

1

Introduction to PL/SQL

ORACLE®

Copyright © 2006, Oracle. All rights reserved.

Objectives

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

- **Explain the need for PL/SQL**
- **Explain the benefits of PL/SQL**
- **Identify the different types of PL/SQL blocks**
- **Use *iSQL*Plus* as a development environment for PL/SQL**
- **Output messages in PL/SQL**

ORACLE®

Copyright © 2006, Oracle. All rights reserved.

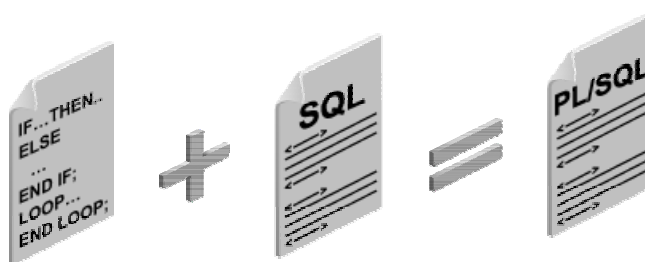
Lesson Aim

This lesson introduces PL/SQL and PL/SQL programming constructs. You learn about the benefits of PL/SQL. You also learn to use *iSQL*Plus* as a development environment for PL/SQL.

What Is PL/SQL?

PL/SQL:

- **Stands for Procedural Language extension to SQL**
- **Is Oracle Corporation's standard data access language for relational databases**
- **Seamlessly integrates procedural constructs with SQL**



ORACLE

Copyright © 2006, Oracle. All rights reserved.

What Is PL/SQL?

Structured Query Language (SQL) is the primary language used to access and modify data in relational databases. There are only a few SQL commands, so you can easily learn and use them. Consider an example:

```
SELECT first_name, department_id, salary FROM employees;
```

The SQL statement shown above is simple and straightforward. However, if you want to alter any data that is retrieved in a conditional manner, you soon encounter the limitations of SQL.

Consider a slightly modified problem statement: For every employee retrieved, check the `department_id` and the salary. Depending on the department's performance and also the employee's salary, you may want to provide varying bonuses to the employees.

Looking at the problem, you know that you have to execute the preceding SQL statement, collect the data, and apply logic to the data. One solution is to write a SQL statement for each department to give bonuses to the employees in that department. Remember that you also have to check the salary component before deciding the bonus amount. This makes it a little complicated. You now feel that it would be much easier if you had conditional statements. PL/SQL is designed to meet such requirements. It provides a programming extension to already-existing SQL.

About PL/SQL

PL/SQL:

- **Provides a block structure for executable units of code. Maintenance of code is made easier with such a well-defined structure.**
- **Provides procedural constructs such as:**
 - **Variables, constants, and types**
 - **Control structures such as conditional statements and loops**
 - **Reusable program units that are written once and executed many times**

ORACLE

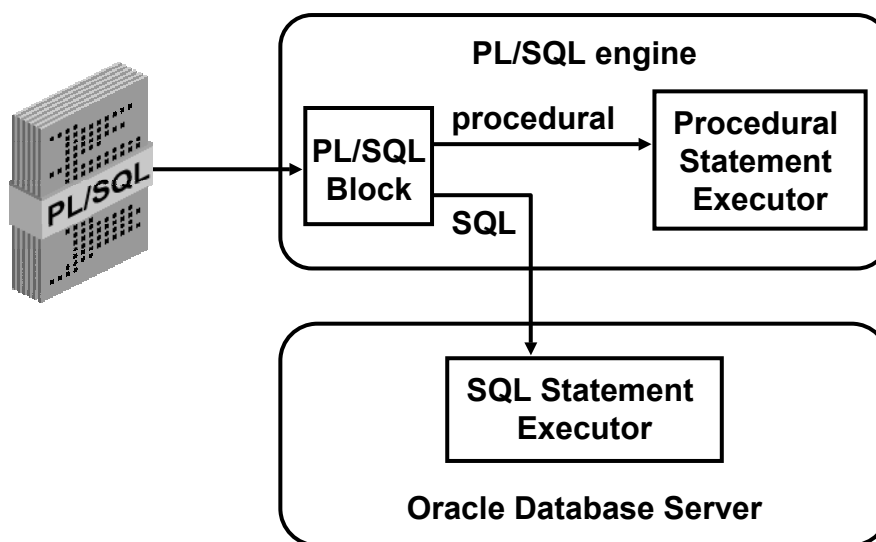
Copyright © 2006, Oracle. All rights reserved.

