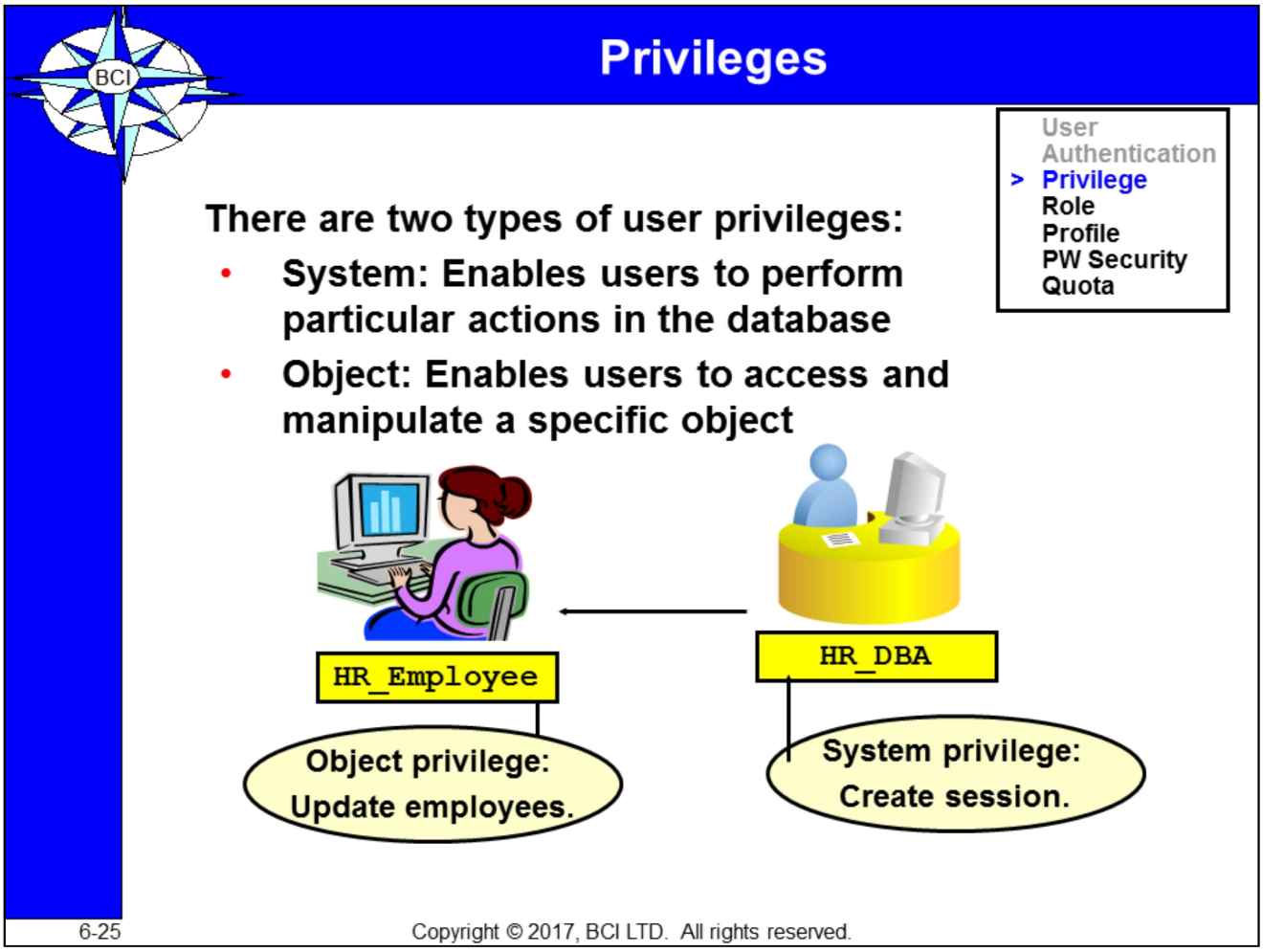
On the other hand, you could remove any privileges given to the user, including CREATE SESSION, have another DBA grant the appropriate privileges on the user's schema to others, then expire and lock the user's account.

To find out which objects are owned by a user you wish to drop, you can query the DBA_OBJECTS view. If you get zero rows returned, you can safely drop the user.

```
SELECT OBJECT NAME, OBJECT_TYPE from DBA_OBJECTS
WHERE OWNER = 'USERNAME';
```

On the other hand, if you do get a list of objects back, you should also query the DBA_DEPENDENCIES view to determine which objects are dependent on those owned by the user you wish to drop.


BCI
Lab 6-1: Creating Users
6-24
Copyright © 2017, BCI LTD. All rights reserved.

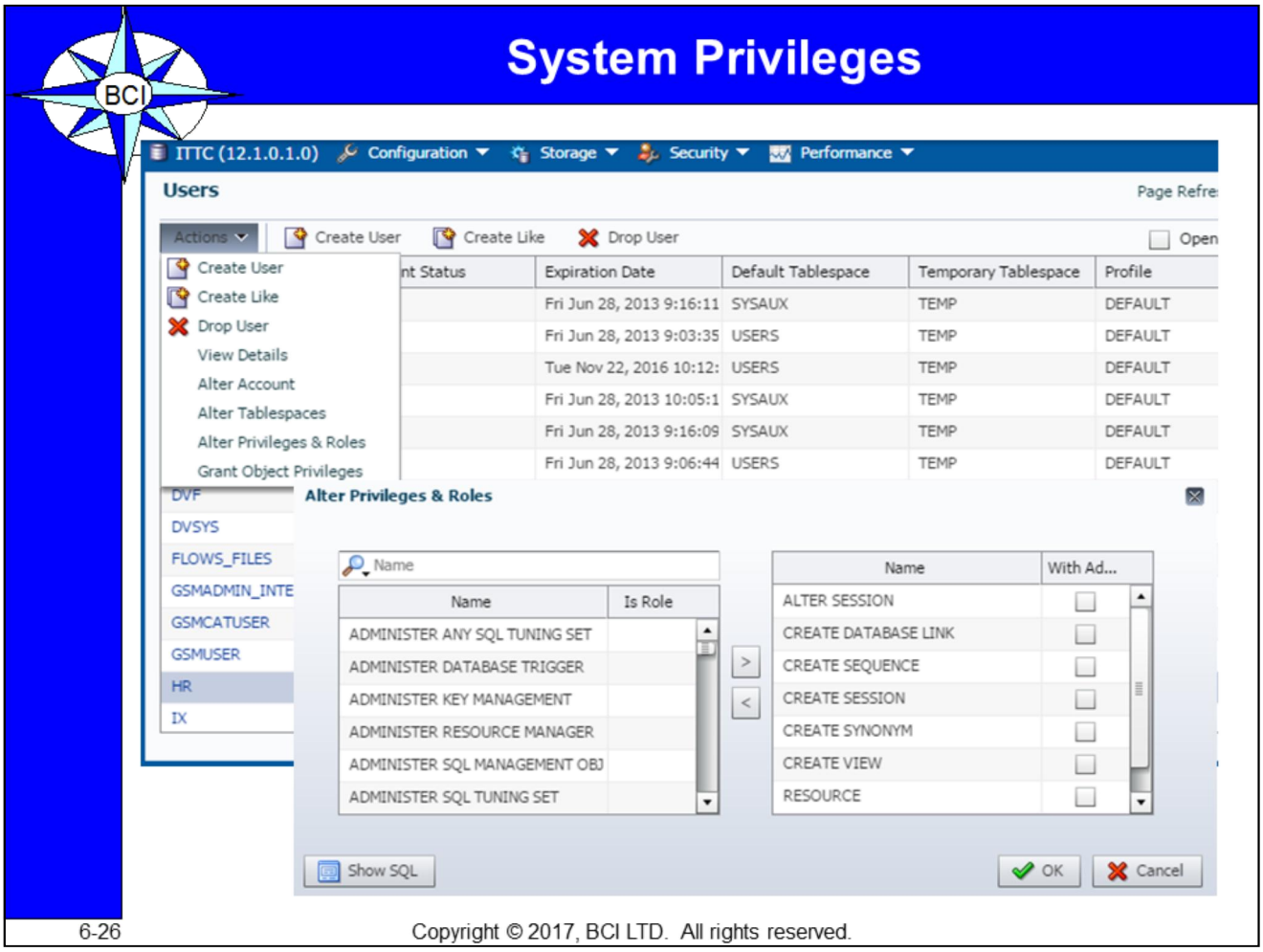


Privileges

A privilege is a right to execute a particular type of SQL statement or to access another user's object. The Oracle database enables you to control what users can or cannot do within the database. Privileges are divided into two categories:

System privileges: Each system privilege allows a user to perform a particular database operation or class of database operations. For example, the privilege to create tablespaces is a system privilege. System privileges can be granted by the administrator or by someone who explicitly gives permission to administer the privilege. There are more than a hundred distinct system privileges. Many system privileges contain the `ANY` clause.

Object privileges: Object privileges allow a user to perform a particular action on a specific object, such as a table, view, sequence, procedure, function, or package. Without specific permission, users can access only their own objects. Object privileges can be granted by the owner of an object, by the administrator, or by someone who has been explicitly given permission to grant privileges on the object.



System Privileges

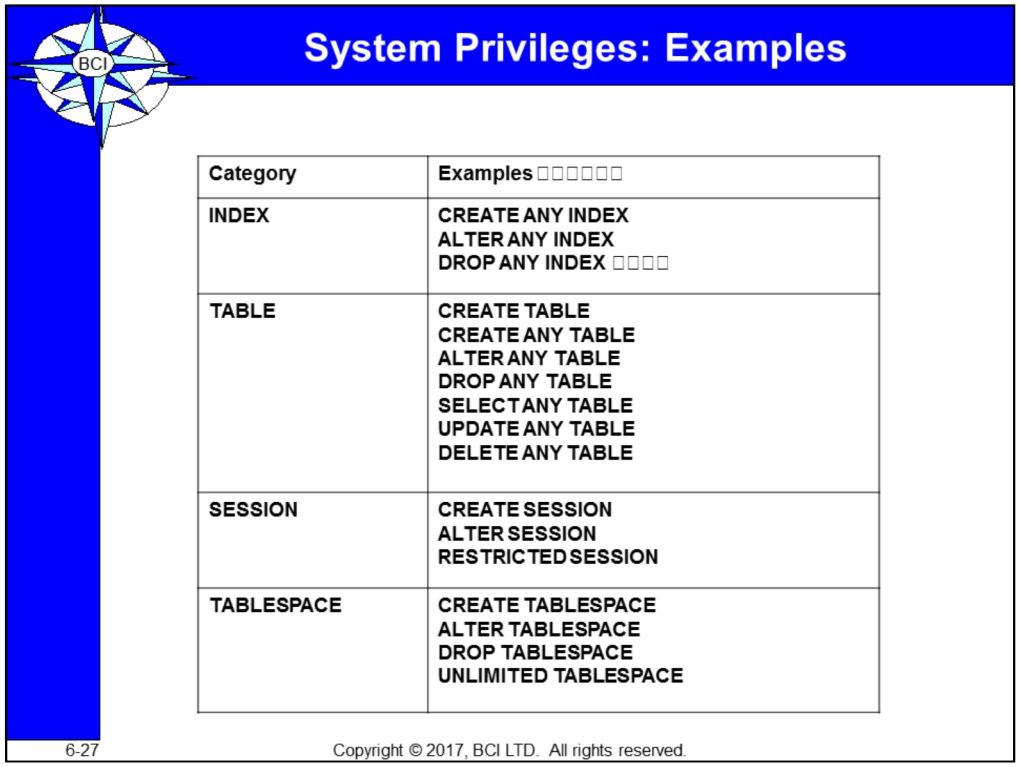
To grant system privileges, click the Alter Privileges & Roles tab on the User page. Select the appropriate privileges from the list of available privileges, and move them to the Selected System Privileges list by clicking the Move arrow.

Granting a privilege with the `ANY` clause means that the privilege crosses schema lines. For example, the `CREATE TABLE` privilege allows you to create a table but only within your own schema. The `SELECT ANY TABLE` privilege allows you to select from tables owned by other users.

Selecting the Admin Option check box enables you to administer the privilege and grant the system privilege to other users.

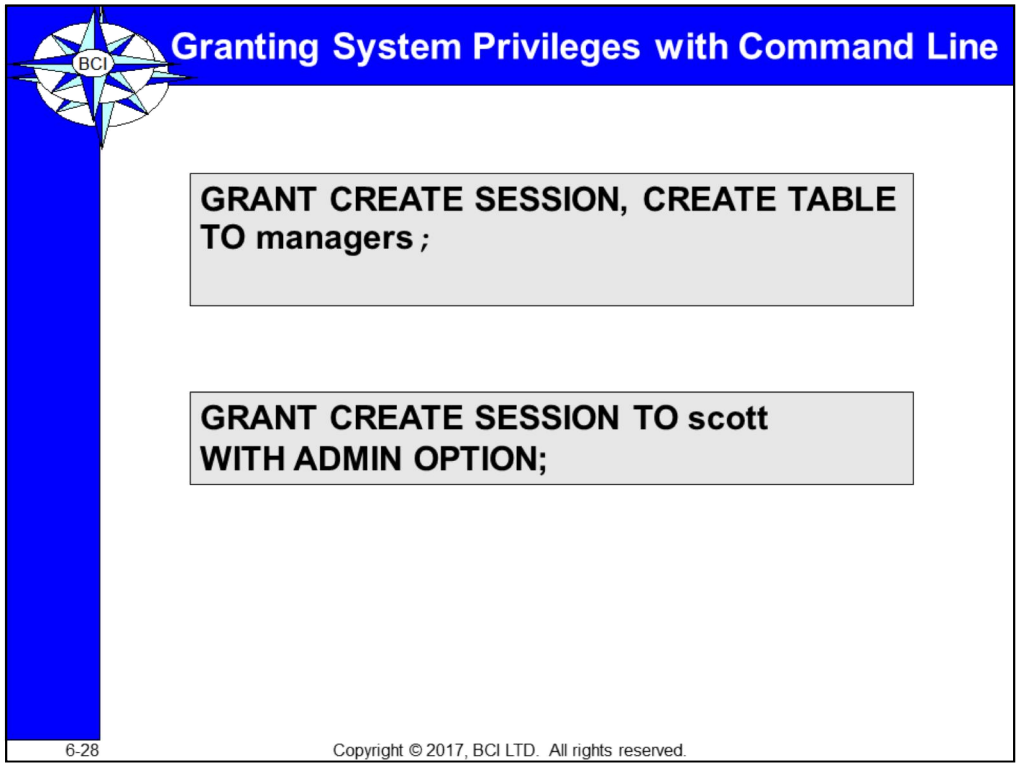
Carefully consider security requirements before granting system permissions. Some system privileges are usually granted only to administrators:

`RESTRICTED SESSION`: This privilege allows you to log in even if the database has been opened in restricted mode.



Listed below are the most commonly granted System Privileges in Oracle 12c and their description:

<INSERT C:\Courseware\ITTC-040 Phase I Oracle 12c\Images\Mod5-2.doc>



The syntax for assigning system privileges is as follows:

```
GRANT privilege [, privilege, …]
TO username [, username, …]
[WITH ADMIN OPTION];
```

As you can see, it is possible to grant multiple privileges to multiple users at the same time. The privileges granted to a user are immediately available to the user. This means that the user does not need to disconnect from the instance and log in again in order for the privilege change to take effect. Simply granting the privilege lets the user make use of it right away.

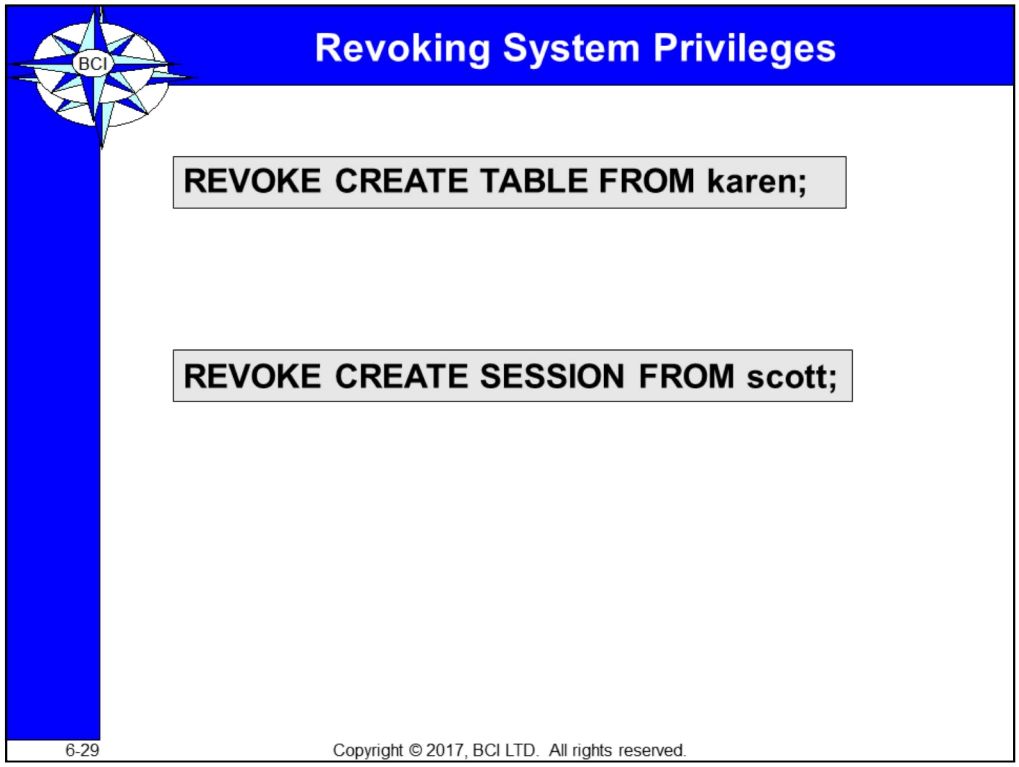
Just because a user has been granted a system privilege does not always mean that the user can exercise that privilege. For example, the user may have the create table privilege, but no quota available on the TOOLS tablespace:

```
CREATE TABLE mynewtable
(ID number,
Name varchar2(40));

CREATE TABLE mynewtable
*
```

```
ERROR at line 1:
ORA-01950: no privileges on tablespace 'TOOLS'
```

When a user is granted a system privilege, the person granting the privilege, (typically the DBA) also has the option to allow the person receiving the privilege (typically the user) to grant the same privilege to other users. If this is the result desired, the person granting the privilege (the DBA) can grant the privilege using the WITH ADMIN OPTION. When privileges are granted WITH ADMIN OPTION, this means that the grantor has decided that the grantee can be fully trusted by him as well as by the user that granted him the system privilege in the first place. In essence, all users holding a system privilege WITH ADMIN OPTION are considered equal and can grant and revoke that privilege from anyone, including the person who granted it to them in the first place.

