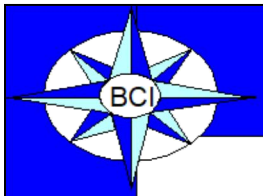


4

Managing the Oracle Instance

Terminal Learning Objective

ACTION: Manage the Oracle Instance.

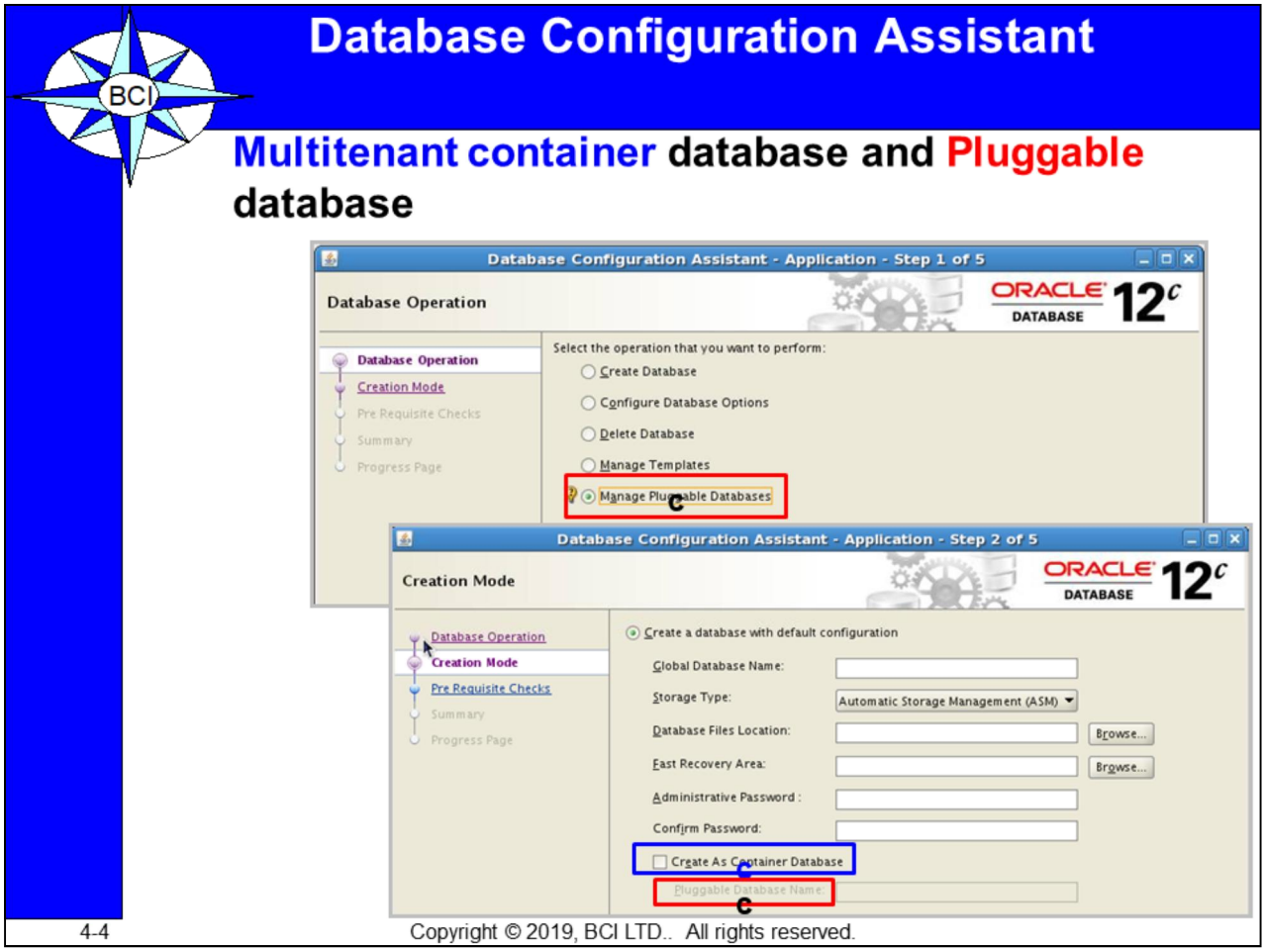
CONDITION: Given a student handout and Oracle DBA Handbook.

STANDARD: Students will successfully start and stop the Oracle Database.

Lesson Objectives

After completing this lesson, you should be able to do the following:

- Start and stop the Oracle database and components
- Access a database with SQL*Plus and SQL Developer
- Modify database initialization parameters
- Describe the stages of database startup
- Describe the database shutdown options
- View the alert log
- Access dynamic performance views



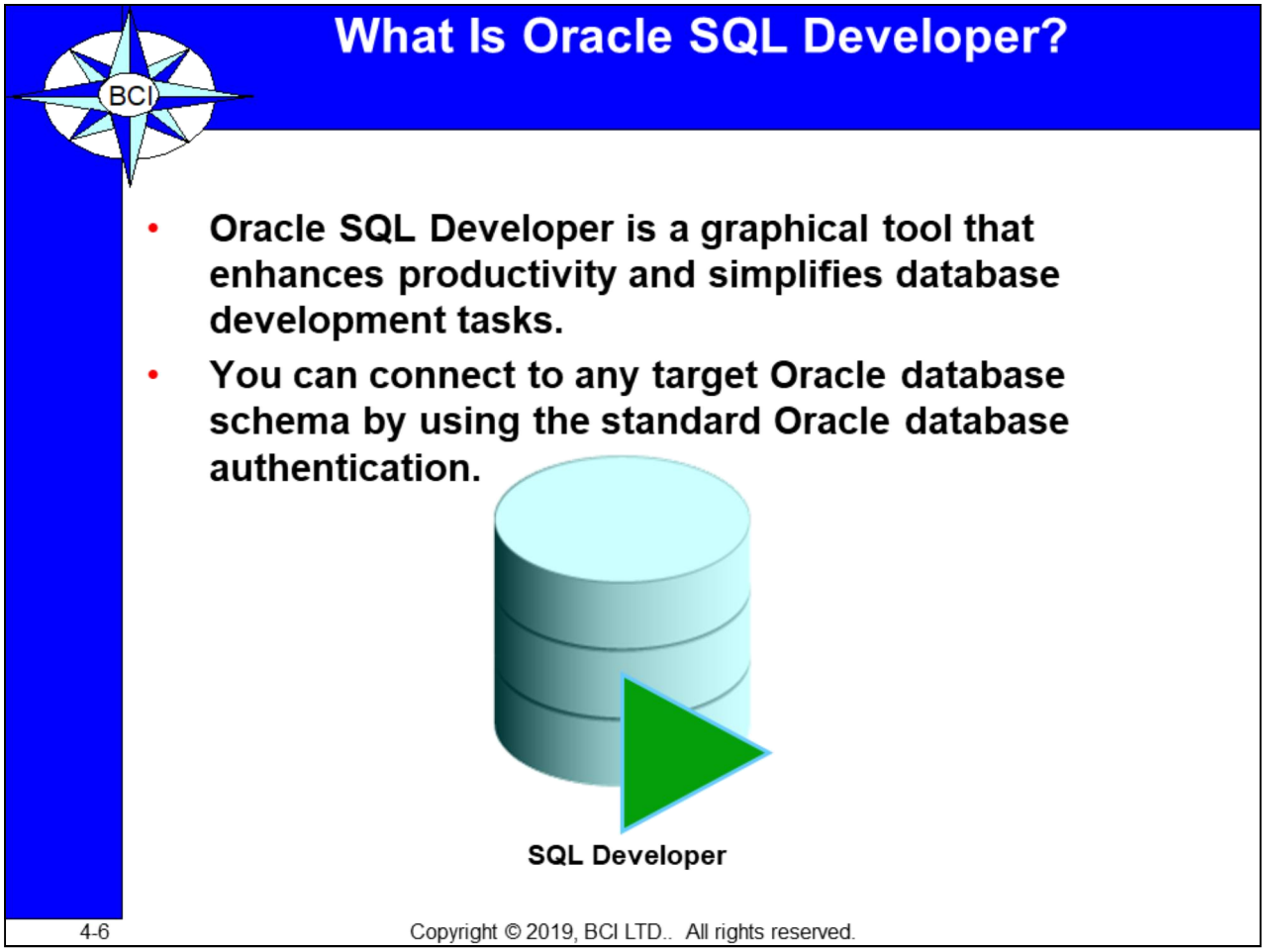
The Database Configuration Assistant in Oracle Database 12*c* allows the creation and management of new types of database:

- Multitenant container database
- Pluggable database

The creation and management of these new types of database are covered in the following module titled « Multitenant Container Databases and Pluggable Databases».

Lesson Agenda

- Start and stop the Oracle database and components
- **Access a database with SQL Developer and**
- **SQL*PLUS or SQLcl**
- Modify database initialization parameters
- Describe the stages of database startup
- Describe the database shutdown options
- View the alert log
- Access dynamic performance views



What Is Oracle SQL Developer?

Oracle SQL Developer is a free graphical tool designed to improve your productivity and simplify the development of everyday database tasks. With just a few clicks, you can easily create and debug stored procedures, test SQL statements, and view optimizer plans.

Oracle SQL Developer, the visual tool for database development, simplifies the following tasks:

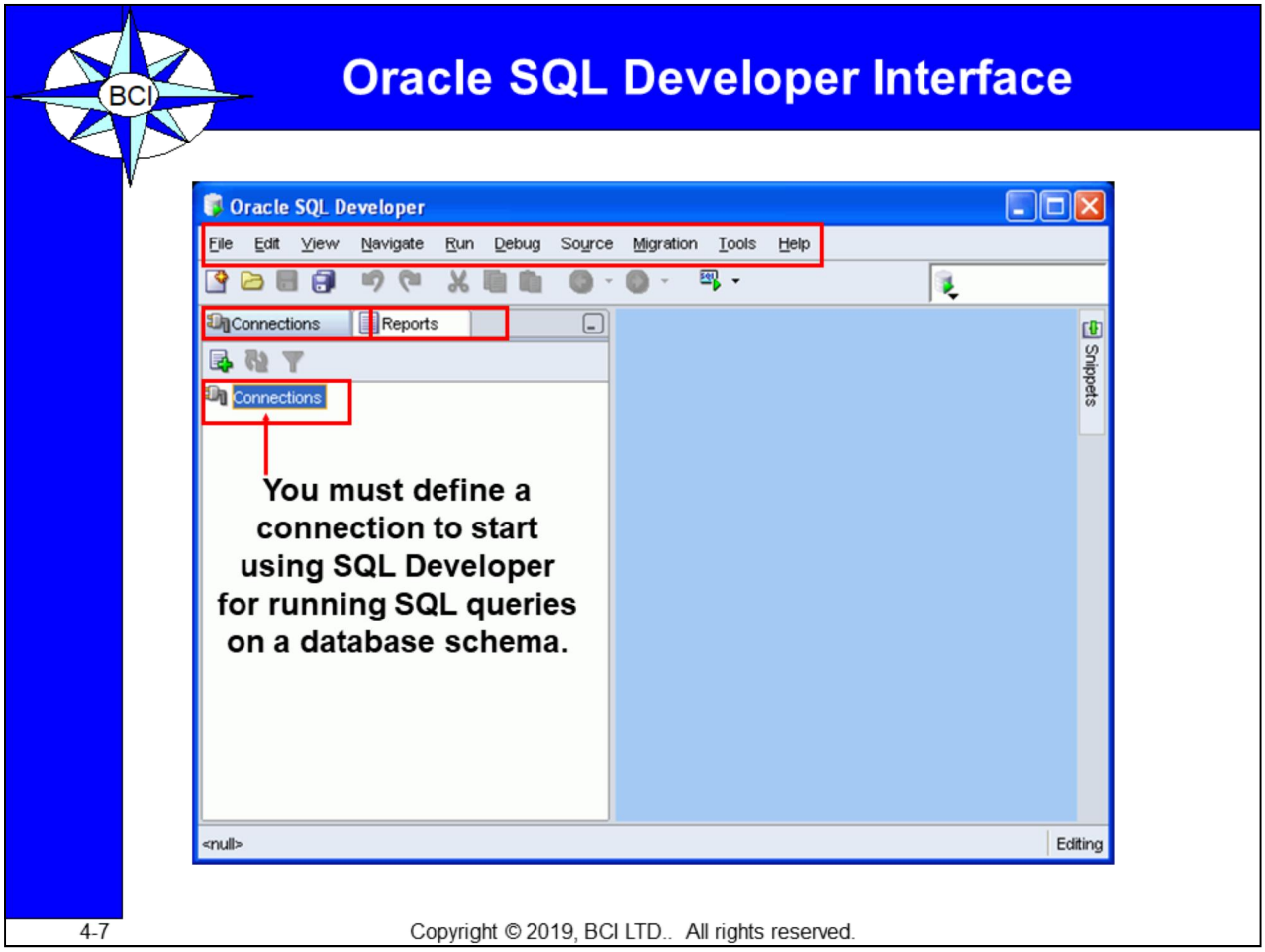
- Browsing and managing database objects
- Executing SQL statements and scripts
- Editing and debugging PL/SQL statements
- Creating reports

You can connect to any target Oracle database schema by using the standard Oracle database authentication. When connected, you can perform operations on objects in the database.

Note: The Oracle SQL Developer, Release 1.2 is called the *Migration release* because it tightly integrates with *Developer Migration Workbench*. Therefore, it provides users with a single point to browse database objects and data in third-party databases, and to migrate from these databases to Oracle. You can also connect to schemas for selected third-party (non-Oracle) databases, such as MySQL, Microsoft SQL Server, and Microsoft Access, and

view metadata and data in these databases.

Additionally, Oracle SQL Developer, Release 1.2 includes support for Oracle Application Express, Release 3.0.1 (Oracle APEX).



Oracle SQL Developer Interface

Oracle SQL Developer has two main navigation tabs:

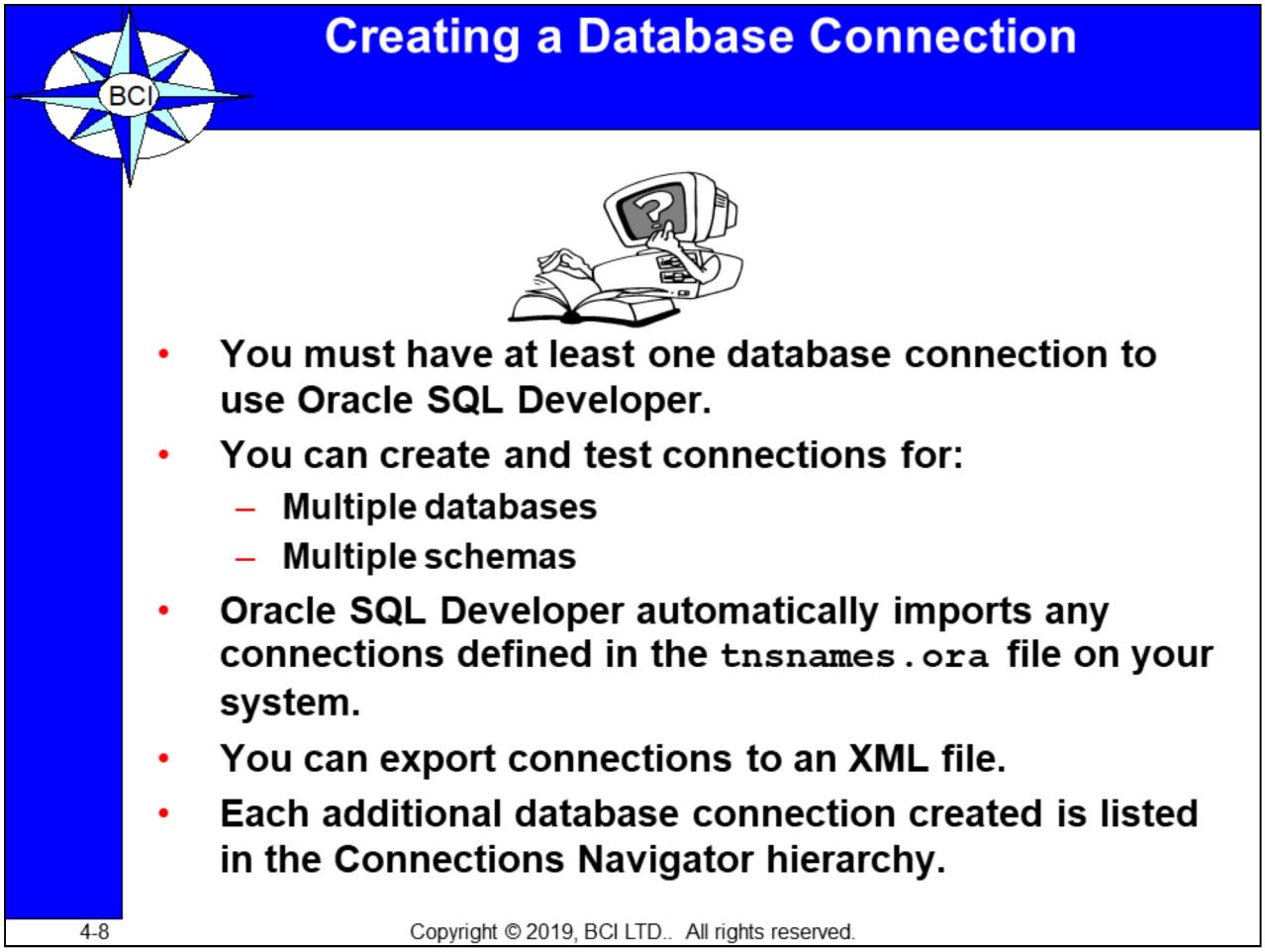
- **Connections tab:** By using this tab, you can browse database objects and users to which you have access.
- **Reports tab:** By using this tab, you can run predefined reports, or create and add your own reports.

Oracle SQL Developer uses the left pane for navigation to find and select objects, and the right pane to display information about selected objects. You can customize many aspects of the appearance and behavior of Oracle SQL Developer by setting preferences. The menus at the top contain standard entries, plus entries for features specific to Oracle SQL Developer.

1. **View:** Contains options that affect what is displayed in the Oracle SQL Developer interface
2. **Navigate:** Contains options for navigating to panes and in the execution of subprograms
3. **Run:** Contains the Run File and Execution Profile options that are relevant when a function or procedure is selected
4. **Debug:** Contains options relevant when a function or procedure is selected for debugging

5. **Source:** Contains options for use when editing functions and procedures
6. **Migration:** Contains options related to migrating third-party databases to Oracle
7. **Tools:** Invokes tools such as SQL*Plus, Preferences, and SQL Worksheet

Note: You must define at least one connection to be able to connect to a database schema and issue SQL queries or run procedures/functions.



Creating a Database Connection

A connection is an Oracle SQL Developer object that specifies the necessary information for connecting to a specific database as a specific user of that database. To use Oracle SQL Developer, you must have at least one database connection, which may be existing, created, or imported.

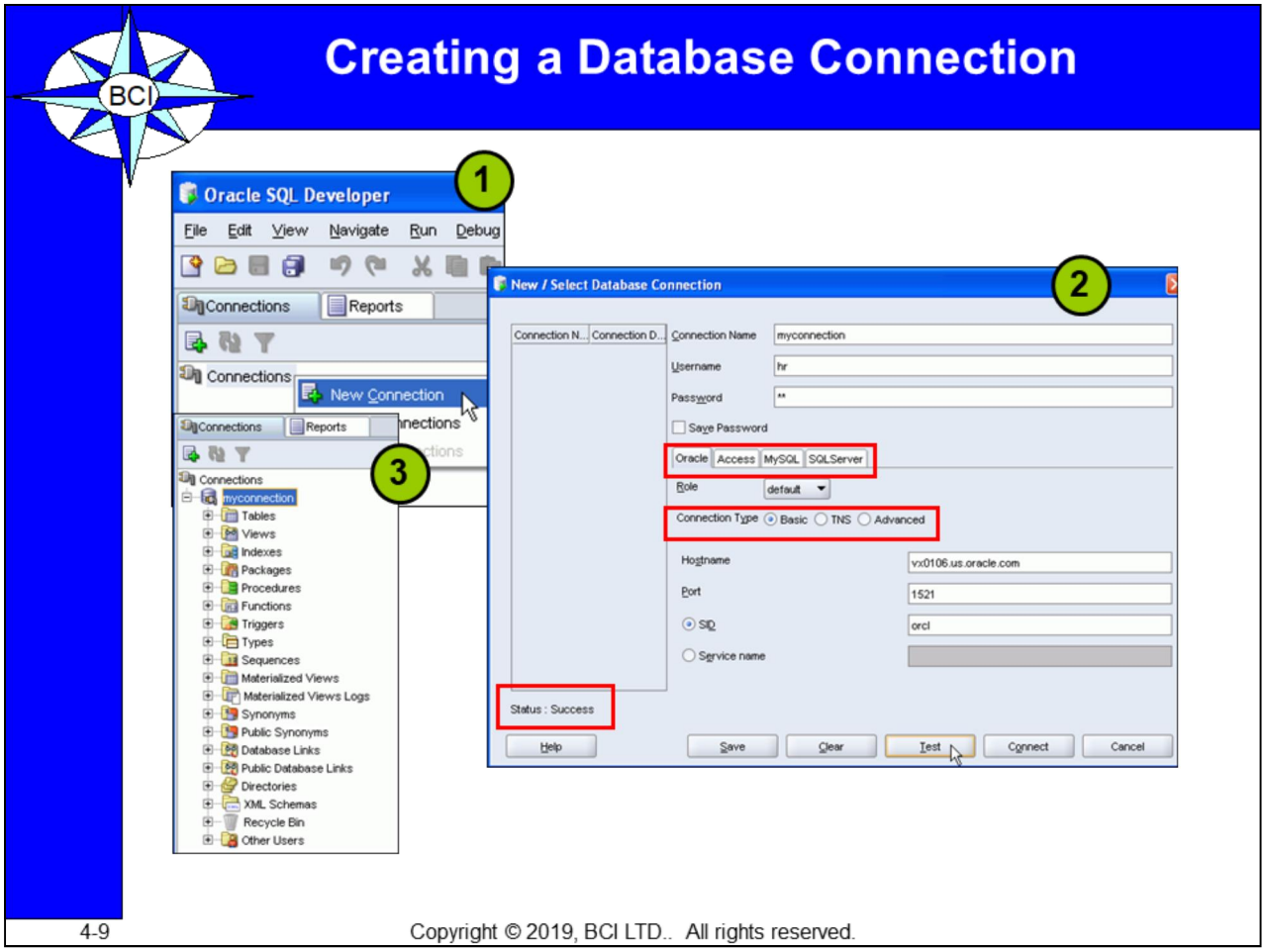
You can create and test connections for multiple databases and for multiple schemas.

By default, the `tnsnames.ora` file is located in the `$ORACLE_HOME/network/admin` directory. But, it can also be in the directory specified by the `TNS_ADMIN` environment variable or the registry value. When you start Oracle SQL Developer and display the Database Connections dialog box, Oracle SQL Developer automatically imports any connections defined in the `tnsnames.ora` file on your system.

Note: On Windows systems, if the `tnsnames.ora` file exists but Oracle SQL Developer is not using its connections, define `TNS_ADMIN` as a system environment variable.

You can export connections to an XML file so that you can reuse it later.

You can create additional connections as different users to the same database or to connect to the different databases.



Creating a Database Connection (continued)

To create a database connection, perform the following steps:

1. On the Connections tabbed page, right-click Connections and select New Connection.
2. In the New/Select Database Connection window, enter the connection name. Enter the username and password of the schema that you want to connect to.
    1. From the Role drop-down list, you can select either *default* or SYSDBA (you will select SYSDBA for the `sys` user or any user with DBA privileges).
    2. You can select the connection type as:
    - Basic: In this type, you enter the host name and system identifier (SID) for the database that you want to connect to. The Port is already set to 1521. Or, you can also enter the Service name directly if you are using a remote database connection.
    - TNS: You select any one of the database aliases imported from the `tnsnames.ora` file
    - Advanced: You define a custom JDBC URL to connect to the database.
    3. Click Test to make sure that the connection has been set correctly.
    4. Click Connect.

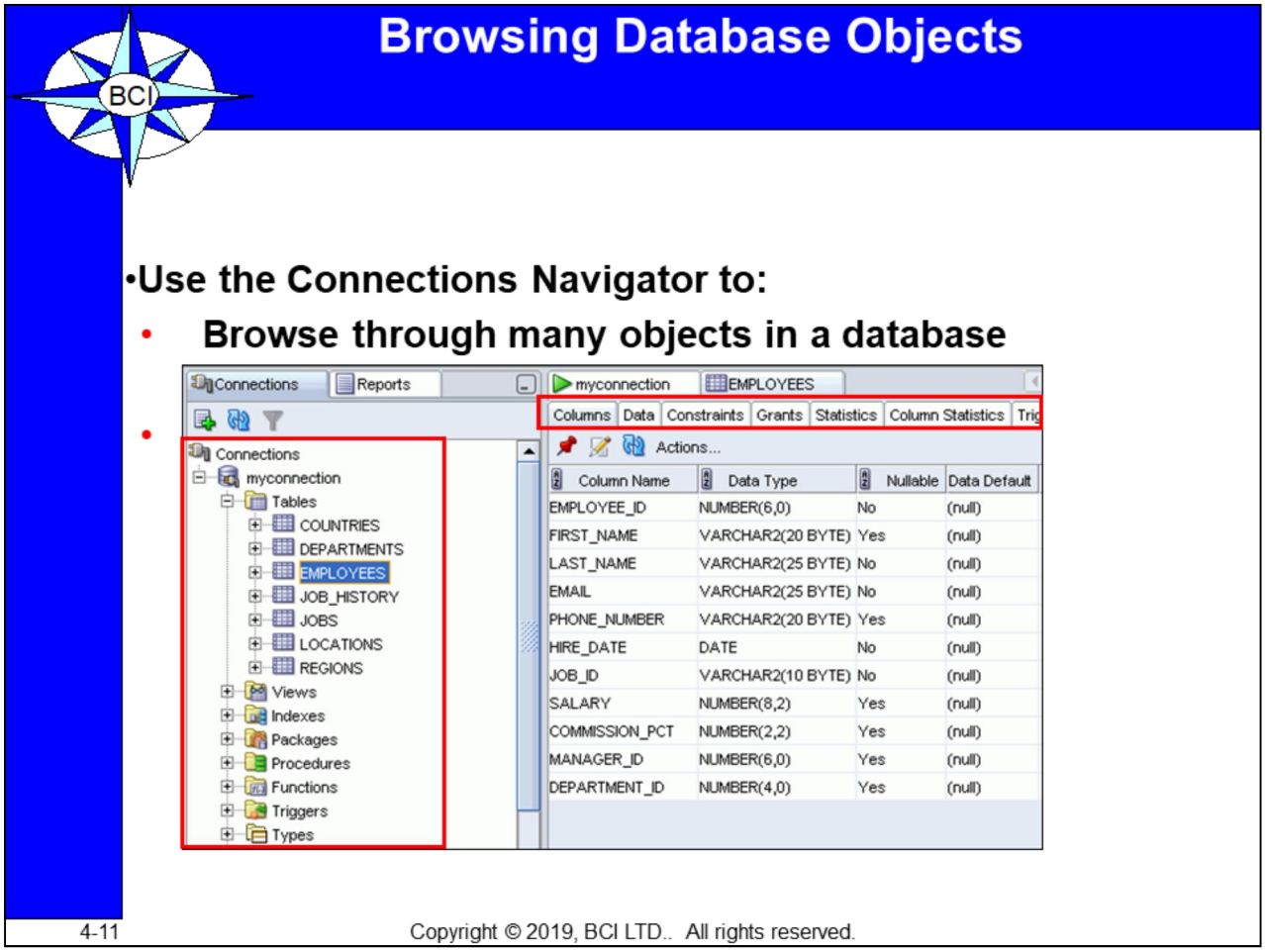
If you select the Save Password check box, the password is saved to an XML file. So, after

you close the Oracle SQL Developer connection and open it again, you will not be prompted for the password.

Creating a Database Connection (continued)

3. The connection gets added in the Connections Navigator. You can expand the connection to view the database objects and view object definitions, for example, dependencies, details, statistics, and so on.

Note: From the same New/Select Database Connection window, you can define connections to non-Oracle data sources by using the Access, MySQL, and SQL Server tabs. However, these connections are read-only connections that enable you to browse objects and data in that data source.



Browsing Database Objects

After you have created a database connection, you can use the Connections Navigator to browse through many objects in a database schema including Tables, Views, Indexes, Packages, Procedures, Triggers, Types, and so on.

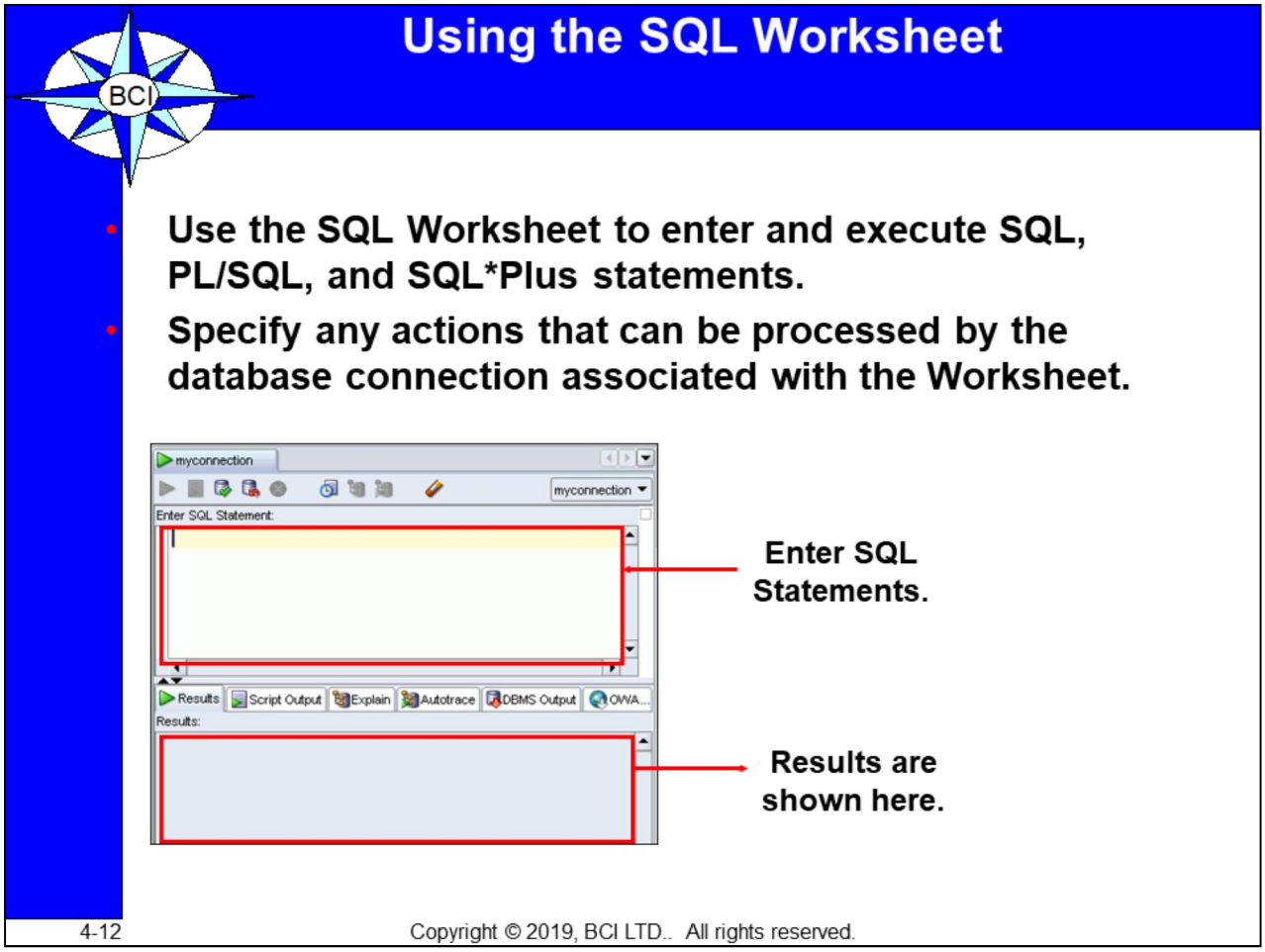
Oracle SQL Developer uses the left pane for navigation to find and select objects and the right pane to display information about the selected objects. You can customize many aspects of the appearance of Oracle SQL Developer by setting preferences.

