

Oracle Database 10g: Program with PL/SQL

Description:

This class is applicable to Oracle8i, Oracle9i and Oracle Database 10g users. This course introduces students to PL/SQL and helps them understand the benefits of this powerful programming language. In the class, students learn to create PL/SQL blocks of application code that can be shared by multiple forms, reports, and data management applications. Students learn to create anonymous PL/SQL blocks, stored procedures, and functions. They learn about declaring variables and trapping exceptions. Students will also learn to develop stored procedures, functions, packages and database triggers. Students will learn to manage PL/SQL program units manage dependencies, manipulate large objects, and use some of the Oracle-supplied packages. Students use iSQL*Plus to develop these program units. Demonstrations and hands-on practice reinforce the fundamental concepts.

Objectives:

- Create simple procedures and functions
- Handle runtime errors
- Use PL/SQL programming constructs and conditionally control code flow (loops, control structures, and explicit cursors)
- Categorize and Use the Oracle supplied PL/SQL packages to generate screen output, file output, web output, and mail output
- Describe the features and syntax of PL/SQL
- Create triggers to solve business challenges
- Manage dependencies between PL/SQL subprograms
- Write PL/SQL code to interface with the database
- Schedule PL/SQL jobs to run independently
- Design PL/SQL anonymous blocks that execute efficiently
- Design PL/SQL packages to group and contain related constructs
- Write dynamic SQL for more coding flexibility

Audiences:

- - PL/SQL Developer
- - Database Designers
- - Forms Developer
- - Technical Consultant

Required Prerequisites:

- Oracle Database 10g: Introduction to SQL

Topics:

Introduction to PL/SQL

- 1 - What is PL/SQL
- 2 - PL/SQL Environment
- 3 - Benefits of PL/SQL
- 4 - Overview of the Types of PL/SQL blocks
- 5 - Create and Execute a Simple Anonymous Block
- 6 - Generate Output from a PL/SQL Block
- 7 - iSQL*Plus as PL/SQL Programming Environment

Declaring PL/SQL Identifiers

- 1 - Identify the Different Types of Identifiers in a PL/SQL subprogram
- 2 - Use the Declarative Section to Define Identifiers
- 3 - List the Uses for Variables
- 4 - Store Data in Variables
- 5 - Declare PL/SQL Variables

Writing Executable Statements

- 1 - Describe Basic Block Syntax Guidelines
- 2 - Use Literals in PL/SQL
- 3 - Customize Identifier Assignments with SQL Functions
- 4 - Use Nested Blocks as Statements
- 5 - Reference an Identifier Value in a Nested Block
- 6 - Qualify an Identifier with a Label
- 7 - Use Operators in PL/SQL
- 8 - Use Proper PL/SQL Block Syntax and Guidelines

Interacting with the Oracle Server

- 1 - Identify the SQL Statements You Can Use in PL/SQL
- 2 - Include SELECT Statements in PL/SQL
- 3 - Retrieve Data in PL/SQL with the SELECT Statement
- 4 - Avoid Errors by Using Naming Conventions When Using Retrieval and DML Statements
- 5 - Manipulate Data in the Server Using PL/SQL
- 6 - The SQL Cursor concept
- 7 - Use SQL Cursor Attributes to Obtain Feedback on DML
- 8 - Save and Discard Transactions

Writing Control Structures

- 1 - Control PL/SQL Flow of Execution
- 2 - Conditional processing Using IF Statements
- 3 - Conditional Processing CASE Statements
- 4 - Handle Nulls to Avoid Common Mistakes
- 5 - Build Boolean Conditions with Logical Operators
- 6 - Use Iterative Control with Looping Statements

Working with Composite Data Types

- 1 - Learn the Composite Data Types of PL/SQL Records and Tables
- 2 - Use PL/SQL Records to Hold Multiple Values of Different Types
- 3 - Inserting and Updating with PL/SQL Records

4 - Use INDEX BY Tables to Hold Multiple Values of the Same Data Type

Using Explicit Cursors

- 1 - Cursor FOR Loops Using Subqueries
- 2 - Increase the Flexibility of Cursors By Using Parameters
- 3 - Use the FOR UPDATE Clause to Lock Rows
- 4 - Use the WHERE CURRENT Clause to Reference the Current Row
- 5 - Use Explicit Cursors to Process Rows
- 6 - Explicit Cursor Attributes
- 7 - Cursors and Records

Handling Exceptions

- 1 - Handling Exceptions with PL/SQL
- 2 - Predefined Exceptions
- 3 - Trapping Nonpredefined Oracle Server Errors
- 4 - Functions that Return Information on Encountered Exceptions
- 5 - Trapping User-Defined Exceptions
- 6 - Propagate Exceptions
- 7 - Use the RAISE_APPLICATION_ERROR Procedure To Report Errors To Applications