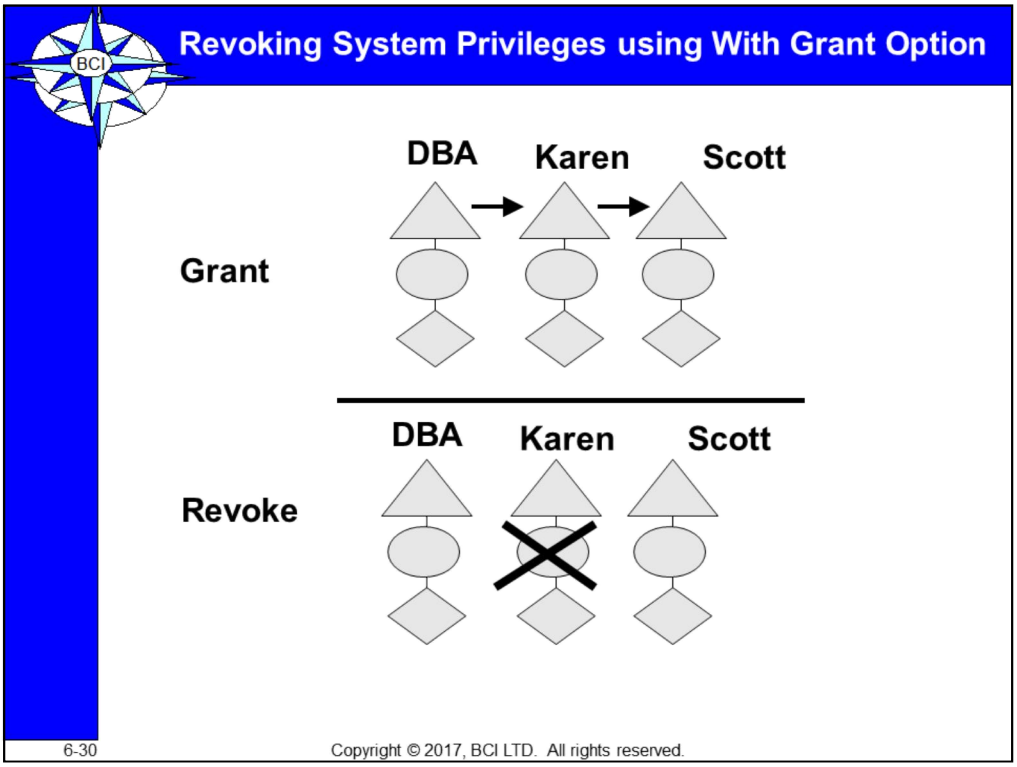


If you do not want anyone to continue to have a system privilege granted to them, you can use the REVOKE command or Enterprise Manager to remove the privileges granted. The syntax of the REVOKE command to revoke system privileges is very similar to that of granting it and can be used to revoke one or more privileges from one or more users/grantees, as follows:

REVOKE privilege [, privilege, …]

FROM username [, username, …];

It is important to note one side effect when the WITH ADMIN OPTION is specified at the time a system privilege is granted. While the DBA may revoke the privilege granted to the user WITH ADMIN OPTION, if the user has granted that same privilege to others, it is not removed from those users that were granted the privilege.

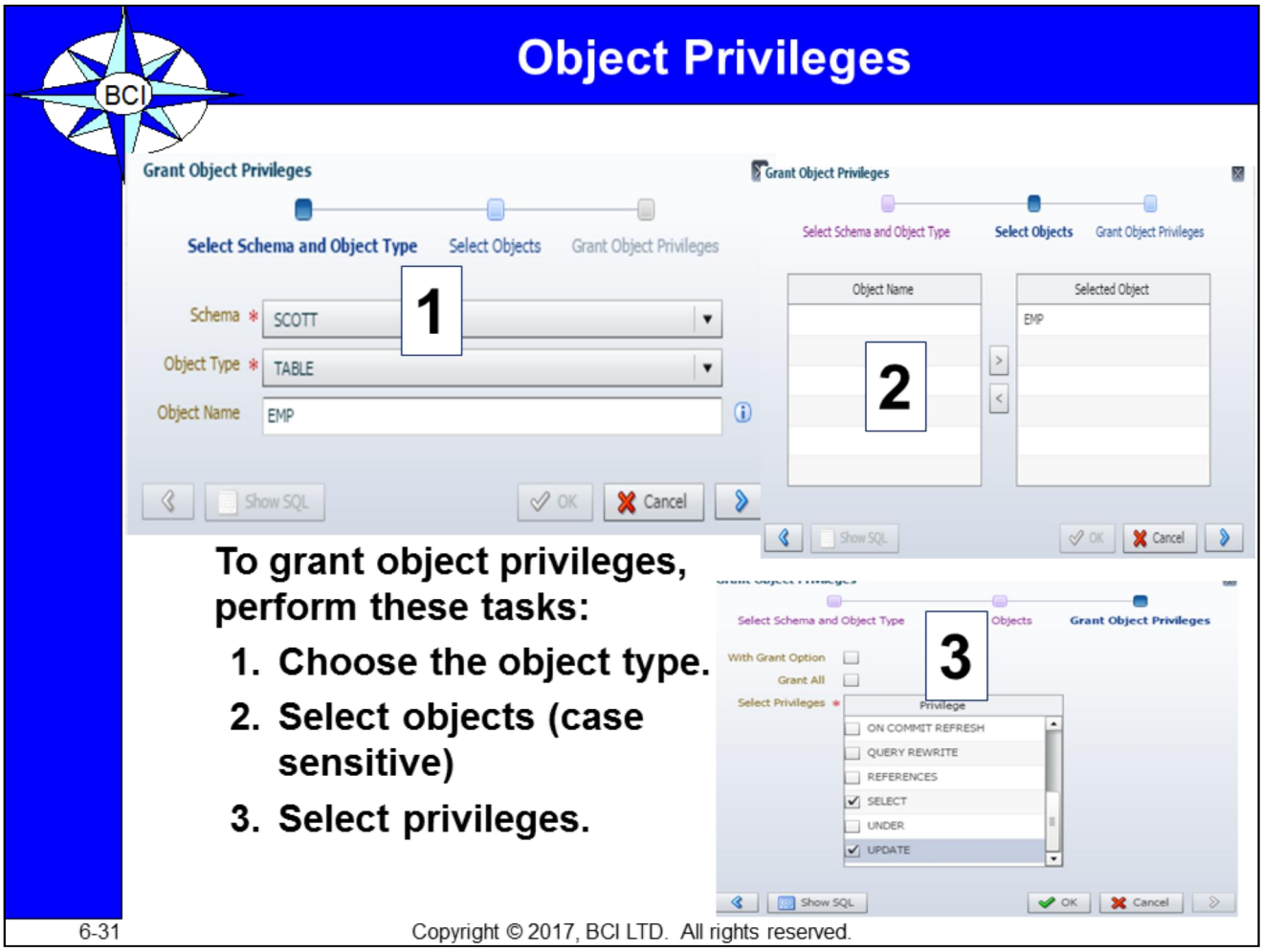


In the above example, the DBA granted a system privilege WITH ADMIN OPTION to Karen. Karen then granted the same privilege to Scott. The DBA revoked Karen's privilege; however, Scott still has the privilege.

Revoking system privileges from a user will not cascade to anyone the revokee granted the same system privilege if he had been granted the privilege WITH ADMIN OPTION.

If you want to find out which system privileges you have been granted, you can query the DBA_SYS_PRIVS (for all system privileges granted to all users) and USR_SYS_PRIVS (for system privileges granted to the currently logged-on user) data dictionary view. These views will let you see which privileges you have been granted and whether or not they have been granted WITH ADMIN OPTION. Only those system privileges that have been granted will appear on the list. Any privileges that have been revoked will not be listed, as Oracle does not keep track of permissions denied a user. The default for Oracle is to deny all actions except those explicitly granted; therefore, only those explicitly granted are listed.

To determine which privileges have been granted to individual users, you can use Enterprise Manager to display the user information, including privileges and quotas.

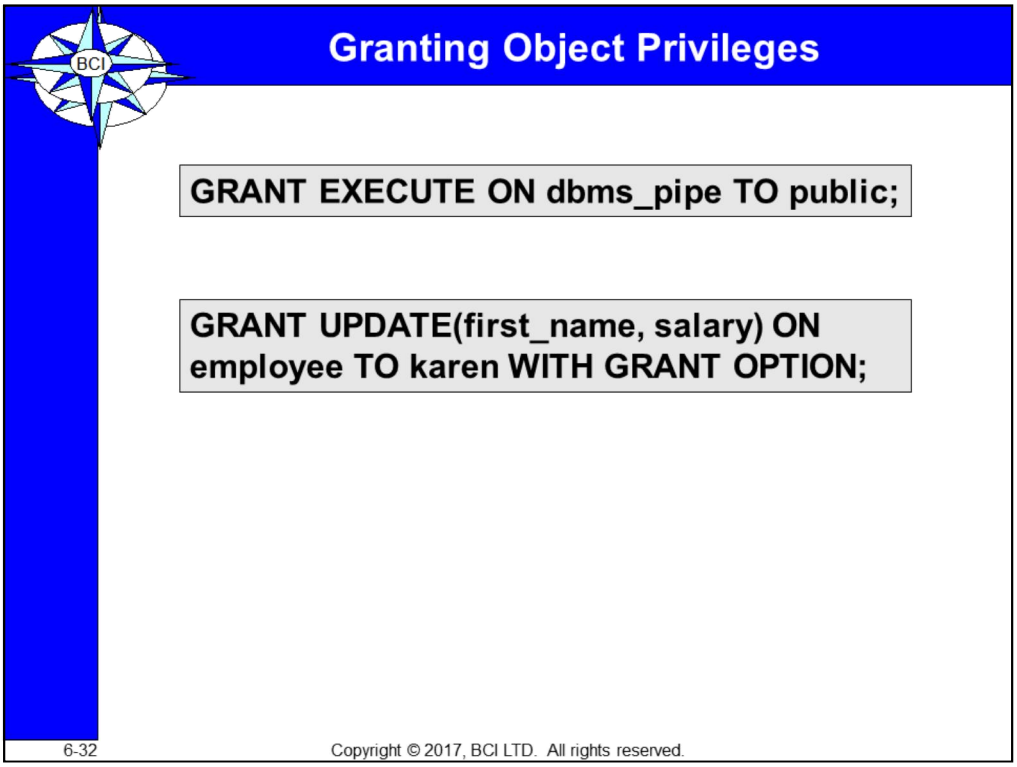


Object Privileges

To grant object privileges, click the Object Privileges tab on the Edit User page. Select the type of object you want to grant privileges on, and click the Add button. Choose the objects you want to grant privileges on by either entering *<username.object name>* or selecting them from the list.

Next, select the appropriate privileges from the Available Privileges list, and click the Move button. When you have finished selecting privileges, click OK.

Back on the Edit User page, select the Grant check box if this user is allowed to grant other users the same access.

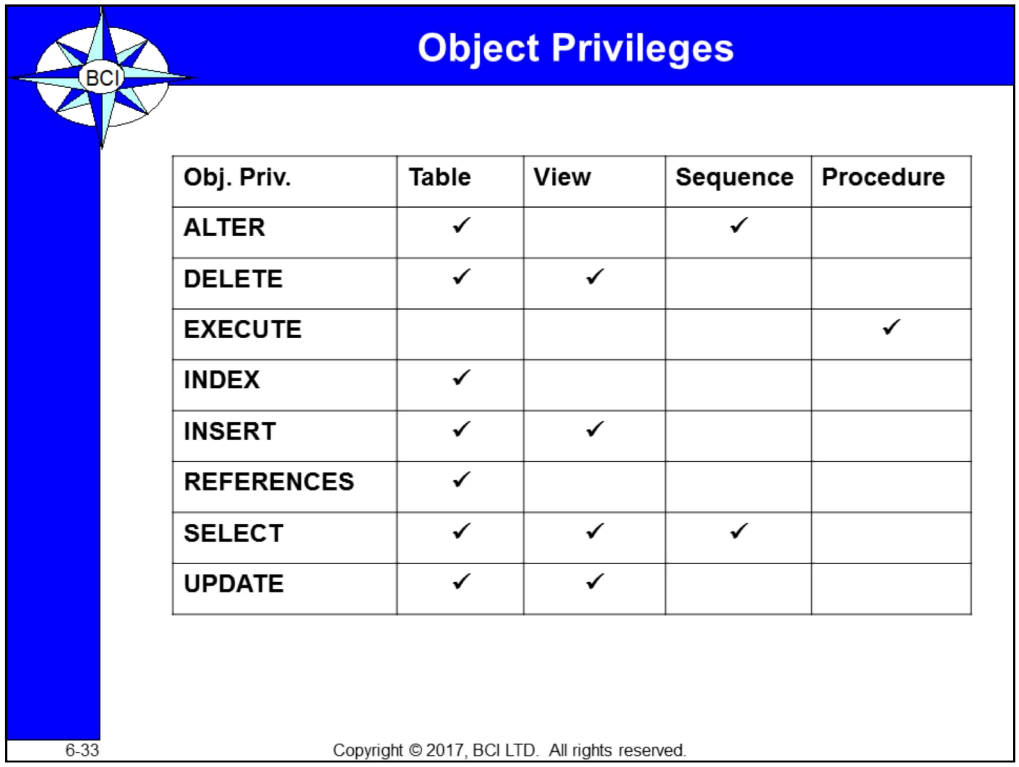


Users in Oracle can also be granted object privileges. Object privileges allow a user to manipulate data in the database or perform an action on an object, such as executing a stored procedure. Unlike system privileges, which are granted by the DBA, object privileges need to be granted by the owner of the object.

The syntax to assign object privileges is as follows:

GRANT privilege [, privilege, …] | ALL [(column[, column, …])]
ON objectname
TO user | role | PUBLIC
[WITH GRANT OPTION];

The major difference in the syntax between system and object privileges is that the keyword ON needs to be specified to determine which object the privileges apply to. Furthermore, object privileges for views and tables can also specify which column of the view or table they should be applied to. The keyword ALL specifies that all privileges that apply to the object should be granted. The privilege can be granted to a user, a role (to be discussed later), or the keyword PUBLIC, which means all users in the database.



The types of privileges that can be granted depend on the object that they are being granted on. For example, if makes no sense to grant the SELECT privilege to a stored procedure, while the SELECT privilege makes perfect sense on a table. The object privileges that can be granted and the object they can be granted to are outlined below.