About PL/SQL

PL/SQL defines a block structure for writing code. Maintaining and debugging the code is made easier with such a structure. One can easily understand the flow and execution of the program unit.

PL/SQL offers modern software engineering features such as data encapsulation, exception handling, information hiding, and object orientation. It brings state-of-the-art programming to the Oracle server and toolset. PL/SQL provides all the procedural constructs that are available in any third-generation language (3GL).

PL/SQL Environment



ORACLE®

Copyright © 2006, Oracle. All rights reserved.

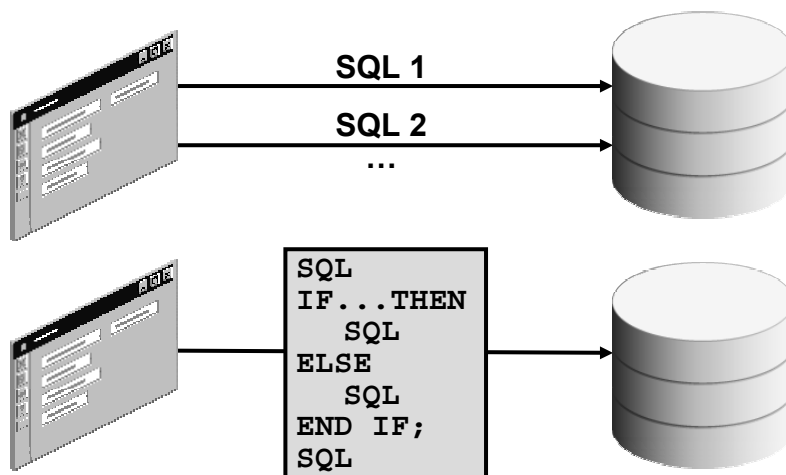
PL/SQL Environment

The slide shows the PL/SQL execution environment in the Oracle database server. A PL/SQL block contains procedural statements and SQL statements. When you submit the PL/SQL block to the server, the PL/SQL engine first parses the block. The PL/SQL engine identifies the procedural statements and SQL statements. It passes the procedural statements to the procedural statement executor and passes the SQL statements to the SQL statement executor individually.

The diagram in the slide shows the PL/SQL engine within the database server. The Oracle application development tools can also contain a PL/SQL engine. The tool passes the blocks to its local PL/SQL engine. Therefore, all procedural statements are executed locally and only the SQL statements are executed in the database. The engine used depends on where the PL/SQL block is being invoked from.

Benefits of PL/SQL

- **Integration of procedural constructs with SQL**
- **Improved performance**



ORACLE

Copyright © 2006, Oracle. All rights reserved.

Benefits of PL/SQL

Integration of procedural constructs with SQL: The most important advantage of PL/SQL is the integration of procedural constructs with SQL. SQL is a nonprocedural language. When you issue a SQL command, your command tells the database server *what* to do. However, you cannot specify *how* to do it. PL/SQL integrates control statements and conditional statements with SQL, giving you better control of your SQL statements and their execution. Earlier in this lesson, you saw an example of the need for such integration.

Improved performance: Without PL/SQL, you would not be able to logically combine SQL statements as one unit. If you have designed an application containing forms, you may have many different forms with fields in each form. When a form submits the data, you may have to execute a number of SQL statements. SQL statements are sent to the database one at a time. This results in many network trips and one call to the database for each SQL statement, thereby increasing network traffic and reducing performance (especially in a client/server model).

With PL/SQL, you can combine all these SQL statements into a single program unit. The application can send the entire block to the database instead of sending the SQL statements one at a time. This significantly reduces the number of database calls. As the slide illustrates, if the application is SQL intensive, you can use PL/SQL blocks to group SQL statements before sending them to the Oracle database server for execution.