You can see the definition of the objects broken into tabs of information that is pulled out of the data dictionary. For example, if you select a table in the Navigator, the details about columns, constraints, grants, statistics, triggers, and so on are displayed in an easy-to-read tabbed page.

If you want to see the definition of the `EMPLOYEES` table as shown in the slide, perform the following steps:

1. Expand the Connections node in the Connections Navigator.
2. Expand Tables.
3. Click EMPLOYEES. By default, the Columns tab is selected. It shows the column description of the table. By using the Data tab, you can view the tables data and also

enter new rows, update data, and commit these changes to the database.



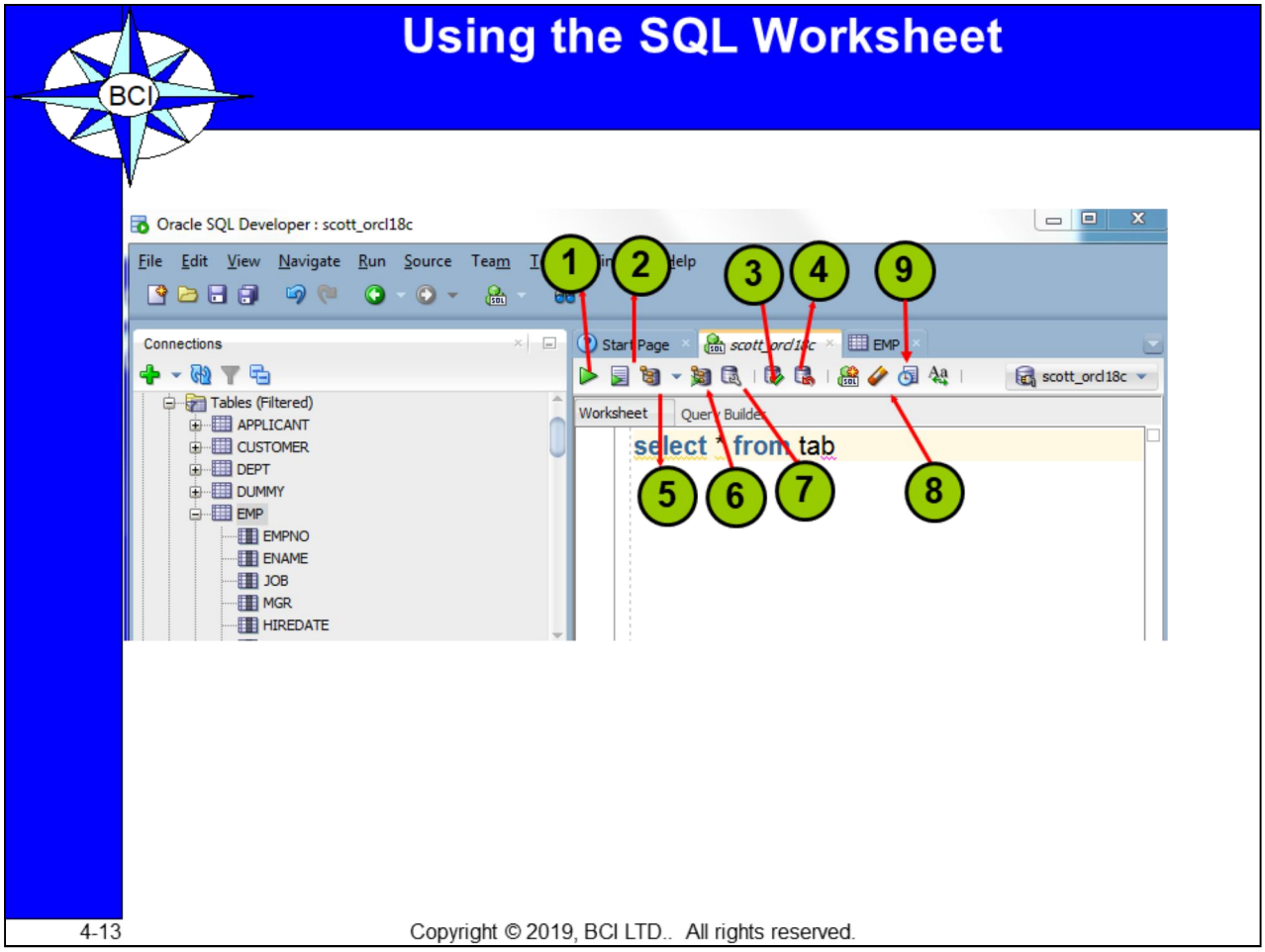
Using the SQL Worksheet

When you connect to a database, a SQL Worksheet window for that connection is automatically opened. You can use the SQL Worksheet to enter and execute SQL, PL/SQL, and SQL*Plus statements. All SQL and PL/SQL commands are supported as they are passed directly from the SQL Worksheet to the Oracle database. However, the SQL*Plus commands used in Oracle SQL Developer have to be interpreted by the SQL Worksheet before being passed to the database.

The SQL Worksheet currently supports a number of SQL*Plus commands. Those commands that are not supported by the SQL Worksheet are ignored and not sent to the Oracle database. Through the SQL Worksheet, you can execute SQL statements and some of the SQL*Plus commands.

You can display a SQL Worksheet by using any of the following two options:

- Select Tools > SQL Worksheet.
- Click the Open SQL Worksheet icon available on the main toolbar.



Using the SQL Worksheet (continued)

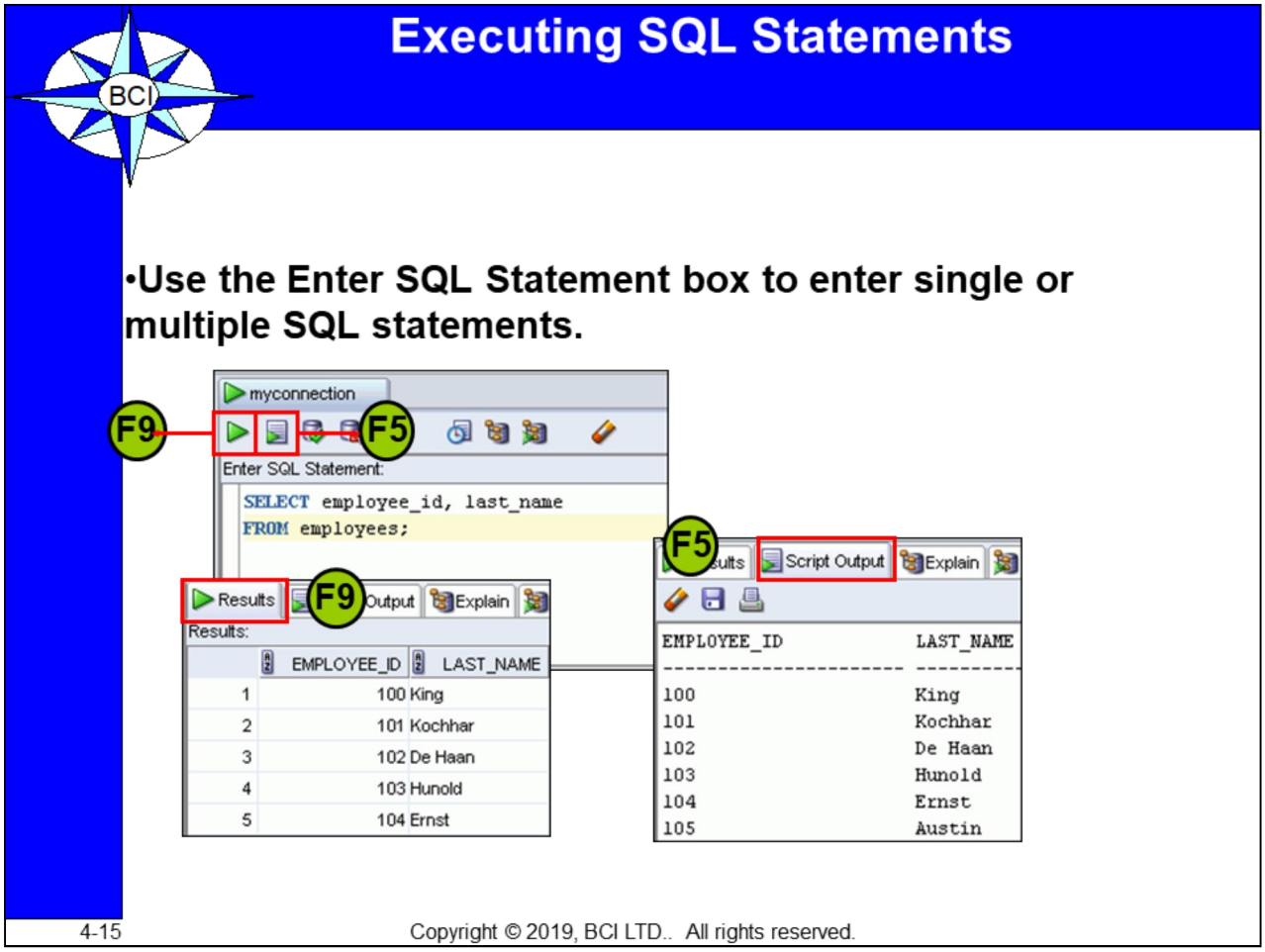
You may want to use shortcut keys or icons to perform certain tasks, such as executing a SQL statement, running a script, or viewing the history of the SQL statements that you have executed.

You can use the SQL Worksheet toolbar that contains icons to perform the following tasks:

1. **Execute Statement:** This executes the statement at the cursor in the Enter SQL Statement box. Alternatively, you can press [F9]. The output is generally shown in a formatted manner in the Results tab page.
2. **Run Script:** This executes all statements in the Enter SQL Statement box using the Script Runner. The output is generally shown in the conventional script format in the Scripts tab page.
3. **Commit:** This writes any changes to the database and ends the transaction.
4. **Rollback:** This discards any changes to the database, without writing them to the database, and ends the transaction.
5. **Explain Plan:** Provides optimizer statistics (COST, IO etc.) on statements currently being executed.

Using the SQL Worksheet (continued)

6. **Autotrace:** This displays trace-related information when you execute the SQL statement by clicking the Autotrace icon. This information can help you to identify the SQL statements that will benefit from tuning.
7. **SQL Tuning:** This generates the execution plan, which you can see by clicking the Explain tab.
8. **Clear:** This erases the statement or statements in the Enter SQL Statement box. Alternatively, press and hold [Ctrl] + [D] to erase the statements.
9. **SQL History:** This displays a dialog box with information about the SQL statements that you have executed.



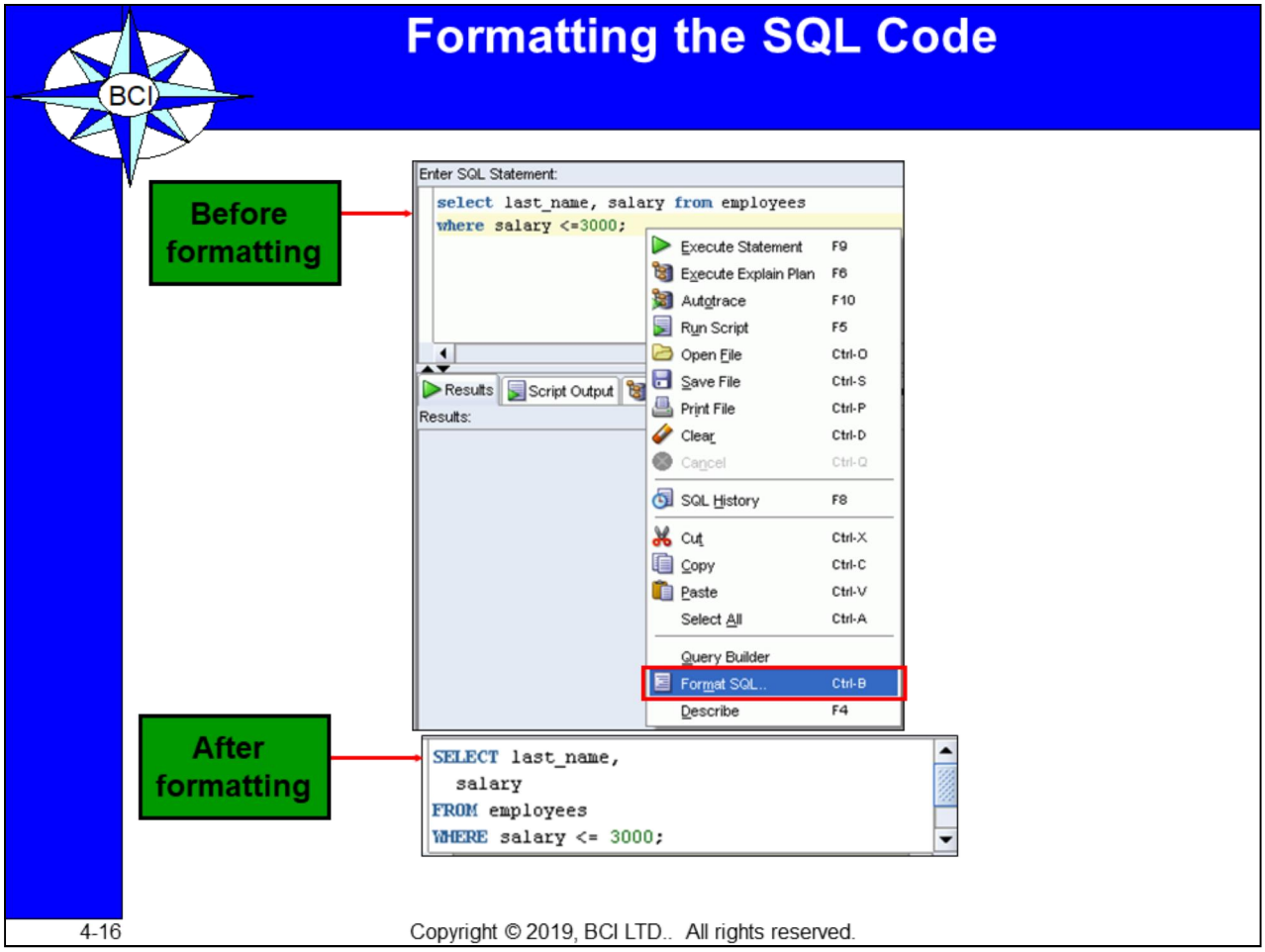
Executing SQL Statements

In the SQL Worksheet, you can use the Enter SQL Statement box to enter a single or multiple SQL statements. For a single statement, the semicolon at the end is optional.

When you enter the statement, the SQL keywords are automatically highlighted. To execute a SQL statement, ensure that your cursor is within the statement and click the Execute Statement icon. Alternatively, you can press [F9].

To execute multiple SQL statements and see the results, click the Run Script icon. Alternatively, you can press [F5].

The example in the slide shows the difference in output for the same query when F9 key or Execute Statement is used versus the output when F5 or Run Script is used.

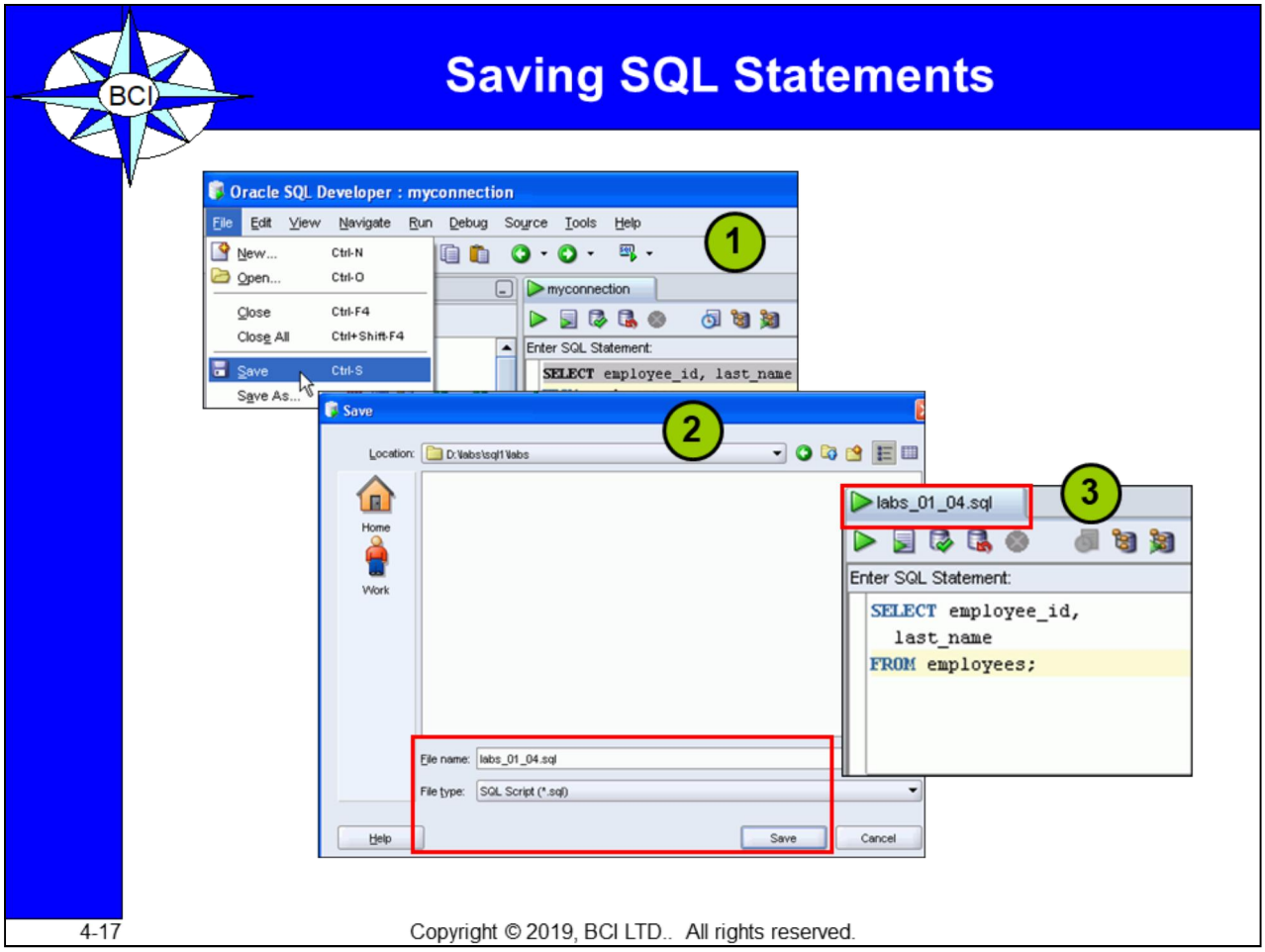


Formatting the SQL Code

You may want to beautify the indentation, spacing, capitalization, and the line separation of the SQL code. Oracle SQL Developer has the feature for formatting the SQL code.

To format the SQL code, right-click in the statement area, and select Format SQL.

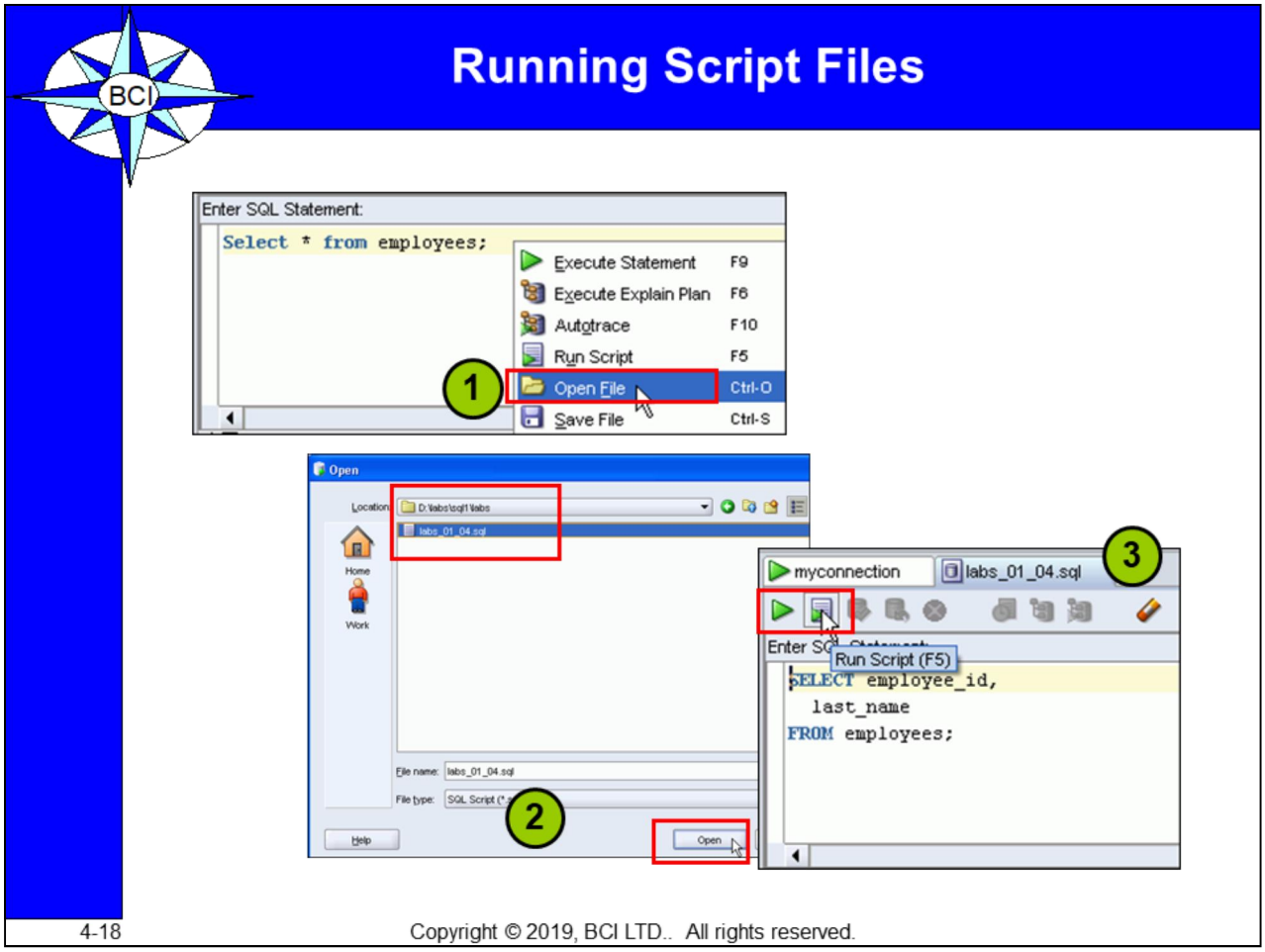
In the example in the slide, before formatting, the keywords are not capitalized and the statement is not properly indented in the SQL code. After formatting, the SQL code is beautified with the keywords capitalized and the statement properly indented.



Saving SQL Statements

In most of the practices that you will perform, you will need to save a particular query in the SQL Worksheet as a `.sql` file. To do so, perform the following:

1. From the File menu, select Save or Save As (if you are renaming a current `.sql` script) or you can right-click the SQL Worksheet and select Save File. Alternatively, you can press and hold [CTRL] + [S].
2. In the Save dialog box, enter the appropriate filename. Make sure the extension is `.sql` or the File type is selected as SQL Script (*.sql). Click Save.
   Note: For this course, you need to save your `sql` scripts in the `D:\labs\sql1\labs` folder.
3. The SQL Worksheet is renamed to the filename that you saved the script as. Make sure you do not enter any other SQL statements in the same worksheet. To continue with other SQL queries, open a new worksheet.

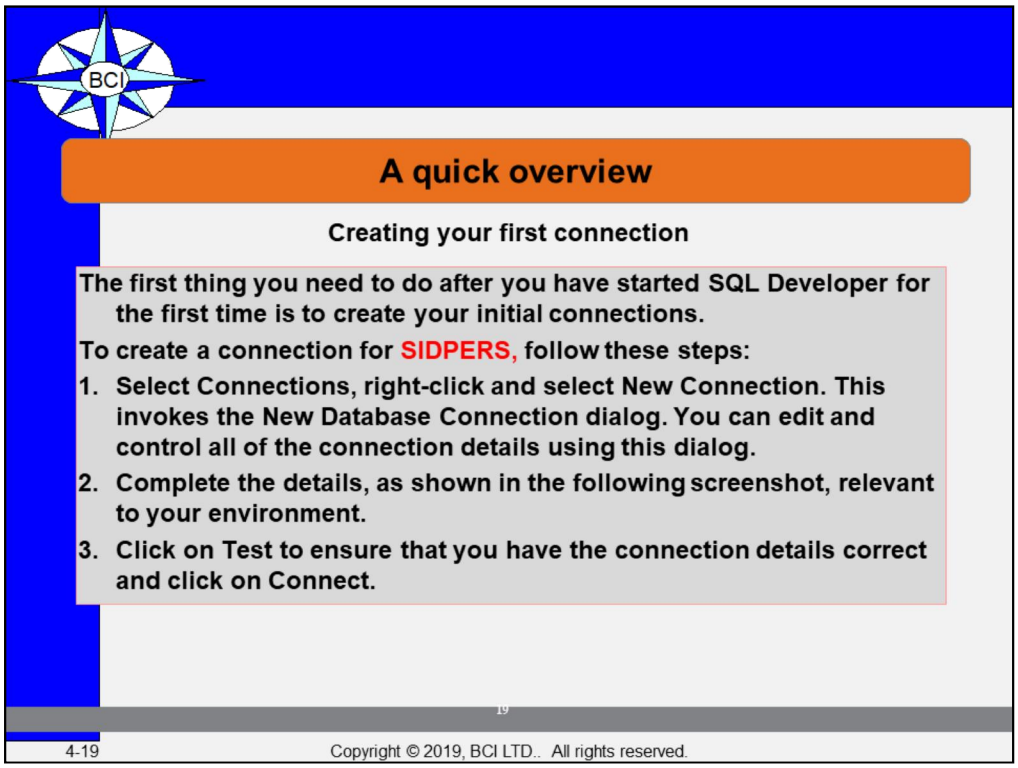


Running Script Files

To run the saved `.sql` script files, perform the following:

1. Right-click the SQL Worksheet and select Open File, or select Open from the File menu. Alternatively, you can press and hold [CTRL] + [O].
2. In the Open dialog box, move to the `D:\labs\sql\labs` folder, or to the location in which you saved the script file, select the file and click Open.
3. The script file opens in a new worksheet. Now, you can run the script by either clicking the Execute Statement icon or the Run Script icon. Again, make sure you do not enter any other SQL statements in the same worksheet. To continue with other SQL queries, open a new worksheet.
   Note: You may want to set the default directory to `D:\labs\sql1` folder, so that every time you try to open or save a script, SQL Developer chooses the same path to look for scripts. From Tools menu, select Preferences. In the Preferences dialog box, expand Database and select Worksheet Parameters. In the right pane, click Browse to set the default path to look for scripts and click OK.

Note: For more details on how to use the Oracle SQL Developer GUI interface for other data objects creation and data retrieval tasks, refer to Appendix G “Performing DML and DDL Operations Using the Oracle SQL Developer GUI.”



To complete this quick walk-through, you need to know the username and password of the SYSTEM user. You also need to know the location of the database, whether this is the machine name or the IP address, and the database SID.

To begin, start SQL Developer. The very first time you start SQL Developer, you'll be asked if you want to migrate from a previous version. Select No and allow the tool to start up.

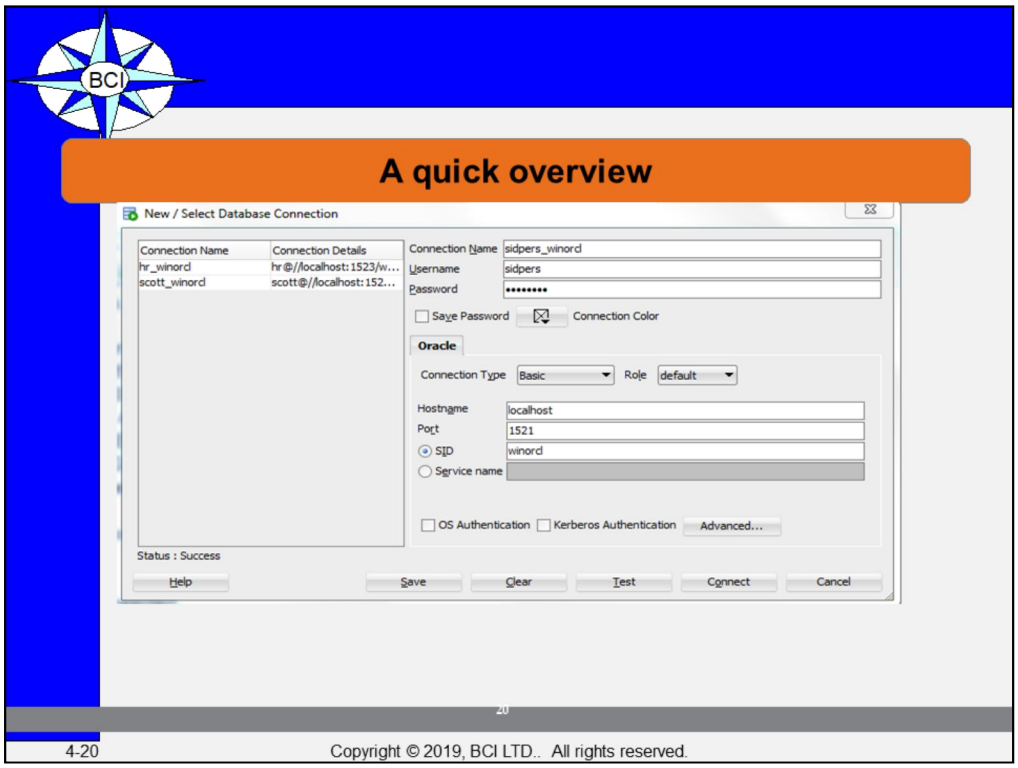
The first thing you need to do after you have started SQL Developer for the first time is to create your initial connections.

