

Oracle® PL/SQL Programming: Hands-On - 5 Days

Course 493 Overview

- You Will Learn How To**
- Develop efficient PL/SQL programs to access Oracle databases
 - Create stored procedures and functions for maximum reuse and minimum code maintenance
 - Design modular applications using packages
 - Manage data retrieval for front-end applications
 - Bulk bind collections to increase the speed of data movement operations
 - Invoke native dynamic SQL to develop high-level abstract code

Course Benefits The Oracle PL/SQL language—a flexible procedural extension to SQL—increases productivity, performance, scalability, portability and security. In this course, you gain the practical knowledge to write PL/SQL programs. You learn to build stored procedures, design and execute modular applications, and increase the efficiency of data movement.

Who Should Attend Programmers and others working with PL/SQL. A working knowledge of SQL and PL/SQL at the level of Course 926, "Oracle Database 11g Comprehensive Introduction", or Course 925, "SQL Programming Language Introduction", is assumed.

Hands-On Training Extensive hands-on exercises provide experience writing modular PL/SQL code. Exercises include:

- Encapsulating data manipulation statements in stored procedures and packages
- Performing complex data manipulation with cursors
- Leveraging EXCEPTIONs to handle runtime errors
- Creating triggers to handle data integrity and derivation
- Utilising weak and strong cursor variables for dynamic SQL
- Denormalising data with user-written functions
- Creating global variables in bodiless packages for session processing

Oracle® PL/SQL Programming: Hands-On - 5 Days

Course 493 Outline

Introduction and Overview

PL/SQL fundamentals

- Declaring variables
- Anchoring variables to database definitions
- Flow control constructs

Oracle 10g and 11g PL/SQL features

- PL/Scope in Oracle 11g
- CASE statement process flow
- Referencing PL/SQL records in DML
- Improving performance with native compilation
- Multiset operators for collections

Data Manipulation Techniques

Maintaining data with DML statements

- Employing the RETURNING INTO clause
- Solving the fetch-across-commit problem

Managing data retrieval with cursors

- Implications of explicit and implicit cursors
- Cursor attributes
- Simplifying cursor processing with cursor FOR LOOPS
- Embedding cursor expressions in SELECT statements

Cursor variables

- Strong vs. weak cursor variables
- Passing cursor variables to other programs
- Defining REF CURSORS in packages

Developing Well-Structured and Error-Free Code

Error handling using EXCEPTIONS

- Predefined and user EXCEPTIONS
- Propagation and scope
- "Retrying" problem transactions with EXCEPTION processing

Debugging PL/SQL blocks

- Simplifying testing and debugging with conditional compilation
- Interpreting compiler messages
- Applying structured testing techniques
- Building and applying a test bed
- Leveraging the debugging facilities in SQL Developer

Achieving Maximum Reusability

Writing stored procedures and functions

- Calling and invoking server-side logic
- Passing input and output parameters
- Implementing an autonomous transaction

- Definer rights vs. invoker rights
- Wrapping the source code

Coding user-written functions

- Calling PL/SQL functions from SQL
- Building table-valued functions

Developing safe triggers

- Employing :OLD and :NEW variables
- Implementing complex business rules
- Avoiding unreliable trigger constructs
- Exploiting schema and database triggers

Exploiting Complex Datatypes

Collection types

- PL/SQL tables, nested tables, VARRAYs
- Stepping through dense and non-consecutive collections

Bulk binding for high performance

- Moving data into and out of PL/SQL blocks
- BULK COLLECT INTO and FORALL
- BULK cursor attributes
- BULK EXCEPTION handling

Invoking Native Dynamic SQL

Finessing the compiler

- The EXECUTE IMMEDIATE statement
- The RETURNING INTO clause

Types of dynamic SQL

- Building SQL statements during runtime
- Autogenerating standard code

Package Tips and Techniques

- Package structure: SPEC and BODY
- Eliminating dependency problems
- Overloading for polymorphic effects
- Evaluating application frameworks
- Bodiless packages for all application definitions
- Declaring and using persistent global variables