Creating Stored Procedures

- 1 - Describe PL/SQL blocks and subprograms
- 2 - Describe the uses of procedures
- 3 - Create procedures
- 4 - Differentiate between formal and actual parameters
- 5 - List the features of different parameter modes
- 6 - Create procedures with parameters and invoke a procedure
- 7 - Handle exceptions in procedures
- 8 - View source code in the data dictionary

Creating Stored Functions

- 1 - Describe stored functions
- 2 - List the CREATE OR REPLACE FUNCTION syntax
- 3 - Identify the steps to create a stored function
- 4 - Create a stored function in iSQL*Plus and execute a stored function
- 5 - Identify the advantages of using stored functions in SQL statements
- 6 - Identify the restrictions of calling functions from SQL statements
- 7 - Describe how procedures and functions differ

Creating Packages

- 1 - List the benefits or using PL/SQL packages
- 2 - Differentiate between a package specification and a package body
- 3 - Create packages
- 4 - Include public and private constructs in a package
- 5 - Call public and private constructs in a package
- 6 - Remove packages

Using More Package Concepts

- 1 - Overload procedure and function definitions
- 2 - Use forward declarations
- 3 - Create a one-time package initialization block
- 4 - Follow the persistent state of constructs in packages
- 5 - Use PL/SQL tables and records in packages
- 6 - Wrap code to hide the source

Utilizing Oracle Supplied Packages in Application Development

- 1 - List the various uses for the Oracle supplied packages
- 2 - Reuse pre-packaged code to complete various tasks from developer to DBA purposes
- 3 - Use the DESCRIBE command to view the package specifications and overloading
- 4 - Describe how DBMS_OUTPUT works
- 5 - Use UTL_FILE to direct output to operating system files
- 6 - Use the HTP package to generate a simple web page
- 7 - Describe the main features of UTL_MAIL
- 8 - Call the DBMS_SCHEDULER package to schedule PL/SQL code to run

Dynamic SQL and Metadata

- 1 - Describe using native dynamic SQL
- 2 - List the execution flow of SQL
- 3 - Write dynamic SQL using the EXECUTE IMMEDIATE syntax
- 4 - Write dynamic SQL with the DBMS_SQL package
- 5 - Generate DDL from metadata using the DBMS_METADATA package

Design Considerations for PL/SQL Code

- 1 - Standardize constants with a constant package
- 2 - Standardize exceptions with an exception package
- 3 - Write PL/SQL code that uses local subprograms
- 4 - Use the NOCOPY compiler hint to pass parameters by reference
- 5 - Use the PARALLEL ENABLE hint for optimization
- 6 - Use the AUTONOMOUS TRANSACTION pragma to run independent transactions within a single transaction
- 7 - Set the AUTHID directive to execute programs with the privileges of the calling user instead of the creating user
- 8 - Use bulk binding for multi-row operations

Managing Dependencies

- 1 - Describe dependent and referenced objects
- 2 - Track procedural dependencies with dictionary views
- 3 - Predict the effect of changing a database object upon stored procedures and functions
- 4 - Manage local and remote procedural dependencies

Manipulating Large Objects

- 1 - Describe a LOB object
- 2 - Create and maintain LOB data types
- 3 - Differentiate between internal and external LOBs
- 4 - Use the DBMS_LOB PL/SQL package to control LOBs
- 5 - Describe the use of temporary LOBs

Creating Triggers

- 1 - Describe different types of triggers
- 2 - Describe database triggers and their use
- 3 - Create database triggers
- 4 - Describe database trigger firing rules
- 5 - Remove database triggers

Applications for Triggers

- 1 - Create database and system event triggers
- 2 - Create triggers on DDL statements
- 3 - Use the CALL statement in triggers to invoke procedures
- 4 - Explain the rules for reading and writing to tables with triggers
- 5 - Describe business application scenarios for implementing with triggers
- 6 - Manage trigger code

Understanding and Influencing the PL/SQL Compiler

- 1 - Describe native compilation and interpreted compilation
- 2 - List the features of native compilation
- 3 - Switch between native and interpreted compilation for compiled PL/SQL code
- 4 - Set the parameters to control aspects of PL/SQL compilation
- 5 - Write a query to retrieve information from the dictionary views on how the PL/SQL code is compiled
- 6 - Explain the compiler warning mechanism
- 7 - List the steps to use the compiler warnings
- 8 - Use DBMS_WARNING to implement compiler warnings