To create a connection for SYSTEM, follow these steps:

Select Connections, right-click and select New Connection. This invokes the New Database Connectiondialog. You can edit and control all of the connection details using this dialog.

Complete the details, as shown in the following screenshot, relevant to your environment.

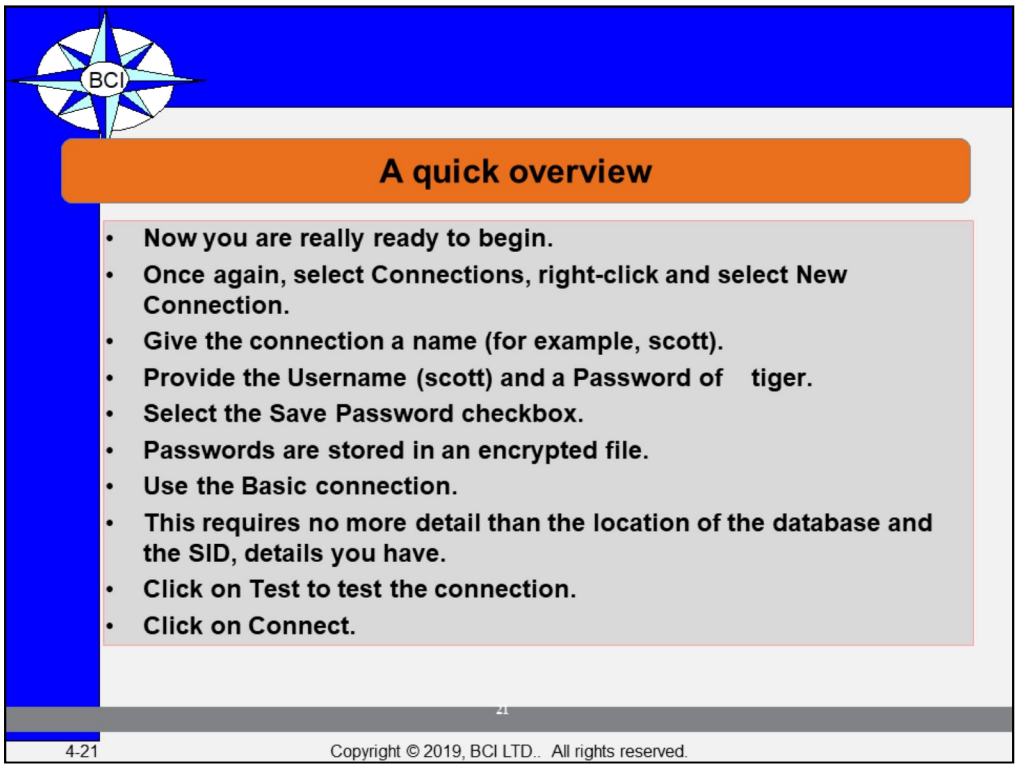
Click on Test to ensure that you have the connection details correct and click on Connect.



You are now connected as SIDPERS. Use this connection to verify your other users, by continuing with following steps.

Select the new connection you have created, expand the node, and scroll down to Other Users.

Expand Other Users and find the user HR. Right-click on it and select Edit User. Verify that the account for HR is unlocked and the Password has not expired, that is, the properties Account is Locked and Password Expired are deselected. If either of these is selected, deselect them. You can change the password for HR at this point too. It's good practice to modify the passwords of the shipped sample schemas once you have unlocked them.



Now you are really ready to begin.

Once again, select Connections, right-click and select New Connection.

Give the connection a name (for example, HR_11g).

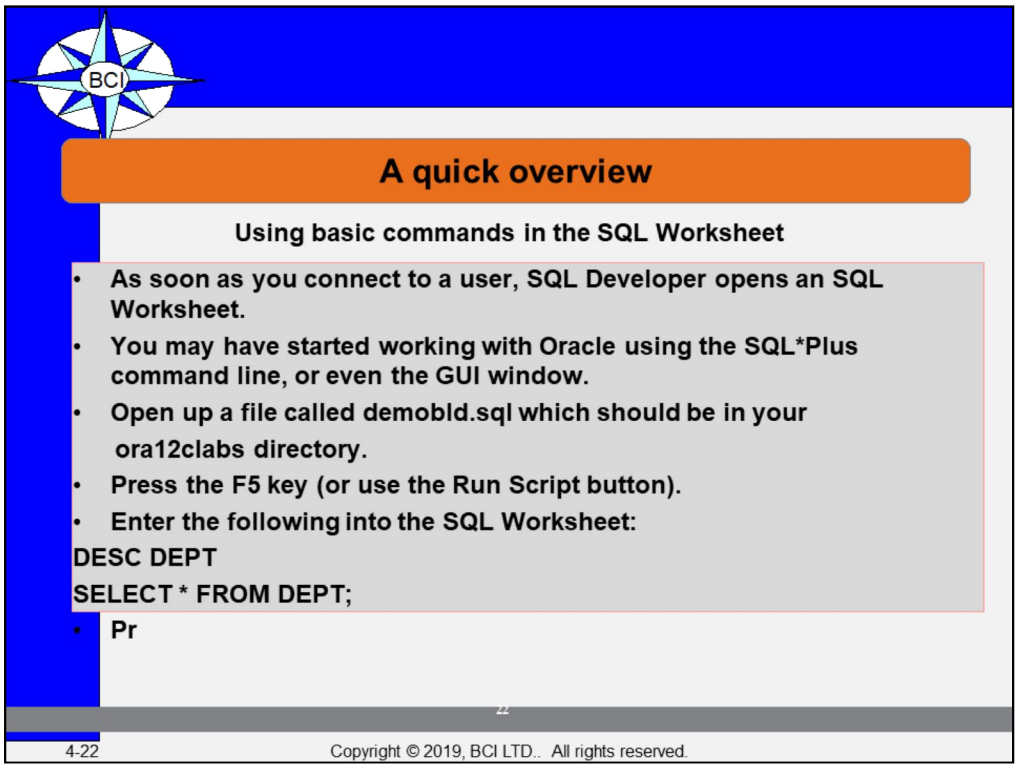
Provide the Username (HR) and a Password. If you are working on Oracle Database 12c, be aware that passwords are now case sensitive.

Select the Save Password checkbox. This makes life easy while you are working with SQL Developer. Passwords are stored in an encrypted file. However, you should always be aware of saving passwords and possible security implications this may have.

Use the Basic connection. This requires no more detail than the location of the database and the SID, details you have.

Click on Test to test the connection.

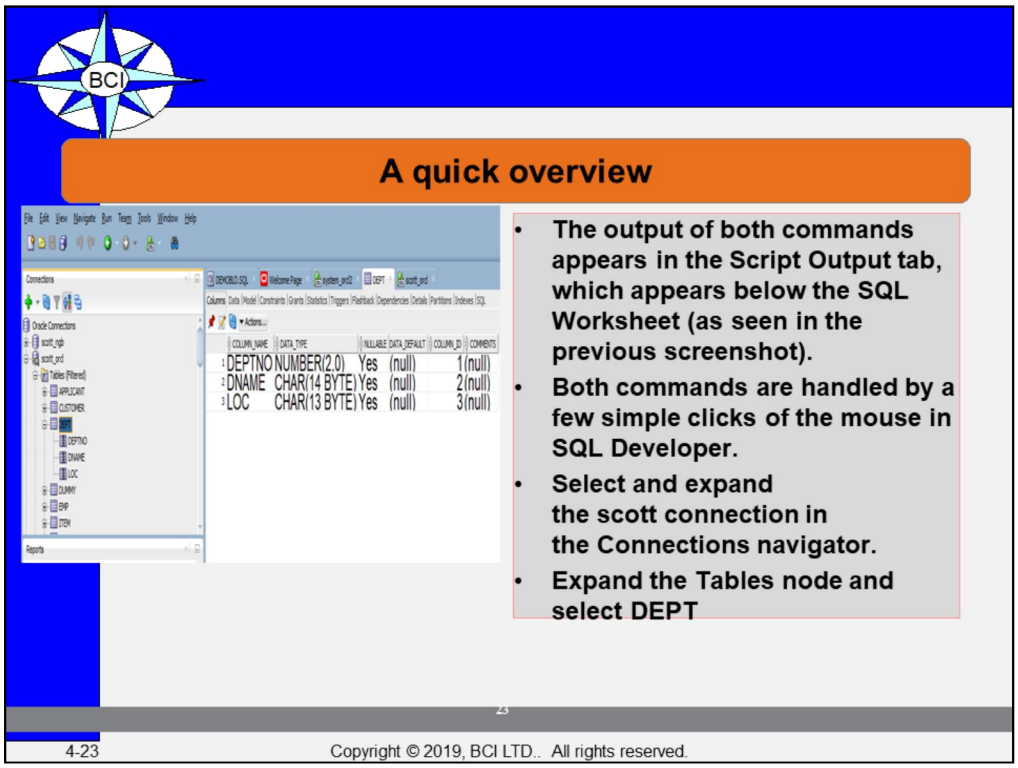
Click on Connect.



As soon as you connect to a user, SQL Developer opens an SQL Worksheet. You may have started working with Oracle using the SQL*Plus command line, or even the GUI window. Either way, you'd start with a selection of SQL*Plus and SQL commands.

Enter the following into the SQL Worksheet:

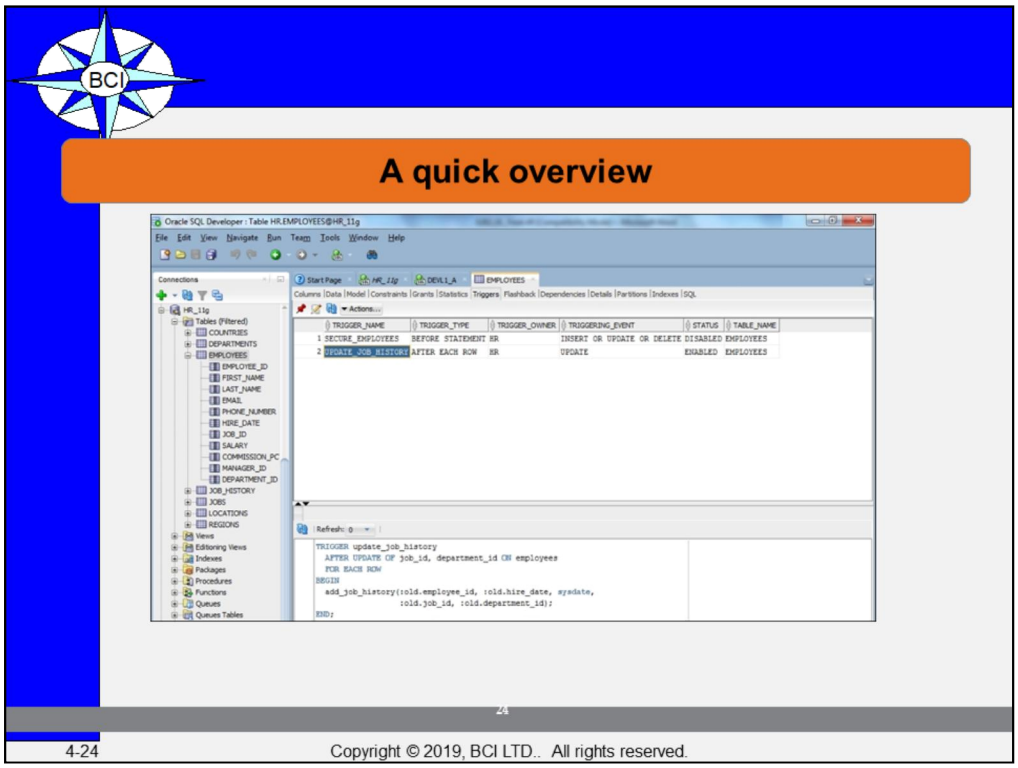
Press the F5 key (or use the Run Script button).



The output of both commands appears in the Script Output tab, which appears below the SQL Worksheet (as seen in the previous screenshot). Both commands are handled by a few simple clicks of the mouse in SQL Developer.

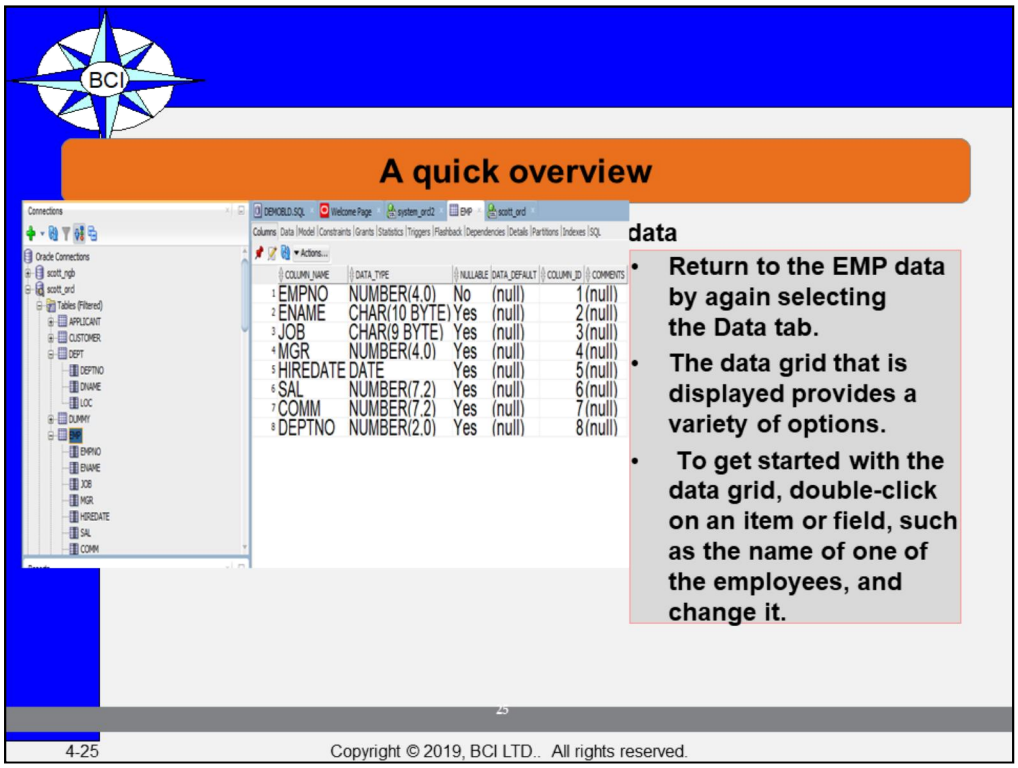
Select and expand the HR_11g connection in the Connections navigator. Expand the Tables node and select DEPARTMENTS.

The DEPARTMENTS tab now opens, displaying a list of the column names and details. These are the same details as given by the DESC (describe) SQL*Plus command that you entered in the SQL Worksheet. It also provides additional detail, such as the Primary Key and column comments.



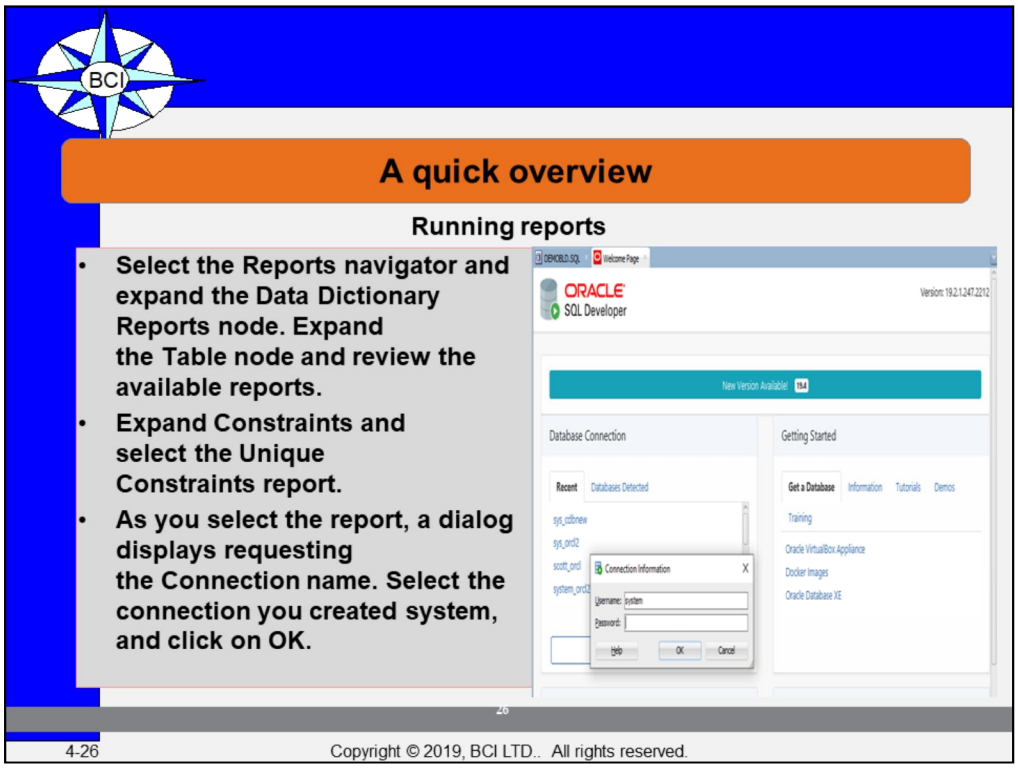
Select the Data tab and notice that you now see the output from your second command. These two tabs are included with a number of other tabs, each with additional details about the DEPT table. You would need to write a number of SQL queries in order to get the additional detail from the data dictionary if you were working in SQL*Plus.

Select the EMP table. Notice that the new table, EMPLOYEES, immediately replaces the previous DEPT table with its details. Select the Triggers tab, and select one of the triggers. The trigger and related trigger detail is displayed in a master-detail window, as shown in the following screenshot:



Return to the EMP data by again selecting the Data tab. The data grid that is displayed provides a variety of options. To get started with the data grid, double-click on an item or field, such as the name of one of the employees, and change it. Tab out of the field and notice that the change is applied to the data grid and an asterisk (*) flags the record. Commit and Rollback buttons are available to send the change to the database, or to undo your action. Roll back the changes.

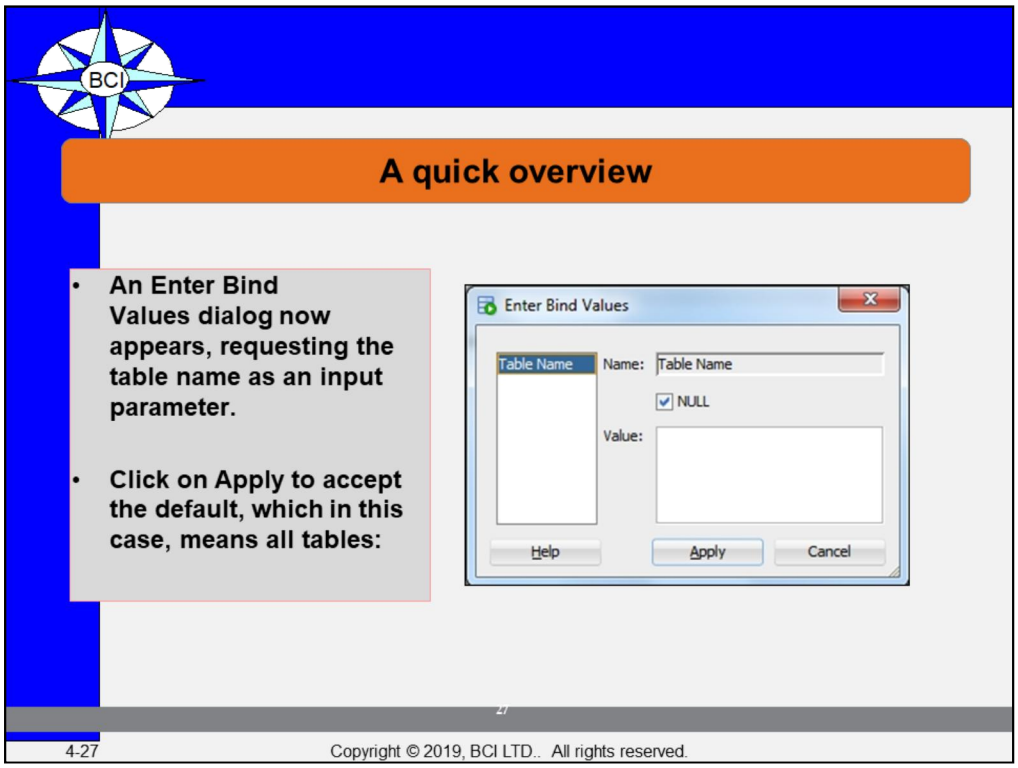
Once again, you get feedback, this time in the Messages Log, as shown in the following screenshot:



To run your report, perform the following steps:

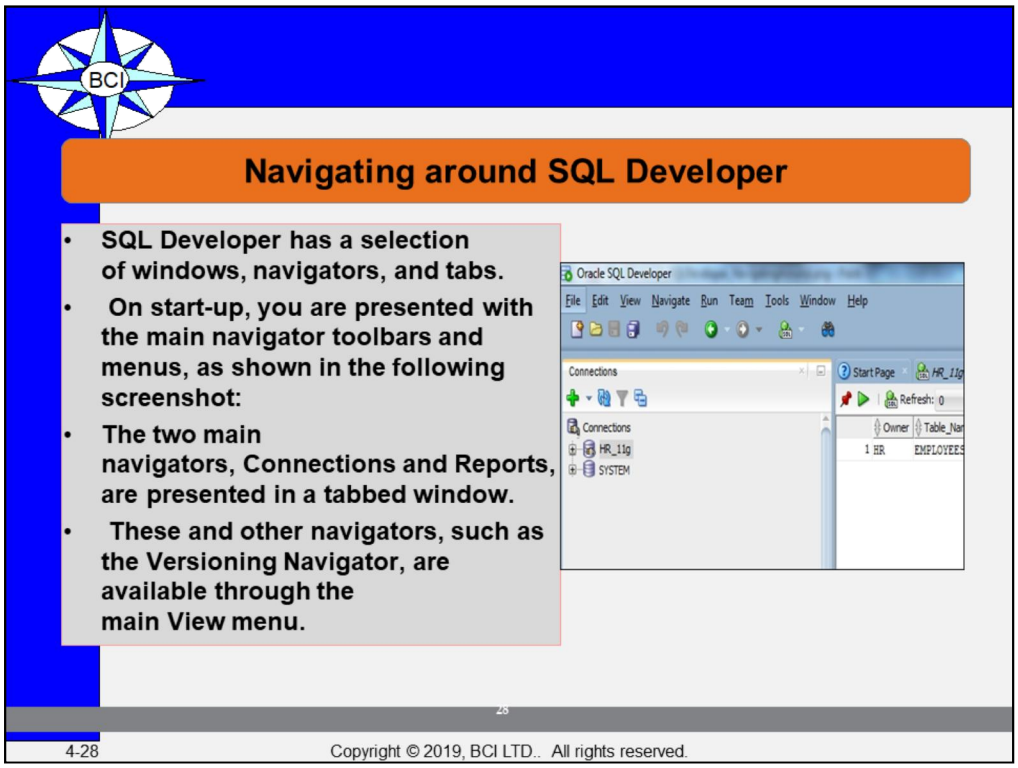
Select the Reports navigator and expand the Data Dictionary Reports node. Expand the Table node and review the available reports.
Expand Constraints and select the Unique Constraints report.

As you select the report, a dialog displays requesting the Connection name. Select the connection youcreated, HR_11g, and click on OK.



An Enter Bind Values dialog now appears, requesting the table name as an input parameter. Click on Applyto accept the default, which in this case, means all tables:Note

Run the same report for any user by selecting the Connections drop-down list on the right-hand side.

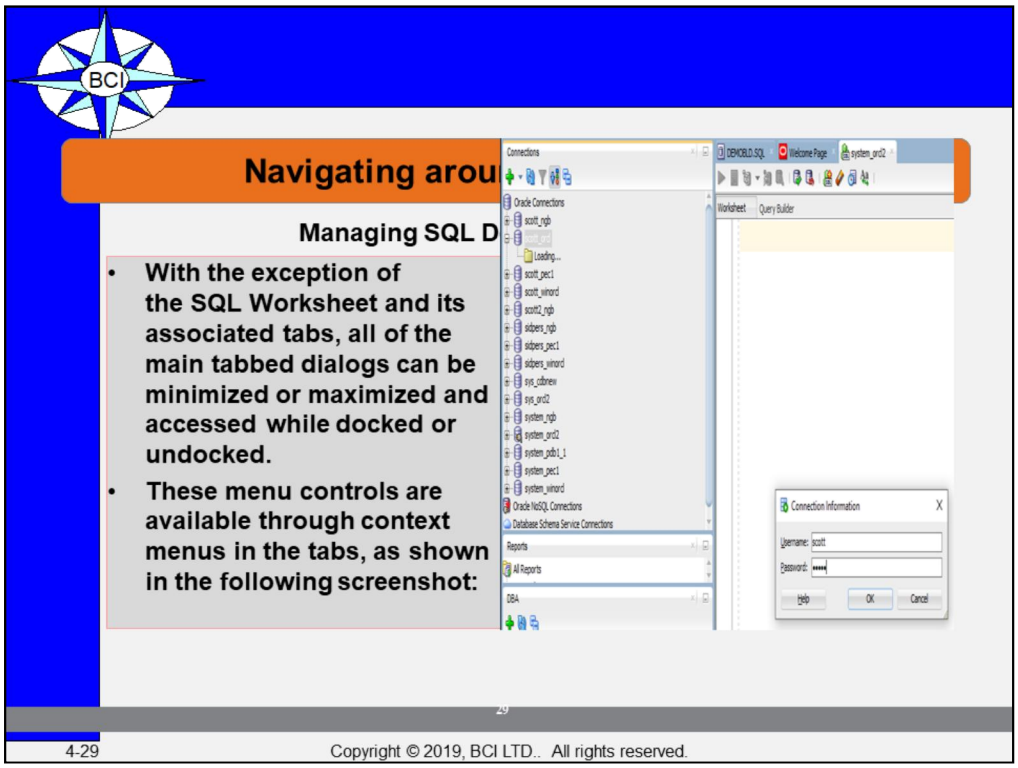


SQL Developer has a selection of windows, navigators, and tabs. On start-up, you are presented with the main navigator toolbars and menus, as shown in the following screenshot:

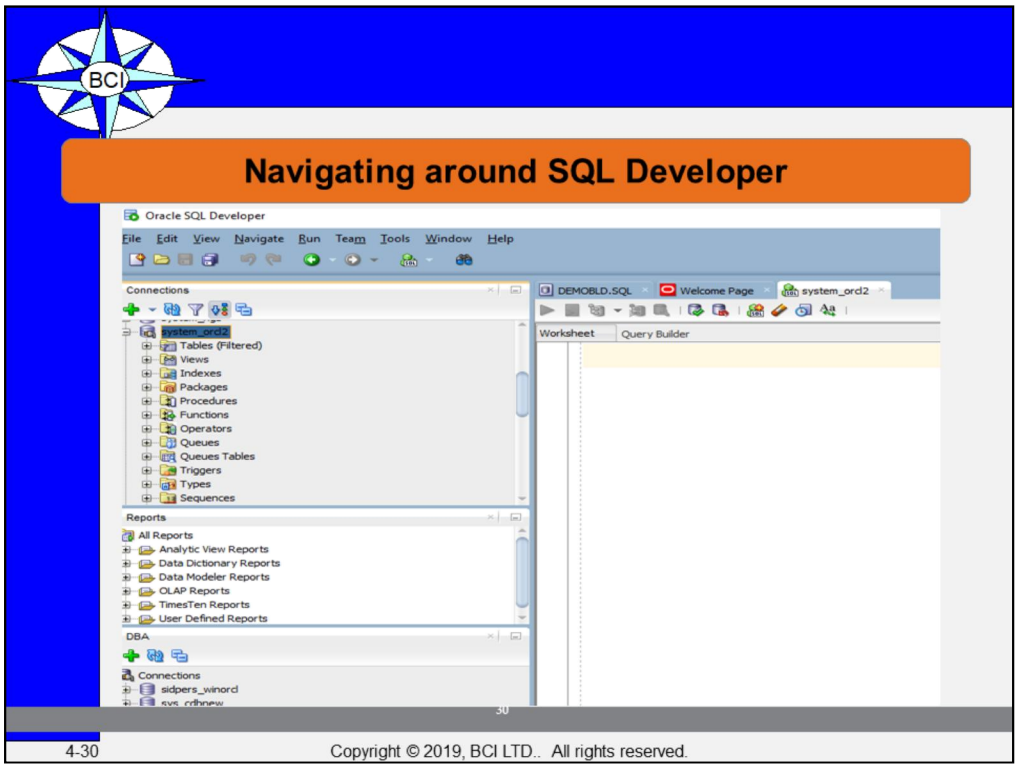
The two main navigators, Connections and Reports, are presented in a tabbed window. These and other navigators, such as the Versioning Navigator, are available through the main View menu. You can also open windows such as Snippets, Recent Objects, and Find DB Objects using the View menu.

Note

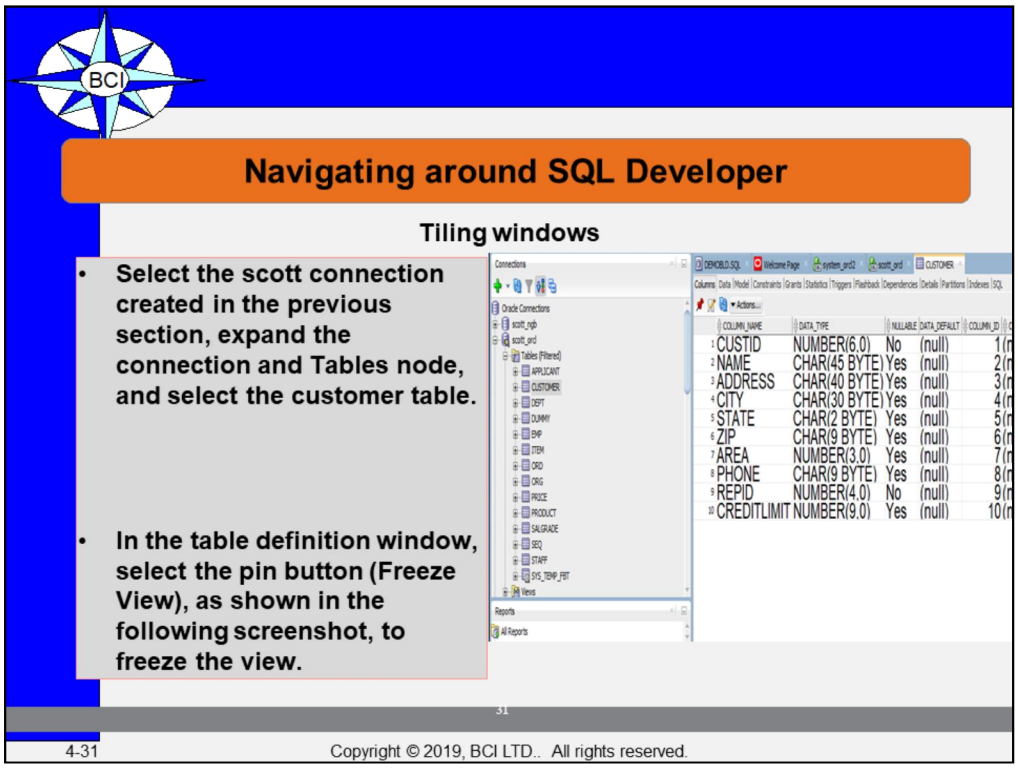
Any navigators that you open during a session, and that are still open when you close the product, are automatically opened when you restart the product



With the exception of the SQL Worksheet and its associated tabs, all of the main tabbed dialogs can be minimized or maximized and accessed while docked or undocked. These menu controls are available through context menus in the tabs, as shown in the following screenshot:You can rearrange tabbed windows by selecting and dragging the tab into place. Once any window is minimized, roll your mouse over the minimized tab to display a floating window that stays active while your mouse lies over it and rolls back into place when you move off. This is very useful when working with temporary windows such as Snippetsand Find DB Object. The following screenshot shows the floating window for the Snippets dialog. If you roll the mouse over the area, you can work in the window (for example, navigating about until you have located the snippet of code you are after, and then dragging the code onto the worksheet). The window will minimize out of the way once you have moved off it.