Benefits of PL/SQL

- **Modularized program development**
- **Integration with Oracle tools**
- **Portability**
- **Exception handling**

ORACLE

Copyright © 2006, Oracle. All rights reserved.

Benefits of PL/SQL (continued)

Modularized program development: A basic unit in all PL/SQL programs is the block. Blocks can be in a sequence or they can be nested in other blocks. Modularized program development has the following advantages:

- You can group logically related statements within blocks.
- You can nest blocks inside larger blocks to build powerful programs.
- You can break your application into smaller modules. If you are designing a complex application, PL/SQL allows you to break down the application into smaller, manageable, and logically related modules.
- You can easily maintain and debug the code.

Integration with tools: The PL/SQL engine is integrated in Oracle tools such as Oracle Forms, Oracle Reports, and so on. When you use these tools, the locally available PL/SQL engine processes the procedural statements; only the SQL statements are passed to the database.

Benefits of PL/SQL (continued)

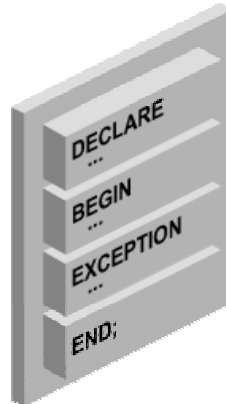
Portability: PL/SQL programs can run anywhere an Oracle server runs, irrespective of the operating system and the platform. You do not need to tailor them to each new environment. You can write portable program packages and create libraries that can be reused in different environments.

Exception handling: PL/SQL enables you to handle exceptions efficiently. You can define separate blocks for dealing with exceptions. You will learn more about exception handling later in the course.

PL/SQL shares the same data type system as SQL (with some extensions) and uses the same expression syntax.

PL/SQL Block Structure

- **DECLARE (optional)**
 - Variables, cursors, user-defined exceptions
- **BEGIN (mandatory)**
 - SQL statements
 - PL/SQL statements
- **EXCEPTION (optional)**
 - Actions to perform when errors occur
- **END; (mandatory)**



ORACLE

Copyright © 2006, Oracle. All rights reserved.

PL/SQL Block Structure

The slide shows a basic PL/SQL block. A PL/SQL block consists of three sections:

- **Declarative (optional):** The declarative section begins with the keyword **DECLARE** and ends when the executable section starts.
- **Executable (required):** The executable section begins with the keyword **BEGIN** and ends with **END**. Observe that **END** is terminated with a semicolon. The executable section of a PL/SQL block can in turn include any number of PL/SQL blocks.
- **Exception handling (optional):** The exception section is nested within the executable section. This section begins with the keyword **EXCEPTION**.

PL/SQL Block Structure (continued)

In a PL/SQL block, the keywords `DECLARE`, `BEGIN`, and `EXCEPTION` are not terminated by a semicolon. However, the keyword `END`, all SQL statements, and PL/SQL statements must be terminated with a semicolon.

Section	Description	Inclusion
Declarative (<code>DECLARE</code>)	Contains declarations of all variables, constants, cursors, and user-defined exceptions that are referenced in the executable and exception sections	Optional
Executable (<code>BEGIN ...</code> <code>END</code>)	Contains SQL statements to retrieve data from the database; contains PL/SQL statements to manipulate data in the block	Mandatory
Exception (<code>EXCEPTION</code>)	Specifies the actions to perform when errors and abnormal conditions arise in the executable section	Optional

Block Types

Anonymous	Procedure	Function
<pre>[DECLARE] BEGIN --statements [EXCEPTION] END;</pre>	<pre>PROCEDURE name IS BEGIN --statements [EXCEPTION] END;</pre>	<pre>FUNCTION name RETURN datatype IS BEGIN --statements RETURN value; [EXCEPTION] END;</pre>

ORACLE

Copyright © 2006, Oracle. All rights reserved.