<INSERT C:\Courseware\ITTC-040 Phase I Oracle 12c\Images\Mod5-3.doc>

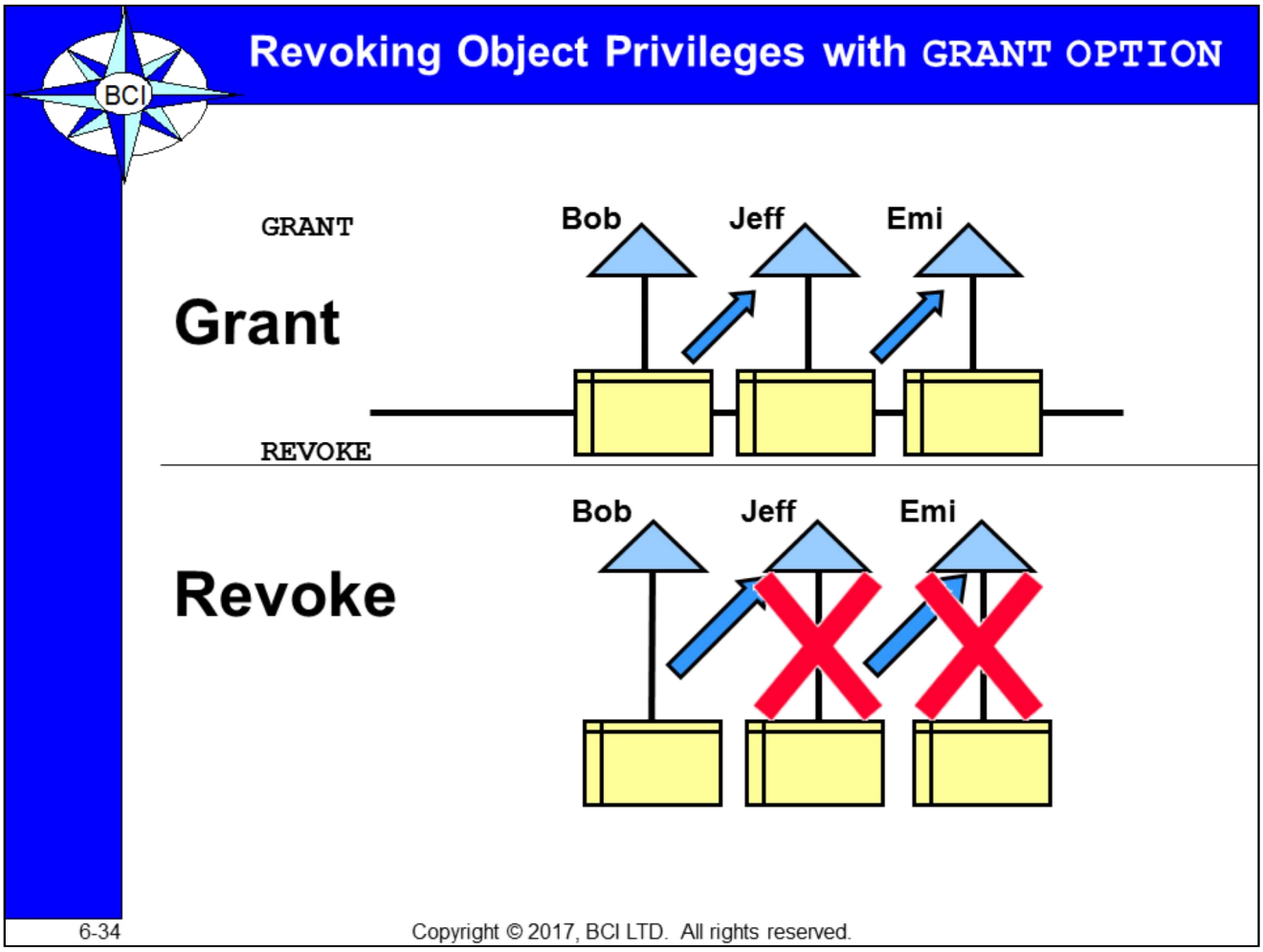
For example, you cannot issue an ALTER VIEW command, so therefore the ALTER privilege cannot apply to a view, and so on.

One privilege that needs an explanation is the REFENCES privilege. The REFERENCES privilege is one that can be granted to a user to create a FOREIGN KEY on a column of a table. By granting the user the REFERENCES privilege, you do not need to allow the user to be able to see that data, as they would with the SELECT privilege, but are only allowing them to reference the data in the FOREIGN KEY. The SELECT permission alone is not sufficient to create a FOREIGN KEY or view that references a column in the table; the REFERENCES permission is also required. Even if the user has the SELECT permission on the table, the creation of a FOREIGN KEY or view will fail without the REFERENCES permission.

IMPORTANT! It is generally not recommended, and often frowned upon, to grant object privileges at the column level. The management of many column-level

privileges can become quite time consuming, as well as confusing. Generally it is recommended that when you need to assign privileges to only certain columns of a table, you should create a view including only those columns, then grant the appropriate permission on the view itself. This way, if you drop the view or remove permission from the view for a user, the management is easier and cleaner.

Similar to the WITH ADMIN OPTION of system privileges, the WITH GRANT OPTION on object privileges allows a user granted the privilege to grant it to someone else. The reason for doing this is to minimize the administrative burden of granting object privileges.



Revoking Object Privileges

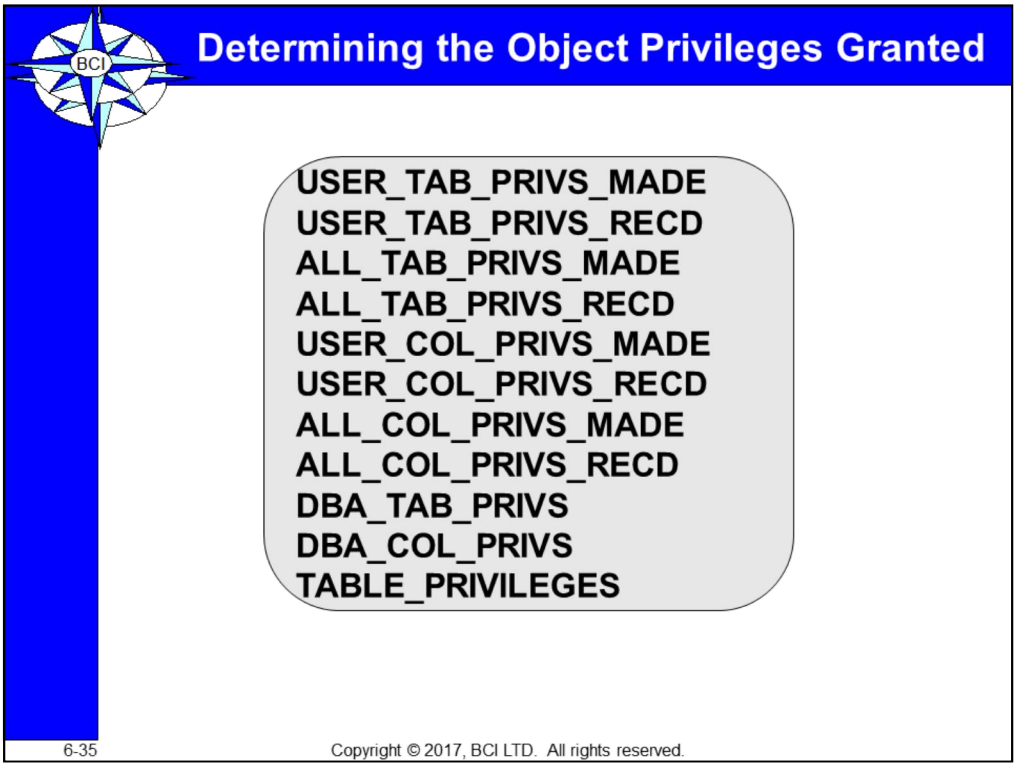
Cascading effects can be observed when revoking a system privilege that is related to a data manipulation language (DML) operation. For example, if the `SELECT ANY TABLE` privilege is granted to a user, and that user has created procedures that use the table, all procedures that are contained in the user's schema must be recompiled before they can be used again.

Revoking object privileges also cascades when given `WITH GRANT OPTION`.

Read through the following steps that illustrate this:

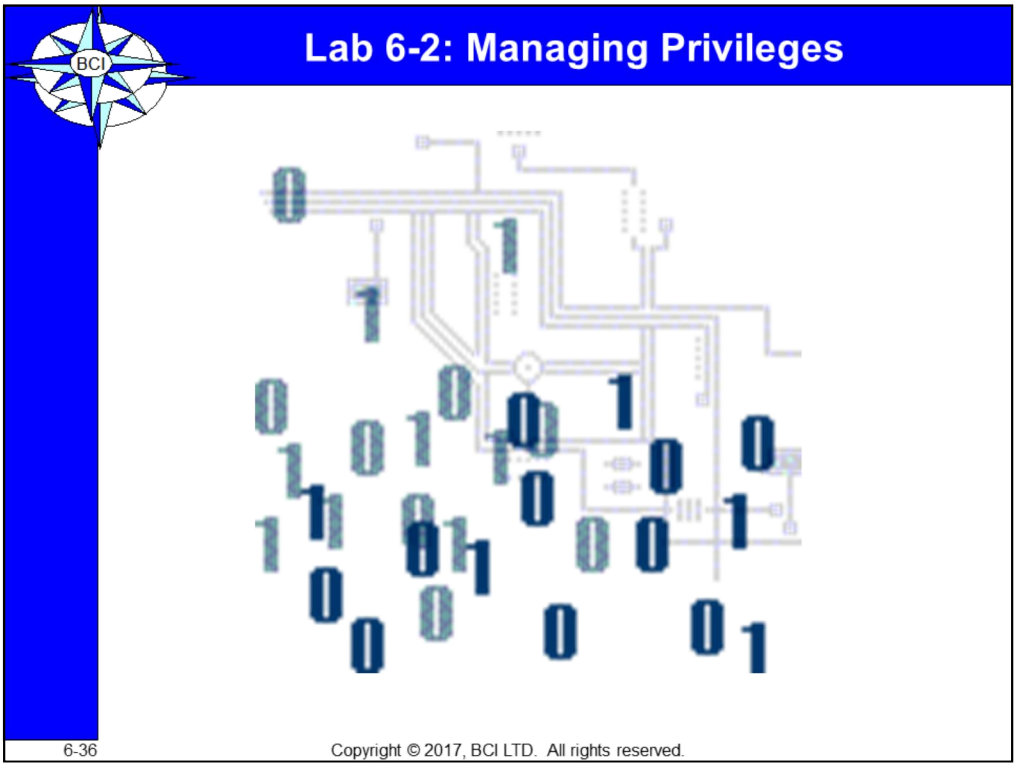
Scenario

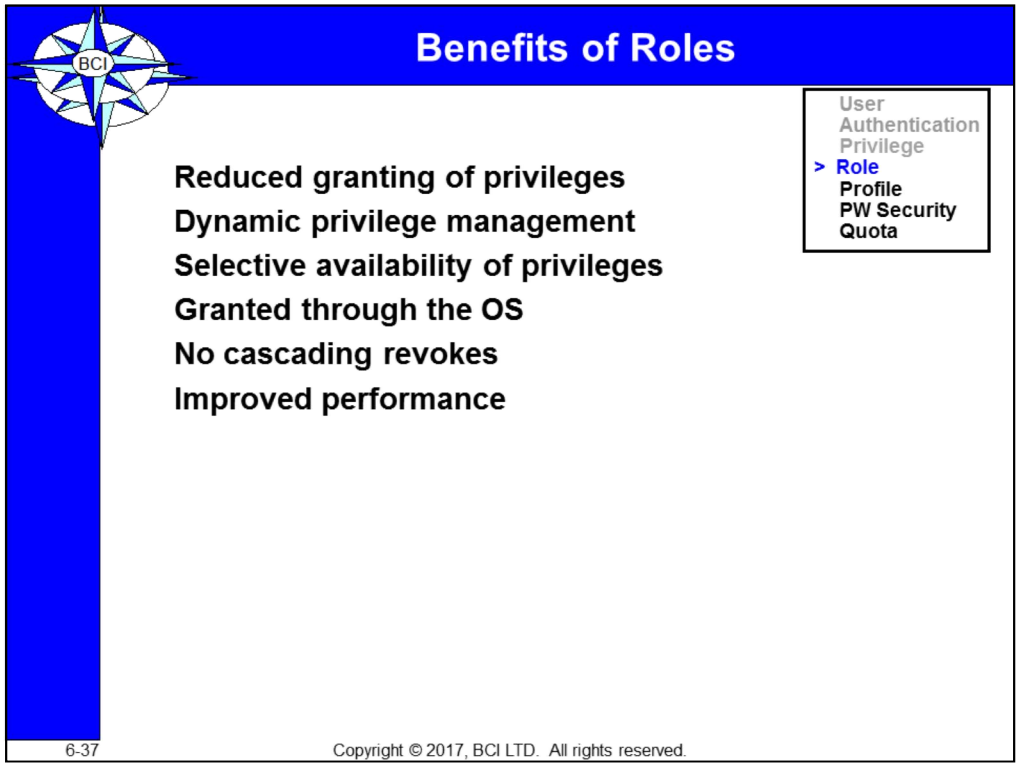
1. Jeff is granted the `SELECT` object privilege on `EMPLOYEES` with `GRANT OPTION`.
2. Jeff grants the `SELECT` privilege on `EMPLOYEES` to Emi.
3. Later, the `SELECT` privilege is revoked from Jeff. This revoke is cascaded to Emi as well.



As for the system privileges, Oracle also allows a user to determine which object privileges have been granted to him/her by querying the data dictionary, or by using Enterprise Manager. These are some of the data dictionary views that can be used to determine which privileges have been granted to a user or granted by the user:

- USER_TAB_PRIVS_MADE
- USER_TAB_PRIVS_RECD
- ALL_TAB_PRIVS_MADE
- ALL_TAB_PRIVS_RECD
- USER_COL_PRIVS_MADE
- USER_COL_PRIVS_RECD
- ALL_COL_PRIVS_MADE
ALL_COL_PRIVS_RECD
- DBA_TAB_PRIVS
- DBA_COL_PRIVS


Lab 6-2: Managing Privileges
BCI
6-36
Copyright © 2017, BCI LTD. All rights reserved.



Oracle provides a mechanism to group permissions together and then assign the whole group of permissions to a user: the ROLE.

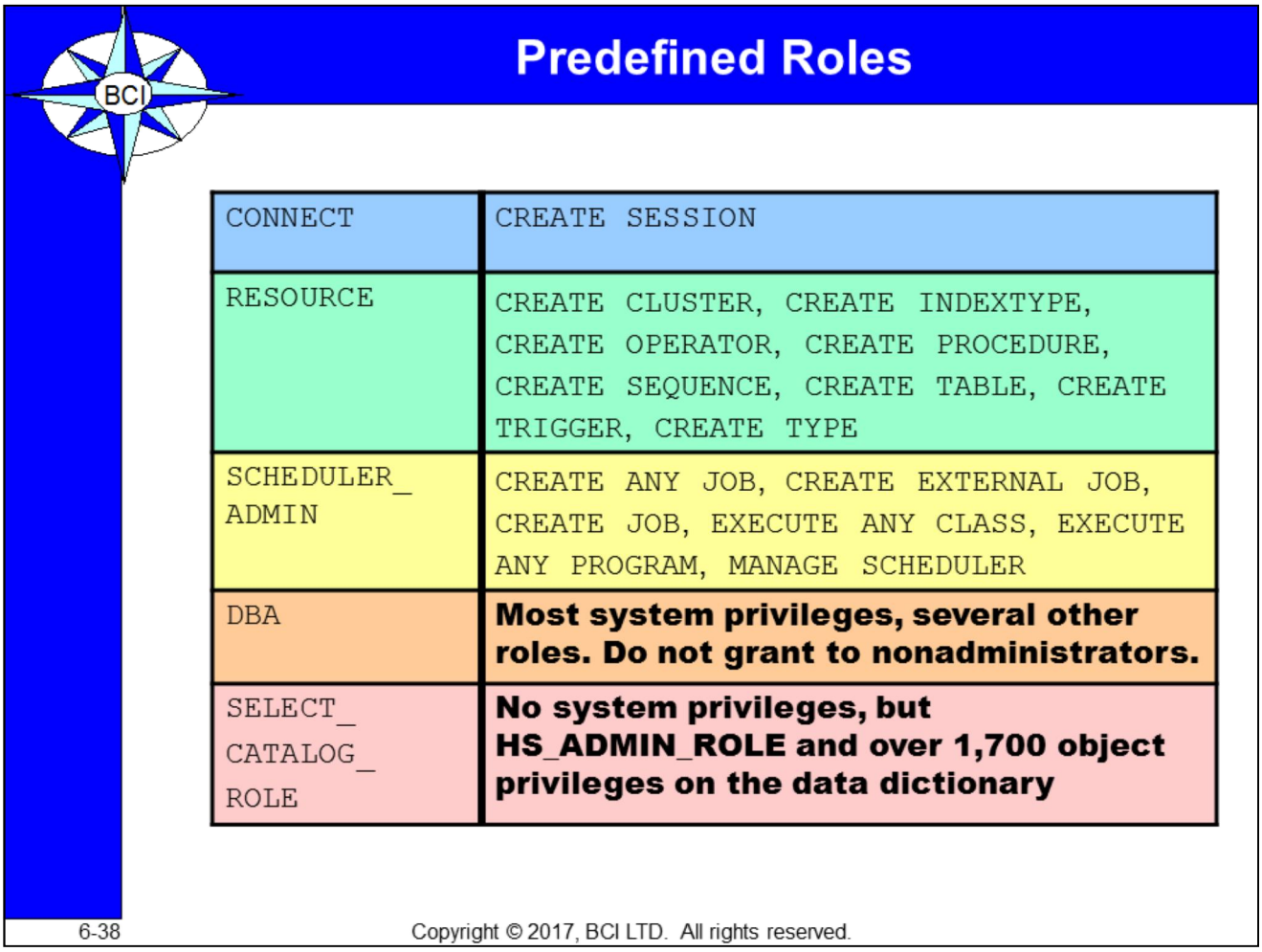
A role is a container that holds privileges. The main benefit of a role is that it simplifies the process of granting privileges to users. To make the process efficient, a DBA creates a role and then grants all of the privileges required by a user to perform a task to the role. If another user comes along that needs to perform the same task, instead of granting that user the permission explicitly, the DBA grants the user the role. Any privileges that have been granted to a role that has been granted to a user automatically apply to the user. Similarly, if you need to grant new privileges to users or revoke existing privileges from users, and if these were granted to a role rather than users, you need to grant or revoke the privileges only once – at the role level – instead of numerous times. Furthermore, those privileges granted will be automatically active once the grant or revoke takes place. Changes to role privileges are dynamically modified for all users holding the role.