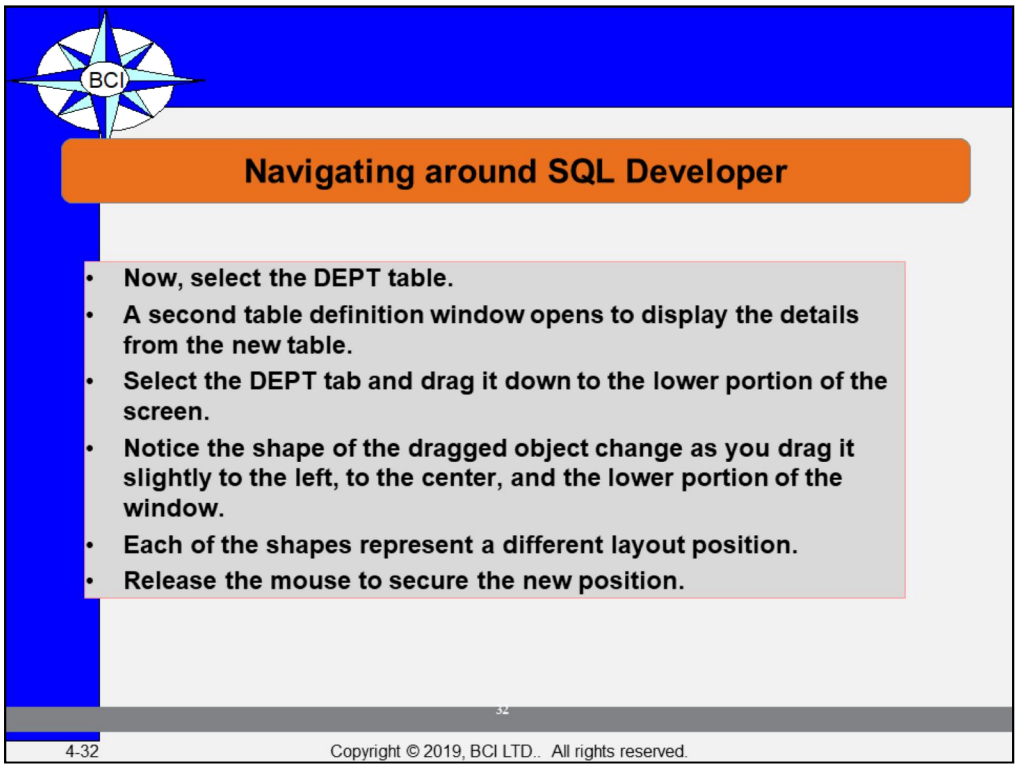
You can undock the floating window, move it off to one side, and keep it undocked while you work with the SQL Worksheet. In a dual-monitor setup, you can drag the floating window onto one monitor, while working with the SQL Worksheet on the other monitor.



Let's create another user called Scott with a password of <password>. Now let's run a script called demobld.sql. Once you start working with connections, you will have more windows and tabs to deal with, especially if you have more than one connection created.

Select the scott connection created in the previous section, expand the connection and Tables node, and select EMP.

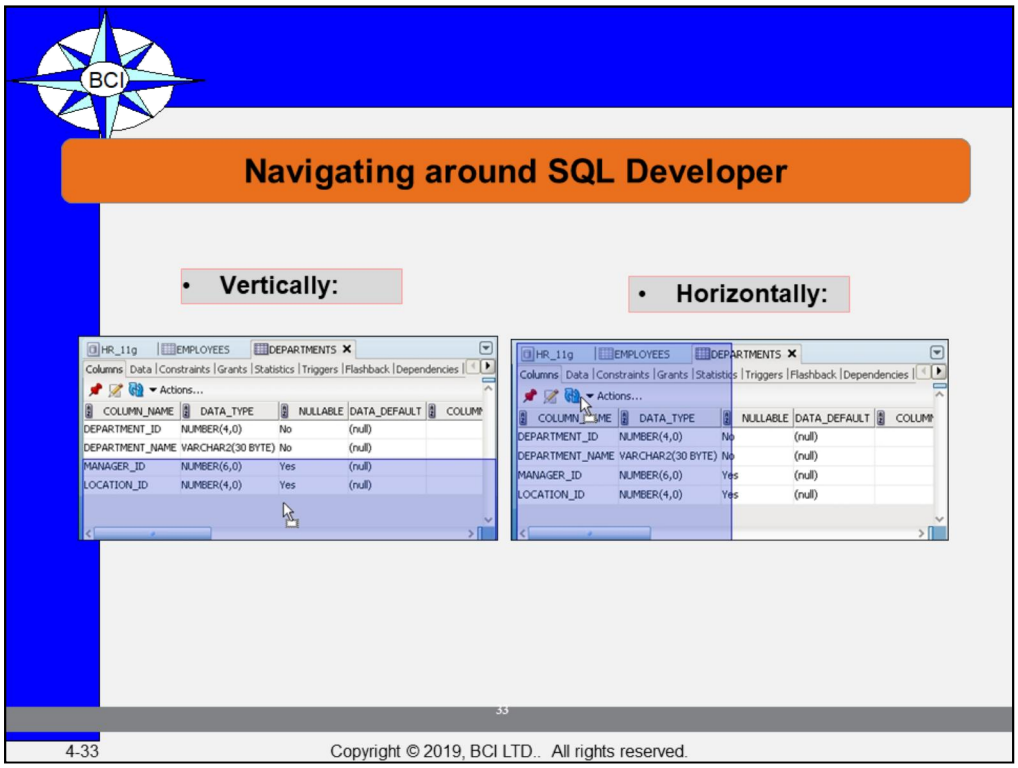
In the table definition window, select the pin button (Freeze View), as shown in the following screenshot, to freeze the view.

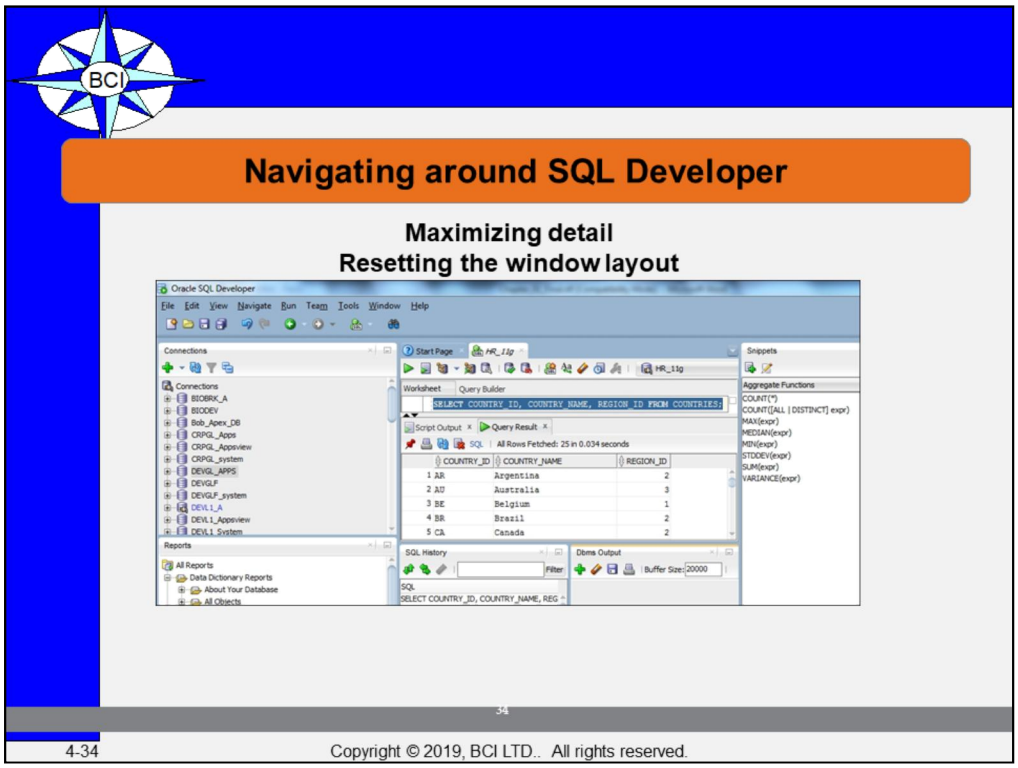


Now, select the DEP table. A second table definition window opens to display the details from the new table.

Select the DEPT tab and drag it down to the lower portion of the screen. Notice the shape of the dragged object change as you drag it slightly to the left, to the center, and the lower portion of the window.

Each of the shapes represent a different layout position. Release the mouse to secure the new position. The screenshots that follow display two of the available positions:


Navigating around SQL Developer
Vertically:
Horizontally:
HR_11g
EMPLOYEES
DEPARTMENTS
Columns
Data
Constraints
Grants
Statistics
Triggers
Flashback
Dependencies
Actions...
COLUMN_NAME
DATA_TYPE
NULLABLE
DATA_DEFAULT
DEPARTMENT_ID NUMBER(4,0) No (null)
DEPARTMENT_NAME VARCHAR2(30 BYTE) No (null)
MANAGER_ID NUMBER(6,0) Yes (null)
LOCATION_ID NUMBER(4,0) Yes (null)
33
4-33
Copyright © 2019, BCI LTD.. All rights reserved.



Maximizing detail

Almost all of the tabs in SQL Developer will maximize when double-clicked. There are a few that do not follow this rule, such as the tabs related to the SQL Worksheet. In general, this works for top-level tabs, which is any tab you can undock and move about, and not for secondary tabs. To maximize a tab, double-click on the tab. A second double-click will reverse the process.

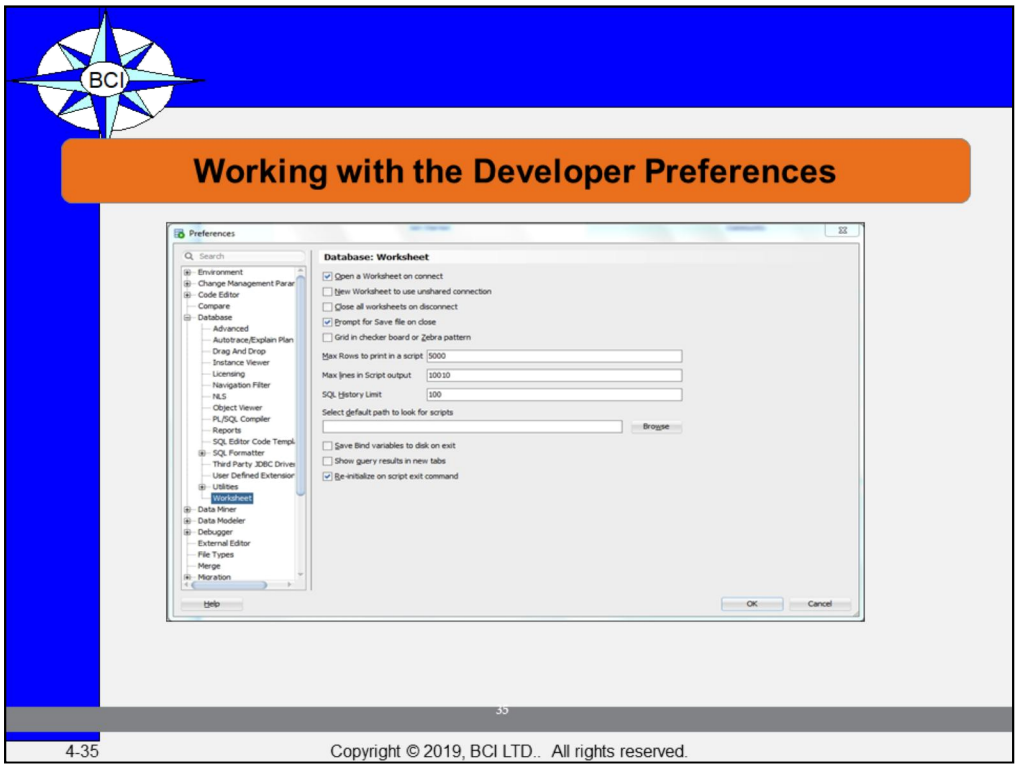
Note

Double-click on the tab to maximize a top-level tab. Double-click again to revert to the previous layout.

Resetting the window layout

If you move your windows about a great deal, you may find that you want to get things back to the default settings.

The example in the following screenshot displays the standard docked Connections and Reports windows to the left. The Reports window, by default, docks below the connections. We have also docked the Snippets window to the right. These windows fill the columns to the left and right, leaving a central window for the editors and log files.



This automatic opening of the SQL Worksheet is controlled by a preference: Open a Worksheet on connect. Go to Tools | Preferences, expand the Database node, and select Worksheet.

Tip

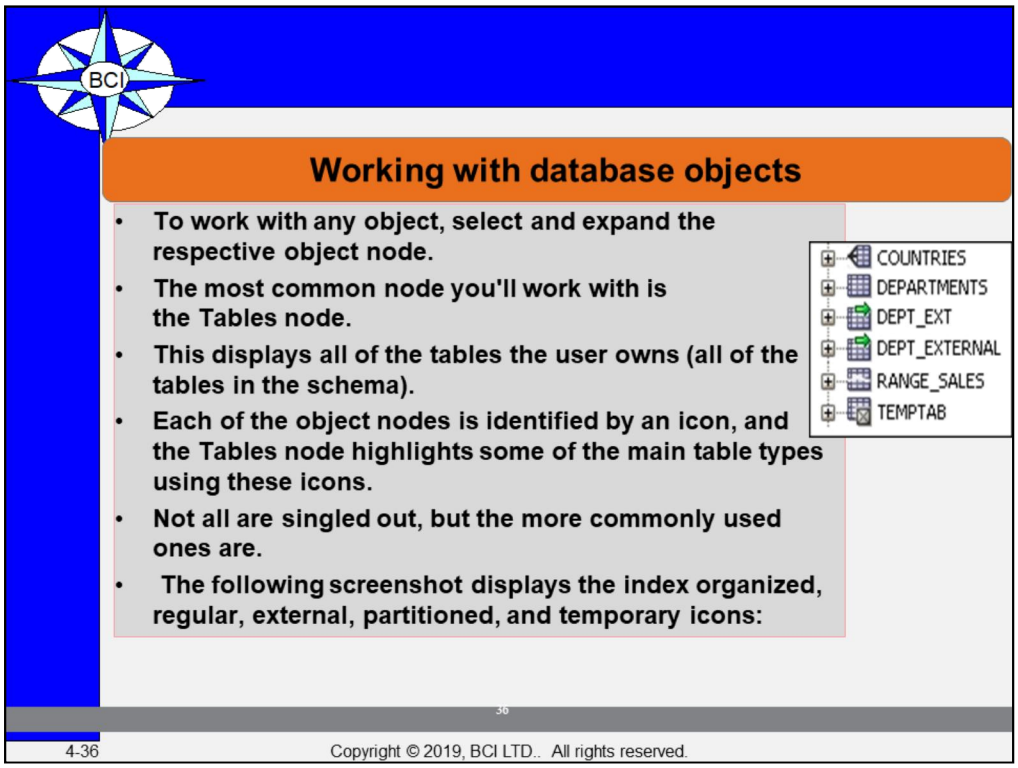
Reconnecting users

When doing administrative work with users, it can help to disconnect the user you are working with before making the changes and reconnect them afterwards. Some actions, such as dropping users or renaming connections, will not be possible without first disconnecting the connection.

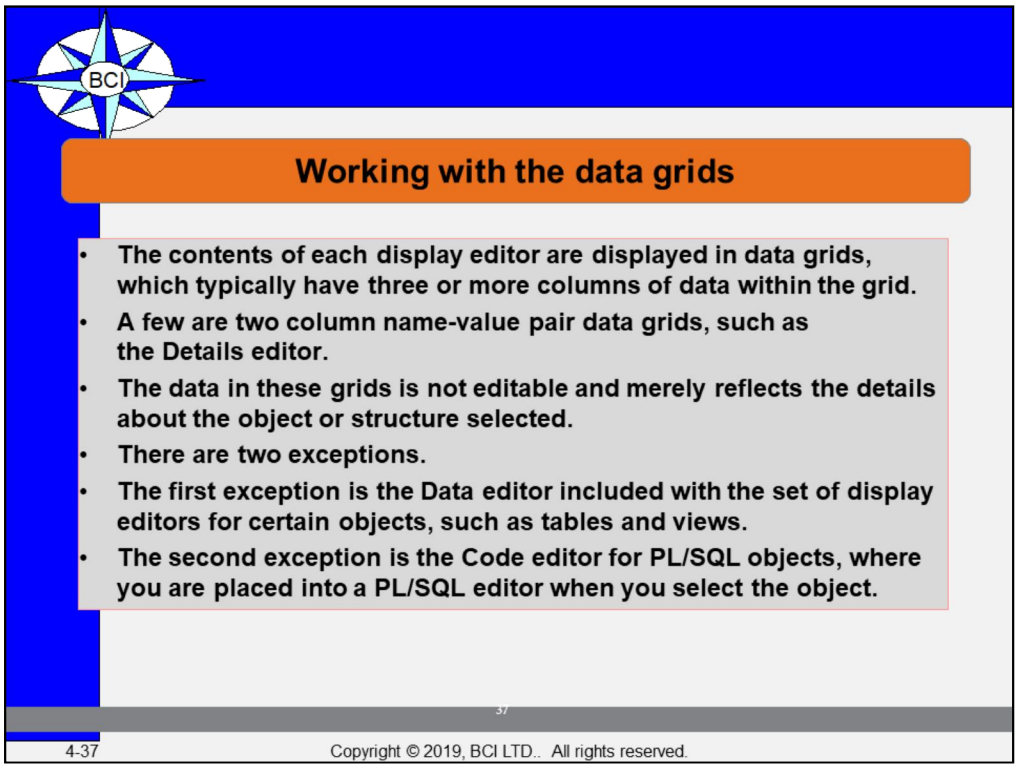
Database schema or user?

The Oracle Concepts Guide states: A schema is a collection of database objects. A schema is owned by a database user and has the same name as that user.

Throughout the text, we use schema and user interchangeably. For the most part, we refer to the SYSTEM and HRschemas, meaning the collection of database objects. When closely or directly related to an activity we use "user", as the code does this. For example, consider DROP user HR cascade. It is a piece of code that drops all of the objects in the schema and the user itself.

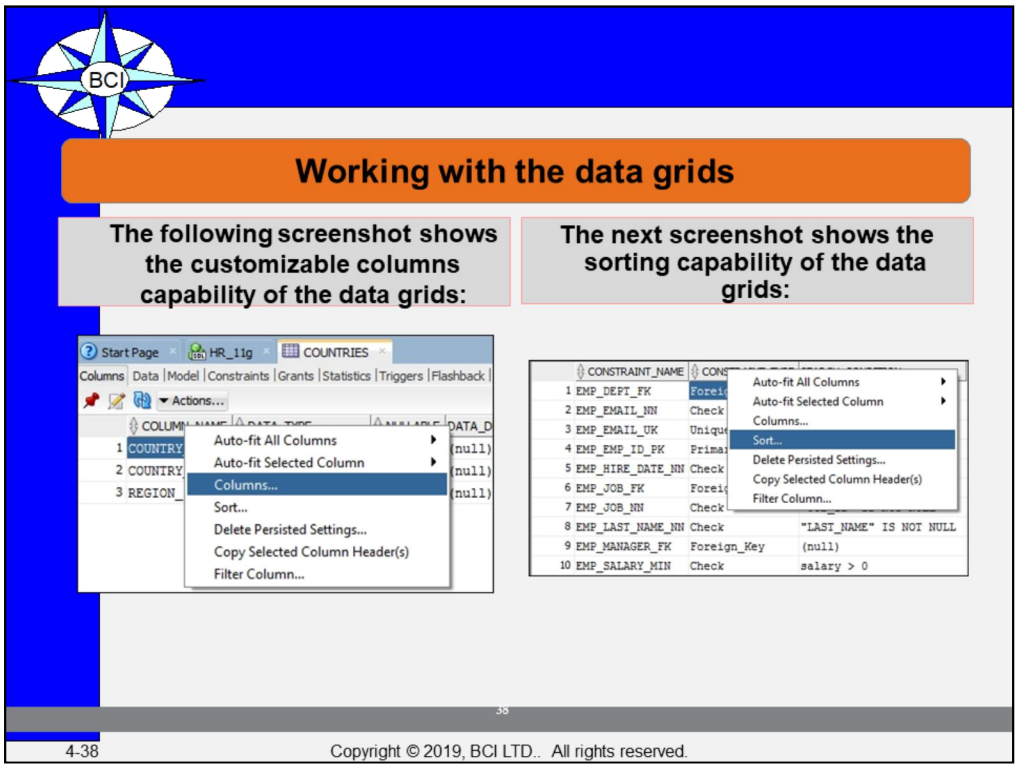


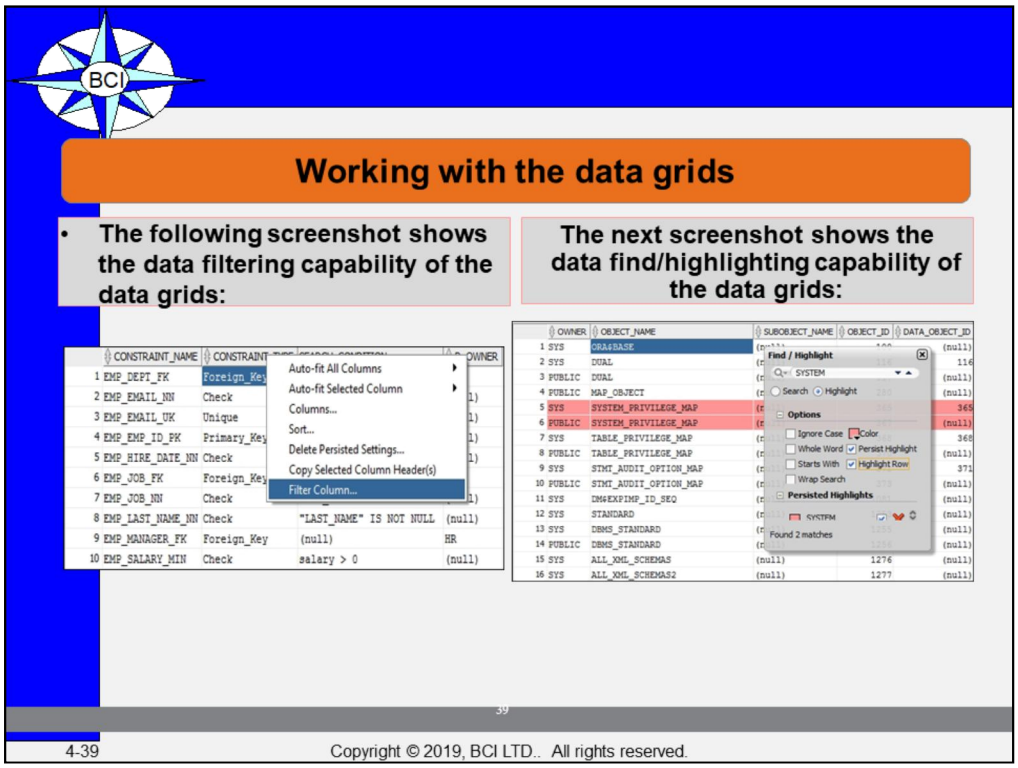
To work with any object, select and expand the respective object node. The most common node you'll work with is the Tables node. This displays all of the tables the user owns (all of the tables in the schema). Each of the object nodes is identified by an icon, and the Tables node highlights some of the main table types using these icons. Not all are singled out, but the more commonly used ones are. If you expand the HR Tables node, then the COUNTRIES table, which in the sample is an index-organized table, is identified by the slightly different table icon used. Partitioned tables are also distinguished from regular, simple tables using icons. The following screenshot displays the index organized, regular, external, partitioned, and temporary icons:

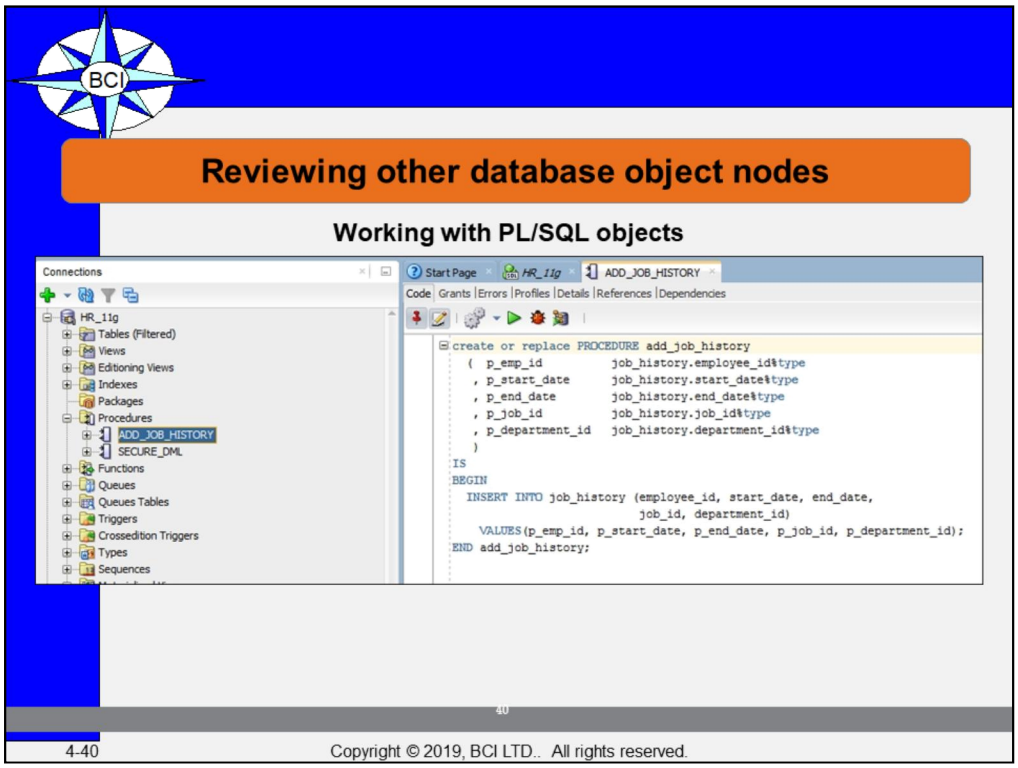


The contents of each display editor are displayed in data grids, which typically have three or more columns of data within the grid. A few are two column name-value pair data grids, such as the Details editor. The data in these grids is not editable and merely reflects the details about the object or structure selected. There are two exceptions. The first exception is the Data editor included with the set of display editors for certain objects, such as tables and views.
The Data editor displays the instance data for a table and, depending on the object, this data can be edited and the changes can be committed to the database. The second exception is the Code editor for PL/SQL objects, where you are placed into a PL/SQL editor when you select the object.

Data grids throughout SQL Developer have context menus on the column's headings and the data grid itself. You can control the layout and what data is displayed by using these two context menus. For the remaining portion of this section we'll review the various options on these context menus.







Reviewing other database object nodes

As you select each of the other database objects, you'll notice that the set of display editors varies considerably.

It would be tedious to single out each of the object nodes and describe them here. The display editors and data grids behave the same for each of them.

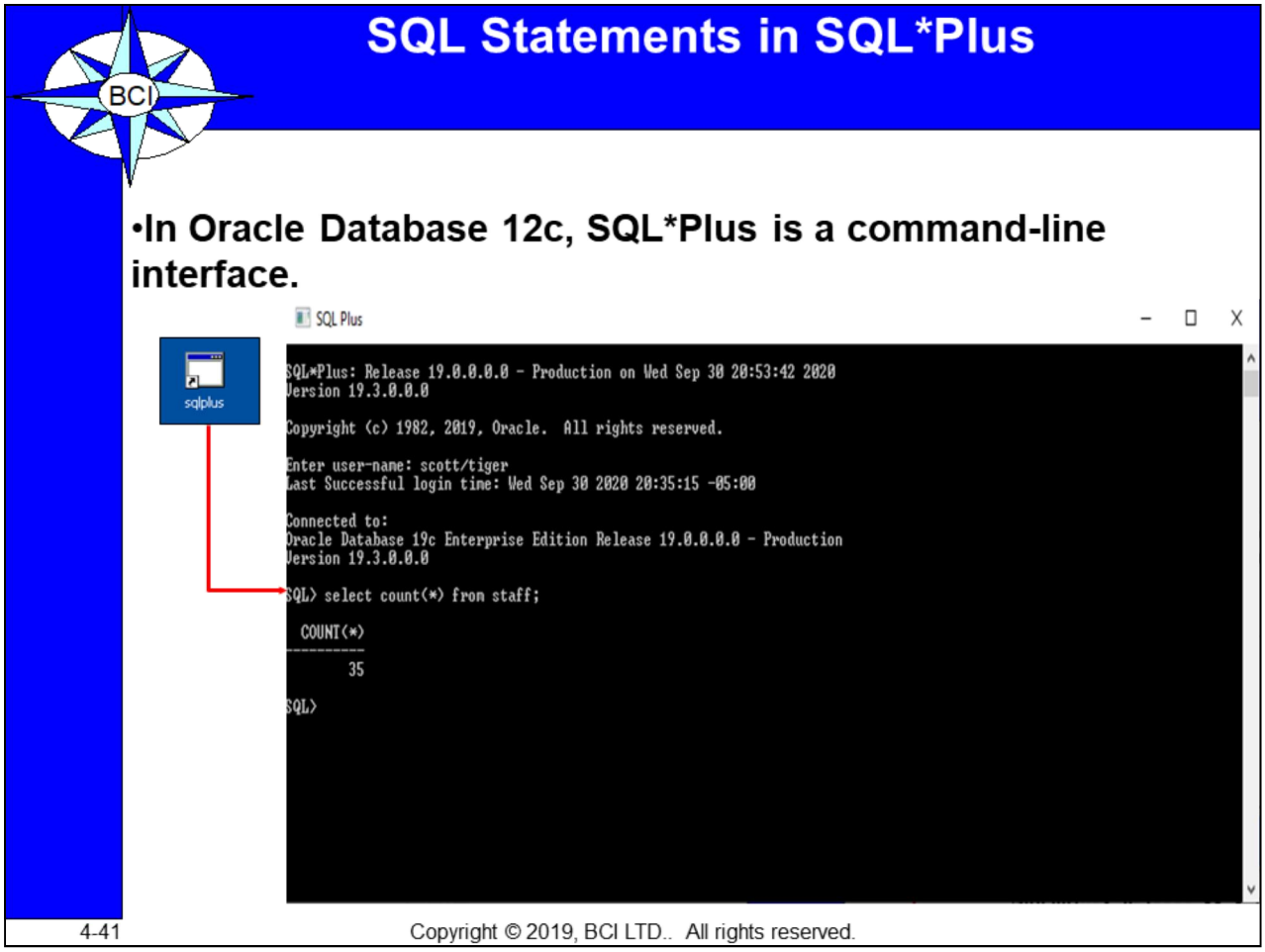
Working with PL/SQL objects

Triggers, functions, procedures, and packages all have their own separate nodes in the Connections navigator. A single-click on any object in these PL/SQL nodes opens as editable PL/SQL code, as shown here:

In the preceding screenshot, the initial Code editor is the editable PL/SQL code editor. This Code editor is included in the set of display editors for the selected procedure.