Block Types

A PL/SQL program comprises one or more blocks. These blocks can be entirely separate or nested within another block. There are three types of blocks that make up a PL/SQL program. They are:

- Anonymous blocks
- Procedures
- Functions

Anonymous blocks: Anonymous blocks are unnamed blocks. They are declared inline at the point in an application where they are to be executed and are compiled each time the application is executed. These blocks are not stored in the database. They are passed to the PL/SQL engine for execution at run time. Triggers in Oracle Developer components consist of such blocks. These anonymous blocks get executed at run time because they are inline. If you want to execute the same block again, you have to rewrite the block. You are unable to invoke or call the block that you wrote earlier because blocks are anonymous and do not exist after they are executed.

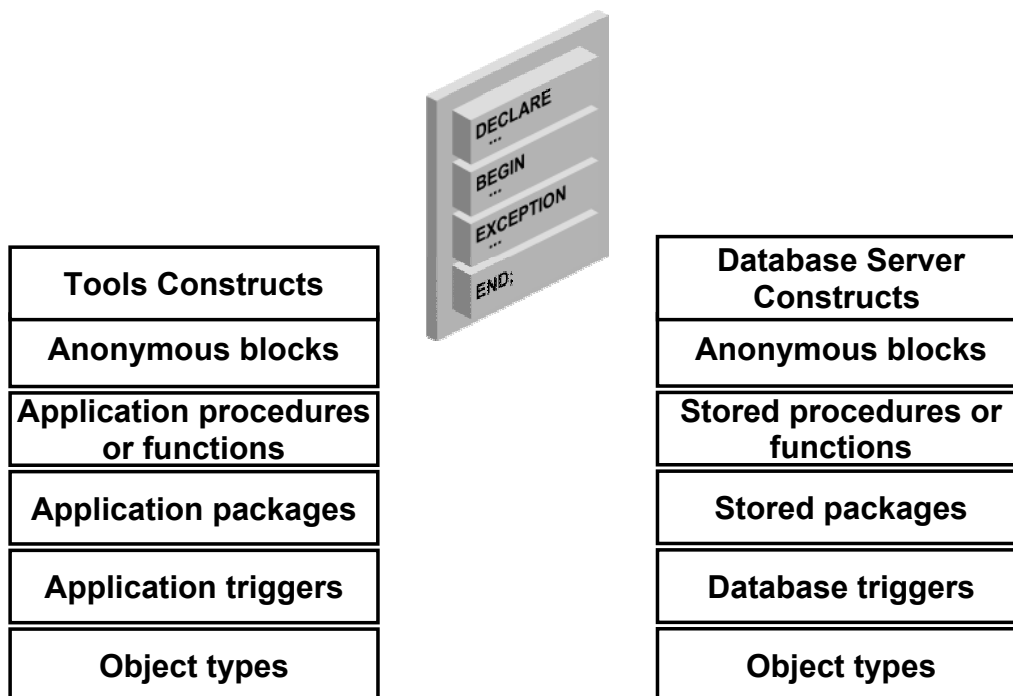
Block Types (continued)

Subprograms: Subprograms are complementary to anonymous blocks. They are named PL/SQL blocks that are stored in the database. Because they are named and stored, you can invoke them whenever you want (depending on your application). You can declare them either as procedures or as functions. You typically use a procedure to perform an action and a function to compute and return a value.

You can store subprograms at the server or application level. Using Oracle Developer components (Forms, Reports), you can declare procedures and functions as part of the application (a form or report) and call them from other procedures, functions, and triggers within the same application whenever necessary.

Note: A function is similar to a procedure, except that a function must return a value.

Program Constructs



ORACLE

Copyright © 2006, Oracle. All rights reserved.