When you grant privileges to users, those privileges will be available no matter how the user accesses the database. This means that someone using a front-end client application that presents preconfigured forms may need the same level of privileges as someone connecting to the instance using iSQL*Plus and performing interactive queries. The issue with this is that a user of the front-end application could also

connect to the instance and perform deletes, or other data manipulation, that may be more controlled through the front-end application. Roles, which can be selectively enabled and disabled, allow you to provide the user with additional privileges by enabling them when the user is using the front-end application, but not allow the user to have the same set of privileges if they connect to the instance using iSQL*Plus.

The restriction to roles is that you cannot grant an object privilege WITH GRANT OPTION to a role. You can, however, grant a system privilege WITH ADMIN OPTION to a role. Doing so allows anyone granted the role to grant those system privileges or roles to others.

The evaluation of roles by the database takes less work than evaluating privileges assigned to users directly. This means that roles provide better performance than granting privileges directly to users. This is because the privileges granted to a role can be cached in the data dictionary cache when they are first used and do not need to be reloaded, unless flushed out, the next time a user who has been assigned the role makes use of the privileges. Individual user privileges must be checked against the data dictionary each time a command is sent to the server, and will be cached only if they are frequently used.



Predefined Roles

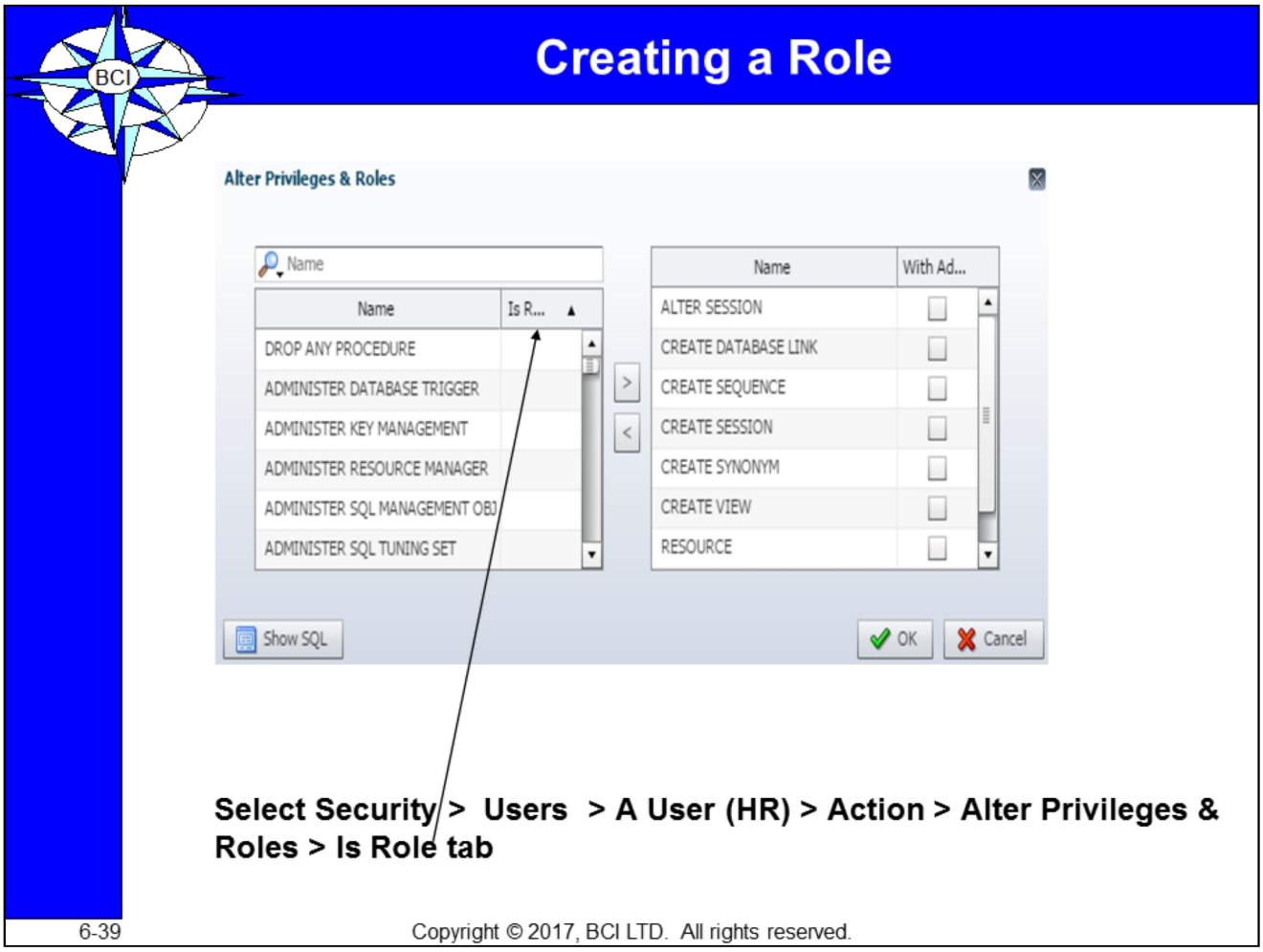
There are several roles that are defined automatically for Oracle databases when you run database creation scripts. `CONNECT` is granted automatically to any user created with Enterprise Manager. In earlier versions of the database (before Oracle Database 12c Release 2), the `CONNECT` role included more privileges, such as `CREATE TABLE` and `CREATE DATABASE LINK`, which have been removed for security reasons.

Note: Be aware that granting the `RESOURCE` role includes granting the `UNLIMITED TABLESPACE` privilege.

Functional Roles

Other roles that authorize you to administer special functions are created when that functionality is installed. For example, `XDBADMIN` contains the privileges required to administer the Extensible Markup Language (XML) database if that feature is installed. `AQ_ADMINISTRATOR_ROLE` provides privileges to administer advanced queuing. `HS_ADMIN_ROLE` includes the privileges needed to administer heterogeneous services. You must not alter the privileges granted to these functional roles without the assistance of Oracle support because you

may inadvertently disable the needed functionality.



Creating a Role

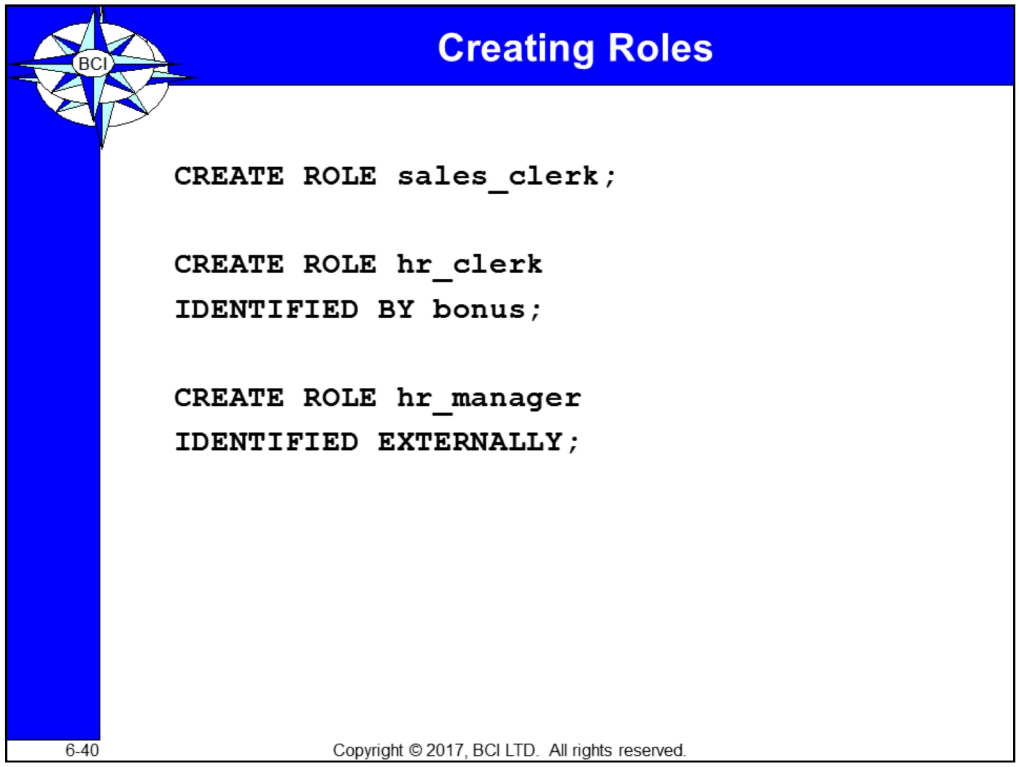
A **role** is a named group of related privileges that are granted to users or to other roles. A DBA manages privileges through roles.

To create a role, perform the following steps:

1. In Enterprise Manager Database Express, select Security > Users > A USER > Actions >

   Alter Privileges & Roles > Select the Role or Roles you want the USER to have.

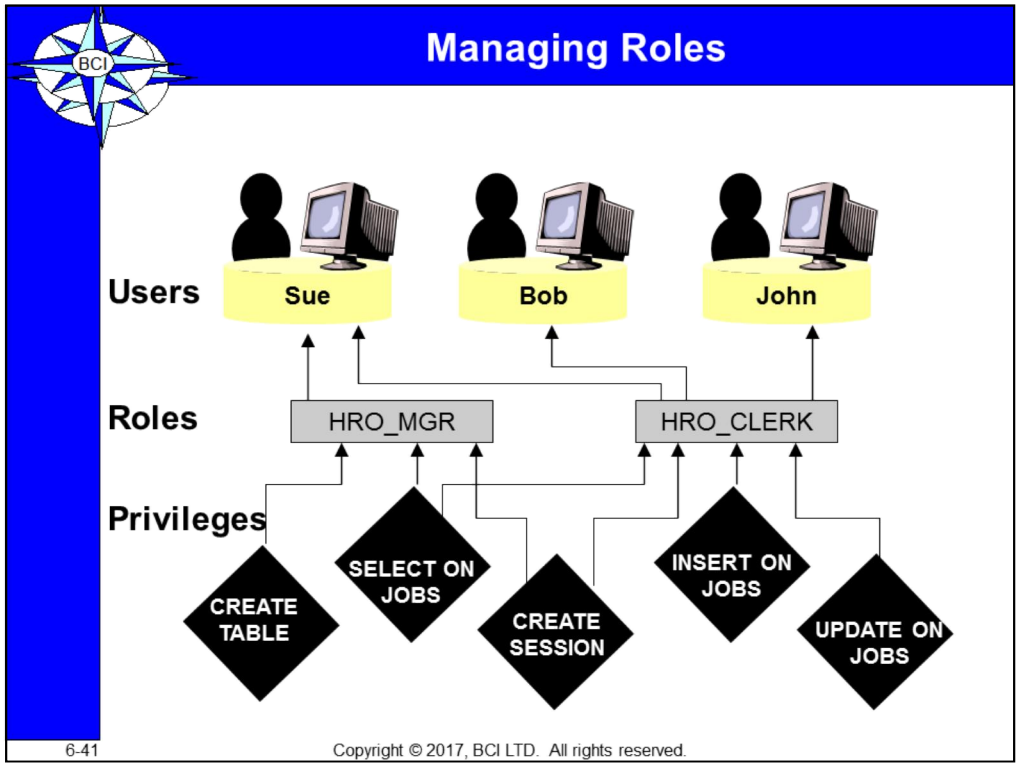
2.. Click the OK button.



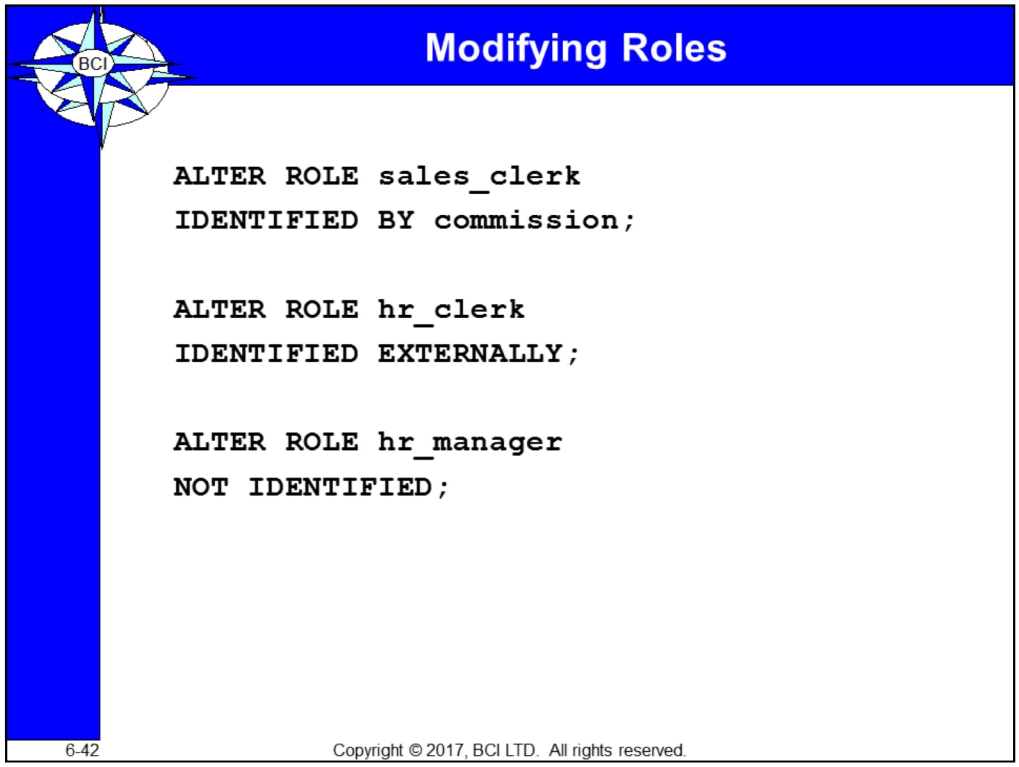
Like most other security-related items in Oracle, you can create roles from the command line using iSQL*Plus or with Enterprise Manager. In order to create a role, you must have been assigned the CREATE ROLE privilege, or the DBA role. The syntax for the CREATE ROLE command is:

```
CREATE ROLE rolename
[NOT IDENTIFIED | IDENTIFIED BY password | EXTERNALLY |
GLOBALLY];
```

The name of each role must be unique in the database and cannot be the same as that of an existing user, since users and roles are both stored in the same place in the data dictionary. When you issue the CREATE ROLE command, the default is to create a role with the name specified and not require any authentication to have the role enabled for the user. However, if you want to enable the role through an application, you can specify a password for the role by using the IDENTIFIED BY clause followed by the password.



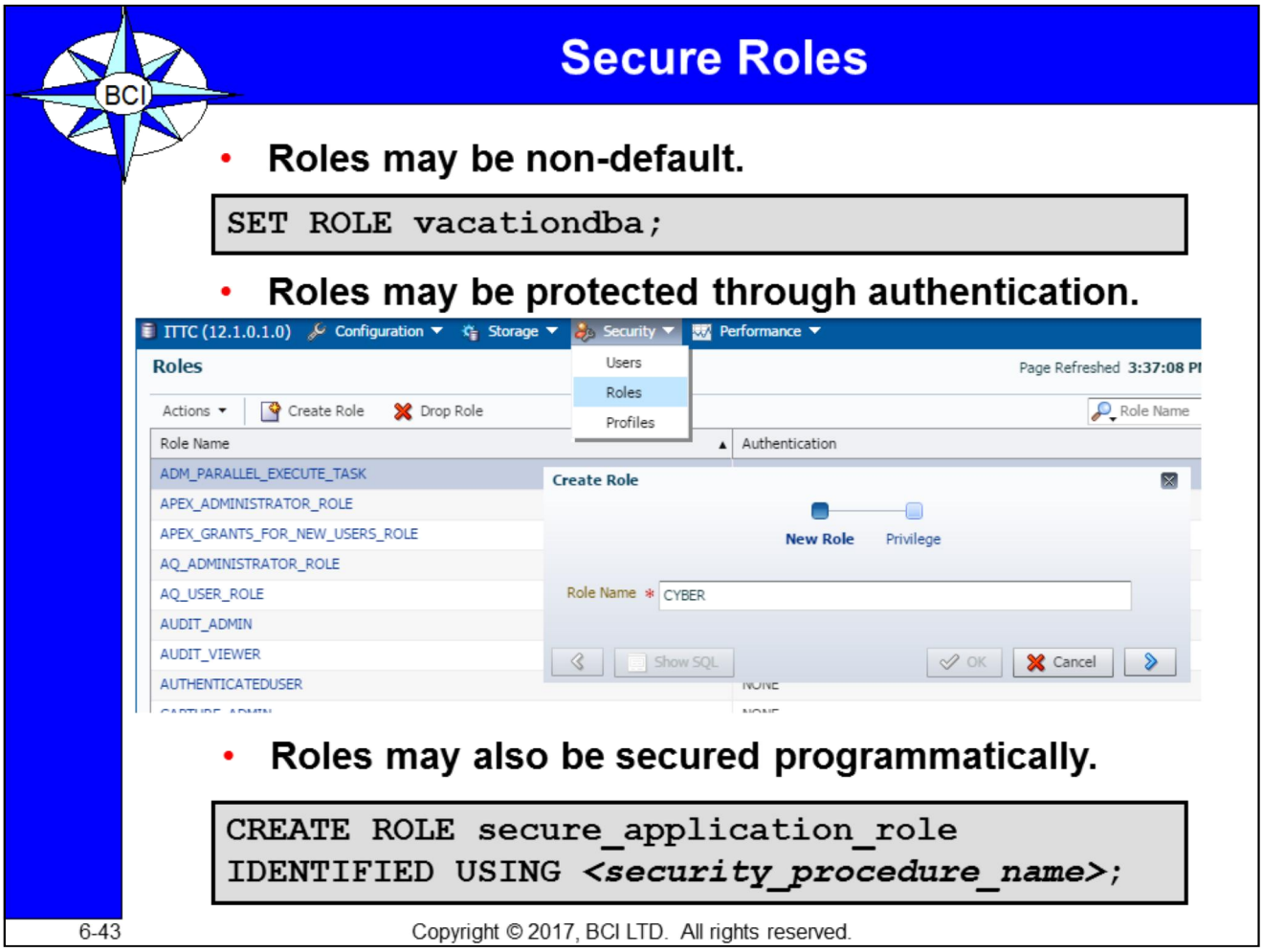
A user can be a member of many roles, as shown on the above slide. The privileges are cumulative.