Tip

Unlike with other database objects, each new selected PL/SQL object opens a new window. In other words, the windows are automatically pinned for the PL/SQL windows. For more information, see the Lesson 4, Working with PL/SQL.



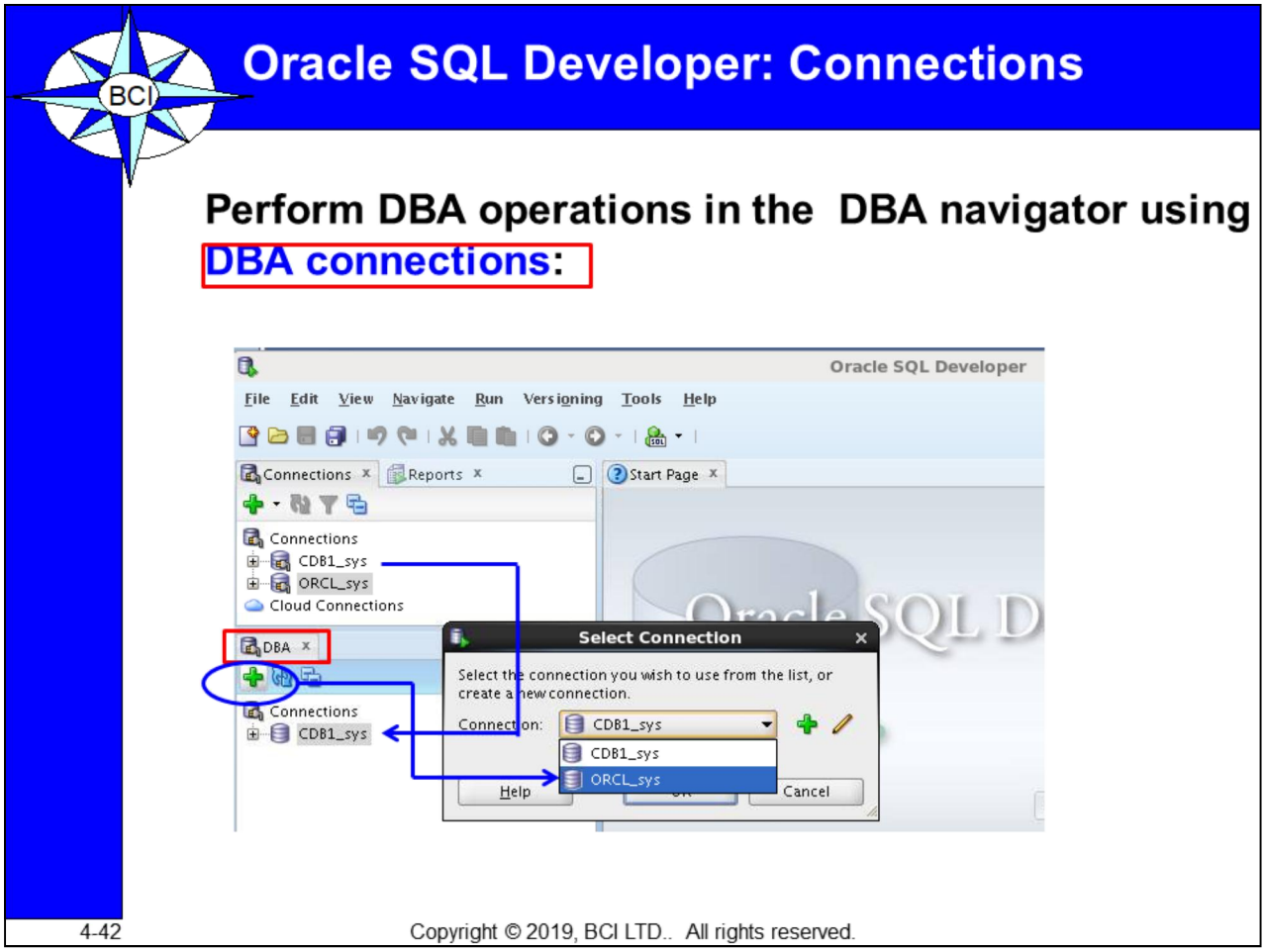
SQL Statements in SQL*Plus

Oracle SQL*Plus is a command-line interface with which you can submit SQL statements and PL/SQL blocks for execution and receive the results in an application or command window.

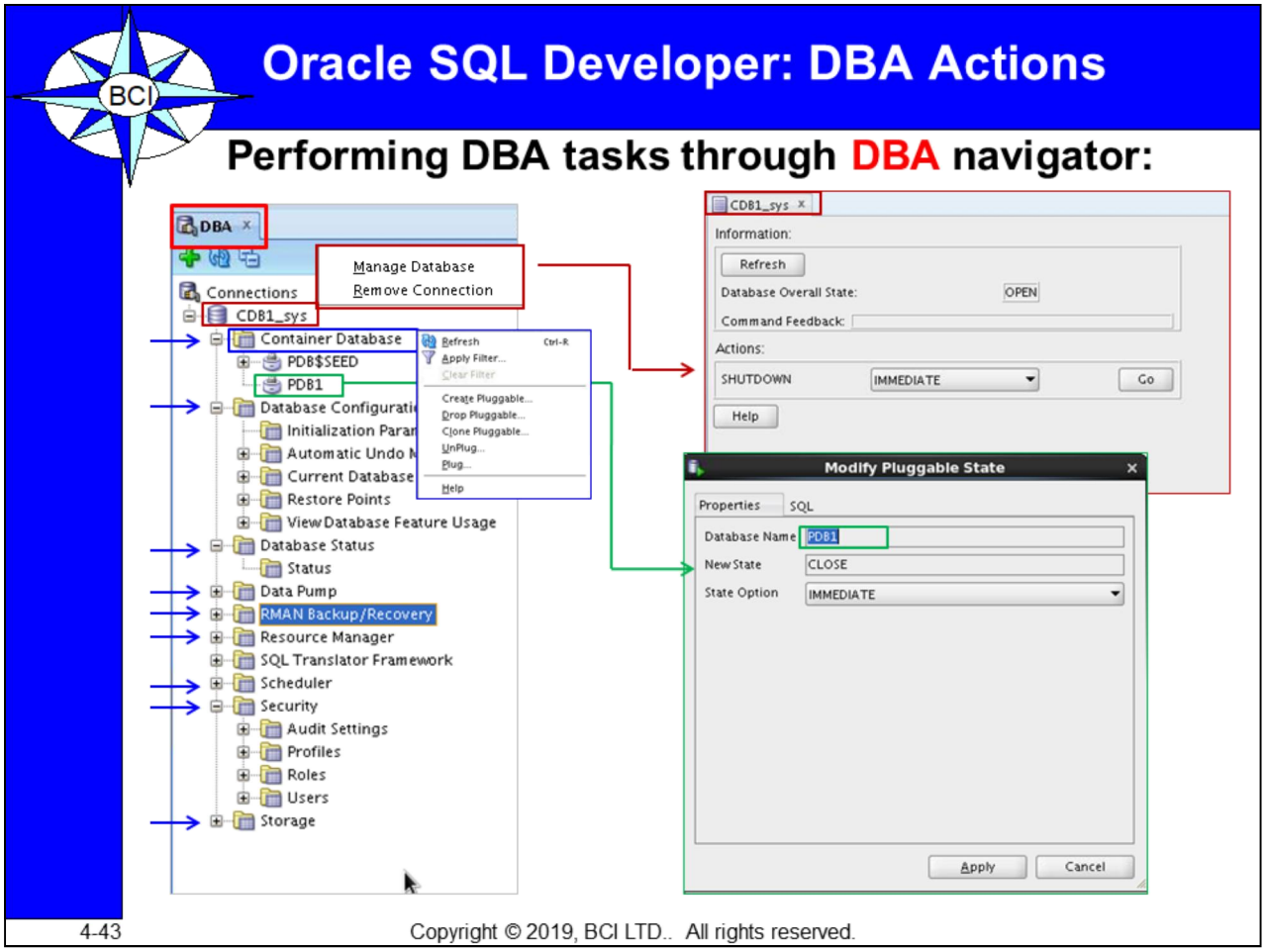
SQL*Plus is:

- Shipped with the database
- Installed on a client and on the database server system
- Accessed through from an icon or the command line

Note: If you do not have access to Oracle SQL Developer and would prefer to use SQL* Plus, the classroom setup provides SQL*Plus icon on your desktop. It may also be useful to use it in cases where Oracle SQL Developer does not support any SQL* Plus command.

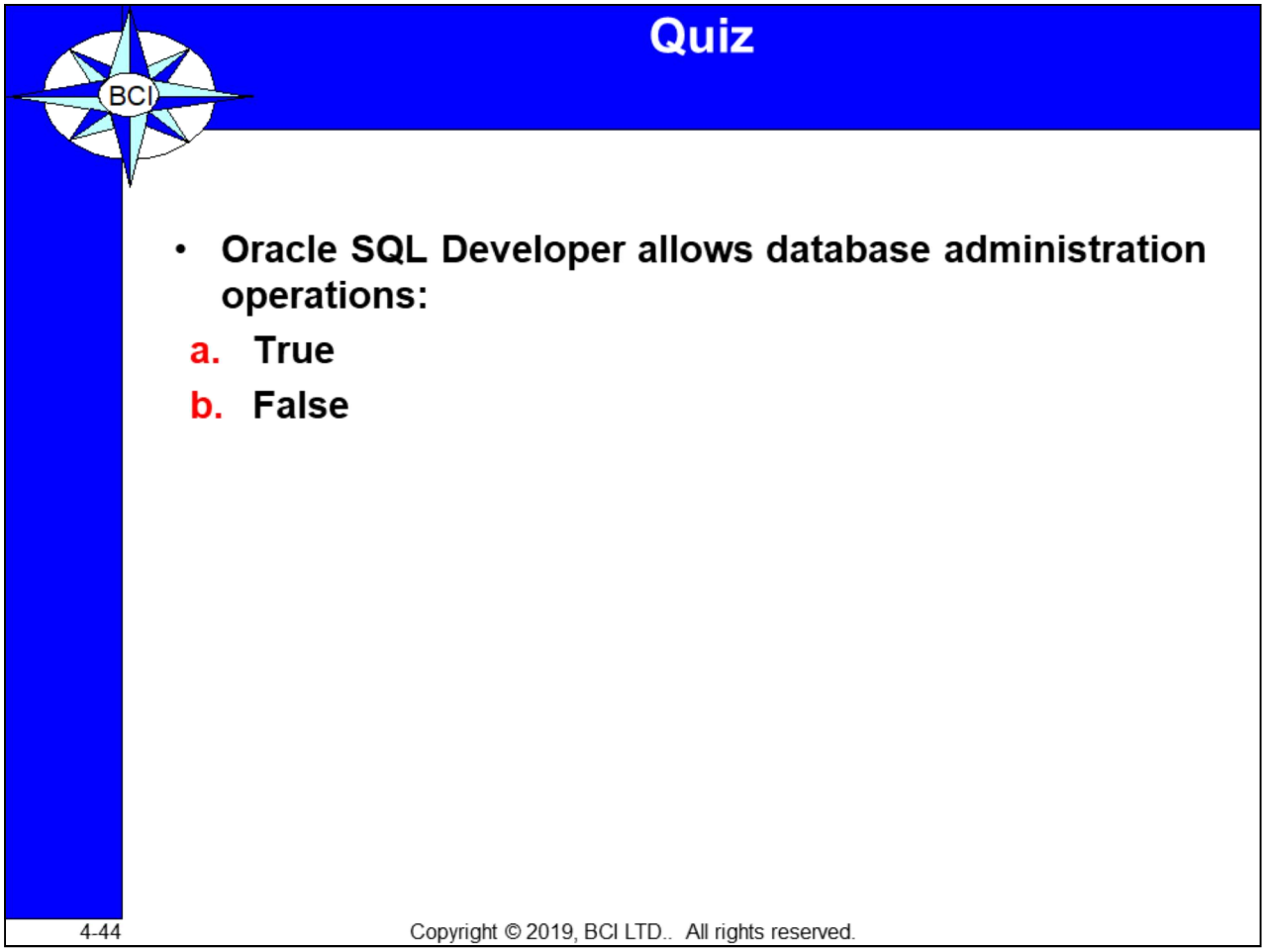


Oracle SQL Developer is a tool that allows stand-alone graphical browsing and development of database schema objects, as well as execution of database administrative tasks.
SQL Developer enables users with database administrator privileges to view and edit certain information relevant to DBAs and perform DBA operations. To perform DBA operations, use the DBA navigator, which is similar to the Connections navigator in that it has nodes for all defined database connections. If the DBA navigator is not visible, select View, then DBA. You should add only connections for which the associated database user has DBA privileges or at least privileges for the desired DBA navigator operations on the specified database.

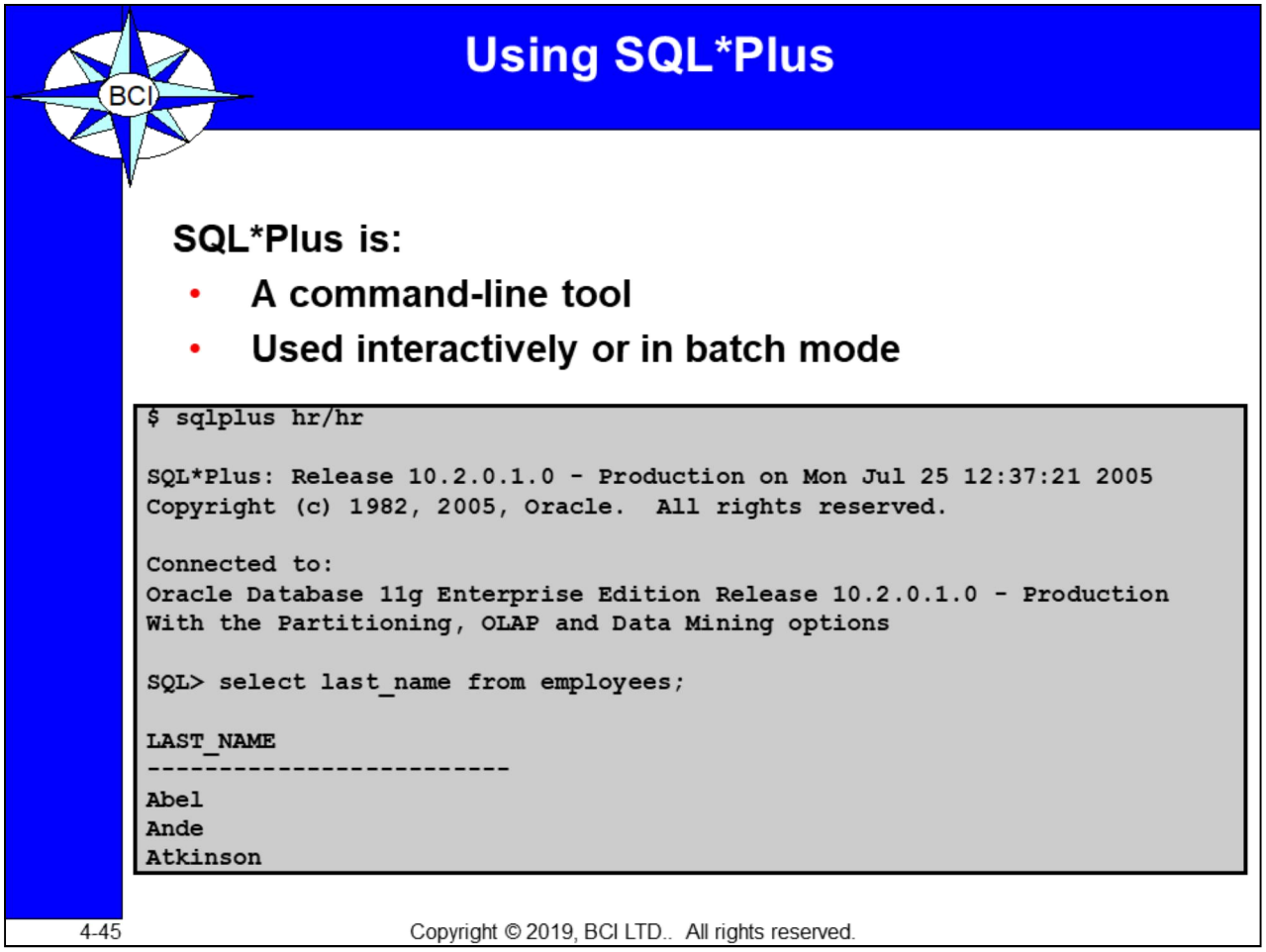


The DBA operations that can be performed are the following:

- Pluggable database startup/shutdown
- Database configuration: Initialization Parameters, Automatic Undo Management, Current Database Properties, Restore Points, View Database Feature Usage
- Database status view
- Data Pump Export and Import jobs
- RMAN Backup/Recovery actions
- Resource Manager configuration
- Scheduler setting
- Security configuration like audit settings, profiles, roles, users
- Storage configuration for archive logs, control files, data files, redo log groups, tablespaces, temporary tablespace groups


Quiz
BCI
• Oracle SQL Developer allows database administration operations:
a. True
b. False
4-44
Copyright © 2019, BCI LTD.. All rights reserved.

Answer: a



Using SQL*Plus

You can use the command-line interface to SQL*Plus to write SQL*Plus, SQL, and PL/SQL commands to:

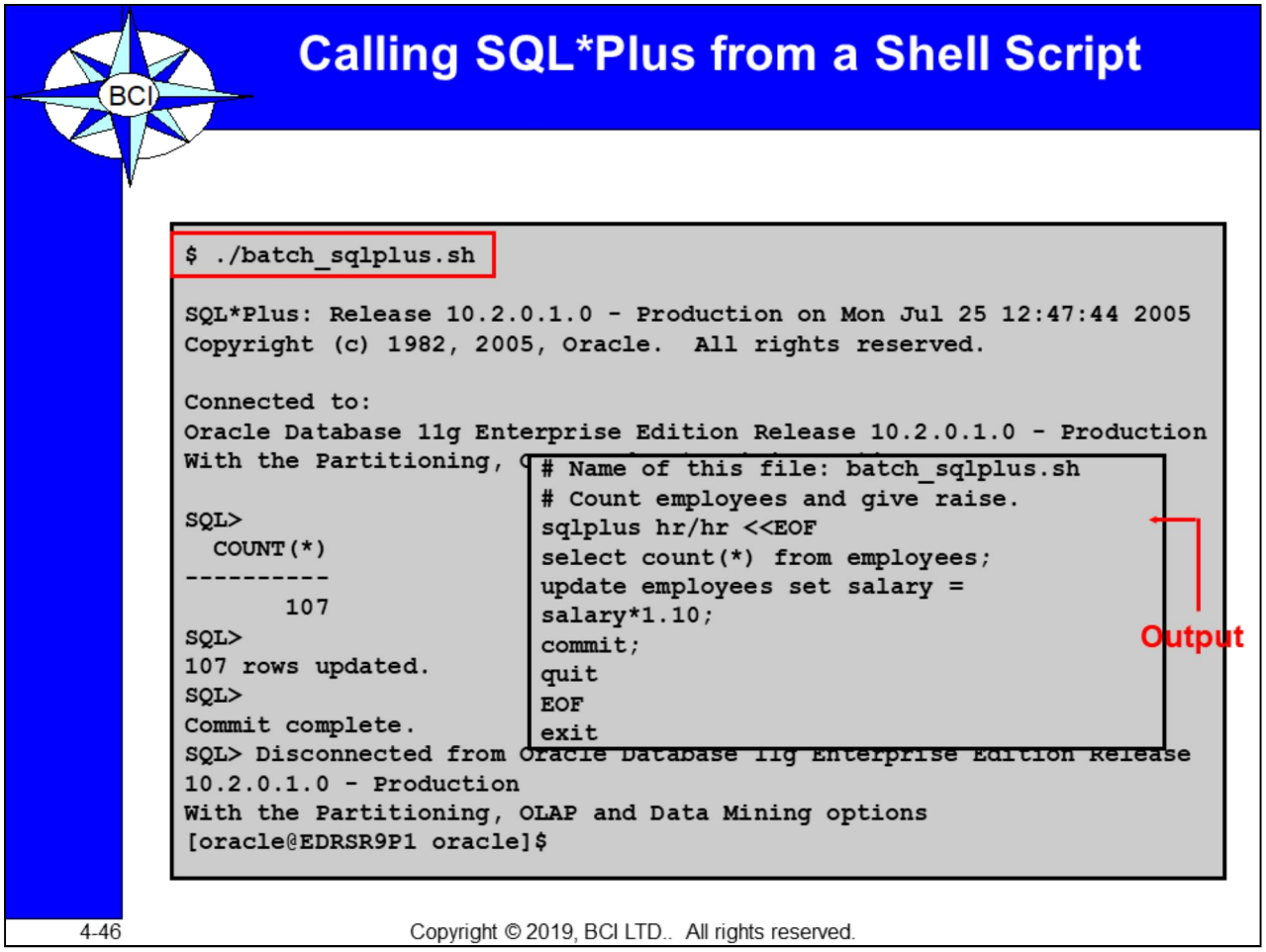
- Enter, edit, run, store, retrieve, and save SQL commands and PL/SQL blocks
- Format, calculate, store, and print query results
- List column definitions for any table
- Send messages to and accept responses from an end user
- Perform database administration

To start SQL*Plus, perform the following steps:

1. Open a terminal window.
2. At the command-line prompt, enter the SQL*Plus command in the form:

```
$ sqlplus /nolog
```

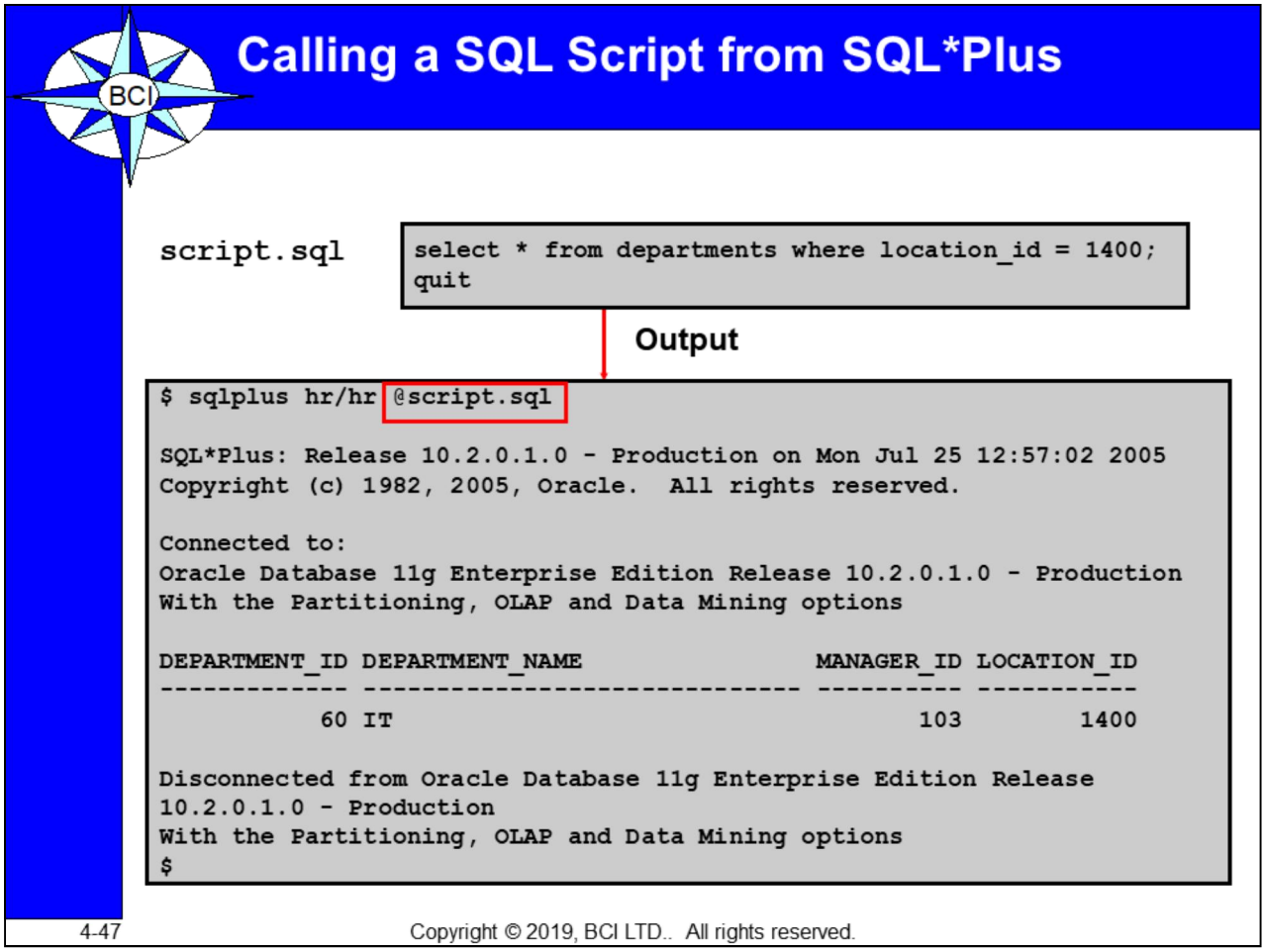
3. Enter `connect` followed by the user you want to connect as.
4. When prompted, enter the user's password.
   SQL*Plus starts and connects to the default database.



Calling SQL*Plus from a Shell Script

You can call SQL*Plus from a shell script or `BAT` file by invoking `sqlplus` and using the operating system scripting syntax for passing parameters.

In this example, the `SELECT`, `UPATE` and `COMMIT` statements are executed, before SQL*Plus returns control to the operating system.



