ORACLE
Academy

Database Programming with PL/SQL – Course Objectives

Overview

This course introduces students to PL/SQL, Oracle's procedural extension language for SQL and the Oracle relational database. Participants explore the differences between SQL and PL/SQL. They also examine the characteristics of PL/SQL and how it is used to extend and automate SQL to administer the Oracle database. This course culminates with a project that challenges students to program, implement, and demonstrate a database solution for a business or organization.

Available Curriculum Languages:

- English

Duration

- Recommended total course time: 180 hours*
- Professional education credit hours for educators who complete Oracle Academy training: 60

** Course time includes instruction, self-study/homework, practices, projects, and assessment*

Target Audiences

Educators

- College/university faculty who teach computer programming or a related subject
- Secondary school teachers who teach computer programming

Students

- Students who wish to learn the techniques and tools to automate database application tasks
- Students who possess basic mathematical, logical, and analytical problem-solving skills
- Novice programmers, as well as those at advanced levels, to learning the PL/SQL programming language to an advanced level

Prerequisites

Required

- Previous experience with at least one programming language

Suggested

- Previous Experience with a database application
- Oracle Academy Course – Database Design and Database Programming with SQL

Suggested Next Courses

- Java Fundamentals
- Java Programming

Lesson-by-Lesson Topics

Section 1 – Fundamentals

- 1-1 Introduction to PL/SQL
  - Describe PL/SQL
  - Differentiate between SQL and PL/SQL
  - Explain the need for PL/SQL
- 1-2 Benefits of PL/SQL
  - List and explain the benefits of PL/SQL
  - List differences between PL/SQL and other programming languages
  - Give examples of how PL/SQL can be used in other Oracle products
- 1-3 Creating PL/SQL Blocks
  - Describe the structure of a PL/SQL block
  - Identify the different types of PL/SQL blocks
  - Identify PL/SQL programming environments
  - Create and execute an anonymous PL/SQL block
  - Output messages in PL/SQL

Section 2 – Defining Variables and Datatypes

- 2-1 Using Variables in PL/SQL
  - List the uses of variables in PL/SQL
  - Identify the syntax for variables in PL/SQL
  - Declare and initialize variables in PL/SQL
  - Assign new values to variables in PL/SQL
- 2-2 Recognizing PL/SQL Lexical Units
  - List and define the different types of lexical units available in PL/SQL
  - Describe identifiers and identify valid and invalid identifiers in PL/SQL
  - Describe and identify reserved words, delimiters, literals, and comments in PL/SQL
- 2-3 Recognizing Data Types
  - Define data type and explain why it is needed
  - List and describe categories of data types
  - Give examples of scalar and composite data types
- 2-4 Using Scalar Data Types
  - Declare and use scalar data types in PL/SQL
  - Define guidelines for declaring and initializing PL/SQL variables
  - Identify the benefits of anchoring data types with the %TYPE attribute
- 2-5 Writing PL/SQL Executable Statements
  - Construct accurate variable assignment statements in PL/SQL
  - Construct accurate statements using built-in SQL functions in PL/SQL
  - Differentiate between implicit and explicit conversions of data types
  - Describe when implicit conversions of data types take place
  - List drawbacks of implicit data type conversions
  - Construct accurate statements using functions to explicitly convert data types
  - Construct statements using operators in PL/SQL
- 2-6 Nested Blocks and Variable Scope
  - Understand the scope and visibility of variables
  - Write nested blocks and qualify variables with labels
  - Describe the rules for variable scope when a variable is nested in a block
  - Recognize a variable scope issue when a variable is used in nested blocks
- 2-7 Good Programming Practices
  - List examples of good programming practices
  - Accurately insert comments into PL/SQL code
  - Create PL/SQL code that follows formatting guidelines to produce readable code