Once a role is created, its authentication method can be changed after the fact using Enterprise Manager or by issuing the ALTER ROLE command:

```
ALTER ROLE rolename
[NOT IDENTIFIED | IDENTIFIED BY password | EXTERNALLY |
GLOBALLY];
```

Changes to role permissions or who it is granted to are done using the standard GRANT and REVOKE commands.



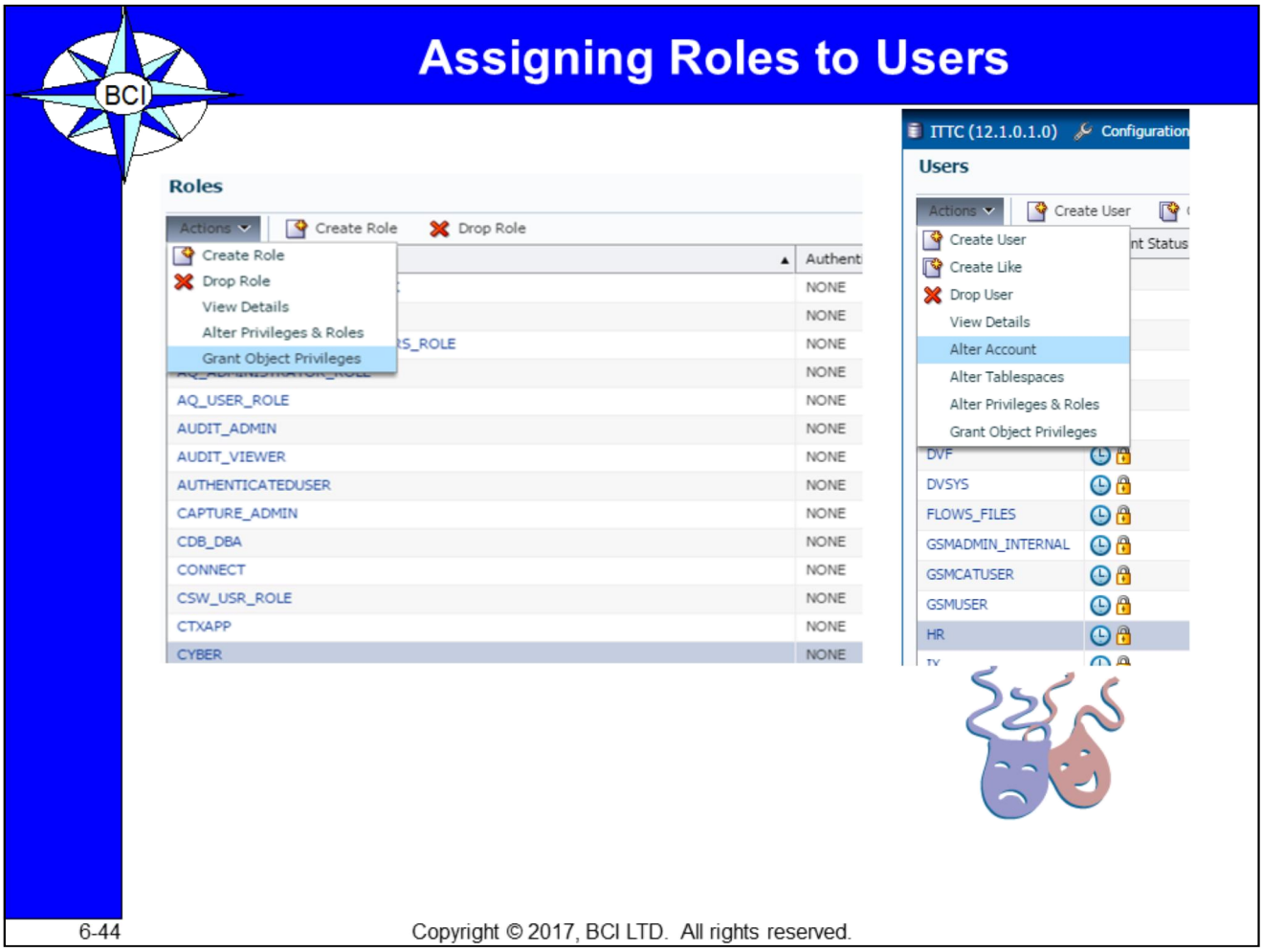
Secure Roles

Roles are usually enabled by default, which means that if a role is granted to a user, that user can exercise the privileges given to that role. It is possible to:

Make a role nondefault. When the role is granted to a user, deselect the DEFAULT check box. The user must now explicitly enable the role before the role's privileges can be exercised.

Have a role require additional authentication. The default authentication for a role is None, but it is possible to have the role require additional authentication before it can be set.

Create secure application roles that can be enabled only by executing a PL/SQL procedure successfully. The PL/SQL procedure can check things such as the user's network address, which program the user is running, time of day, or other elements needed to properly secure a group of permissions.



Assigning Roles to Users

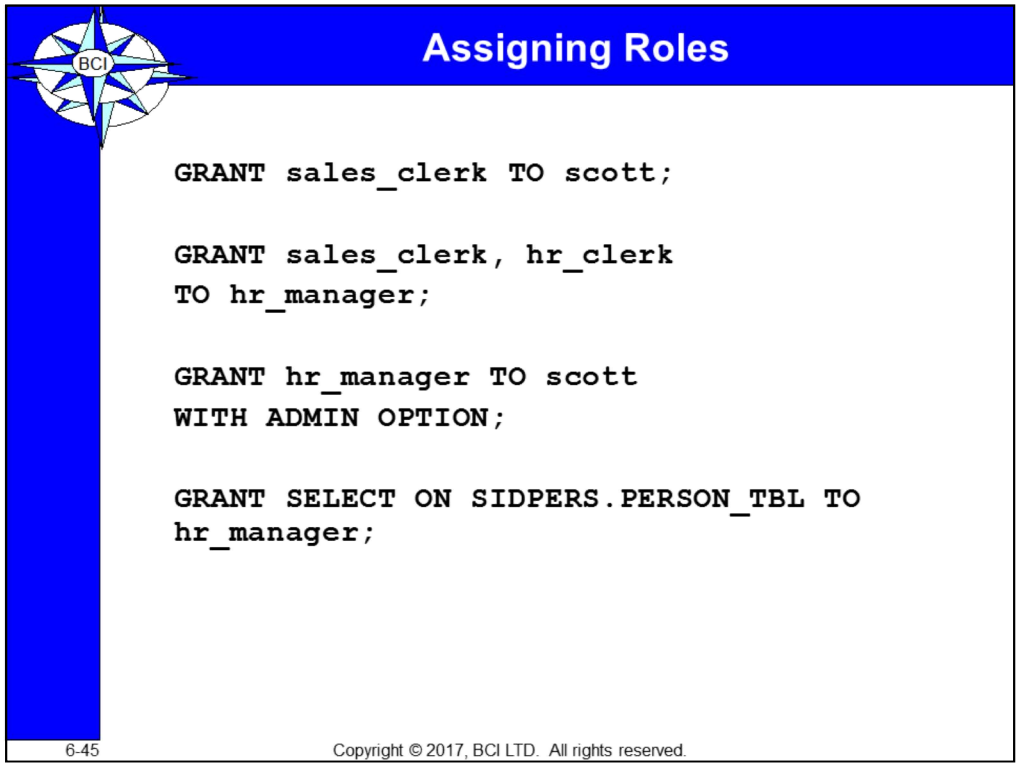
A role is a set of privileges that can be granted to users or to other roles. You can use roles to administer database privileges. You can add privileges to a role and then grant the role to a user. The user can then enable the role and exercise the privileges granted by the role. A role contains all privileges granted to that role and all privileges of other roles granted to it.

By default, Enterprise Manager automatically grants the `CONNECT` role to new users. This allows users to connect to the database and create database objects in their own schemas.

To assign a role to a user, perform the following steps:

1. In Enterprise Manager Database Express, choose Security > USERS >
2. Click the Select Privileges & Roles tab, and then click the Select from the List button.
3. Select the desired role under Available Roles.
4. When you have assigned all appropriate roles, click the OK

button.



You grant permissions to roles just as you grant them to users: using Enterprise Manager or the GRANT command. Permissions are revoked the same way as well: using the REVOKE command. The syntax for granting system permissions to roles is as follows:

```
GRANT system_priv [, system_priv, …]
TO role | PUBLIC [, role | PUBLIC, …]
[WITH ADMIN OPTION];
```

For granting object privileges to roles the syntax is as follows:

```
GRANT ALL [PRIVILEGES] | object_priv [(column, column, …)]
          [, object_priv [(column, column, …(], …]
ON [schema_name.]object_name
TO role | PUBLIC [, role | PUBLIC, …];
```

Remember that you cannot grant object privileges WITH GRANT OPTION to roles, but you can grant system privileges WITH ADMIN OPTION.

To revoke system and object privileges from roles, the syntax is similar to revoking those same privileges from a user. For system privileges:

```
REVOKE system_priv | role_name [, system_priv | role_name, …]
FROM role | PUBLIC [, role | PUBLIC, …];
```

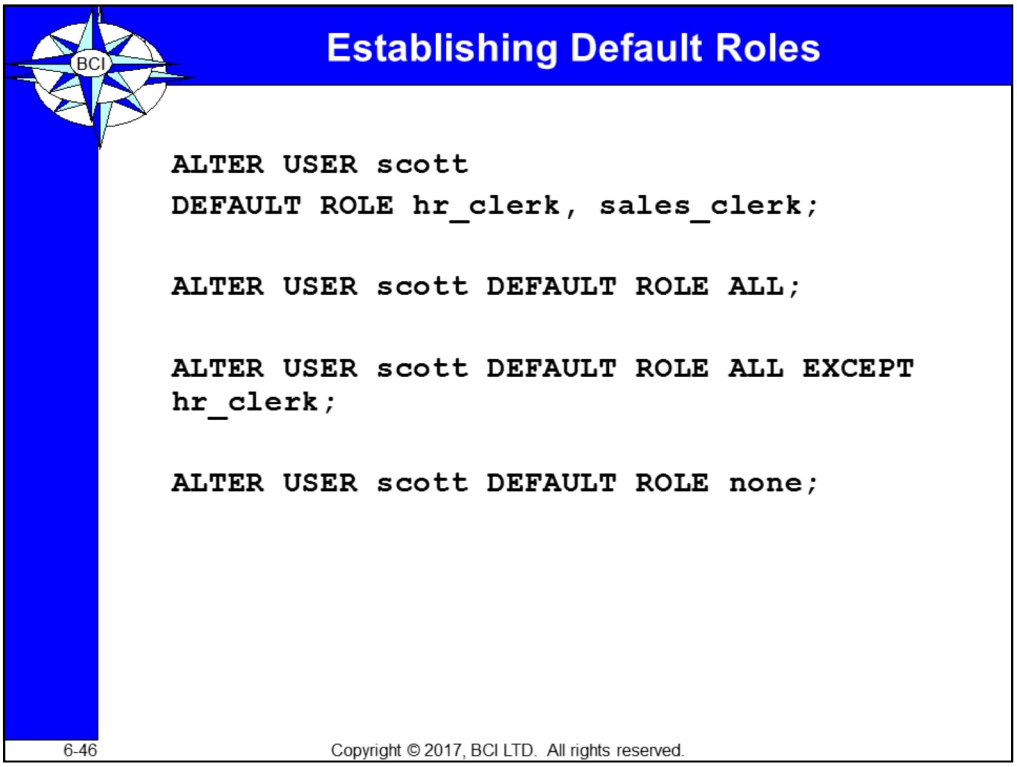
And for object privileges:

```
REVOKE ALL [PRIVILEGES] | object_priv [, object_priv, …]
ON [schema_name.]object_name
FROM role | PUBLIC [, role | PUBLIC, …]
[CASCADE CONSTRAINTS];
```

Once you have created a role and granted the role its further roles, along with the object and system privileges desired, you next need to assign the role to users that you want to inherit all the privileges that the role has. To do so, you use the GRANT command or Oracle Enterprise Manager. The syntax of the GRANT command to grant roles to users (as well as other roles) is:

```
GRANT role_name [, role_name, …]
TO user_name | role | PUBLIC [, user_name | role | PUBLIC, …]
[WITH ADMIN OPTION];
```

Just as you can grant a role to a user, you can also revoke the role to remove all of the role's privileges from the user using the REVOKE command or Oracle Enterprise Manager. If you revoke the role from a user, the role's permissions will not immediately taken away from the user, unless the user disconnects from the instance or disables the role. However, the user will not be able to re-enable the role on the next connection attempt or by using the SET ROLE command once it has been revoked.



Once you grant a role to a user, it is automatically configured to be a default role. This means that when the user connects to the instance, the role will automatically be enabled for the user and any privileges that the role has been granted will be available to the user. However, if you want some of the roles granted to the user to be active only when the user connects to the instance, you need to modify the set of default roles that are automatically enabled.

All roles granted to a user are considered the user's default roles unless otherwise specified by the ALTER USER command or Enterprise Manager. The syntax of the ALTER USER command to manage a user's default role list is as follows:

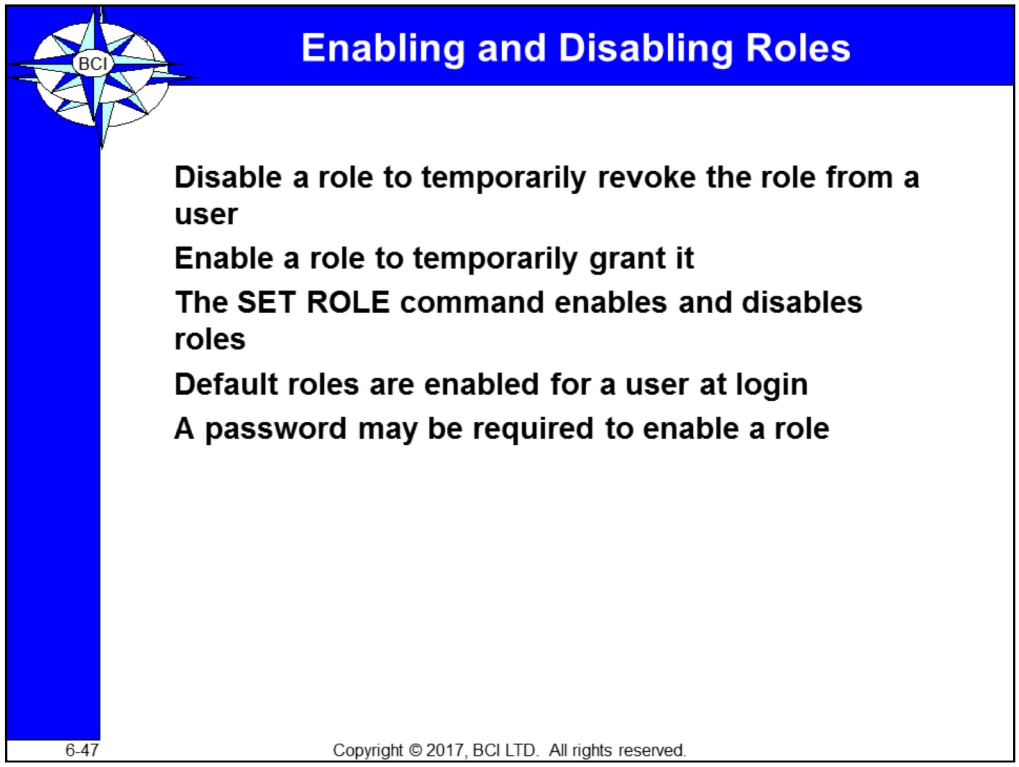
```
ALTER USER username DEFAULT ROLE
          role [, role, …] | ALL [EXCEPT role [, role, …]] |
NONE;
```

If you do not want Oracle to enable all roles that a user has been granted, you must use the ALTER USER command to disable any roles that you do not want the user to have when they connect to the instance. You can then programmatically enable the roles or have the user issue the SET ROLE command to enable those roles that you disabled by default.