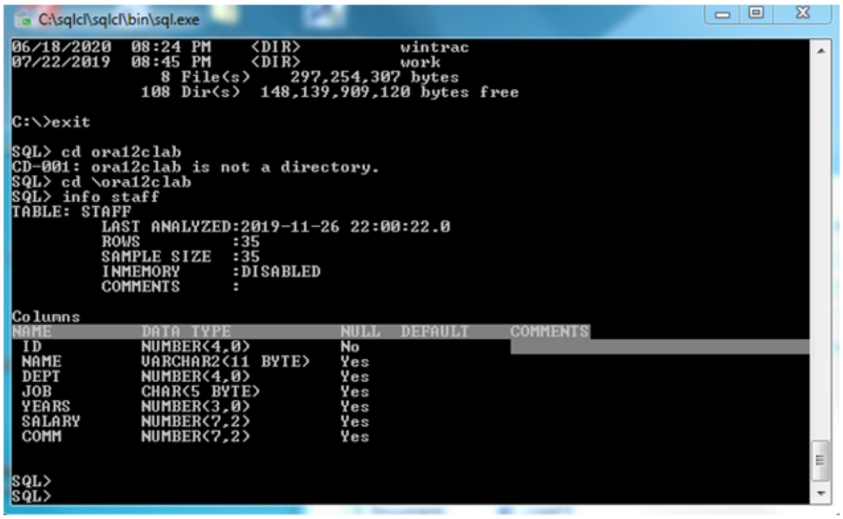
Calling a SQL Script from SQL*Plus

You can call an existing SQL script file from within SQL*Plus. This can be done at the command line when first invoking SQL*Plus, as shown in the slide. It can also be done from inside a SQL*Plus session, simply by using the "@" operator. For example, this runs the script from within an already established SQL*Plus session:

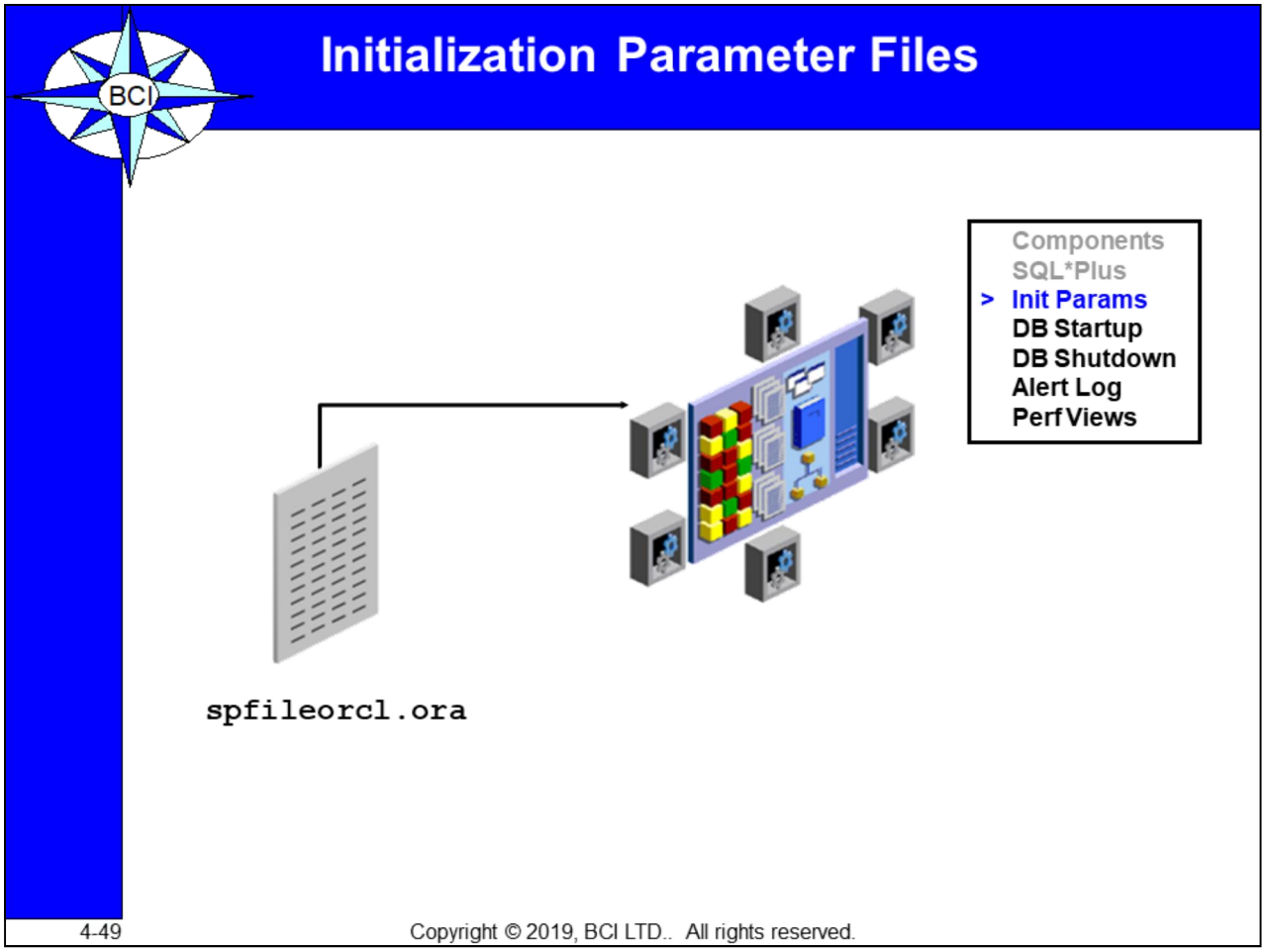
```
SQL> @script.sql
```

SQLcl

A New Feature that is similar to SQL*PLUS



It allows for commands such as: history
info
alias
cd

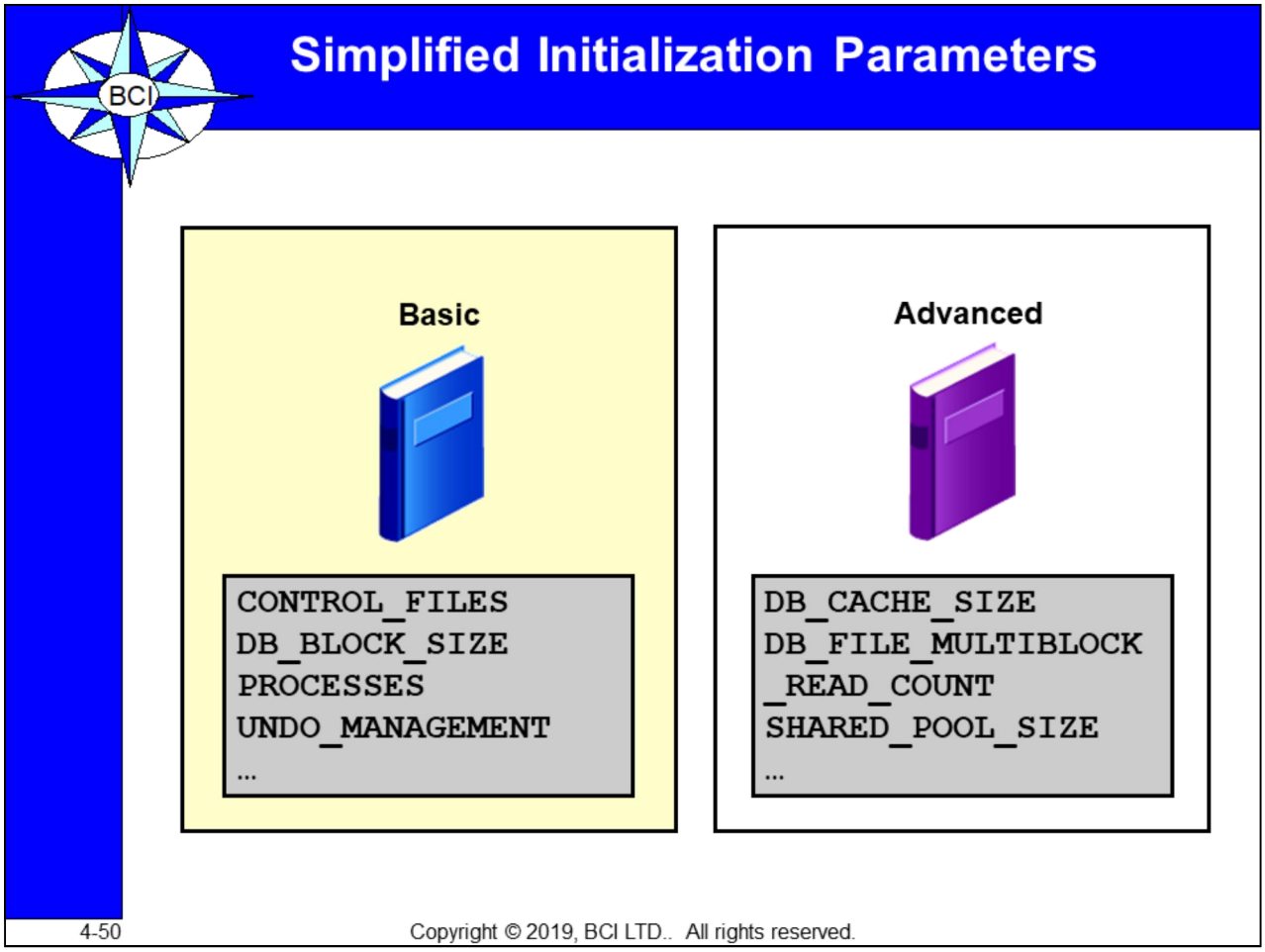


Initialization Parameter Files

When you start the instance, an initialization parameter file is read. There are two types of parameter files:

- **Server parameter file:** This is the preferred type of initialization parameter file. It is a binary file that can be written to and read by the database server and *must not be edited manually*. It resides in the server that the Oracle database is executing on, and is persistent across shutdown and startup. This is often referred to as a server parameter file (SPFILE). The default name of this file, which is automatically sought at startup, is `spfile<SID>.ora`.
- **Text initialization parameter file:** This type of initialization parameter file can be read by the database server, but it is not written to by the server. The initialization parameter settings must be set and changed manually by using a text editor so that they are persistent across shutdown and startup. The default name of this file, which is automatically sought at startup if an SPFILE is not found, is `init<SID>.ora`.

It is recommended that you create an SPFILE as a dynamic means of maintaining initialization parameters. By using an SPFILE, you can store and manage your initialization parameters persistently in a server-side disk file.



Simplified Initialization Parameters

Initialization parameters are divided into two groups: basic and advanced.

In the majority of cases, it is necessary to set and tune only the 32 basic parameters to get reasonable performance from the database. In rare situations, modification of the advanced parameters may be needed to achieve optimal performance.

A basic parameter is defined as one that you are likely to set to keep your database running with good performance. All other parameters are considered to be advanced.

The examples of basic parameters include "destinations" or directory names for specific types of files: `AUDIT_FILE_DEST`, `BACKGROUND_DUMP_DEST`, `CORE_DUMP_DEST`, `DB_CREATE_FILE_DEST`, `DB_CREATE_ONLINE_LOG_DEST_`*n*, `DB_RECOVERY_FILE_DEST`, and `USER_DUMP_DEST`.

Initialization Parameters: Examples

The `CONTROL_FILES` parameter specifies one or more control file names. Oracle strongly recommends that you multiplex and mirror control files. The range of values for this parameter is from 1 to 8 file names (with path names). The default range is OS dependent.

Simplified Initialization Parameters (continued)

Initialization Parameters: Examples (continued)

The DB_BLOCK_SIZE parameter specifies the size (in bytes) of an Oracle database block. This value is set at database creation and cannot be subsequently changed. Range of values: 1024 – 65536 (OS dependent). Default value: 8K (OS dependent).

The DB_CACHE_SIZE parameter specifies the size of the standard block buffer cache. Range of values: At least 16 MB. Default value: 48 MB.

The DB_FILE_MULTIBLOCK_READ_COUNT parameter specifies the maximum number of blocks read during an input/output (I/O) operation involving a full sequential scan. Range of values: Operating system dependent. Default value: 8.

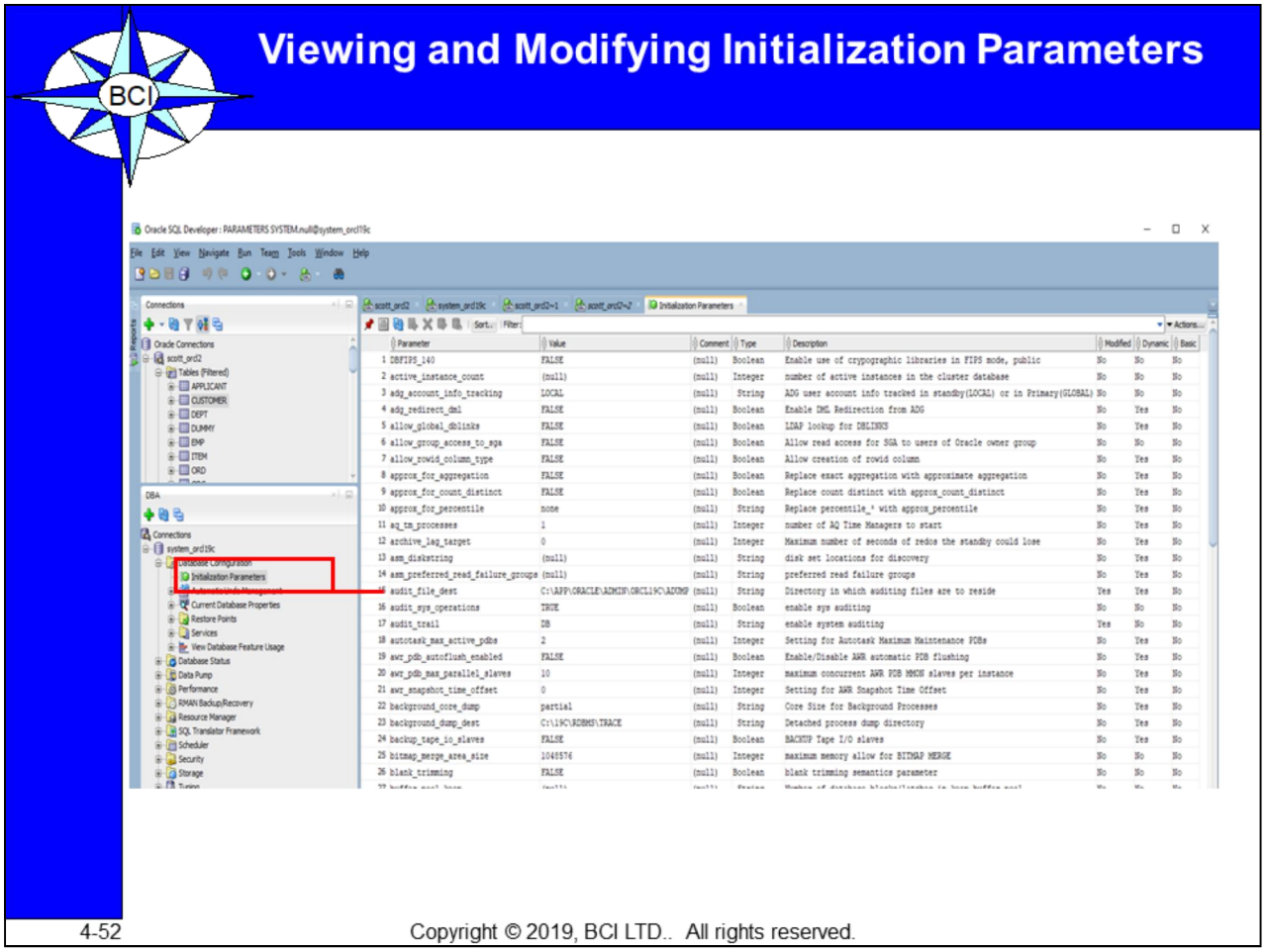
The DB_FILES parameter specifies the maximum number of database files that can be opened for this database. Range of values: MAXDATAFILES – OS dependent. Default value: OS dependent (200 on Solaris).

The PGA_AGGREGATE_TARGET parameter specifies the amount of Program Global Area (PGA) memory allocated to all server processes attached to the instance. Set this parameter to a positive value before enabling the automatic setting of working areas. This memory does not reside in the System Global Area (SGA). The database uses this parameter as a target amount of PGA memory to use. When setting this parameter, subtract the SGA from the total memory on the system available to the Oracle instance. The remaining memory can be assigned to PGA_AGGREGATE_MEMORY. Range of values: Integers plus letter K, M, or G to specify this limit in kilobytes, megabytes, or gigabytes. Minimum value is 10M and maximum value is 400G. Default: "Not Specified," which means that the automatic tuning of work areas is fully disabled.

The PROCESSES parameter specifies the maximum number of OS user processes that can simultaneously connect to an Oracle server. This value should allow for all background processes. Range of values: 6 to an OS-dependent value. Default value: Depends on the PARALLEL_MAX_SERVERS parameter.

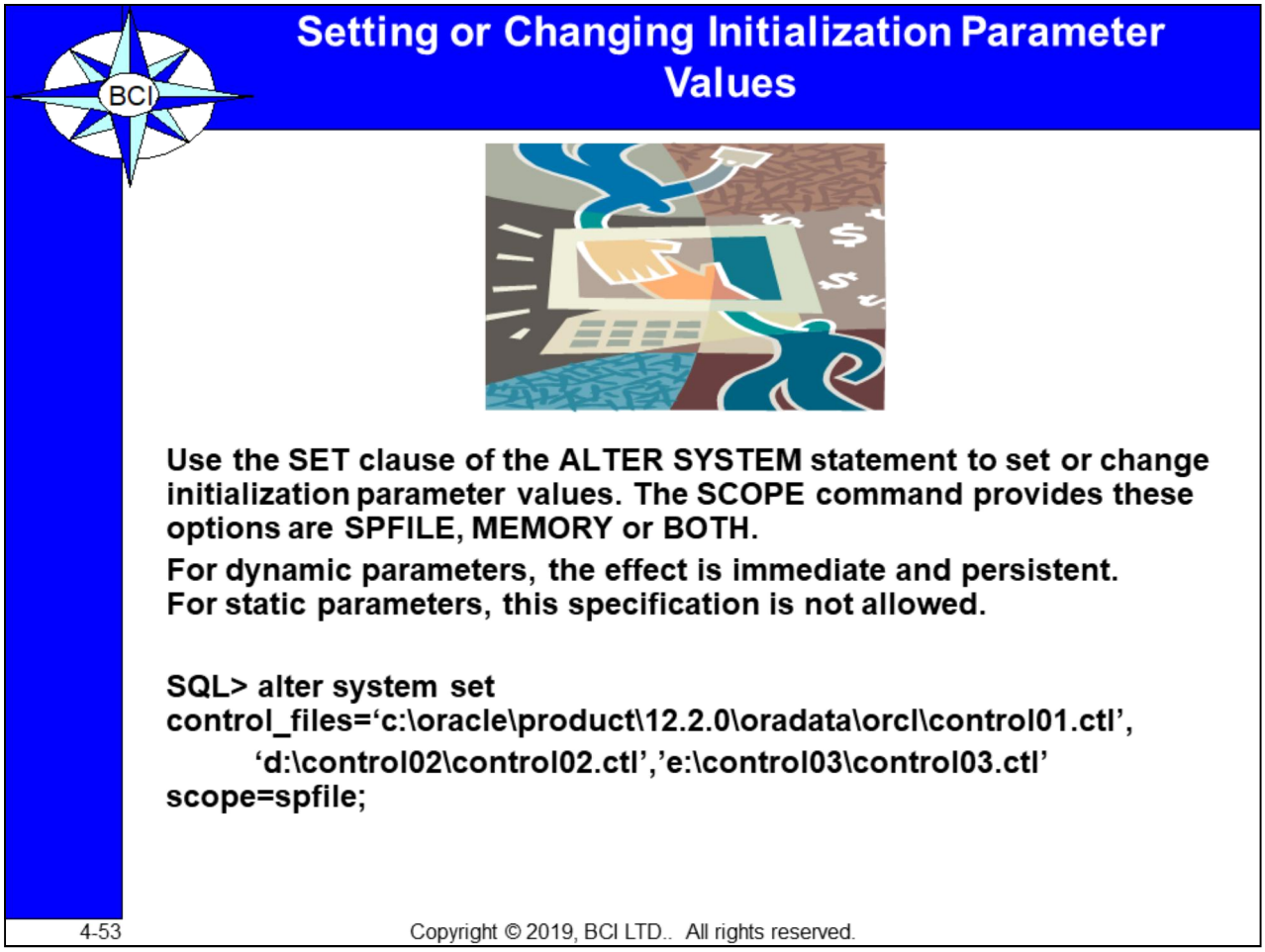
The SHARED_POOL_SIZE parameter specifies the size of the shared pool in bytes. The shared pool contains objects such as shared cursors, stored procedures, control structures, and parallel execution message buffers. Larger values can improve performance in multiuser systems. Range of values: 300 KB – OS dependent. Default value: If 64 bit, then 64 MB, or else 16 MB.

The UNDO_MANAGEMENT parameter specifies which undo space management mode the system should use. When set to AUTO, the instance is started in System Managed Undo (SMU) mode. Otherwise, it is started in Rollback Undo (RBU) mode. In RBU mode, undo space is allocated externally as rollback segments. In SMU mode, undo space is allocated externally as undo tablespaces. Range of values: AUTO or MANUAL. Default value: If the UNDO_MANAGEMENT parameter is omitted when the first instance is started, the default value of MANUAL is used and the instance is started in RBU mode. If it is not the first instance, the instance is started in the same undo mode as all other existing instances.



Viewing and Modifying Initialization Parameters

You can use Enterprise Manager to view and modify initialization parameters by clicking All Initialization Parameters in the Database Configuration region of the Database Administration tabbed page. You select the Initialization Paramter action.

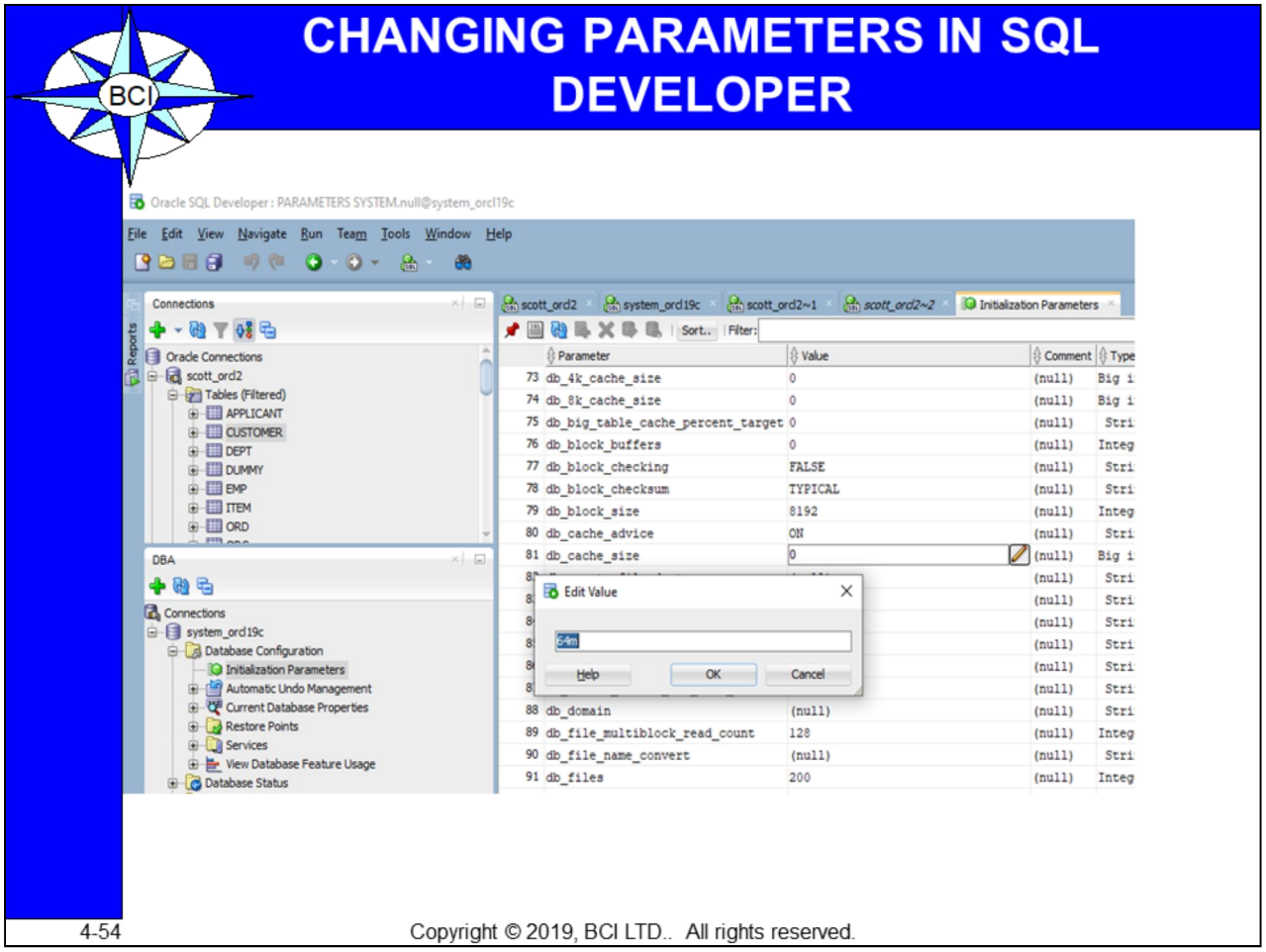


SCOPE = SPFILE The change is applied in the server parameter file only. The effect is as follows:
For dynamic parameters, the change is effective at the next startup and is persistent.
For static parameters, the behavior is the same as for dynamic parameters. This is the only SCOPE specification allowed for static parameters.

SCOPE = MEMORYThe change is applied in memory only. The effect is as follows:

For dynamic parameters, the effect is immediate, but it is not persistent because the server parameter file is not updated.
For static parameters, this specification is not allowed.

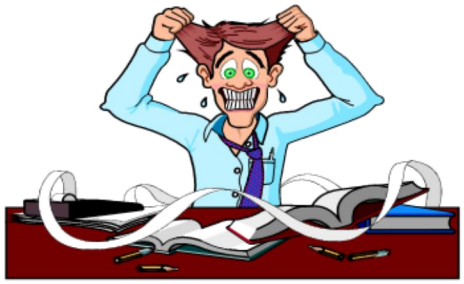
SCOPE = BOTH The change is applied in both the server parameter file and memory. The effect is as follows:



In the DBA role, click on Database Configuration, then click on Initialization Parameters. Move cursor to db_cache_size and click on the value. You can then edit the valaue to

The appropriate number. In this case, 64m.

CHANGING THE INIT.ORA FILE



The init.ora file stores the initialization parameters of Oracle when the database is FIRST created. The values that are currently in effect can be viewed through v$parameter. The init.ora file is read when an instance is started up then it creates a spfile to be used thereafter.

Connect / as sysdba

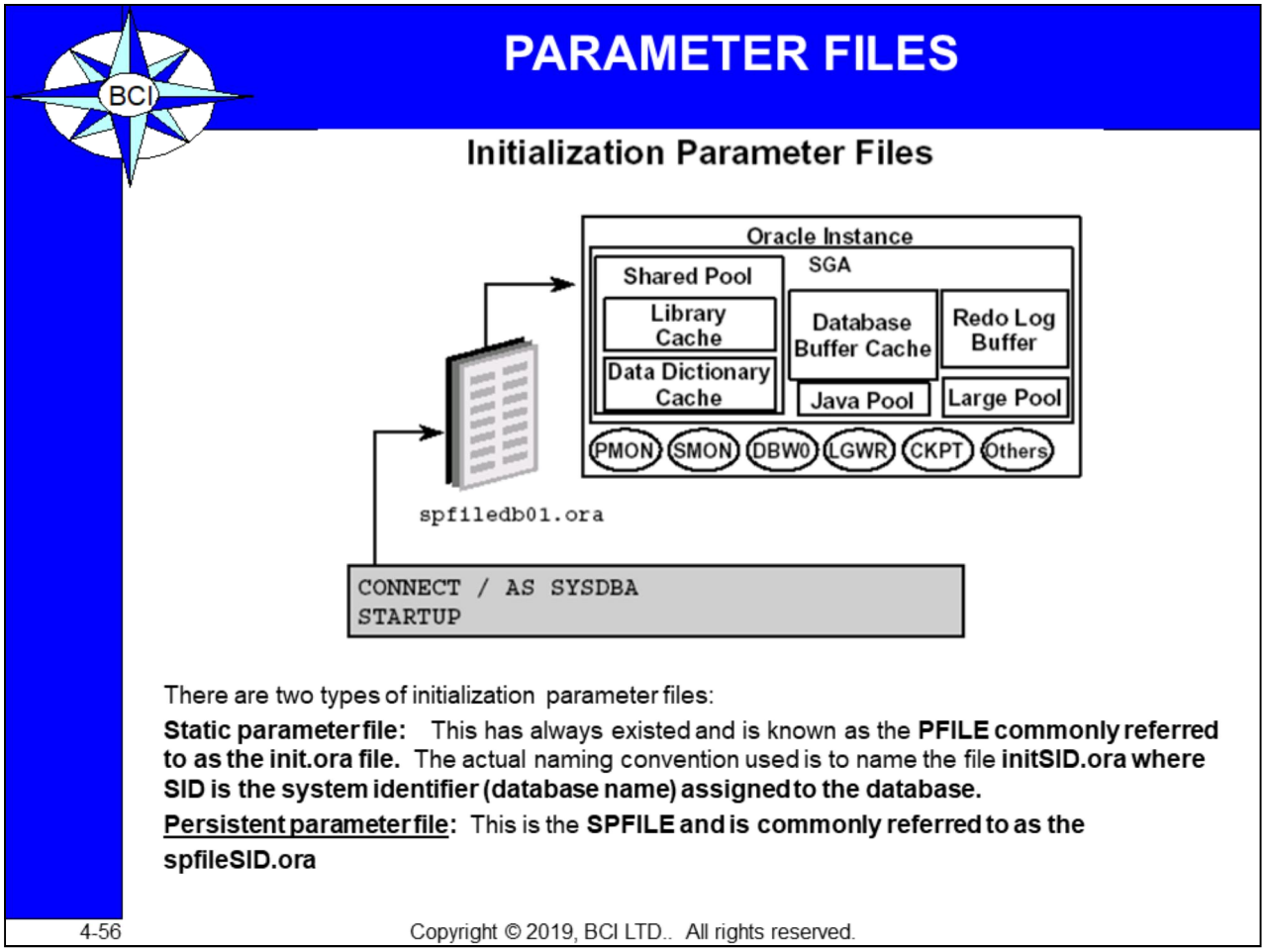
startup

4-55

Default location and name

The default location of init.ora is $ORACLE_HOME/dbs> on unix and %ORACLE_HOME%\database on Windows. On Windows, the location can be changed by changing ORA_%ORACLE_SID%_PFILE. The default name for the file is init$ORACLE_SID.ora (unix) or init%ORACLE_SID%.ora (windows) However, it is possible to start the database with another init.ora file than the default one. In this case, there is no way to determine which init.ora was used when the database is running (at least up to Oracle 11g). Creating a PFILE from an SPFILE

create pfile='/some/path/init.ora' from spfile; In order to execute create pfile one needs the sysdba role.



PFILE

This is a plain text file. I usually maintain this file either by editing it with the vi editor, or by FTPing it to my client computer, modifying it with Notepad, and then FTPing it back to the SOBORA2 server.

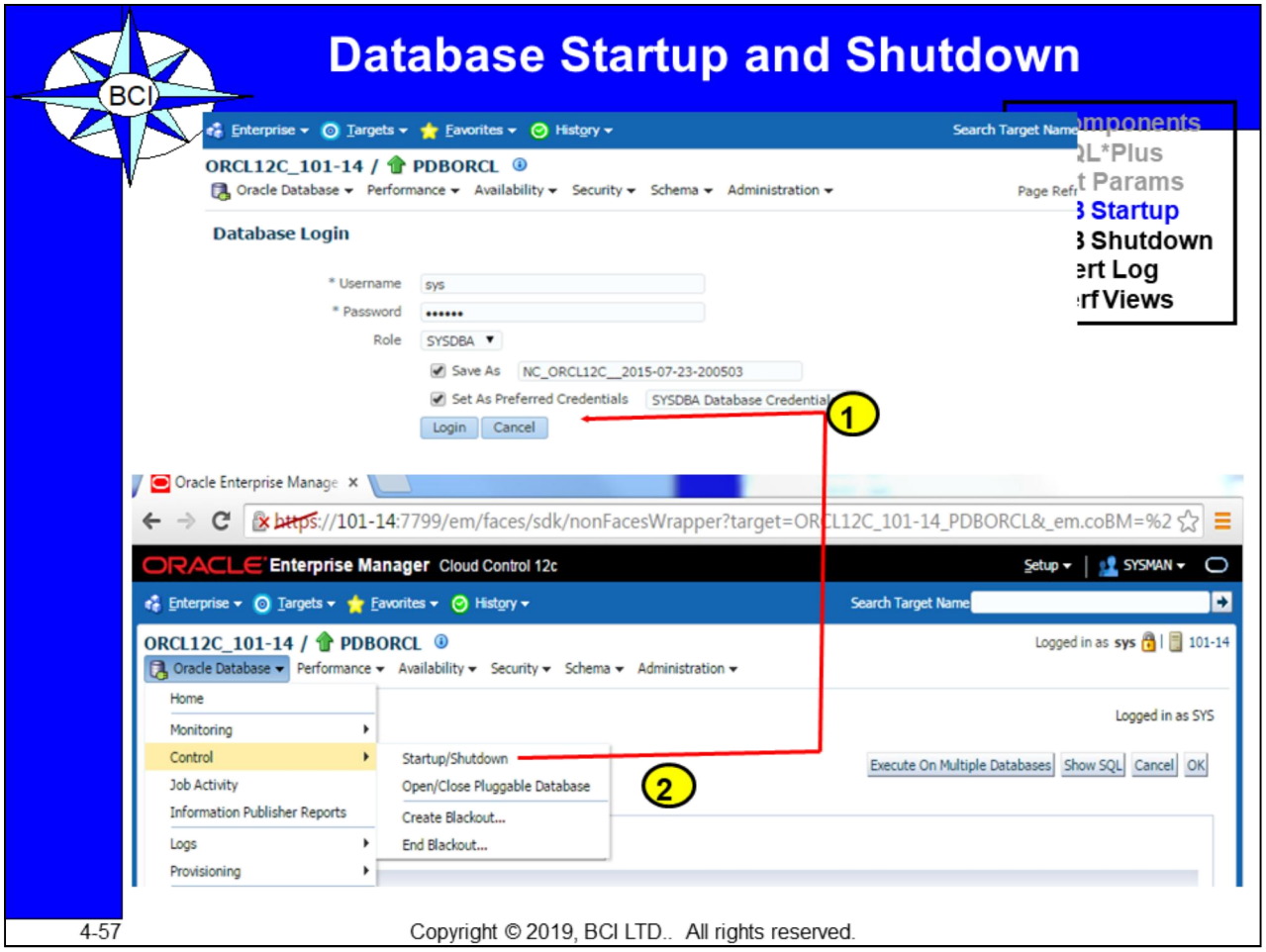
The file is only read during database startup so any modifications take effect the next time the database is started up. This is an obvious limitation since shutting down and starting up an Oracle database is not desirable in a 24/7 operating environment.