Section 3 – Using SQL in PL/SQL

- 3-1 Review of SQL DML
    - Insert data into a database table
    - Update data in a database table
    - Delete data from a database table
    - Merge data into a database table
- 3-2 Retrieving Data in PL/SQL
    - Recognize the SQL statements that can be directly included in a PL/SQL executable block
    - Construct and execute an INTO clause to hold the values returned by a single-row SQL SELECT statement
    - Construct statements to retrieve data that follow good practice guidelines
    - Construct statements that apply good practice guidelines for naming variables
- 3-3 Manipulating Data in PL/SQL
    - Construct and execute PL/SQL statements that manipulate data with DML statements
    - Describe when to use implicit or explicit cursors in PL/SQL
    - Create PL/SQL code to use SQL implicit cursor attributes to evaluate cursor activity
- 3-4 Using Transaction Control Statements
    - Define a transaction and provide an example
    - Construct and execute a transaction control statement in PL/SQL
    - Since Oracle Application Express automatically commits changes, the following information will be presented as if you were issuing the commands in an installed/local environment with the ability to use COMMIT and ROLLBACK

Section 4 – Program Structures to Control Execution Flow

- 4-1 Conditional Control: IF Statements
    - Describe a use for conditional control structures
    - List the types of conditional control structures
    - Construct and use an IF statement
    - Construct and use an IF-THEN-ELSIF-ELSE statement
    - Create PL/SQL to handle the null condition in IF statements
- 4-2 Conditional Control: Case Statements
    - Construct and use CASE statements in PL/SQL
    - Construct and use CASE expressions in PL/SQL
    - Include the correct syntax to handle null conditions in PL/SQL CASE statements
    - Include the correct syntax to handle Boolean conditions in PL/SQL IF and CASE statements
- 4-3 Iterative Control: Basic Loops
    - Describe the need for LOOP statements in PL/SQL
    - Recognize different types of LOOP Statements
    - Create PL/SQL containing a basic loop and an EXIT statement
    - Create PL/SQL containing a basic loop and an EXIT statement with conditional termination
- 4-4 Iterative Control: While and For Loops
    - Construct and use the WHILE looping construct in PL/SQL
    - Construct and use the FOR looping construct in PL/SQL
    - Describe when a WHILE loop is used in PL/SQL
    - Describe when a FOR loop is used in PL/SQL
- 4-5 Iterative Control: Nested Loops
    - Construct and execute PL/SQL using nested loops
    - Label loops and use the labels in EXIT statements
    - Evaluate a nested loop construct and identify the exit point

Section 5 – Using Cursors and Parameters

- 5-1 Introduction to Explicit Cursors
    - Distinguish between an implicit and an explicit cursor
    - Describe why and when to use an explicit cursor in PL/SQL code
    - List two or more guidelines for declaring and controlling explicit cursors
    - Create PL/SQL code that successfully opens a cursor and fetches a piece of data into a variable
    - Use a simple loop to fetch multiple rows from a cursor
    - Create PL/SQL code that successfully closes a cursor after fetching data into a variable
- 5-2 Using Explicit Cursor Attributes
    - Use the %ROWTYPE attribute to define a record based on a cursor
    - Create PL/SQL code to process the row of an active set using record types in cursors
    - Retrieve information about the state of an explicit cursor using cursor attributes
- 5-3 Cursor FOR Loops
    - List and explain the benefits of using cursor FOR loops
    - Create PL/SQL code to declare a cursor and manipulate it in a FOR loop
    - Create PL/SQL code containing a cursor FOR loop using a subquery
- 5-4 Cursors with Parameters
    - List the benefits of using parameters with cursors
    - Create PL/SQL code to declare and manipulate a cursor with a parameter
- 5-5 Using Cursors For Update
    - Create PL/SQL code to lock rows before an update using the appropriate clause
    - Explain the effect of using NOWAIT in a update cursor declaration
    - Create PL/SQL code to use the current row of the cursor in an UPDATE or DELETE statement
- 5-6 Using Multiple Cursors
    - Explain the need for using multiple cursors to produce multilevel reports
    - Create PL/SQL code to declare and manipulate multiple cursors within nested loops
    - Create PL/SQL code to declare and manipulate multiple cursors using parameters

Section 6 – Using Composite Datatypes

- 6-1 User-Defined Records
    - Create and manipulate user-defined PL/SQL records
    - Define a record structure using the %ROWTYPE attribute
- 6-2 Indexing Tables of Records
    - Create an INDEX BY table
    - Create an INDEX BY table of records
    - Describe the difference between records, tables, and tables of records