If you grant the user a role that requires a password, and if you make that role a default role, the user will not be required to enter a password in order to make use of the privileges granted to the role. In essence, making a role that has a password a default role for the user bypasses the password requirement. In this way some

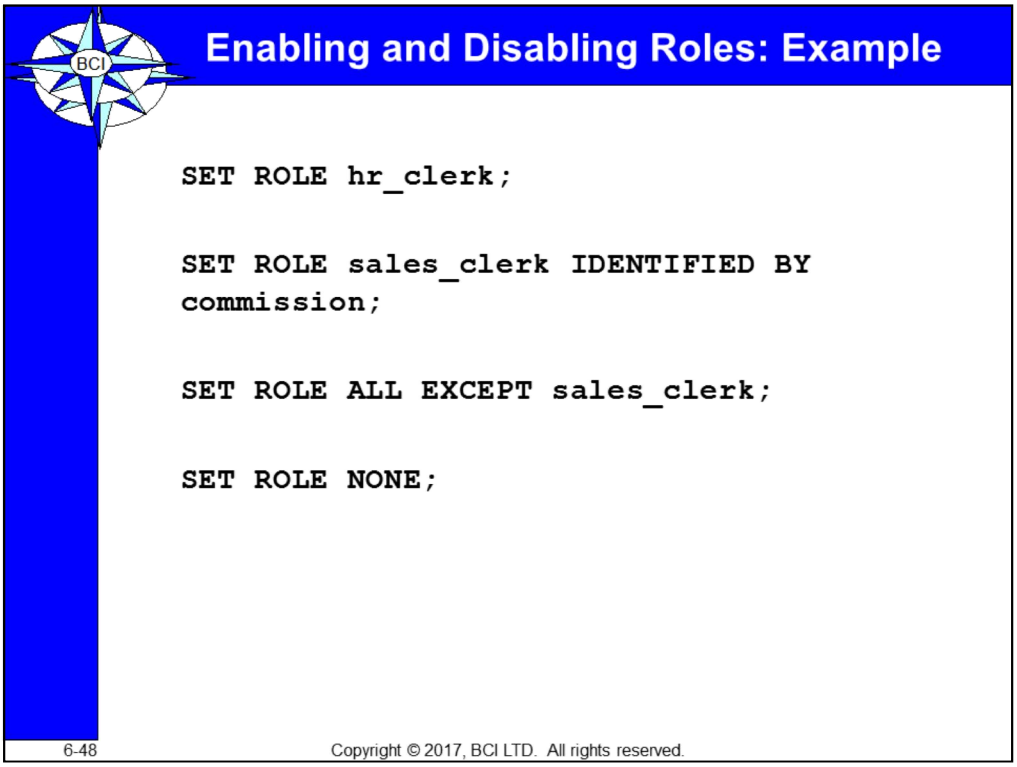
users may have the role and its privileges when they connect, by default, while other users will be required to enable the role manually and specify a password in order to access the privileges granted the role.

You can also disable all roles that have been assigned to the user by using the NONE option when specifying which roles are default roles. After doing so, all roles granted to the user will be disabled and will need to be enabled using the SET ROLE command. The user will have only the capability to perform actions according to those system and object privileges that have been assigned directly to him/her, or to PUBLIC.

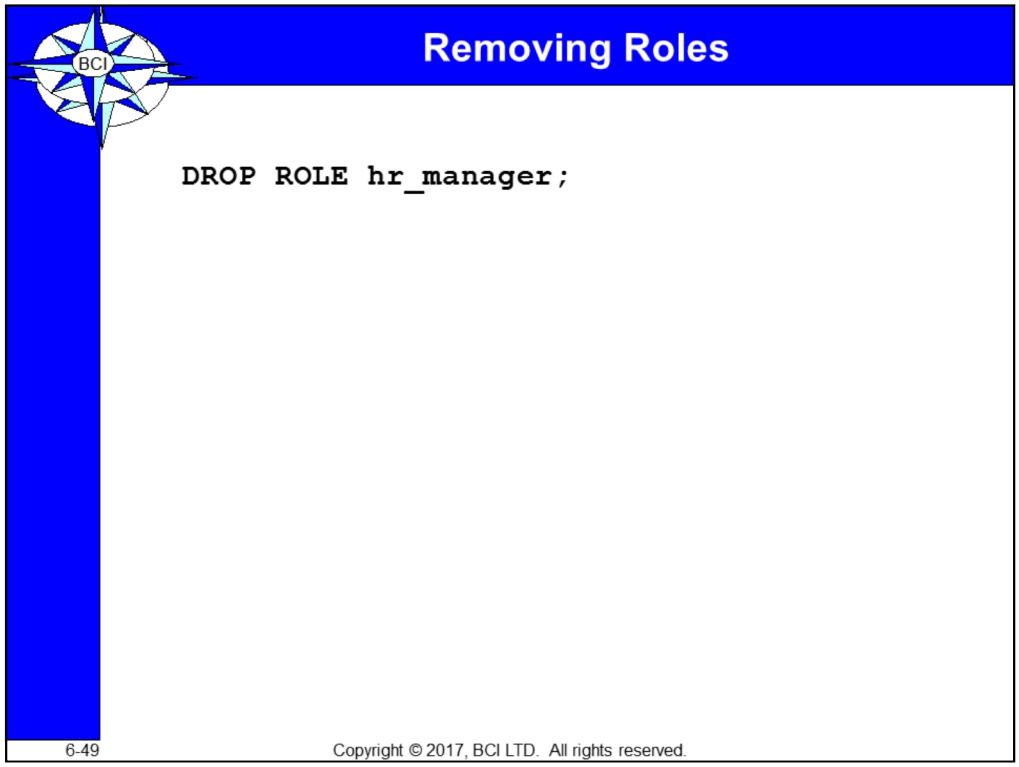


One of the major benefits of roles is the ability to selectively grant and revoke a set of privileges by enabling and disabling roles that contain them. While a user is connected to the instance, your application (typically) can issue the SET ROLE command, or execute the DBMS_SESSION.SET_ROLE package procedure, to enable or disable roles dynamically. A role that was created with a password will need to have the password specified when it is enabled. This allows you to further control the enabling of roles by users by ensuring that roles are enabled only while a particular application is being used to connect to the database. The syntax for the SET ROLE command is as follows:

```
SET ROLE ALL [EXCEPT role_name [, role_name]] | NONE |
role_name [IDENTIFIED BY password] [, role_name [IDENTIFIED
BY password, …];
```



If you want to disable a role for a user, you need to issue the SET ROLE command, or execute the DBMS_SESSION.SET_ROLE procedure a second time, omitting the role that you do not want the user to have enabled. In other words, there is no “UNSET ROLE” command or its equivalent.

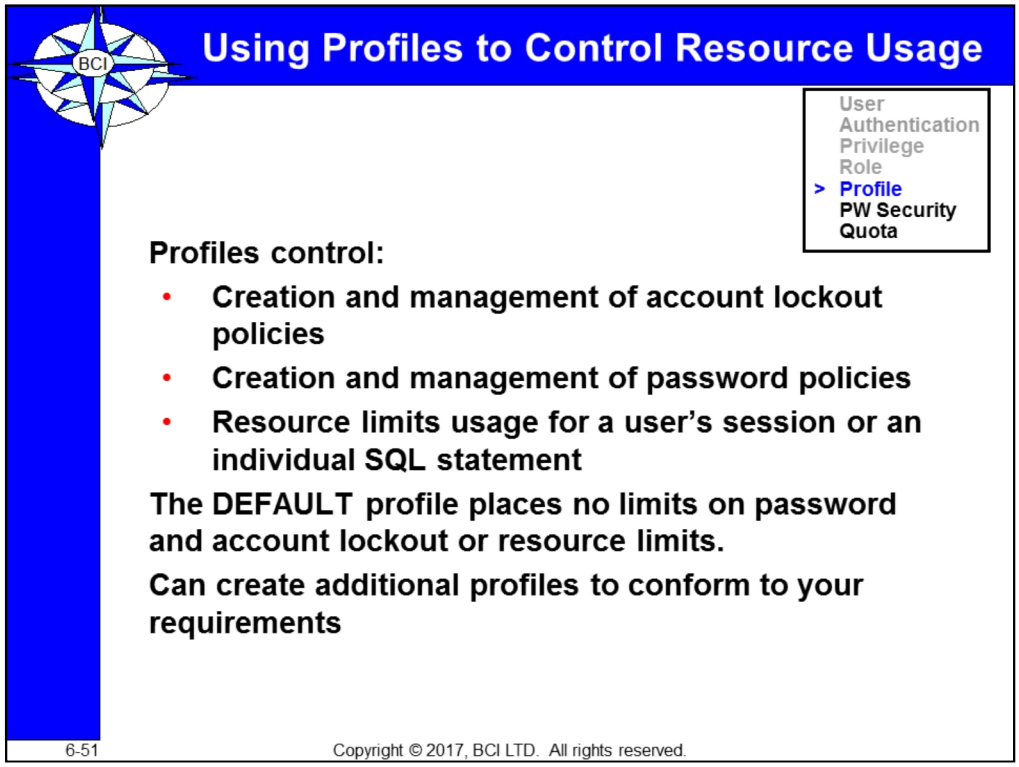


If you no longer need a role that you have been using, you can drop it from the database by issuing the DROP ROLE command, or by using Oracle Enterprise Manager. In order to drop a role, you must be the user who created the role, have been granted the DROP ANY ROLE system privilege, or have been granted the role WITH ADMIN OPTION. If any of these is not true, the command will fail and the role will not be dropped. The syntax of the DROP ROLE command is as follows:

```
DROP ROLE role_name;
```

When you drop a role, any user or role to which the role being dropped has been granted will have it revoked at the time the role is dropped. Any privileges that the role granted its holders will also be revoked at the time the role is dropped.

Lab 6-3: Managing Roles



One aspect of the Oracle security domain deals with ensuring that password management and account lockout policies for the database are adhered to. These may be set at the enterprise level and need to be enforced at the database level. You may need to ensure that a database is available to all users and that no one user is able to invoke a SELECT statement that performs a large query and consumes all system resources, for example. The creation and management of account lockout and password policies, as well as limiting resource usage for a user's session or an individual SQL statement is handled by the use of profiles.

A profile is an Oracle object that allows you to set both password management and resource limits. A single profile is created when you create the database. This profile, called DEFAULT, places no limits on password and account lockout, or on resource utilization. You can change the settings of the DEFAULT profile to conform to your requirements, and they will then be applied to all users in the database assigned the DEFAULT profile.

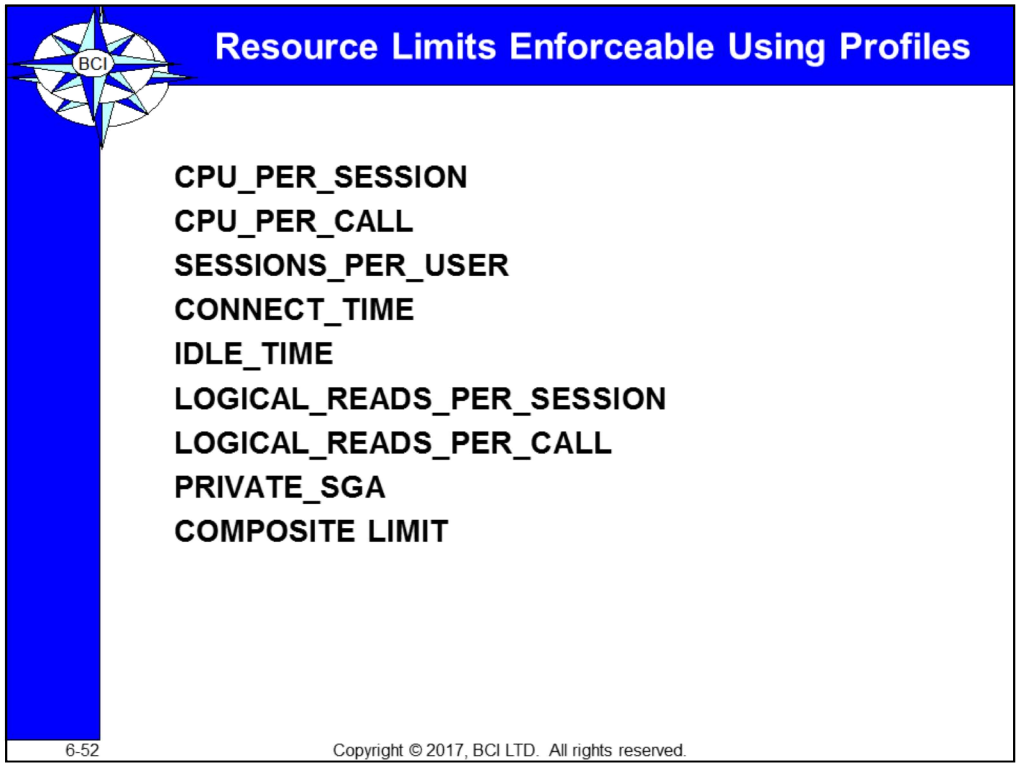
A DBA may create additional profiles dealing with password or account lockout issues, resource management settings, or both. Once created, a profile can be assigned to a user accounts as it is created, or it can be assigned to the user with the ALTER USER command. Any settings in the profile will then apply to the user then next time he/she connects to the database. A user may have only one profile active at one time, so you need to ensure that the settings within the profile match the requirements of each user.

When deciding to make use of profiles, it is important to understand what settings are always applied and which require that you change your database and instance

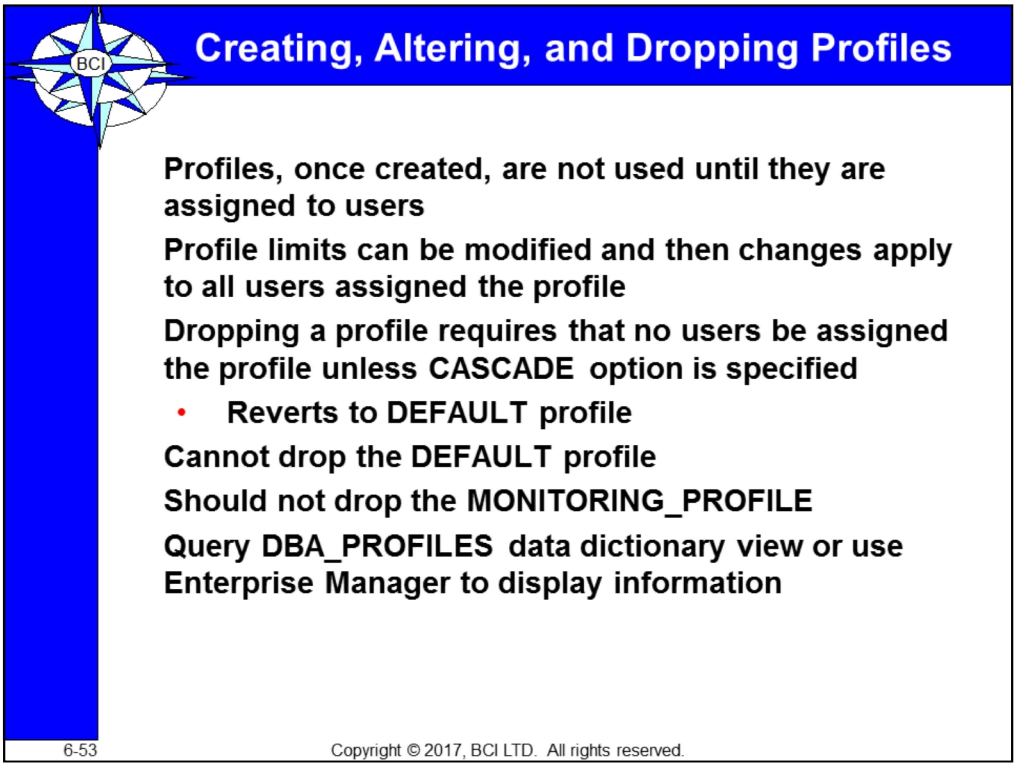
configuration. Because of the very nature of security requirements, Oracle ensures that password management and account lockout settings in profiles are always enforced. Any settings dealing with security policy are considered important enough that simply configuring them enables them. The utilization of system resources, such as CPU and disk I/O, is not automatically enforced. In order to have these aspects of a profile limit a user's actions, you need to enable them by setting them in the parameter file (SPFILE or INIT.ORA file) or by changing the value of the RESOURCE_LIMIT initialization parameter to TRUE with the ALTER SYSTEM command. This parameter is FALSE by default. According to the Database STIG, Section B.18.2, The required value for this parameter is TRUE.

The Database STIG states that the following settings will be used in all profiles:

idle_time	15
password_life_time	90
password_reuse_max	10
password_reuse_time	365
failed_login_attempts	3



Profiles also allow you to limit the user of system resources by a user. The limit can be specified for the session (known as a per session limit) or for a single SQL statement (known as a per call limit). The following table lists the resource limits that can be enforced in Oracle 12c:



You can create, alter, and drop profiles using the CREATE PROFILE, ALTER PROFILE, and DROP PROFILE commands or by using Oracle Enterprise Manager. Profiles, once created, are not used until they are assigned to users. Once a profile is created, its limits can be modified and then those changes will apply to all users assigned the profile the next time the users connect to the server (i.e., the changes are not applied to any logged-on users). Dropping a profile requires that no users be assigned the profile, unless the CASCADE option is specified. If you use the CASCADE option of the DROP PROFILE command, users to whom the profile was assigned will automatically have the default profile assigned.