The naming convention followed is to name the file initSID.ora where SID is the system identifier. For example, the PFILE for the departmental SOBORA2 database is named initDBORCL.ora.

When Oracle software is installed, a sample init.ora file is created. You can create one for your database by simply copying the init.ora sample file and renaming it. The sample command shown here creates an init.ora file for a database named USER350. Here the file was copied to the user's HOME directory and named initUSER350.ora.

$ cp $ORACLE_HOME/dbs/init.ora $HOME/initUSER350.ora

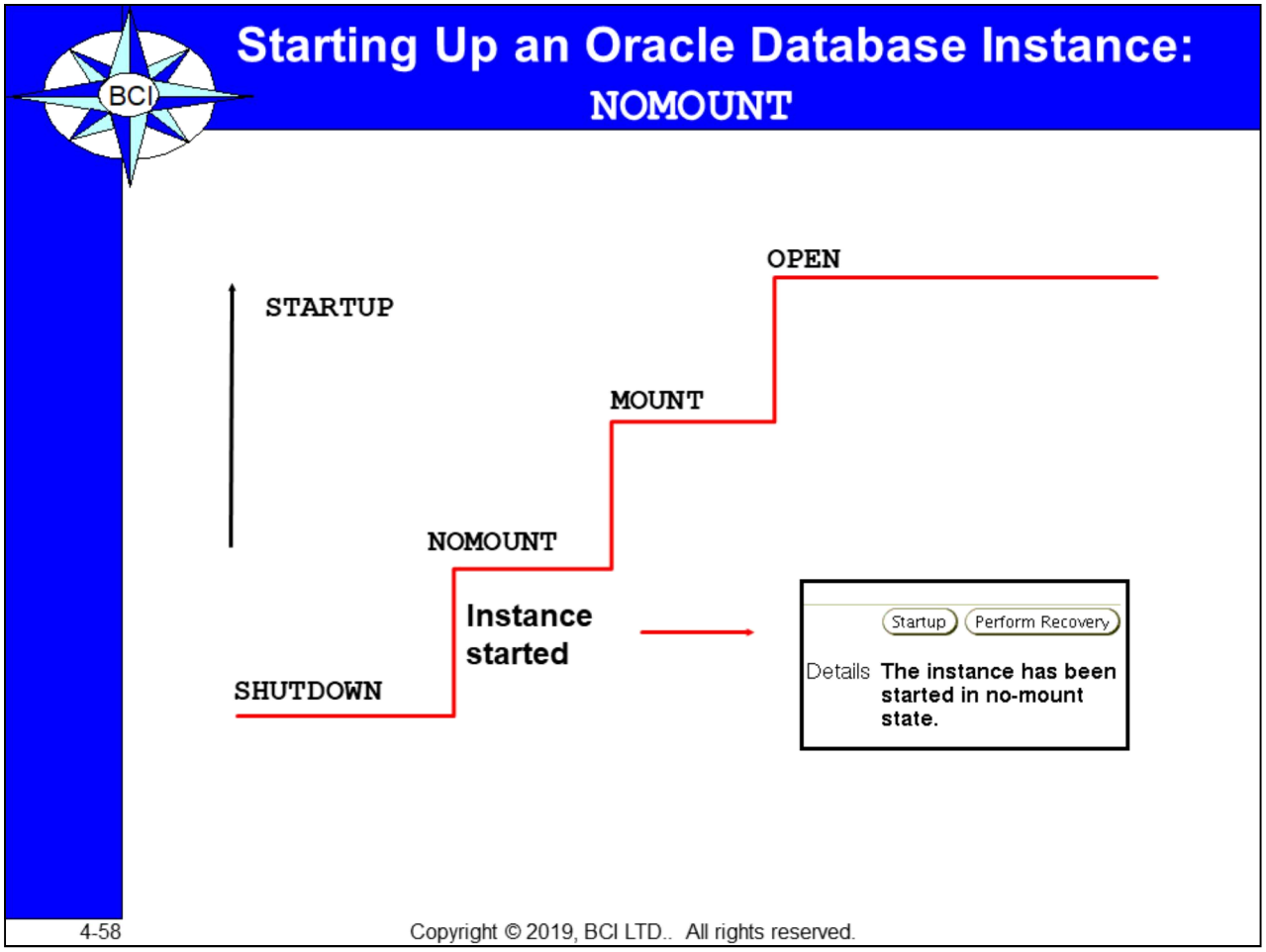
You can also create an init.ora file by typing commands into a plain text file using an editor such as Notepad.



Database Startup and Shutdown

In 12c Cloud Control, you are prompted for credentials that are used for both logging on to the host (the computer on which the database resides) and logging in to the database itself. Enter the credentials. You must be logged on as sys and as sysdba to startup or shutdown the database. From the Oracle Database Menu,

You can then select the Control | Startup/Shutdown action as shown above.



Starting Up an Oracle Database Instance: NOMOUNT

When starting the database instance, select the state in which it starts. The following scenarios describe different stages of starting up an instance.

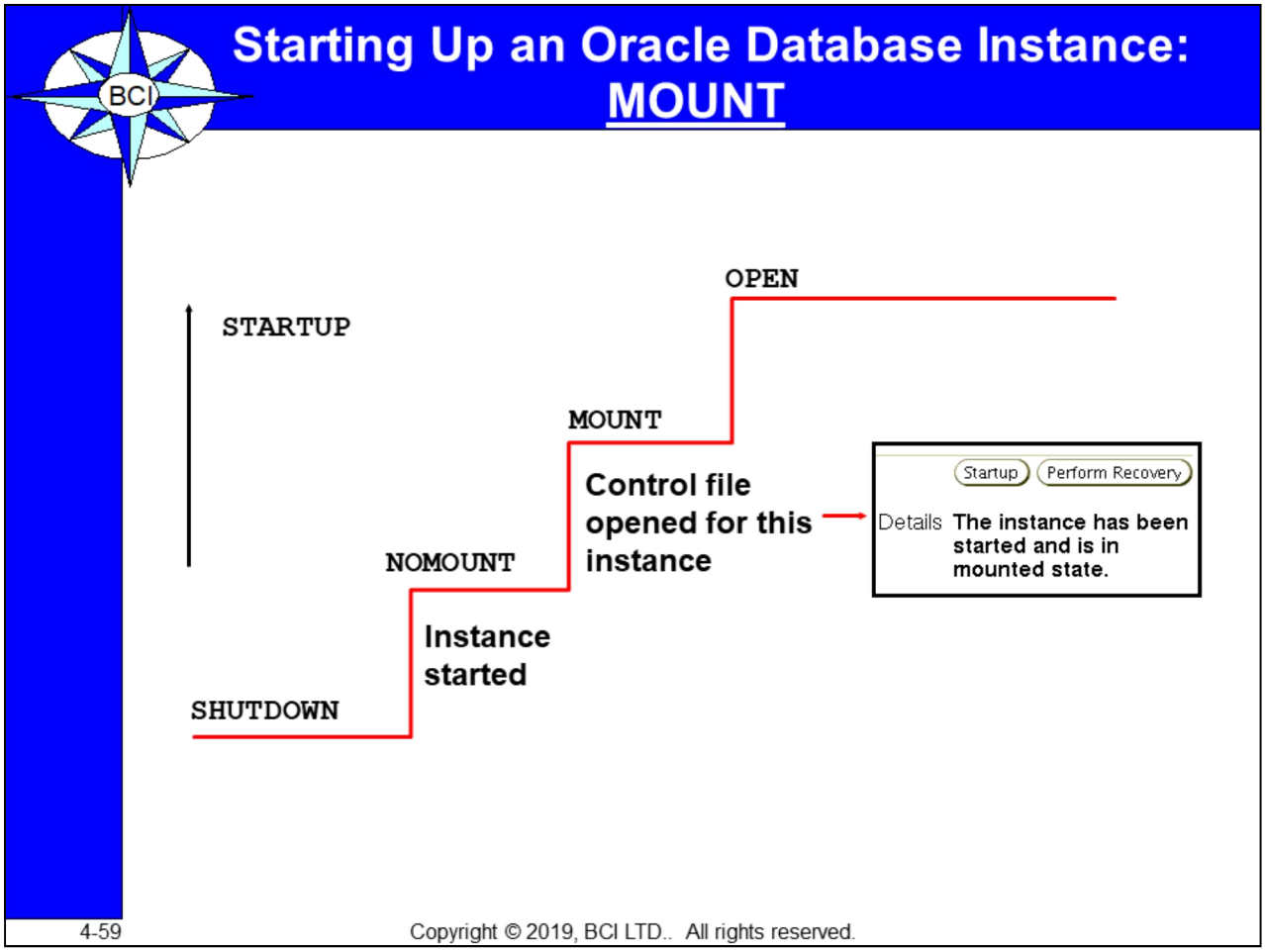
An instance is typically started only in `NOMOUNT` mode during database creation, during re-creation of control files, or during certain backup and recovery scenarios.

Starting an instance includes the following tasks:

- Searching `<oracle_home>/database` for a file of a particular name in this order:
  - `spfile<SID>.ora`
  - If not found, `spfile.ora`
  - If not found, `init<SID>.ora`

    This is the file that contains initialization parameters for the instance. Specifying the `PFILE` parameter with `STARTUP` overrides the default behavior.
- Allocating the SGA
- Starting the background processes
- Opening the `alert<SID>.log` file and the trace files

Note: `SID` is the system ID, which identifies the instance (for example, `ORCL`).



Starting Up an Oracle Database Instance: MOUNT

Mounting a database includes the following tasks:

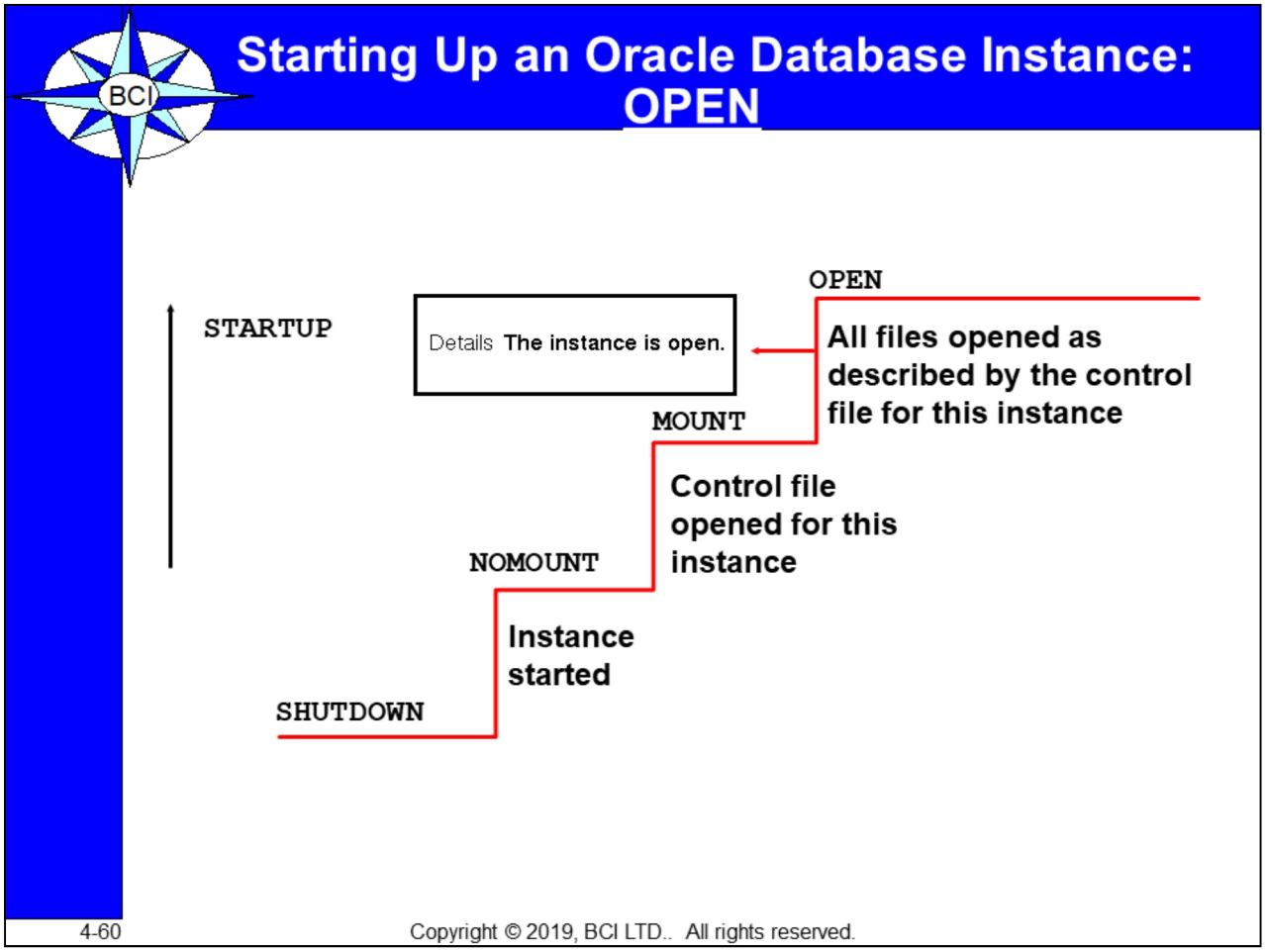
- Associating a database with a previously started instance
- Locating and opening the control files specified in the parameter file
- Reading the control files to obtain the names and statuses of the data files and online redo log files. However, no checks are performed to verify the existence of the data files and online redo log files at this time.

To perform specific maintenance operations, start an instance and mount a database, but do not open the database.

For example, the database must be mounted but must not be opened during the following tasks:

- Renaming data files (Data files for an offline tablespace can be renamed when the database is open.)
- Enabling and disabling online redo log file archiving options
- Performing full database recovery

Note: A database may be left in `MOUNT` mode even though an `OPEN` request has been made. This may be because the database needs to be recovered in some way.



Starting Up an Oracle Database Instance: OPEN

A normal database operation means that an instance is started and the database is mounted and opened. With a normal database operation, any valid user can connect to the database and perform typical data access operations.

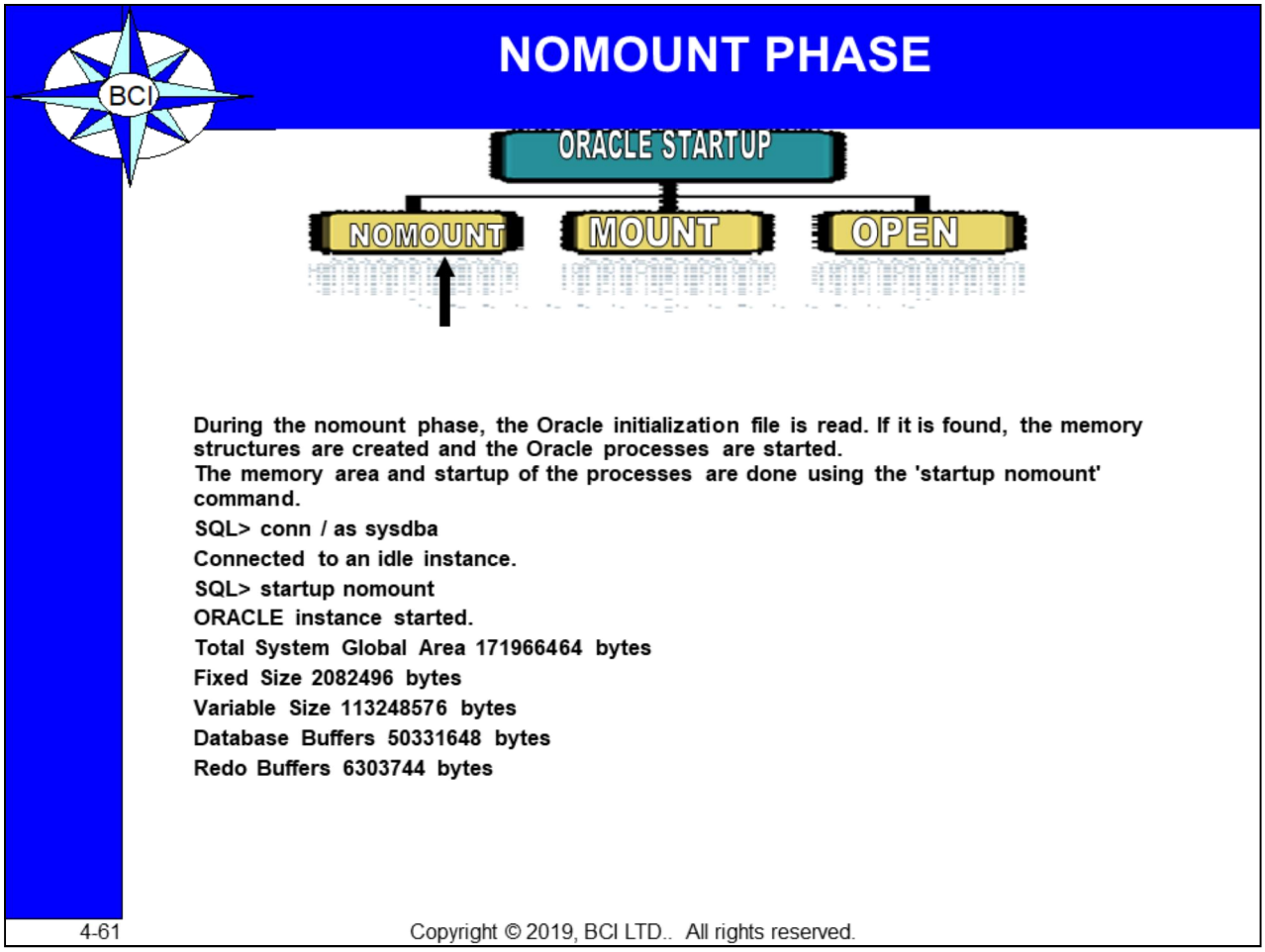
Opening the database includes the following tasks:

- Opening the online data files
- Opening the online redo log files

If any of the data files or online redo log files are not present when you attempt to open the database, then the Oracle server returns an error.

During this final stage, the Oracle server verifies that all the data files and online redo log files can be opened and checks the consistency of the database. If necessary, the System Monitor (SMON) background process initiates instance recovery.

You can start up a database instance in restricted mode so that it is available to users with administrative privileges only. To start an instance in restricted mode, select the “Restrict access to database” option on the Advanced Startup Options page.

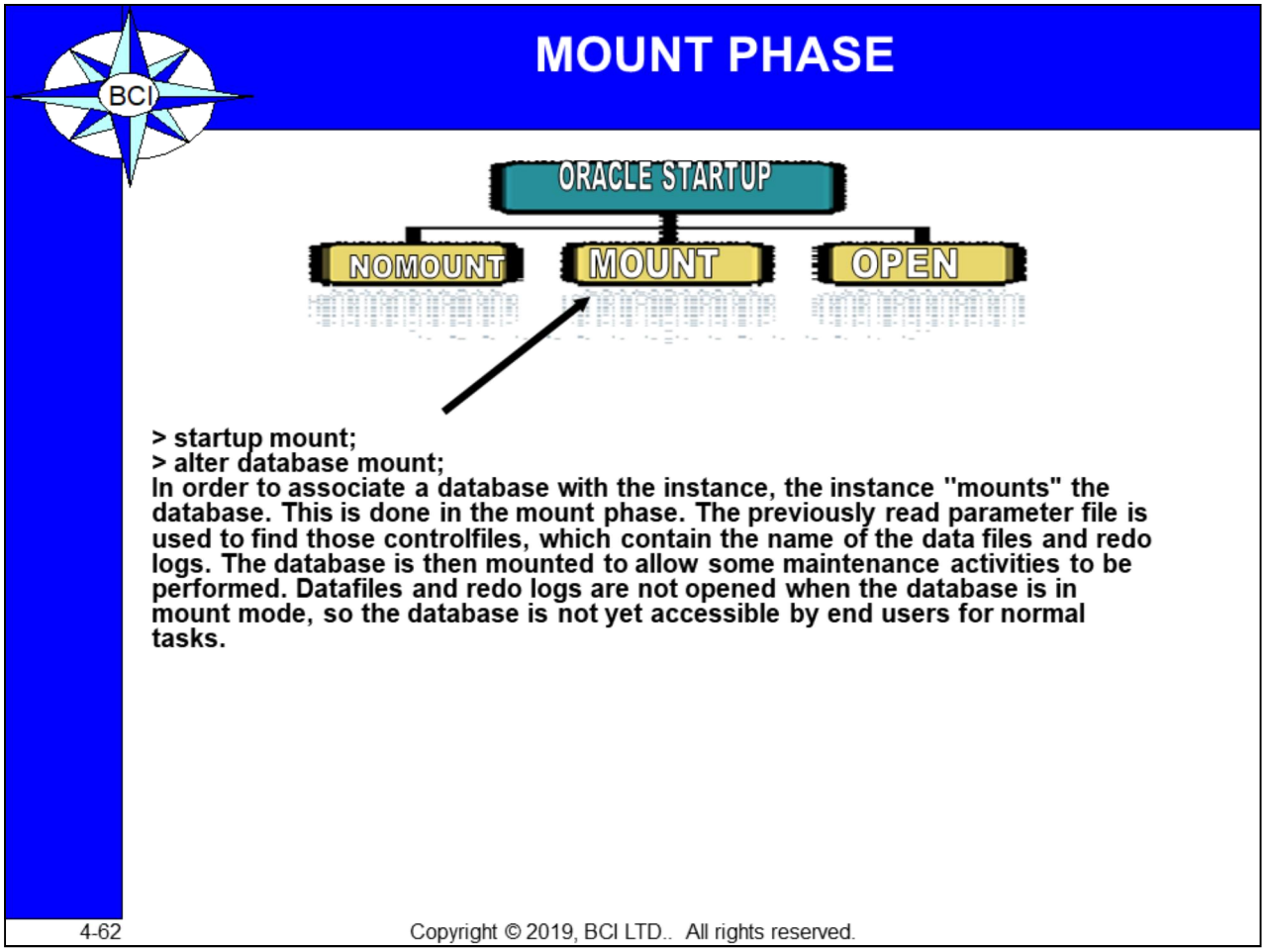


We can check the v$instance view if the instance was started successfully and check the status of the instance through the following command:

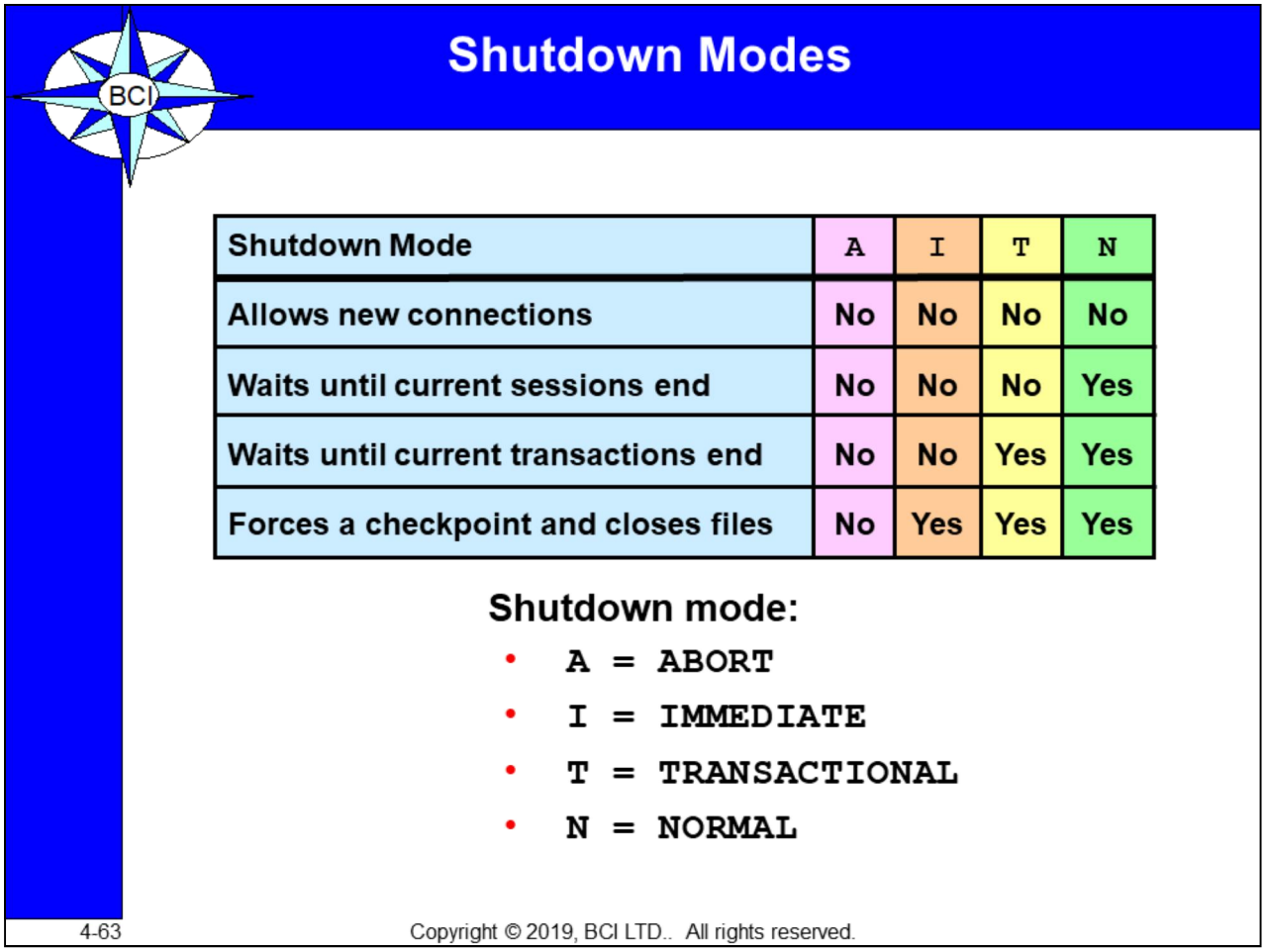
SQL> select instance_name, status from v$instance;

INSTANCE_NAME STATUS ---------- ---------- ORA12c STARTED

Because we only have read the initialization file, we cannot check the status of the database.



Force Option
> startup force
Over time, you will run into situations where Oracle has not shutdown properly and you are unable to restart it. In these rare instances, you will need to use the force option of the startup command. This will first perform a "shutdown abort" that forces the database to shutdown followed by a database startup. This command actually takes Oracle through three distinct startup phases automatically which are nomount mount open, or you could also choose to explicitly step through these phases:In the nomount phase, the database reads the spfile/pfile and starts up the Oracle Instance, but the database is not yet associated with the newly started instance. This is usually used in cases where you need to re-create the controlfile. The command to perform this isIn order to associate a database with the instance, the instance "mounts" the database. This is done in the mount phase. The previously read parameter file is used to find those controlfiles, which contain the name of the data files and redo logs. The database is then mounted to allow some maintenance activities to be performed. Datafiles and redo logs are not opened when the database is in mount mode, so the database is not yet accessible by end users for normal tasks. When Oracle opens the database in the open phase, it opens the data files and redo logs, making the database available for normal operations. Your redo logs must exist in order for the database to open. If they do not, the resetlogs command must be used to create new redo logs in the location specified in the control files.So that no database changes (inserts, updates, or deletes) can be performed.Only users with both the create session and restricted session privileges will be able to use the database.only the sys and system users can query the database without stopping the database and performing a subsequent startup restrict. The activities of other users continue until they become inactive. Over time, you will run into situations where Oracle has not shutdown properly and you are unable to restart it. In these rare instances, you will need to use the force option of the startup command. This will first perform a "shutdown abort" that forces the database to shutdown followed by a database startup. <

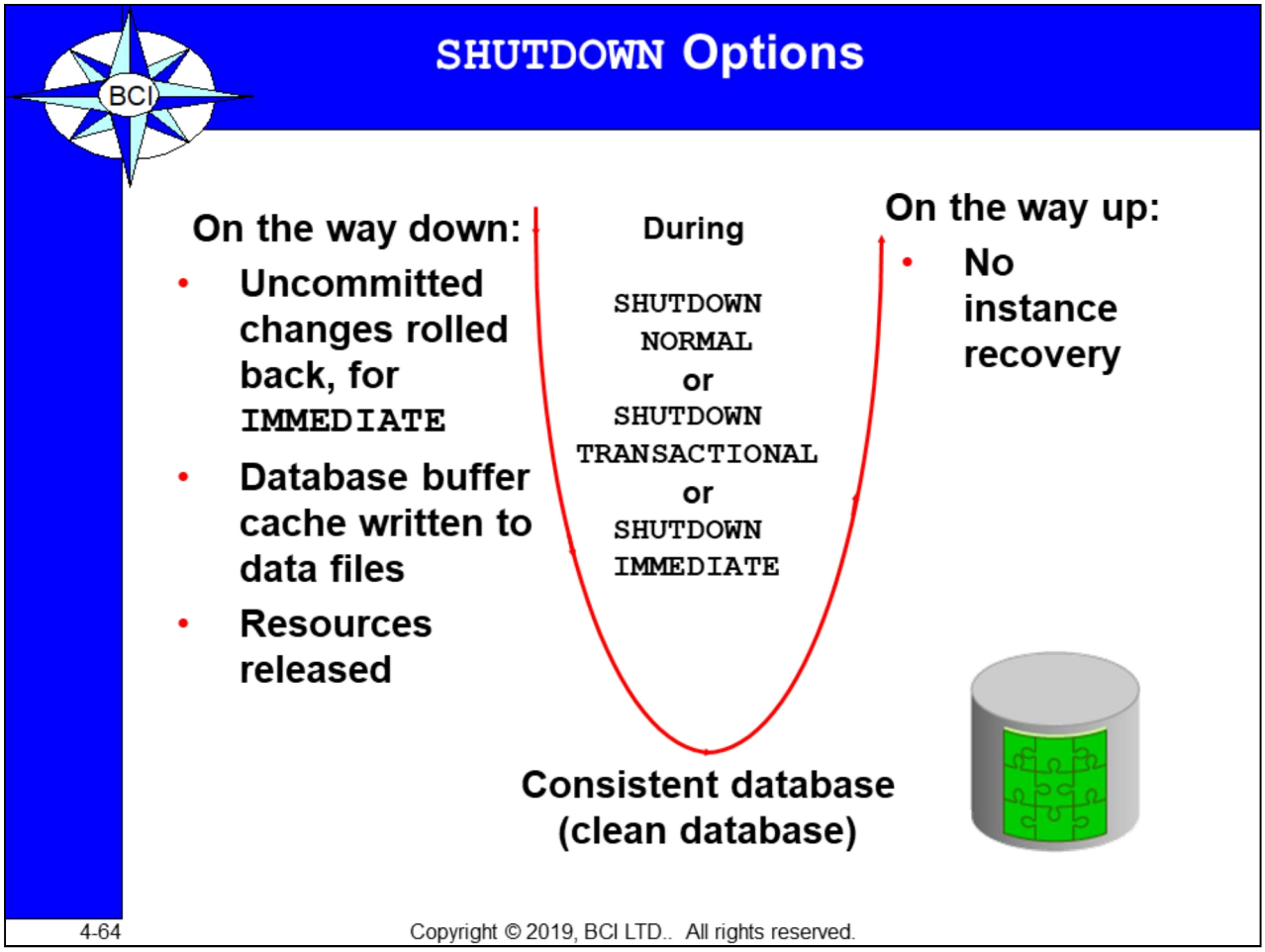


Shutdown Modes

Shutdown modes are progressively more accommodating of current activity in this order:

- `ABORT`: Performs the least amount of work before shutting down. Because this requires recovery before startup, use this only when necessary. This is typically used when no other form of shutdown works, when there are problems when starting the instance, or when you need to shut down immediately because of an impending situation, such as notice of a power outage within seconds.
- `IMMEDIATE`: Is the most typically used option. Uncommitted transactions are rolled back.
- `TRANSACTIONAL`: Allows transactions to finish
- `NORMAL`: Waits for sessions to disconnect

If you consider the amount of time that it takes to perform the shutdown, you find that `ABORT` is the fastest and `NORMAL` is the slowest.



