

Course Outline

Oracle Database: Program with PL/SQL

Duration: 5 days (30 hours)

ORACLE[®]
Education Partner

Learning Objectives:

- Create and debug stored procedures and functions
- Use conditional compilation to customize the functionality in a PL/SQL application without removing any source code
- Design PL/SQL packages to group related constructs
- Create overloaded package subprograms for more flexibility
- Design PL/SQL anonymous blocks that execute efficiently
- Use the Oracle supplied PL/SQL packages to generate screen output, file output and mail output
- Write dynamic SQL for more coding flexibility
- Describe the features and syntax of PL/SQL
- Use PL/SQL programming constructs and conditionally control code flow (loops, control structures, and explicit cursors)
- Manage dependencies between PL/SQL subprograms
- Handle runtime errors
- Create triggers to solve business challenges
- Design PL/SQL code for predefined data types, local subprograms, additional pragmas, standardized constants and exceptions

Target Audience:

- PL/SQL Developer
- Forms Developer
- Application Developers
- Database Administrators
- Technical Consultant
- Portal Developer
- System Analysts
- Developer

Prerequisites:

- Oracle Database: Introduction to SQL (combination of Oracle Database: SQL Fundamentals I and Oracle Database: SQL Fundamentals II listed)
- Oracle Database: SQL Fundamentals I
- Previous programming experience

Topics Covered:

- Introduction
 - Course Objectives
 - Course Agenda

- Describe the Human Resources (HR) Schema
- PL/SQL development environments available in this course
- Introduction to SQL Developer
- Introduction to PL/SQL
 - Overview of PL/SQL
 - Identify the benefits of PL/SQL Subprograms
 - Overview of the types of PL/SQL blocks
 - Create a Simple Anonymous Block
 - How to generate output from a PL/SQL Block?
- Declare PL/SQL Identifiers
 - List the different Types of Identifiers in a PL/SQL subprogram
 - Usage of the Declarative Section to Define Identifiers
 - Use variables to store data
 - Identify Scalar Data Types
 - The %TYPE Attribute
 - What are Bind Variables?
 - Sequences in PL/SQL Expressions
- Write Executable Statements
 - Describe Basic PL/SQL Block Syntax Guidelines
 - Learn to Comment the Code
 - Deployment of SQL Functions in PL/SQL
 - How to convert Data Types?
 - Describe Nested Blocks
 - Identify the Operators in PL/SQL
- Interaction with the Oracle Server
 - Invoke SELECT Statements in PL/SQL
 - Retrieve Data in PL/SQL
 - SQL Cursor concept
 - Avoid Errors by using Naming Conventions when using Retrieval and DML Statements
 - Data Manipulation in the Server using PL/SQL
 - Understand the SQL Cursor concept
 - Use SQL Cursor Attributes to Obtain Feedback on DML
 - Save and Discard Transactions
- Control Structures
 - Conditional processing using IF Statements
 - Conditional processing using CASE Statements
 - Describe simple Loop Statement
 - Describe While Loop Statement
 - Describe For Loop Statement
 - Use the Continue Statement

- Composite Data Types
 - Use PL/SQL Records
 - The %ROWTYPE Attribute
 - Insert and Update with PL/SQL Records
 - INDEX BY Tables
 - Examine INDEX BY Table Methods
 - Use INDEX BY Table of Records

- Explicit Cursors
 - What are Explicit Cursors?
 - Declare the Cursor
 - Open the Cursor
 - Fetch data from the Cursor
 - Close the Cursor
 - Cursor FOR loop
 - The %NOTFOUND and %ROWCOUNT Attributes
 - Describe the FOR UPDATE Clause and WHERE CURRENT Clause

- Exception Handling
 - Understand Exceptions
 - Handle Exceptions with PL/SQL
 - Trap Predefined Oracle Server Errors
 - Trap Non-Predefined Oracle Server Errors
 - Trap User-Defined Exceptions
 - Propagate Exceptions
 - RAISE_APPLICATION_ERROR Procedure

- Stored Procedures
 - Create a Modularized and Layered Subprogram Design
 - Modularize Development With PL/SQL Blocks
 - Understand the PL/SQL Execution Environment
 - List the benefits of using PL/SQL Subprograms
 - List the differences between Anonymous Blocks and Subprograms
 - Create, Call, and Remove Stored Procedures
 - Implement Procedures Parameters and Parameters Modes
 - View Procedure Information

- Stored Functions and Debugging Subprograms
 - Create, Call, and Remove a Stored Function
 - Identify the advantages of using Stored Functions
 - Identify the steps to create a stored function
 - Invoke User-Defined Functions in SQL Statements