To create a profile, issue the CREATE PROFILE command, whose syntax is as follows when dealing with resource limits:

```
CREATE PROFILE profile_name LIMIT
          [SESSIONS_PER_USER                      value]
          [CPU_PER_SESSION                        value]
          [CPU_PER_CALL
          value]
          [CONNECT_TIME
          value]
          [IDLE_TIME
          value]
          [LOGICAL_READS_PER_SESSION    value]
          [LOGICAL_READS_PER_CALL                 value]
```

```
          [COMPOSITE_LIMIT                        value]
[PRIVATE_SGA                                      bytes [K|M]]
```

Password limits will be discussed in the next section.

You can specify the keyword UNLIMITED for any profile limit to not have any limitation imposed for that resource. Specifying the keyword DEFAULT for any limit when creating or altering a profile will assign the value of the DEFAULT profile for the resources. This way, you can have some limits specific to a profile and others at the same value for all users. If the DEFAULT profile imposes a limit, specifying DEFAULT for the profile you are creating will impose the same limit.

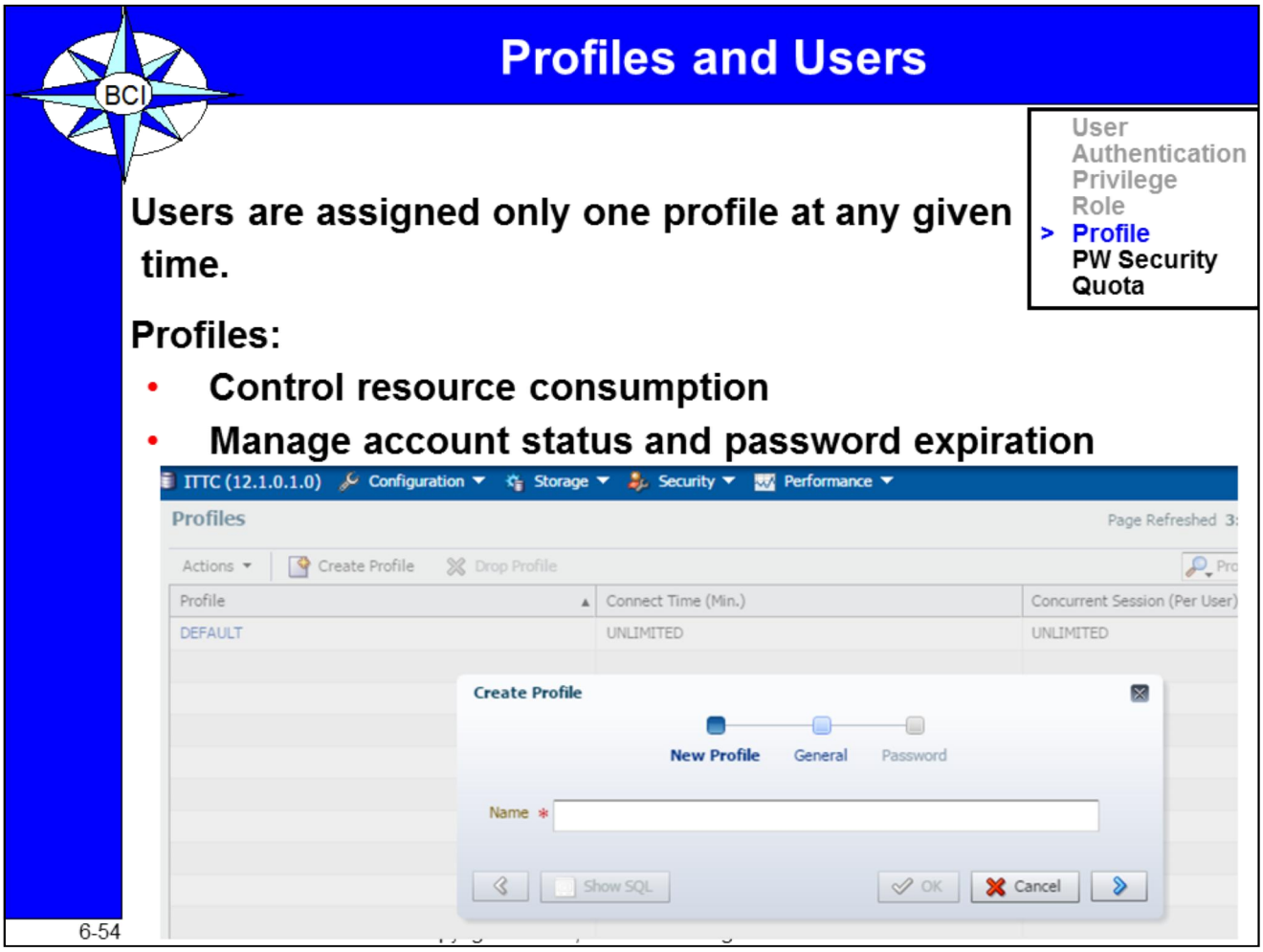
To modify a profile, you issue the ALTER PROFILE command, whose options are the same as the CREATE PROFILE command. As stated earlier, any changes to the profile limits do not take effect until the next time the user to whom the profile applies logs on. Any existing user session will not be affected by the profile change.

The DROP PROFILE command deletes a profile and optionally, if the CASCADE option is specified, assigns the DEFAULT profile to any users for whom the profile was active when it was dropped. The syntax of the DROP PROFILE command is:

```
DROP PROFILE profile_name [CASCADE];
```

You cannot drop the DEFAULT profile, since it must exist in the database. You also should not drop the MONITORING_PROFILE, because it is needed by Oracle to perform system monitoring functions in the database.

If you want to determine the currently configured profile values, you can query the DBA_PROFILES data dictionary view or use Enterprise Manager to display the information.



Profiles and Users

Profiles impose a named set of resource limits on database usage and instance resources. Profiles also manage the account status and place limitations on users' passwords (length, expiration time, and so on). Every user is assigned a profile and may belong to only one profile at any given time. If users have already logged in when you change their profile, the change does not take effect until their next login.

The default profile serves as the basis for all other profiles. As illustrated in the slide, limitations for a profile can be implicitly specified (as in CPU/Session), be unlimited (as in CPU/Call), or reference whatever setting is in the default profile (as in Connect Time).

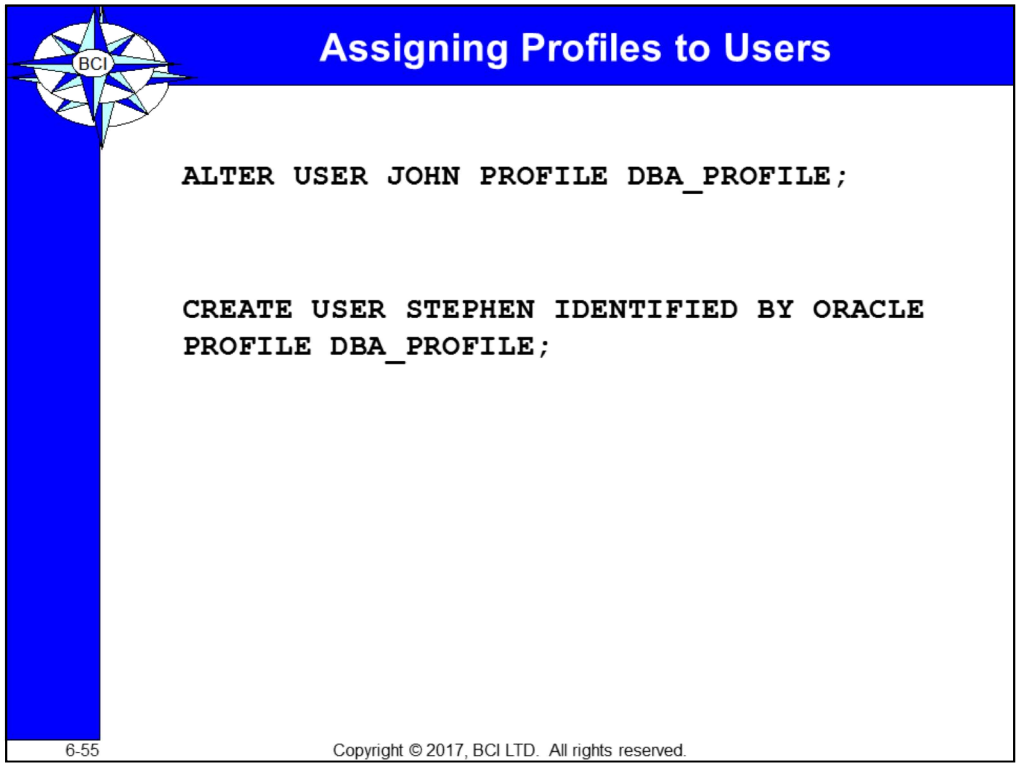
Profiles cannot impose resource limitations on users unless the `RESOURCE_LIMIT` initialization parameter is set to `TRUE`. With `RESOURCE_LIMIT` at its default value of `FALSE`, profile limitations are ignored.

Profiles enable the administrator to control the following system resources:

CPU: CPU resources may be limited on a per-session or per-call basis. A CPU/Session limitation of 1,000 means that if any individual session that

uses this profile consumes more than 10 seconds of CPU time (CPU time limitations are in hundredths of a second.), then that session receives an error and is logged off:

> ORA-02392: exceeded session limit on CPU usage, you are being logged off



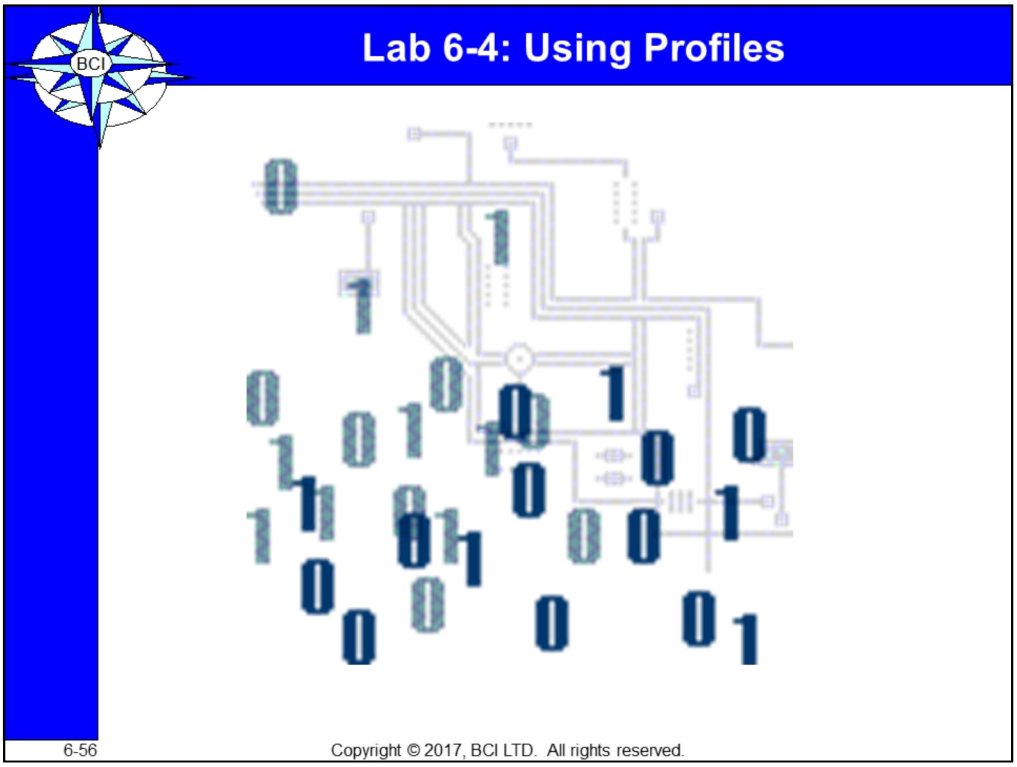
Users are assigned profiles when they are either created or altered via the command interface or Enterprise Manager. You can use the ALTER USER command at any time to assign a profile to an existing user. Only one profile is active for a user at any time, so it is not possible for a user to be assigned more than one profile. The profile limits apply each time the user connects to the instance and creates a session, and they are enforced for the duration of the session.

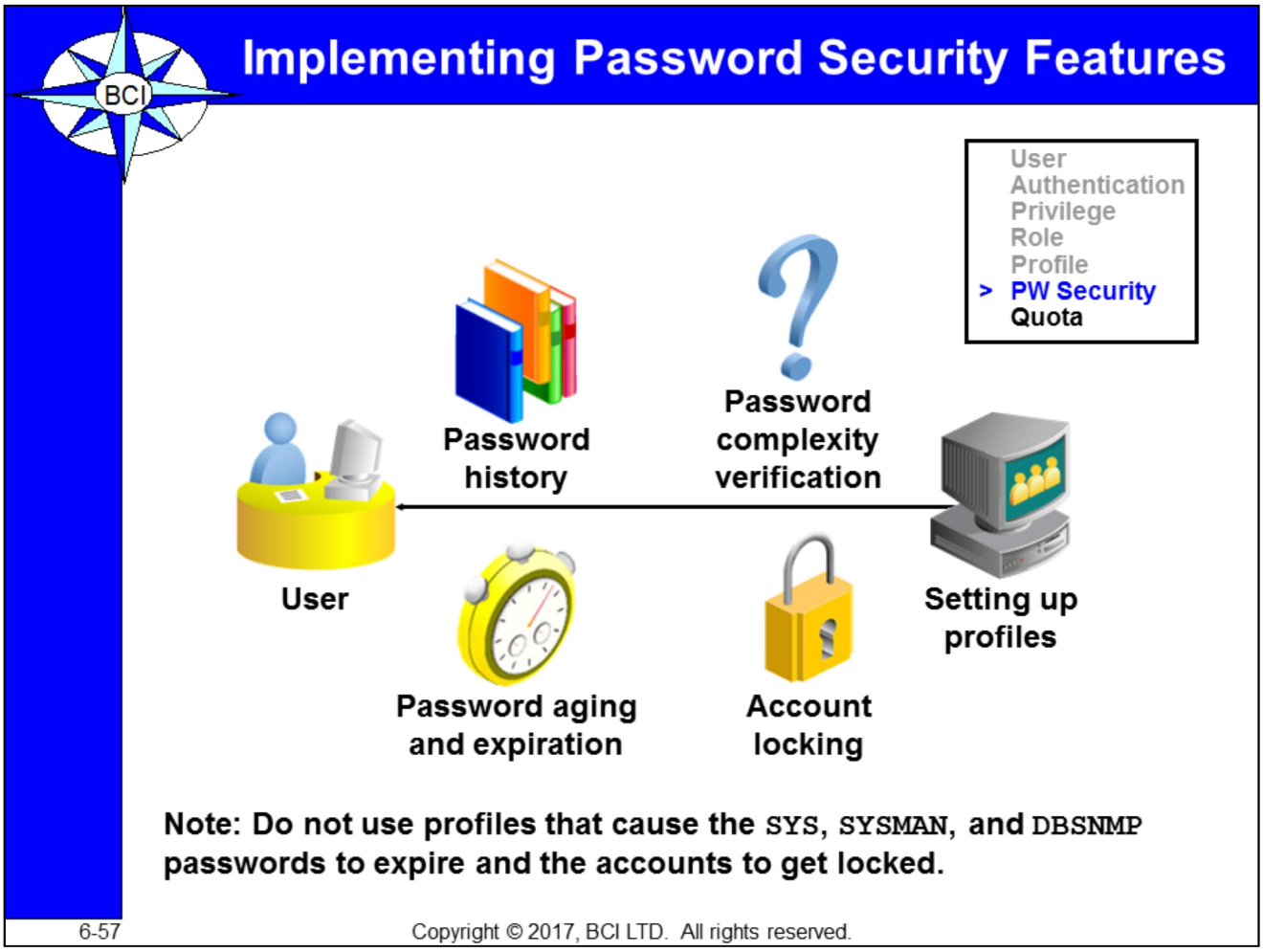
To assign a profile called DBA_PROFILE to the user John, who already exists in the database, you can issue the following command:

```
ALTER USER JOHN PROFILE DBA_PROFILE;
```

If you create a new user called Stephen and want to assign him the DBA_PROFILE, you can issue the following command:

```
CREATE USER STEPHEN IDENTIFIED BY ORACLE PROFILE DBA_PROFILE;
```


Lab 6-4: Using Profiles
BCI
6-56
Copyright © 2017, BCI LTD. All rights reserved.



Implementing Password Security Features

Oracle password management is implemented with user profiles. Profiles can provide many standard security features including the following:

Account locking: Enables automatic locking of accounts for a set duration when users fail to log in to the system in the specified number of attempts.

The `FAILED_LOGIN_ATTEMPTS` parameter specifies the number of failed login attempts before the lockout of the account.