SHUTDOWN Options

SHUTDOWN NORMAL

Normal is the default shutdown mode. A normal database shutdown proceeds with the following conditions:

- No new connections can be made.
- The Oracle server waits for all users to disconnect before completing the shutdown.
- Database and redo buffers are written to disk.
- Background processes are terminated and the SGA is removed from memory.
- The Oracle server closes and dismounts the database before shutting down the instance.
- The next startup does not require an instance recovery.

SHUTDOWN TRANSACTIONAL

A transactional shutdown prevents clients from losing data, including the results from their current activity. A transactional database shutdown proceeds with the following conditions:

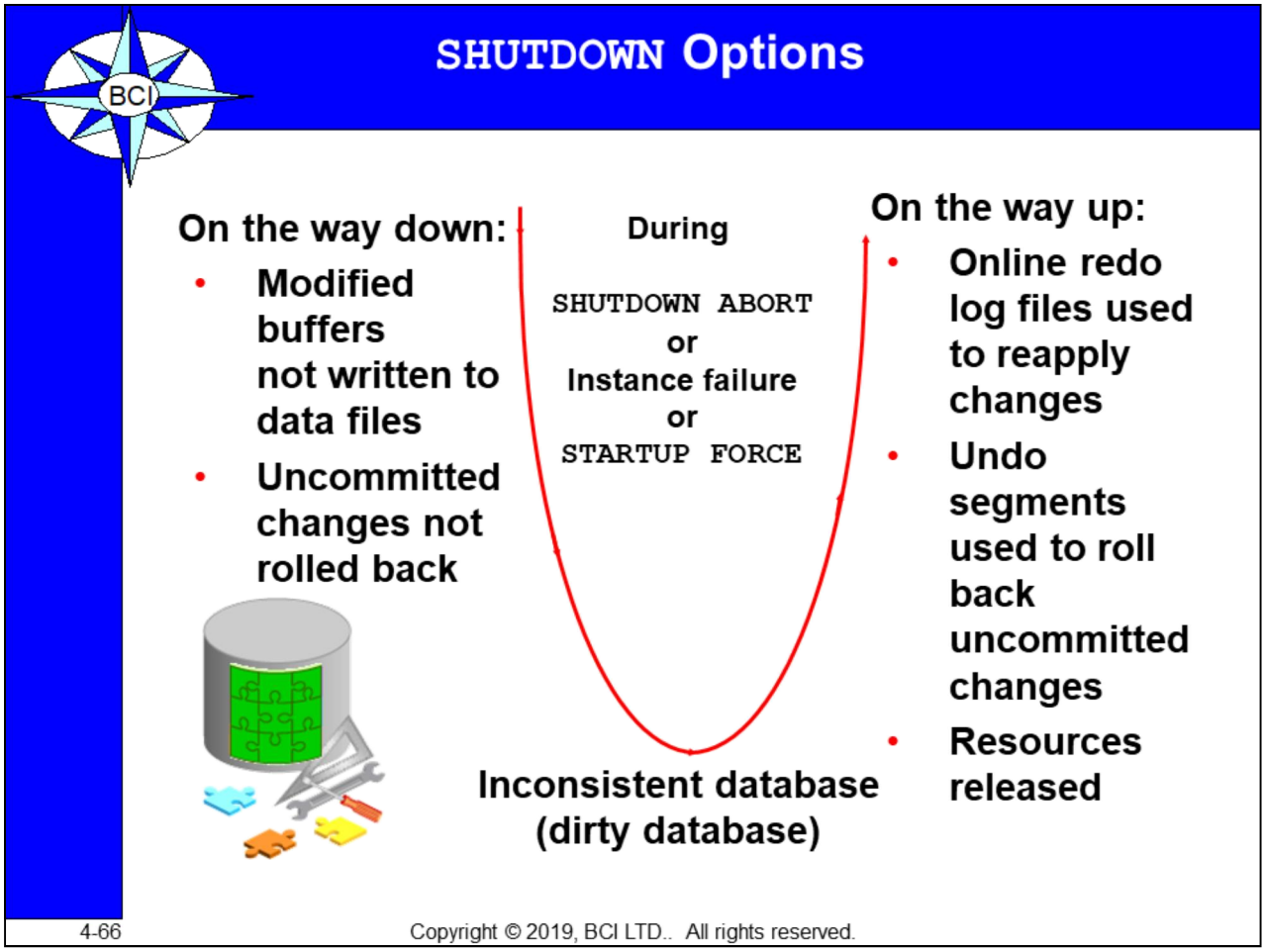
- No client can start a new transaction on this particular instance.
- A client is disconnected when the client ends the transaction that is in progress.
- When all transactions have been completed, a shutdown occurs immediately.
- The next startup does not require an instance recovery.

SHUTDOWN Options (continued)

SHUTDOWN IMMEDIATE

Immediate database shutdown proceeds with the following conditions:

- Current SQL statements being processed by the Oracle database are not completed.
- The Oracle server does not wait for the users who are currently connected to the database to disconnect.
- The Oracle server rolls back active transactions and disconnects all connected users.
- The Oracle server closes and dismounts the database before shutting down the instance.
- The next startup does not require an instance recovery.



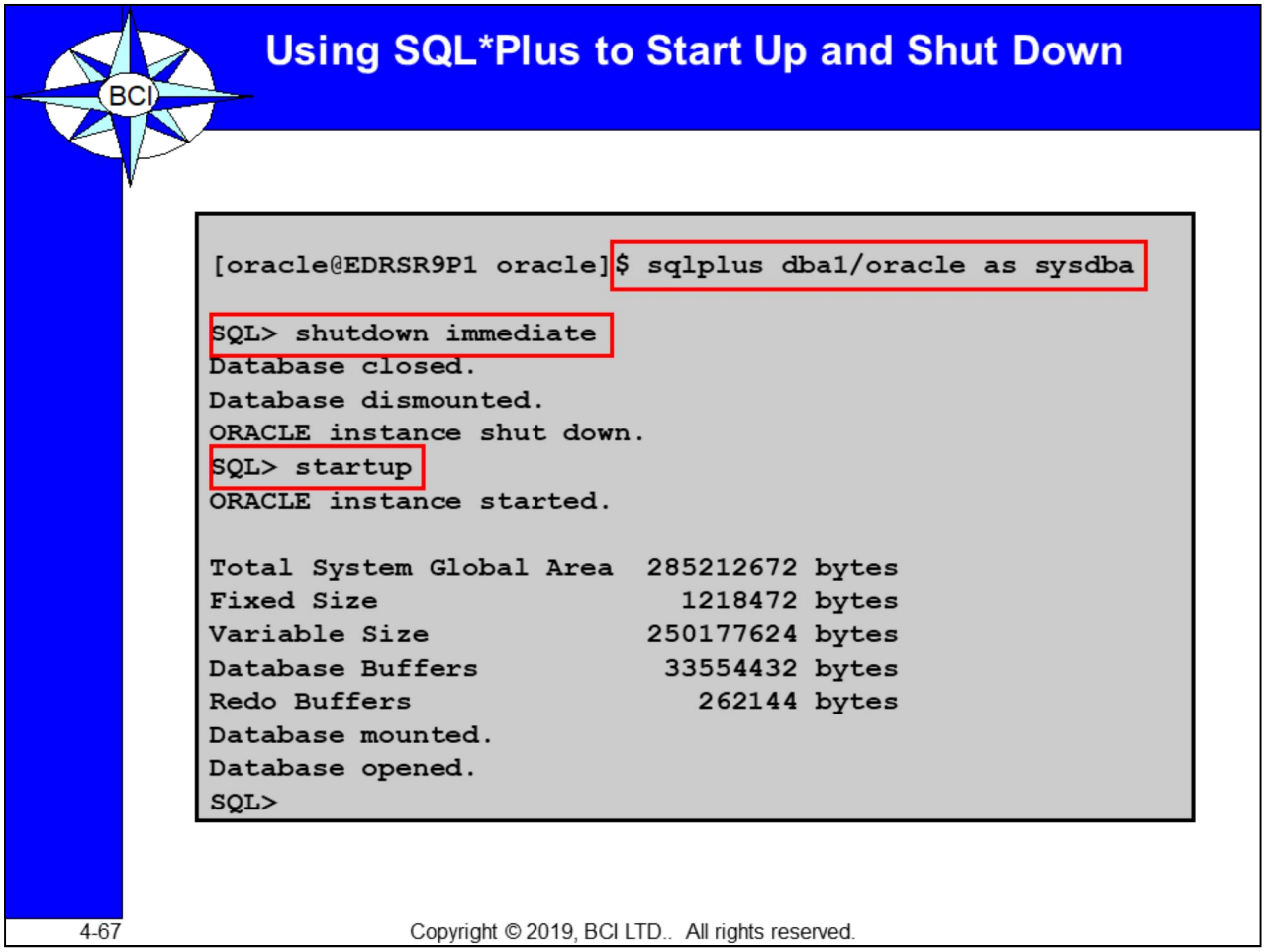
SHUTDOWN Options (continued)

SHUTDOWN ABORT

If the NORMAL and IMMEDIATE shutdown options do not work, you can abort the current database instance. Aborting an instance proceeds with the following conditions:

- Current SQL statements being processed by the Oracle server are immediately terminated.
- The Oracle server does not wait for users currently connected to the database to disconnect.
- Database and redo buffers are not written to disk.
- Uncommitted transactions are not rolled back.
- The instance is terminated without closing the files.
- The database is not closed or dismounted.
- The next startup requires instance recovery, which occurs automatically.

Note: It is not advisable to back up a database that is in an inconsistent state.



Using SQL*Plus to Start Up and Shut Down

You can also use SQL*Plus to start up, shut down, and otherwise change the state of the database. To use SQL*Plus for these tasks, you must log in as `SYSDBA` or `SYSOPER`. Then, use the equivalent commands for the Enterprise Manager functionality discussed earlier:

```
SHUTDOWN [NORMAL | TRANSACTIONAL | IMMEDIATE | ABORT ]

STARTUP [FORCE] [RESTRICT] [MOUNT | OPEN | NOMOUNT]
```

This enables you to include startup and shutdown operations as part of a script or batch process that performs tasks on the database, where the database needs to be in a particular state.

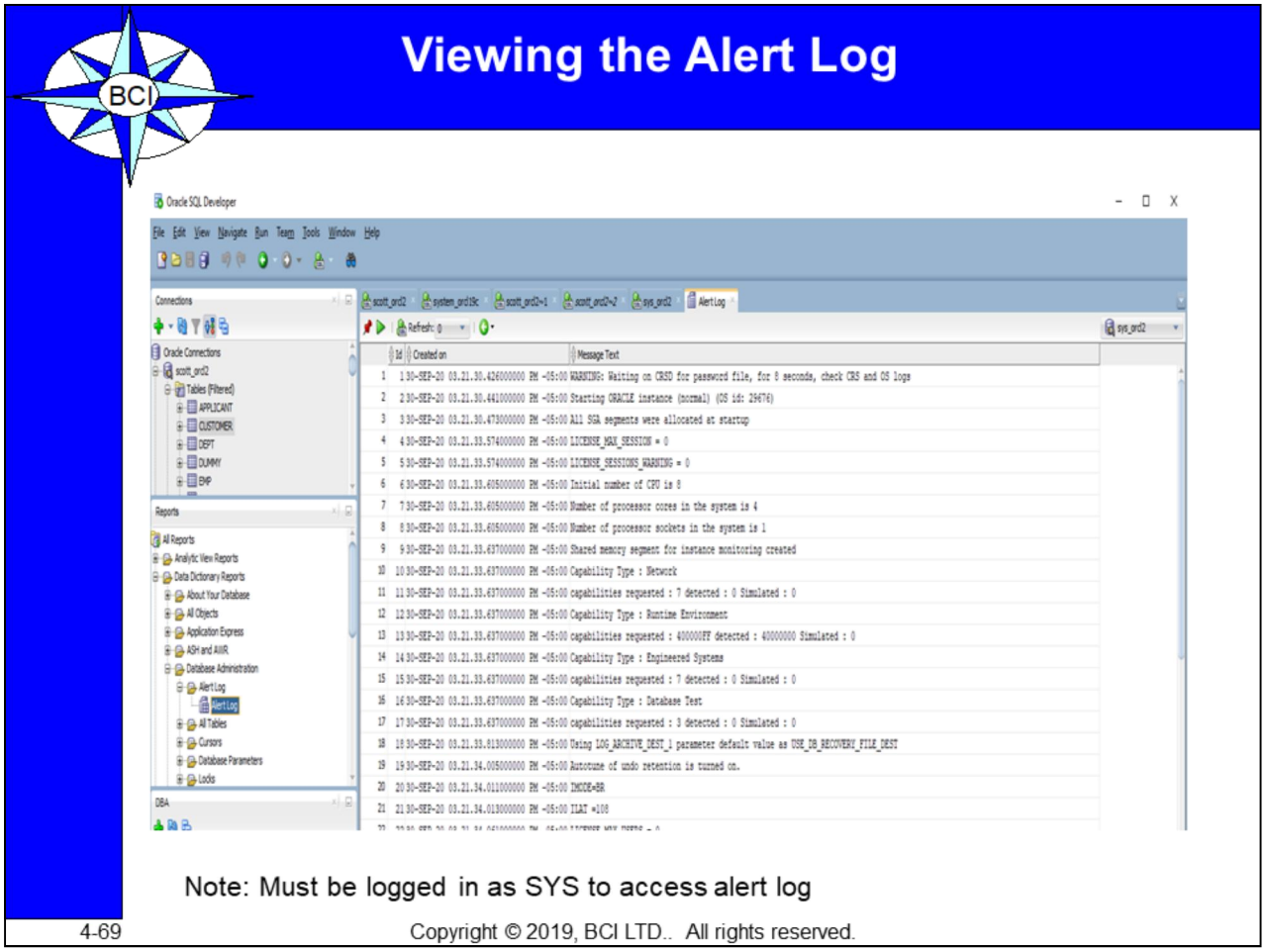
Starting up with a PFILE command

PFILE

This is a plain text file. It is usually maintained by either editing it with the vi editor, or by FTPing it to a client computer, modifying it with Notepad, and then FTPing it back to the Unix server.

The file is only read during database startup so any modifications take effect the next time the database is started up. This is an obvious limitation since shutting down and starting up an Oracle database is not desirable in a 24/7 operating environment.

```
SQL> Startup pfile=/oracle/initorclxx.ora
```



Viewing the Alert Log

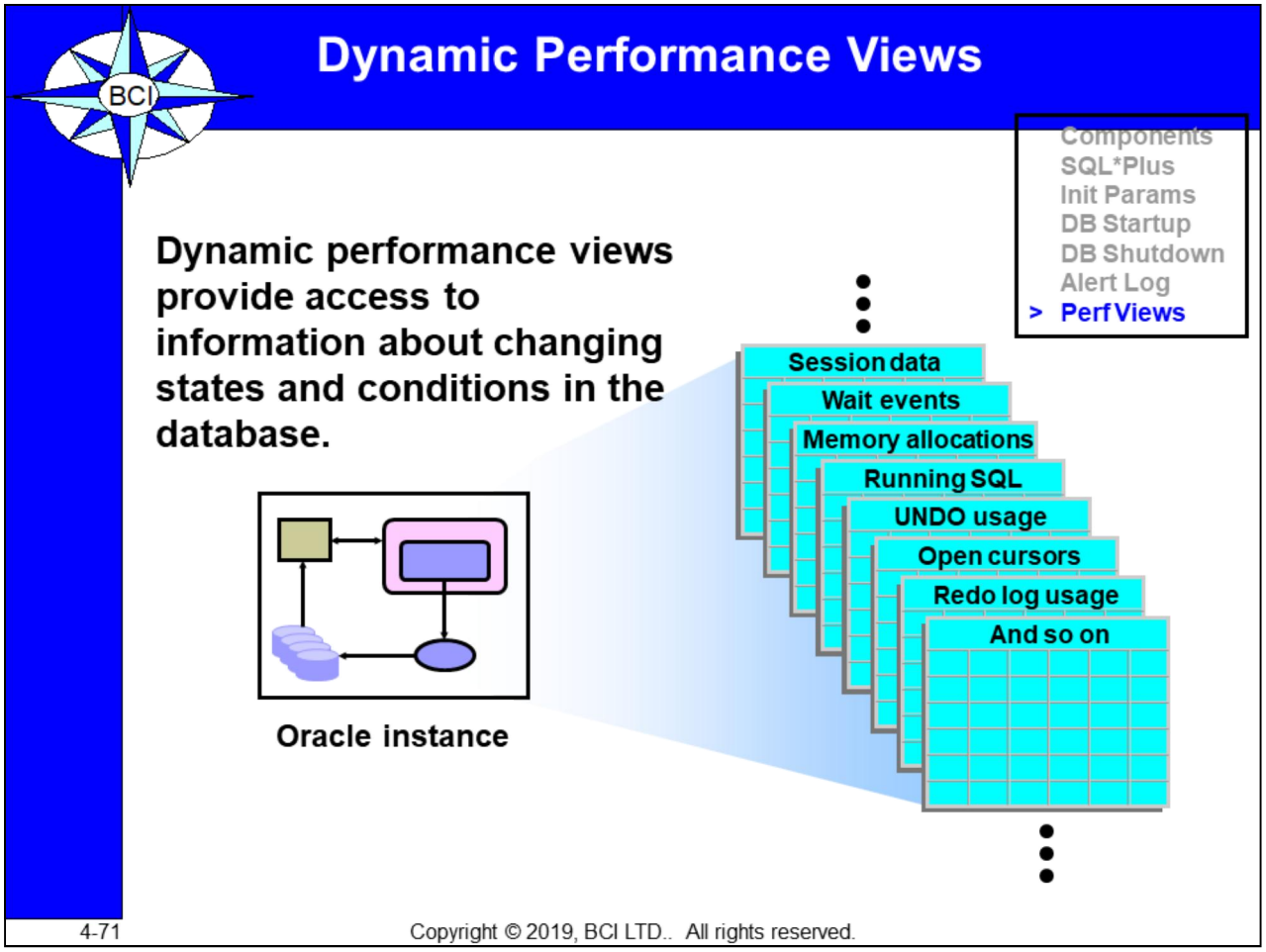
Each database has an `alert_<sid>.log` file. The file is on the server with the database and is stored in the directory specified with the `background_dump_dest` initialization parameter. The alert file of a database is a chronological log of messages and errors, including the following:

- Any nondefault initialization parameters used at startup
- All internal errors (`ORA-600`), block corruption errors (`ORA-1578`), and deadlock errors (`ORA-60`) that occurred
- Administrative operations, such as the SQL statements `CREATE`, `ALTER`, `DROP DATABASE`, and `TABLESPACE`, and the Enterprise Manager or SQL*Plus statements `STARTUP`, `SHUTDOWN`, `ARCHIVE LOG`, and `RECOVER`
- Several messages and errors relating to the functions of shared server and dispatcher processes
- Errors during the automatic refresh of a materialized view

Enterprise Manager monitors the alert log file and notifies you of critical errors. You can also view the log to see noncritical error and informative messages. The file can grow to an unmanageable size. You can occasionally back up the alert file and delete the current alert file. When the database attempts to write to the alert file again, it re-creates a new one.

Viewing the Alert History

The Alert History page displays a chart that shows the alert history of the current database in segments of time, which you designate. An alert indicates a potential problem: either a warning or critical threshold for a monitored metric, or that a target is no longer available.



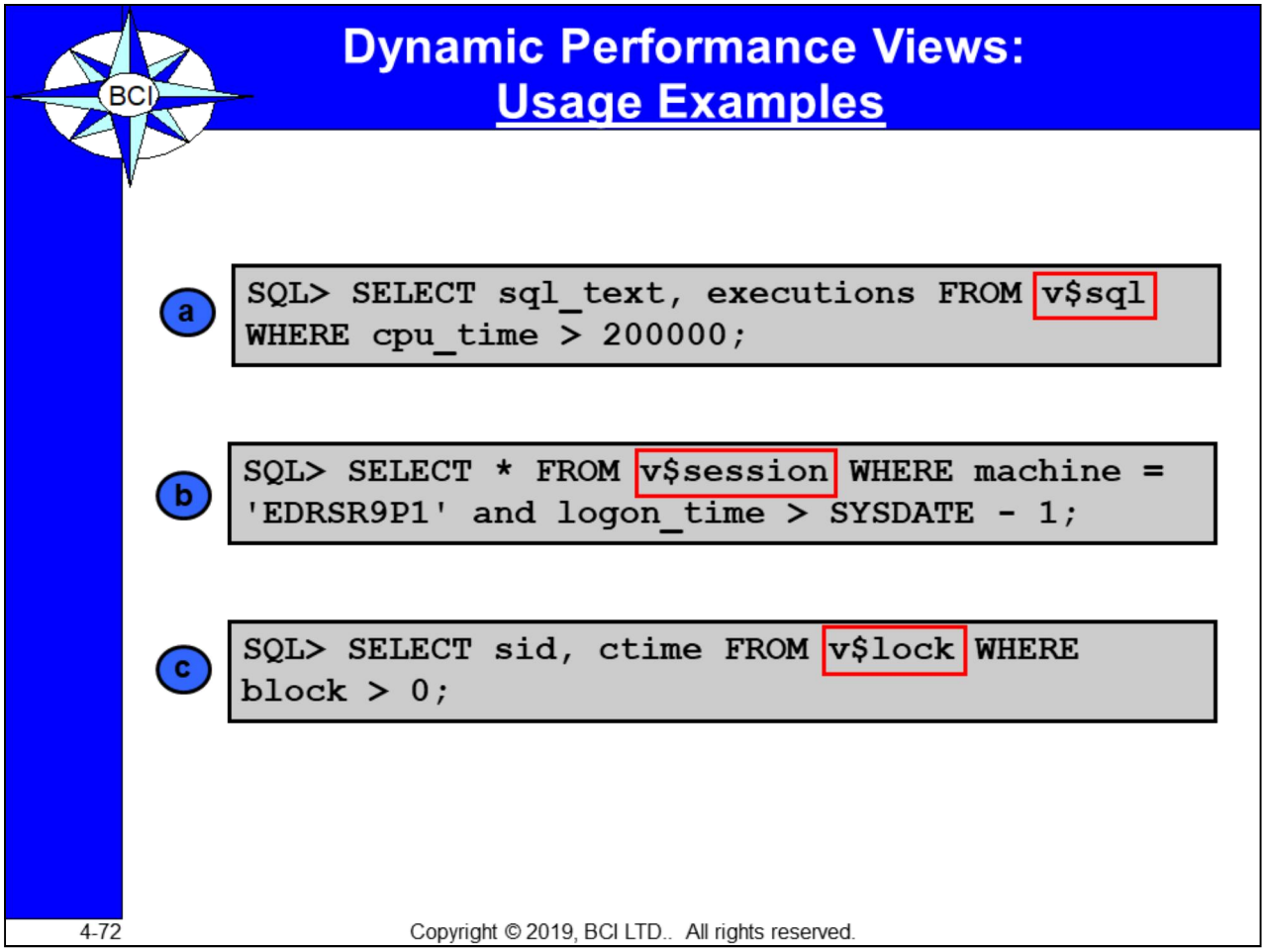
Dynamic Performance Views

The Oracle database also maintains a more dynamic set of data about the operation and performance of the database instance. These dynamic performance views are based on virtual tables that are built from memory structures inside the database server. That is, they are not conventional tables that reside in a database. This is why some of them can show you data before a database is mounted or open.

Dynamic performance views include information about:

- Sessions
- File states
- Progress of jobs and tasks
- Locks
- Backup status
- Memory usage and allocation
- System and session parameters
- SQL execution
- Statistics and metrics

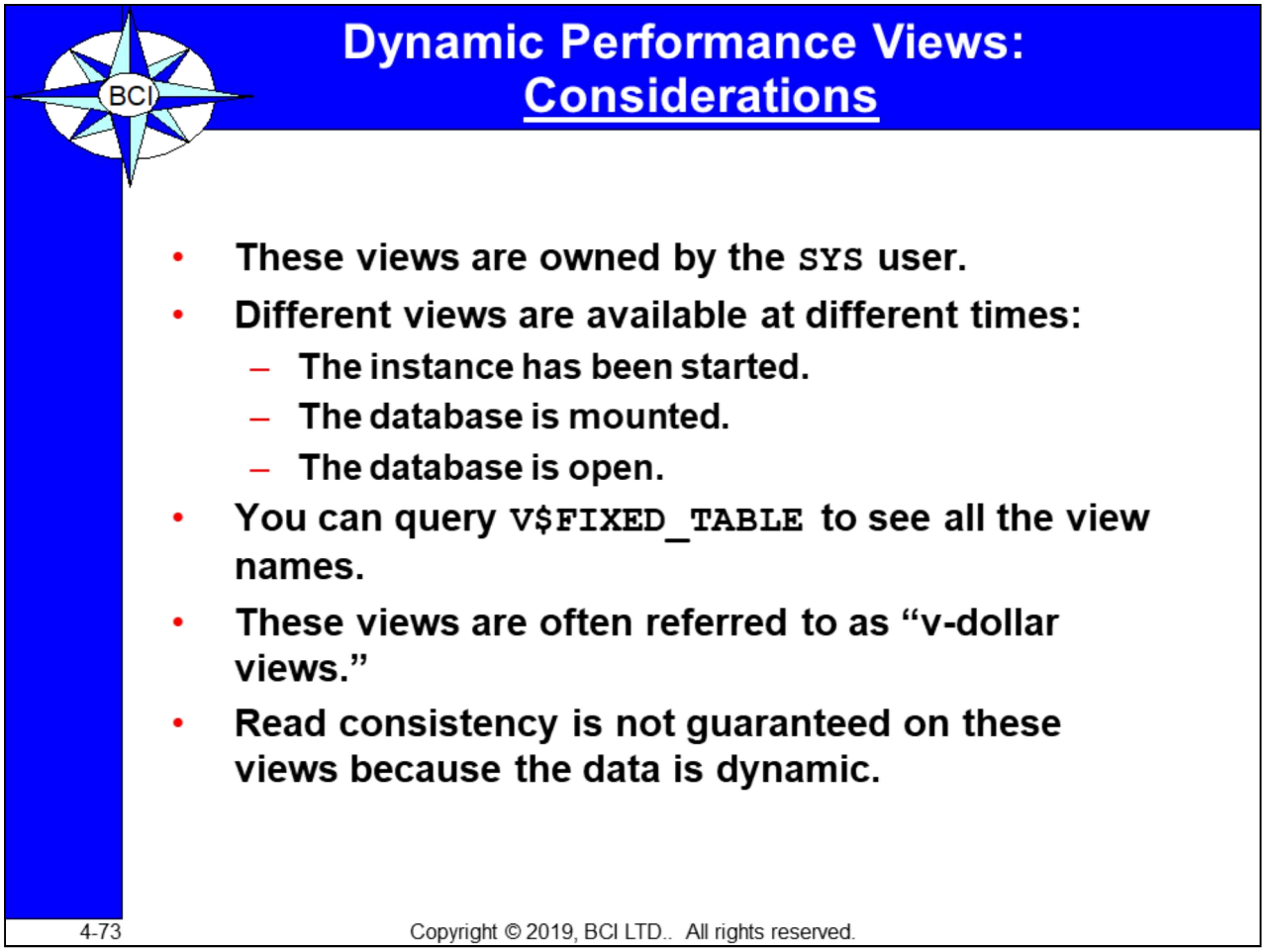
Note: The `DICT` and `DICT_COLUMNS` views also contain the names of these dynamic performance views.



Dynamic Performance Views: Usage Examples

A frequent user of these views is Enterprise Manager, but users can also query these views as needed. The three examples shown in the slide answer the following questions:

a. What are the SQL statements and their associated number of executions where the CPU time consumed is greater than 200,000 microseconds?
b. What sessions logged in from the `EDRSR9P1` computer within the last day?
c. What are the session IDs of any sessions that are currently holding a lock that is blocking another user, and how long has that lock been held?



Dynamic Performance Views: Considerations

Some dynamic views contain data that is not applicable to all states of an instance or database. For example, if an instance has just been started, but no database is mounted, you can query `V$BGPROCESS` to see the list of background processes that are running. But you cannot query `V$DATAFILE` to see the status of database data files because it is the mounting of a database that reads the control file to find out about the data files associated with a database.

Practical Exercise: Managing the Oracle Instance

This practice covers the following topics:

- Navigating in Enterprise Manager
- Viewing and modifying initialization parameters
- Stopping and starting the database instance
- Viewing the alert log
- Connecting to the database by using SQL*Plus and *SQL DEVELOPER*

Summary

In this lesson, you should have learned how to:

- Start and stop the Oracle database and components
- Use Enterprise Manager and describe its high-level functionality
- Introduce Database Express
- Access a database with SQL*Plus and *i*SQL*Plus
- Modify database initialization parameters
- Describe the stages of database startup
- Describe the database shutdown options
- View the alert log
- Access dynamic performance views

Terminal Learning Objective

ACTION: Manage the Oracle Instance.

CONDITION: Given a student handout and Oracle DBA Handbook.

STANDARD: Students will successfully start and stop the Oracle Database.