Program Constructs

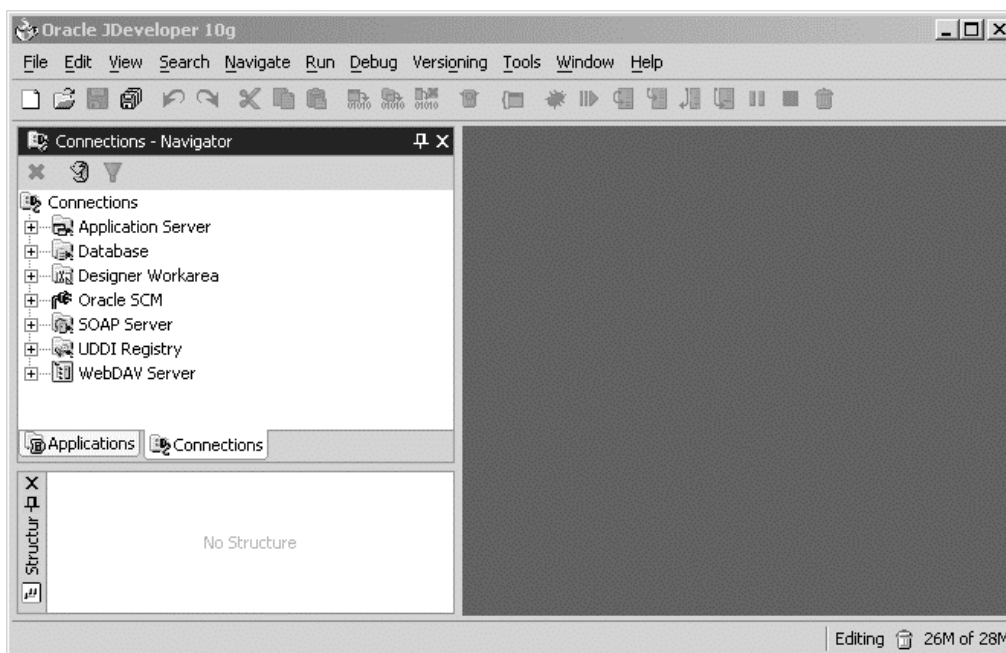
The following table outlines a variety of different PL/SQL program constructs that use the basic PL/SQL block. The program constructs are available based on the environment in which they are executed.

Program Construct	Description	Availability
Anonymous blocks	Unnamed PL/SQL blocks that are embedded within an application or are issued interactively	All PL/SQL environments
Application procedures or functions	Named PL/SQL blocks stored in an Oracle Forms Developer application or shared library; can accept parameters and can be invoked repeatedly by name	Oracle Developer tools components (for example, Oracle Forms Developer, Oracle Reports)
Stored procedures or functions	Named PL/SQL blocks stored in the Oracle server; can accept parameters and can be invoked repeatedly by name	Oracle server or Oracle Developer tools
Packages (application or stored)	Named PL/SQL modules that group related procedures, functions, and identifiers	Oracle server and Oracle Developer tools components (for example, Oracle Forms Developer)

Program Constructs (continued)

Program Construct	Description	Availability
Database triggers	PL/SQL blocks that are associated with a database table and fired automatically when triggered by various events	Oracle server or any Oracle tool that issues the DML
Application triggers	PL/SQL blocks that are associated either with a database table or system events. They are fired automatically when triggered by a DML or a system event respectively.	Oracle Developer tools components (for example, Oracle Forms Developer)
Object types	User-defined composite data types that encapsulate a data structure along with the functions and procedures needed to manipulate the data	Oracle server and Oracle Developer tools

PL/SQL Programming Environments



Copyright © 2006, Oracle. All rights reserved.

PL/SQL Programming Environments

Oracle JDeveloper 10g: An integrated development environment (IDE) that provides end-to-end support for building, testing, and deploying J2EE applications, Web services, and PL/SQL