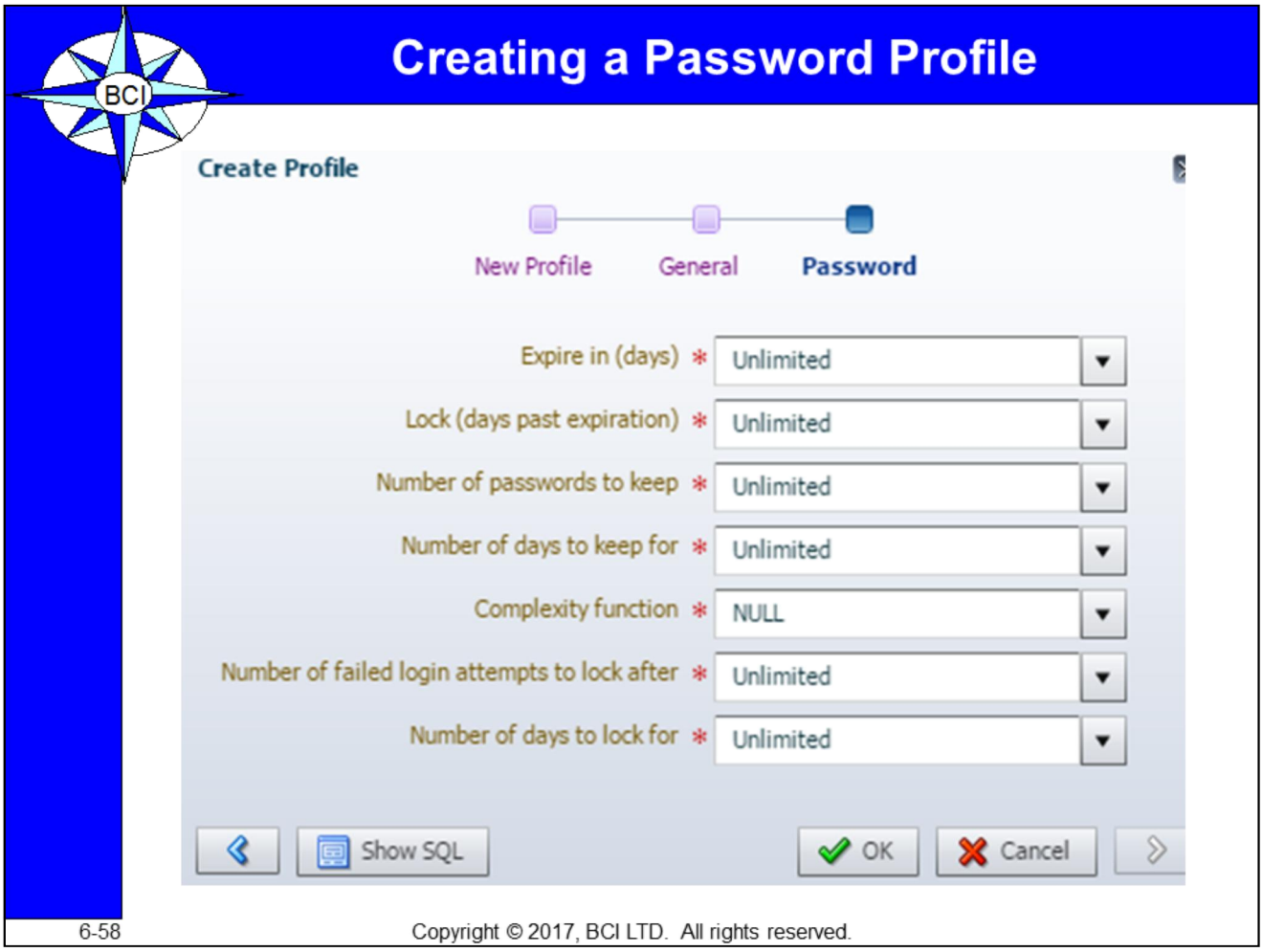
The `PASSWORD_LOCK_TIME` parameter specifies the number of days for which the account is locked after the specified number of failed login attempts.

Password aging and expiration: Enables user passwords to have a lifetime, after which the passwords expire and must be changed

The `PASSWORD_LIFE_TIME` parameter determines the lifetime of the password in days, after which the password expires.

The `PASSWORD_GRACE_TIME` parameter specifies a grace period in days for changing the password after the first successful login after the password has expired.

Note: Expiring passwords and locking the `SYS`, `SYSMAN`, and `DBSNMP` accounts prevent Enterprise Manager from functioning properly. The applications must catch the ‘‘password expired’’ warning message and handle the password change; otherwise, the grace period expires and the user is locked out without knowing the reason.



Creating a Password Profile

To create a password profile, select Security > Profiles, and click the Create button.

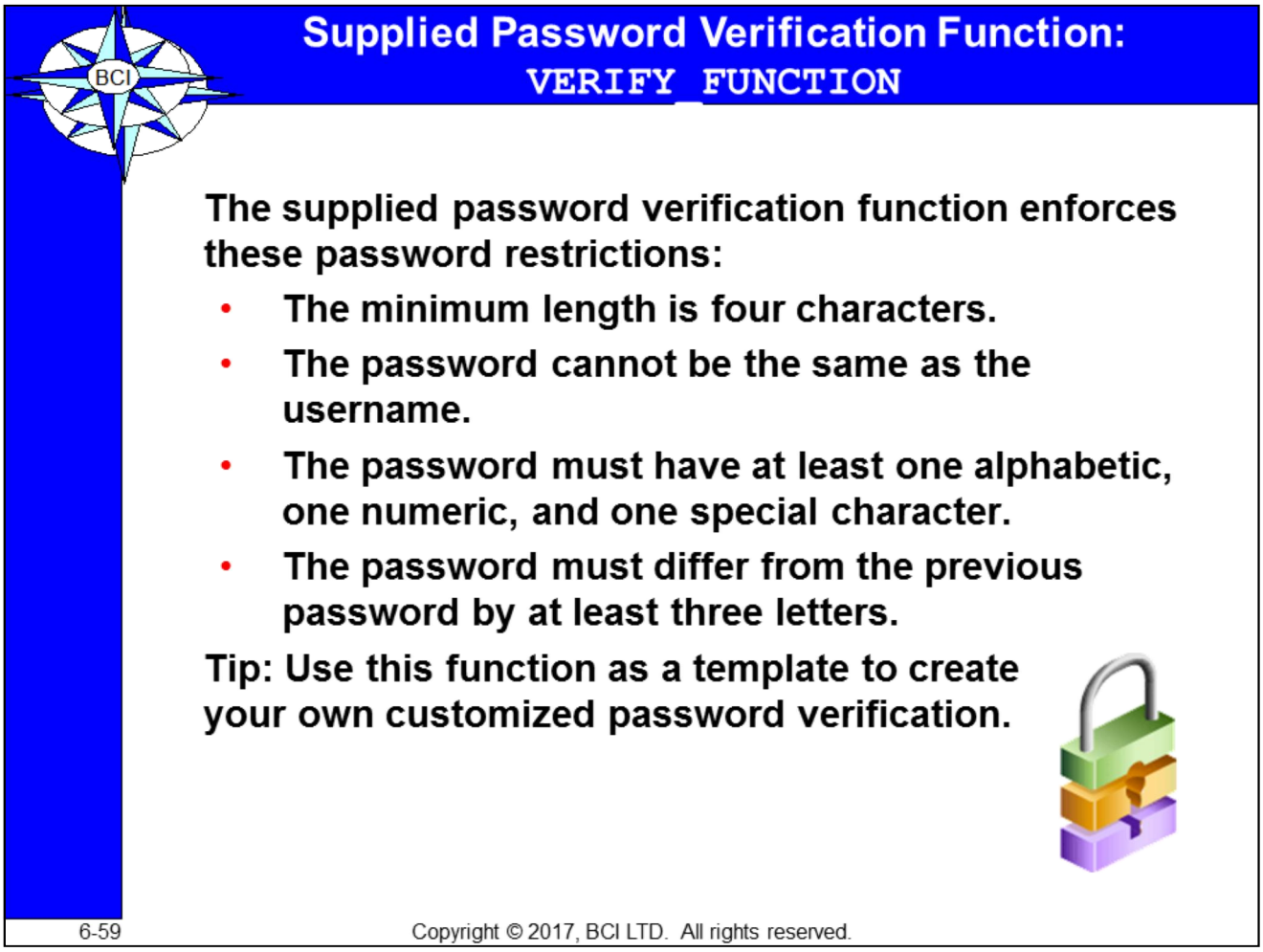
Common values for each of the settings can be chosen from a list of values (Click the flashlight icon to browse.), or you can enter a custom value.

All time periods are expressed in days, but can be expressed as fractions also. There are 1,440 minutes in a day, and so 5/1440 is five minutes.

Enterprise Manager can also be used to edit existing password profiles.

Dropping a Password Profile

In Enterprise Manager, you cannot drop a profile that is used by users. However, if you drop a profile with the `CASCADE` option (for example, in SQL*Plus), then all users who have that profile are automatically assigned the `DEFAULT` profile.



Supplied Password Verification Function: VERIFY_FUNCTION

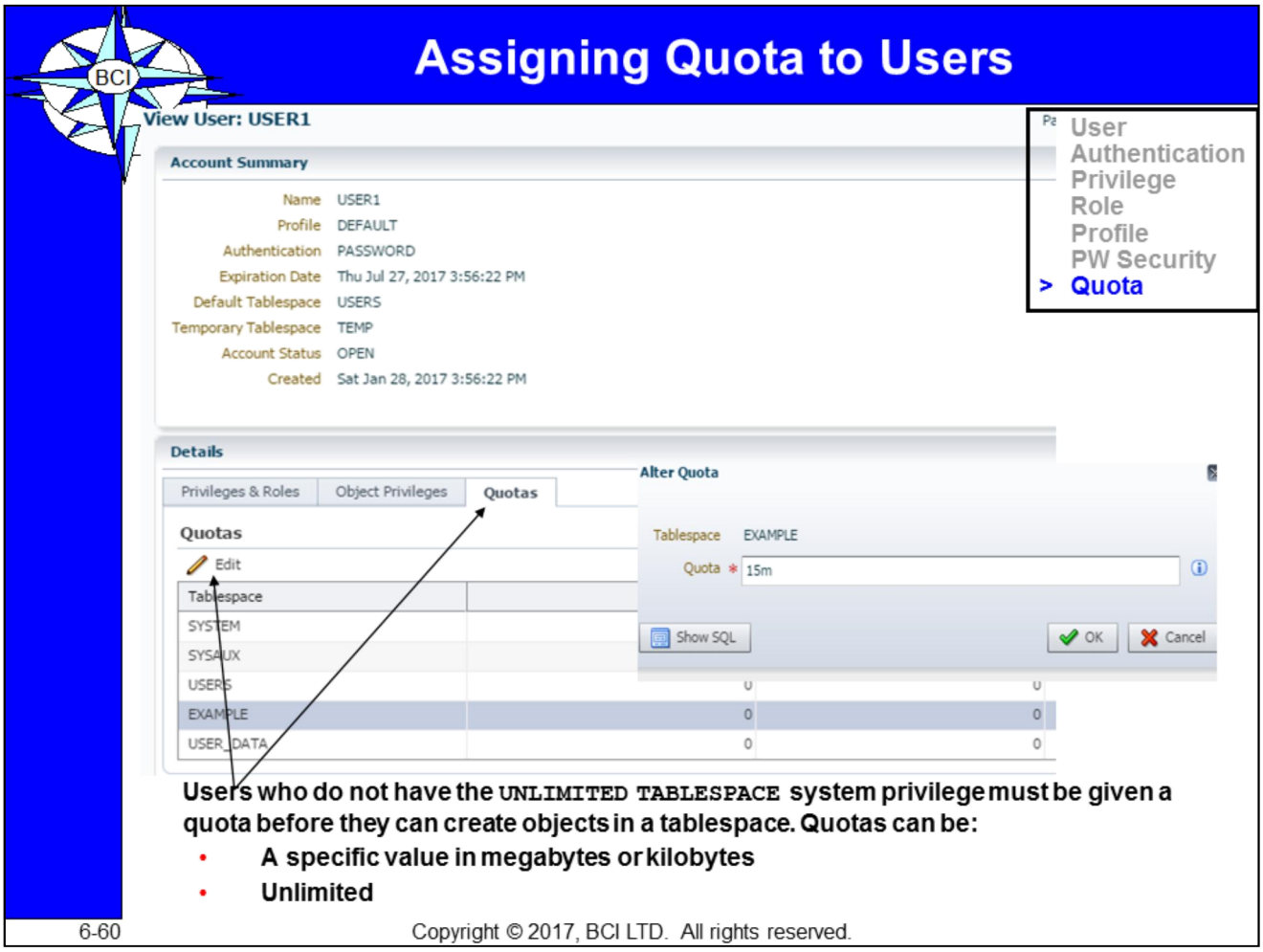
The Oracle server provides a password complexity verification function named VERIFY_FUNCTION. This function is created with the <oracle_home>/rdbms/admin/utlpwdmg.sql script. The password complexity verification function must be created in the SYS schema. It can be used as a template for your customized password verification.

In addition to creating VERIFY_FUNCTION, the utlpwdmg script also changes the DEFAULT profile with the following ALTER PROFILE command:

```
ALTER PROFILE default LIMIT
PASSWORD_LIFE_TIME 60
PASSWORD_GRACE_TIME 10
PASSWORD_REUSE_TIME 1800
PASSWORD_REUSE_MAX UNLIMITED
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LOCK_TIME 1/1440
PASSWORD_VERIFY_FUNCTION verify_function;
```

Remember that when users are created, they are assigned the DEFAULT profile,

unless another profile is specified.



Assigning Quota to Users

Quota is a space allowance in a given tablespace. By default, a user has no quota on any of the tablespaces. You have three options for providing a user quota on a tablespace.

Unlimited: This allows the user to use as much space as is available in the tablespace.

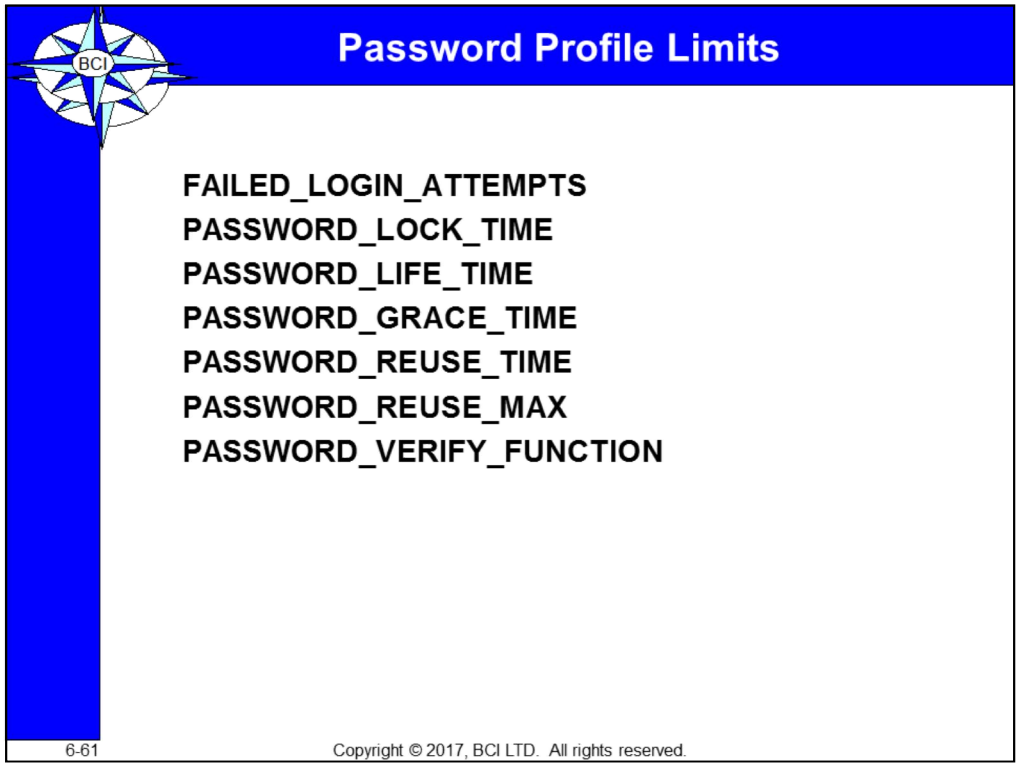
Value: This is a number of kilobytes or megabytes that the user can use. This does not guarantee that the space is set aside for the user. This value can be larger or smaller than the current space that is available in the tablespace.

`UNLIMITED TABLESPACE` system privilege: This system privilege overrides all individual tablespace quotas and gives the user unlimited quota on all tablespaces, including `SYSTEM` and `SYSAUX`. This privilege must be granted with caution.
Note: Be aware that granting the `RESOURCE` role includes granting this privilege.

You must not provide quota to users on the `SYSTEM` or `SYSAUX` tablespace.

Typically, only the `SYS` and `SYSTEM` users must be able to create objects in the `SYSTEM` or `SYSAUX` tablespace.

You do not need quota on an assigned temporary tablespace or any undo tablespaces.



The following table lists the Password Profile Limits and their descriptions:

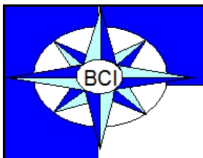
<INSERT C:\Courseware\ITTC-040 Phase I Oracle 12c\Images\Mod5-7.doc>

Summary

- Examine All Aspects of Security
- Security Requirements
- Database Administration Special Privileges
- Security Critical Instance Parameters
- Users and Security
- Create a New User:
  Database Authentication
- Changing User Quota on Tablespace

Summary (Continued)

- Managing Privileges
- System Privileges
- Granting Object Privileges
- Managing Roles
- Using Profiles to Control Resource Usage
- Password Profile Limits
- Password Verify Function



Terminal Learning Objective

ACTION: Maintain Oracle Database User Accounts.

CONDITION: Given a student handout and Oracle DBA Handbook.

STANDARD: Students will successfully create users, roles and perform resource and password management.