Section 7 – Exception Handling

- 7-1 Handling Exceptions
    - Describe several advantages of including exception handling code in PL/SQL
    - Describe the purpose of an EXCEPTION in a PL/SQL block
    - Create PL/SQL code to include an EXCEPTION section
    - List several guidelines for exception handling
- 7-2 Trapping Oracle Server Exceptions
    - Describe and provide an example of an error defined by the Oracle server
    - Describe and provide an example of an error defined by the PL/SQL programmer
    - Differentiate between errors that are handled implicitly and explicitly by the Oracle Server
    - Write PL/SQL code to trap a predefined Oracle Server error
    - Write PL/SQL code to trap a non-predefined Oracle Server error
    - Write PL/SQL code to identify an exception by error code and by error message

- 7-3 Trapping User-Defined Exceptions
    - Write PL/SQL code to name a user-defined exception
    - Write PL/SQL code to raise an exception
    - Write PL/SQL code to handle a raised exception
    - Write PL/SQL code to use RAISE_APPLICATION_ERROR
- 7-4 Recognizing the Scope of Exceptions
    - Describe the scope of an exception
    - Recognize an exception-scope issue when an exception is within nested blocks
    - Describe the effect of exception propagation in nested blocks

Section 8 – Using and Managing Procedures

- 8-1 Creating Procedures
    - Differentiate between anonymous blocks and subprograms
    - Identify benefits of subprograms
    - Define a stored procedure
    - Create a procedure
    - Describe how a stored procedure is invoked
    - List the development steps for creating a procedure
    - Create a nested subprogram in the declarative section of a procedure
- 8-2 Using Parameters in Procedures
    - Describe how parameters contribute to a procedure
    - Define a parameter
    - Create a procedure using a parameter
    - Invoke a procedure that has parameters
    - Differentiate between formal and actual parameters
- 8-3 Passing Parameters
    - List the types of parameter modes
    - Create a procedure that passes parameters
    - Identify three methods for passing parameters
    - Describe the DEFAULT option for parameters

Section 9 – Using and Managing Functions

- 9-1 Creating Functions
    - Define a stored function
    - Create a PL/SQL block containing a function
    - List ways in which a function can be invoked
    - Create a PL/SQL block that invokes a function that has parameters
    - List the development steps for creating a function
    - Describe the differences between procedures and functions
- 9-2 Using Functions in SQL Statements
    - List the advantages of user-defined functions in SQL statements
    - List where user-defined functions can be called from within an SQL statement
    - Describe the restrictions on calling functions from SQL statements
- 9-3 Review of the Data Dictionary
    - Describe the purposes of the Data Dictionary
    - Differentiate between the three types of Data Dictionary view
    - Write SQL SELECT statements to retrieve information from the Data Dictionary
    - Explain the use of DICTIONARY as a Data Dictionary search engine
- 9-4 Managing Procedures and Functions
    - Describe how exceptions are propagated
    - Remove a function and a procedure
    - Use Data Dictionary views to identify and manage stored programs

- 9-5 Review of Object Privileges
    - List and explain several object privileges
    - Explain the function of the EXECUTE object privilege
    - Write SQL statements to grant and revoke object privileges
- 9-6 Using Invoker's Rights and Autonomous Transactions
    - Contrast invoker's rights with definer's rights
    - Create a procedure that uses invoker's rights
    - Create a procedure that executes an Autonomous Transaction

Section 10 – Using and Managing Packages

- 10-1 Creating Packages
    - Describe the reasons for using a package
    - Describe the two components of a package: specification and body
    - Create packages containing related variables, cursors, constants, exceptions, procedures, and functions
    - Create a PL/SQL block that invokes a package construct
- 10-2 Managing Package Concepts
    - Explain the difference between public and private package constructs
    - Designate a package construct as either public or private
    - Specify the appropriate syntax to drop packages
    - Identify views in the Data Dictionary that manage packages
    - Identify guidelines for using packages
- 10-3 Advanced Package Concepts
    - Write packages that use the overloading feature
    - Write packages that use forward declarations
    - Explain the purpose of a package initialization block
    - Create and use a bodiless package
    - Invoke packaged functions from SQL
    - Identify restrictions on using packaged functions in SQL statements
    - Create a package that uses PL/SQL tables and records