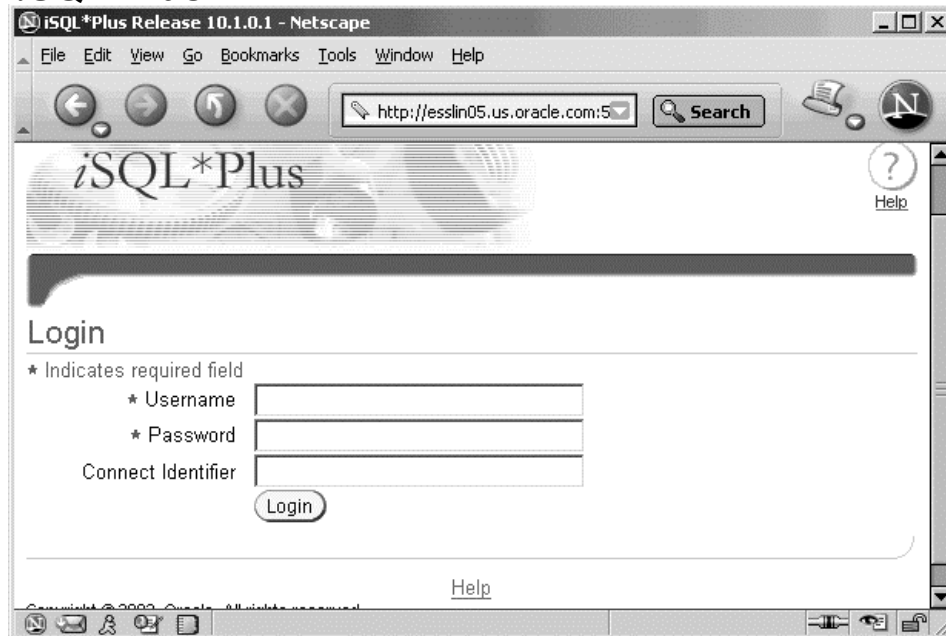
You can use Oracle JDeveloper 10g to do the following:

- Establish connection to the database with a user-friendly wizard
- Browse through the objects in the database you are connected to
- Create database users and objects
- Create, run, and debug PL/SQL programs such as procedures, functions, and packages

Note: Oracle JDeveloper 10g and *iSQL*Plus* can both be used as programming environments. However, this course uses *iSQL*Plus* for all demonstrations and practices.

PL/SQL Programming Environments

iSQL*Plus



PL/SQL Programming Environments (continued)

iSQL*Plus: A browser-based interface to SQL*Plus. You can connect to the local database or remote database by using iSQL*Plus. It enables you to perform all the operations that you can perform with the command-line version of SQL*Plus.

PL/SQL Programming Environments



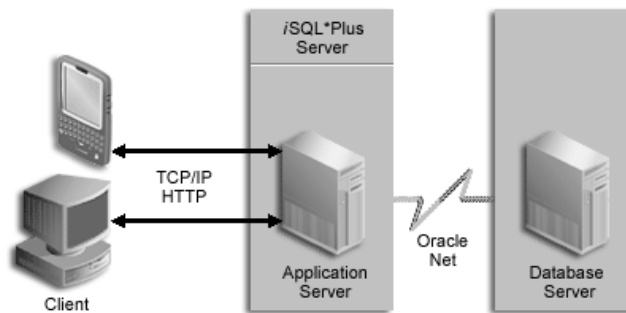
ORACLE

Copyright © 2006, Oracle. All rights reserved.

PL/SQL Programming Environments (continued)

When you log in to *iSQL*Plus*, you see the screen shown in the slide. Note that you have a workspace to enter SQL, PL/SQL, and SQL*Plus statements. Click the Execute button to execute your statements in the workspace. Click the Save Script button when you want to save all the commands in the workspace in a script file. You can save the script as a *.sql file. If you want to execute any script file, click the Load Script button and browse to select the script file. All the statements in the script file are loaded to the workspace and you can click the Execute button to execute the statements. The Clear button is used to clear the workspace.

*iSQL*Plus* Architecture



ORACLE®

Copyright © 2006, Oracle. All rights reserved.

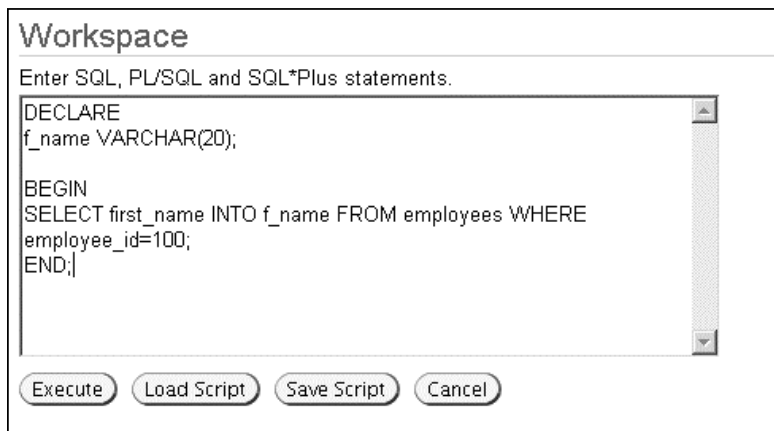
*iSQL*Plus* Architecture

*iSQL*Plus* uses a three-tier model as shown in the slide. The three tiers in the architecture are:

- **Client tier:** The client is a typical HTTP client. Any browser connected to the intranet or Internet can access the *iSQL*Plus* user interface.
- **Middle tier:** The application server forms the middle tier in the *iSQL*Plus* architecture. The application server is installed when the database is installed. The *iSQL*Plus* server must be installed on the same machine as the application server. The middle tier is a Java2 Enterprise Edition (J2EE)–compliant application server. The application server enables communication between *iSQL*Plus* and the database. The three tiers in the architecture need not be on the same machine. However, the HTTP Server and *iSQL*Plus* Server should be on the same machine. *iSQL*Plus* manages a unique identity for each session. The advantage of this is that many concurrent users can use *iSQL*Plus* to access the database.
- **Database tier:** The database tier has the database server. The Oracle Net components enable communication between the *iSQL*Plus* Server and the database.

Create an Anonymous Block

Type the anonymous block in the *iSQL*Plus* workspace:



The screenshot shows the 'Workspace' window in iSQL*Plus. It contains a text area with the following SQL code:

```
DECLARE
f_name VARCHAR(20);

BEGIN
SELECT first_name INTO f_name FROM employees WHERE
employee_id=100;
END;
```

Below the text area are four buttons: 'Execute', 'Load Script', 'Save Script', and 'Cancel'.

Create an Anonymous Block

To create an anonymous block using *iSQL*Plus*, enter the block in the workspace (as shown in the slide). The block has the declarative section and the executable section. You need not pay attention to the syntax of statements in the block; you learn the syntax later in the course. The anonymous block gets the `first_name` of the employee whose `employee_id` is 100 and stores it in a variable called `f_name`.

Execute an Anonymous Block

Click the Execute button to execute the anonymous block:

The screenshot shows the 'Workspace' window in Oracle SQL*Plus. It contains a text area with the following PL/SQL code:

```
DECLARE
f_name VARCHAR(20);

BEGIN
SELECT first_name INTO f_name FROM employees WHERE
employee_id=100;
END;
```

Below the text area are four buttons: 'Execute', 'Load Script', 'Save Script', and 'Cancel'. The 'Execute' button is highlighted.

PL/SQL procedure successfully completed.

ORACLE

Copyright © 2006, Oracle. All rights reserved.

Execute an Anonymous Block

Click the Execute button to execute the anonymous block in the workspace. Note that the message “PL/SQL procedure successfully completed” is displayed after the block is executed.

Test the Output of a PL/SQL Block

- **Enable output in *iSQL*Plus* with the following command:**
`SET SERVEROUTPUT ON`
- **Use a predefined Oracle package and its procedure:**
 - `DBMS_OUTPUT.PUT_LINE`

```
SET SERVEROUTPUT ON
...
DBMS_OUTPUT.PUT_LINE(' The First Name of the
Employee is ' || f_name);
...
```

ORACLE

Copyright © 2006, Oracle. All rights reserved.