- Restrictions when calling Functions
- Control side effects when calling Functions
- View Functions Information
- How to debug Functions and Procedures?
- Packages
 - Listing the advantages of Packages
 - Describe Packages
 - What are the components of a Package?
 - Develop a Package
 - How to enable visibility of a Package's Components?
 - Create the Package Specification and Body using the SQL CREATE Statement and SQL Developer
 - Invoke the Package Constructs
 - View the PL/SQL Source Code using the Data Dictionary
- Deploying Packages
 - Overloading Subprograms in PL/SQL
 - Use the STANDARD Package
 - Use Forward Declarations to solve Illegal Procedure Reference
 - Implement Package Functions in SQL and Restrictions
 - Persistent State of Packages
 - Persistent State of a Package Cursor
 - Control side effects of PL/SQL Subprograms
 - Invoke PL/SQL Tables of Records in Packages
- Implement Oracle-Supplied Packages in Application Development
 - What are Oracle-Supplied Packages?
 - Examples of some of the Oracle-Supplied Packages
 - How does the DBMS_OUTPUT Package work?
 - Use the UTL_FILE Package to Interact with Operating System Files
 - Invoke the UTL_MAIL Package
 - Write UTL_MAIL Subprograms
- Dynamic SQL
 - The Execution Flow of SQL
 - What is Dynamic SQL?
 - Declare Cursor Variables
 - Dynamically Executing a PL/SQL Block
 - Configure Native Dynamic SQL to Compile PL/SQL Code
 - How to invoke DBMS_SQL Package?
 - Implement DBMS_SQL with a Parameterized DML Statement
 - Dynamic SQL Functional Completeness
- Design Considerations for PL/SQL Code
 - Standardize Constants and Exceptions

- Understand Local Subprograms
 - Write Autonomous Transactions
 - Implement the NOCOPY Compiler Hint
 - Invoke the PARALLEL_ENABLE Hint
 - The Cross-Session PL/SQL Function Result Cache
 - The DETERMINISTIC Clause with Functions
 - Usage of Bulk Binding to Improve Performance
- Triggers
- Describe Triggers
 - Identify the Trigger Event Types and Body
 - Business Application Scenarios for Implementing Triggers
 - Create DML Triggers using the CREATE TRIGGER Statement and SQL Developer
 - Identify the Trigger Event Types, Body, and Firing (Timing)
 - Differences between Statement Level Triggers and Row Level Triggers
 - Create Instead of and Disabled Triggers
 - How to Manage, Test and Remove Triggers
- Creating Compound, DDL, and Event Database Triggers
- What are Compound Triggers?
 - Identify the Timing-Point Sections of a Table Compound Trigger
 - Understand the Compound Trigger Structure for Tables and Views
 - Implement a Compound Trigger to Resolve the Mutating Table Error
 - Comparison of Database Triggers to Stored Procedures
 - Create Triggers on DDL Statements
 - Create Database-Event and System-Events Triggers
 - System Privileges Required to Manage Triggers
- PL/SQL Compiler
- What is the PL/SQL Compiler?
 - Describe the Initialization Parameters for PL/SQL Compilation
 - List the new PL/SQL Compile Time Warnings
 - Overview of PL/SQL Compile Time Warnings for Subprograms
 - List the benefits of Compiler Warnings
 - List the PL/SQL Compile Time Warning Messages Categories
 - Setting the Warning Messages Levels: Using SQL Developer, PLSQL_WARNINGS Initialization Parameter, and the DBMS_WARNING Package Subprograms
 - View Compiler Warnings: Using SQL Developer, SQL*Plus, or the Data Dictionary Views
- Manage PL/SQL Code
- What Is Conditional Compilation?
 - Implement Selection Directives
 - Invoke Predefined and User-Defined Inquiry Directives
 - The PLSQL_CCFLAGS Parameter and the Inquiry Directive

- Conditional Compilation Error Directives to Raise User-Defined Errors
- The DBMS_DB_VERSION Package
- Write DBMS_PREPROCESSOR Procedures to Print or Retrieve Source Text
- Obfuscation and Wrapping PL/SQL Code
- Manage Dependencies
 - Overview of Schema Object Dependencies
 - Query Direct Object Dependencies using the USER_DEPENDENCIES View
 - Query an Object's Status
 - Invalidation of Dependent Objects
 - Display the Direct and Indirect Dependencies
 - Fine-Grained Dependency Management in Oracle Database 11g
 - Understand Remote Dependencies
 - Recompile a PL/SQL Program Unit