Section 11 – Getting the Best out of Packages

- 11-1 Persistent State of Package Variables
    - Identify persistent states of package variables
    - Control the persistent state of a package cursor
- 11-2 Using Oracle-Supplied Packages
    - Describe two common uses for the DBMS_OUTPUT server-supplied package
    - Recognize the correct syntax to specify messages for the DBMS_OUTPUT package
    - Describe the purpose for the UTL_FILE server-supplied package.
    - Recall the exceptions used in conjunction with the UTL_FILE server-supplied package.
    - Describe the main features of the UTL_MAIL server-supplied package

Section 12 – Improving PL/SQL Performance

- 12-1 Using Dynamic SQL
    - Recall the stages through which all SQL statements pass
    - Describe the reasons for using dynamic SQL to create an SQL statement
    - List four PL/SQL statements supporting Native Dynamic SQL
    - Describe the benefits of EXECUTE IMMEDIATE over DBMS_SQL for Dynamic SQL
- 12-2 Improving PL/SQL Performance
    - Identify the benefits of the NOCOPY hint and the DETERMINISTIC clause
    - Create subprograms that use the NOCOPY hint and the DETERMINISTIC clause
    - Use Bulk Binding FORALL in a DML statement
    - Use BULK COLLECT in a SELECT or FETCH statement
    - Use the Bulk Binding RETURNING clause

Section 13 – Using and Managing Triggers

- 13-1 Introduction to Triggers
  - Describe database triggers and their uses
  - Define a database trigger
  - Recognize the difference between a database trigger and an application trigger
  - List two or more guidelines for using triggers
  - Compare and contrast database triggers and stored procedures
- 13-2 Creating DML Triggers: Part I
  - Create a DML trigger
  - List the DML trigger components
  - Create a statement level trigger
  - Describe the trigger firing sequence options
- 13-3 Creating DML Triggers: Part II
  - Create a DML trigger that uses conditional predicates
  - Create a row-level trigger
  - Create a row-level trigger that uses OLD and NEW qualifiers
  - Create an INSTEAD OF trigger
  - Create a Compound Trigger
- 13-4 Creating DDL and Database Event Triggers
  - Describe events that cause DDL and database event triggers to fire
  - Create a trigger for a DDL statement
  - Create a trigger for a database event
  - Describe the functionality of the CALL statement
  - Describe the cause of a mutating table
- 13-5 Managing Triggers
  - View trigger information in the Data Dictionary
  - Disable and enable a database trigger
  - Remove a trigger from the database

Section 14 – Recognizing and Managing Dependencies

- 14-1 Introduction to Dependencies
  - Describe the implications of procedural dependencies
  - Contrast dependent objects and referenced objects
  - View dependency information in the data dictionary
  - Use the UTLDTREE script to create the objects required to display dependencies
  - Use the IDEPTREE and DEPTREE views to display dependencies
  - Describe when automatic recompilation occurs
  - List how to minimize dependency failures
- 14-2 Understanding Remote Dependencies
  - Describe remote dependencies
  - List how remote dependencies are controlled
  - Describe when a remote dependency is unsuccessfully recompiled
  - Describe when a remote dependency is successfully recompiled

Section 15 – Using the PL/SQL Compiler

- 15-1 Using PL/SQL Initialization Parameters
  - Describe how PLSQL_CODE_TYPE can improve execution speed
  - Describe how PLSQL_OPTIMIZE_LEVEL can improve execution speed
  - Use USER_PLSQL_OBJECT_SETTINGS to see how a PL/SQL program was compiled
- 15-2 Displaying Compiler Warning Messages
  - Explain the similarities and differences between a warning and an error
  - Compare and contrast the warning levels which can be set by the PLSQL_WARNINGS parameter
  - Set warning levels by calling the DBMS_WARNING server-supplied package from within a PL/SQL program

- 15-3 Using Conditional Compilation
    - Describe the benefits of conditional compilation
    - Create and conditionally compile a PL/SQL program containing selection, inquiry and error directives
    - Create and conditionally compile a PL/SQL program which calls the DBMS_DB_VERSION server-supplied package
- 15-4 Hiding your Source Code
    - Describe the benefits of obfuscated PL/SQL source code
    - Use the DBMS_DDL.CREATE_WRAPPED server-supplied procedure
    - Describe how to use the Wrapper utility to obfuscate PL/SQL source code