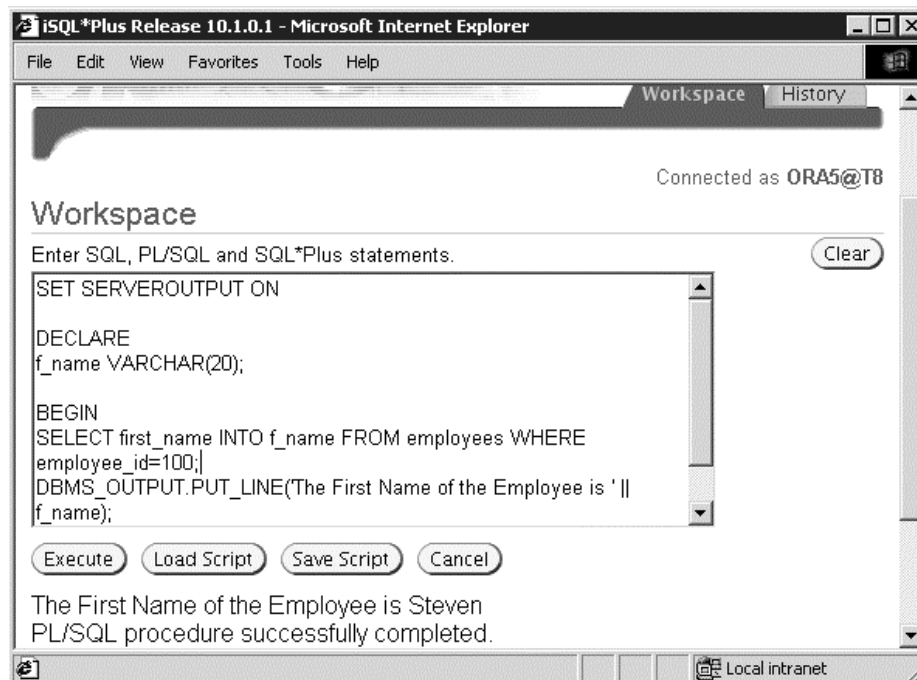
Test the Output of a PL/SQL Block

In the example shown in the previous slide, we have stored a value in the variable `f_name`. However, we have not printed the value. You now learn how to print the value.

PL/SQL does not have built-in input or output functionality. Therefore, we use predefined Oracle packages for input and output. To generate output, you must:

- Enable output in *iSQL*Plus* by using the `SET SERVEROUTPUT ON` command. `SET SERVEROUTPUT ON` is a *SQL*Plus* command that is also supported by *iSQL*Plus*.
- Use the procedure `PUT_LINE` of the package `DBMS_OUTPUT` to display the output. Pass the value that has to be printed as argument to this procedure (as shown in the slide). The procedure then outputs the arguments.

Test the Output of a PL/SQL Block



Copyright © 2006, Oracle. All rights reserved.

Test the Output of a PL/SQL Block (continued)

The slide shows the output of the PL/SQL block after the inclusion of the code for generating output.

Summary

In this lesson, you should have learned how to:

- **Integrate SQL statements with PL/SQL program constructs**
- **Identify the benefits of PL/SQL**
- **Differentiate different PL/SQL block types**
- **Use iSQL*Plus as the programming environment for PL/SQL**
- **Output messages in PL/SQL**

ORACLE

Copyright © 2006, Oracle. All rights reserved.

Summary

PL/SQL is a language that has programming features that serve as an extension to SQL. SQL, which is a nonprocedural language, is made procedural with PL/SQL programming constructs. PL/SQL applications can run on any platform or operating system on which an Oracle server runs. In this lesson, you learned how to build basic PL/SQL blocks.

Practice 1: Overview

This practice covers the following topics:

- **Identifying which PL/SQL blocks execute successfully**
- **Creating and executing a simple PL/SQL block**

ORACLE

Copyright © 2006, Oracle. All rights reserved.

Practice 1: Overview

This practice reinforces the basics of PL/SQL covered in this lesson.

- Exercise 1 is a paper-based exercise in which you identify PL/SQL blocks that execute successfully.
- Exercise 2 involves creating and executing a simple PL/SQL block.

Practice 1

Before you begin this practice, please ensure that you have seen both the viewlets on *iSQL*Plus* usage.

The `labs` folder will be your working directory. You can save your scripts in the `labs` folder. Please take the instructor's help to locate the `labs` folder for this course. The solutions for all practices are in the `soln` folder.

1. Which of the following PL/SQL blocks execute successfully?
 - a.

```
BEGIN
END;
```
 - b.

```
DECLARE
amount INTEGER(10);
END;
```
 - c.

```
DECLARE
BEGIN
END;
```
 - d.

```
DECLARE
amount INTEGER(10);
BEGIN
DBMS_OUTPUT.PUT_LINE(amount);
END;
```
2. Create and execute a simple anonymous block that outputs "Hello World." Execute and save this script as `lab_01_02_soln.sql`.

