Oracle Database 12c: PL/SQL III - Advanced Programming and Tuning

(5 Days)

About this Course

This Oracle Database 12c: PL/SQL III - Advanced Programming and Tuning training class presents the three fundamental pillars of effective implementation of PL/SQL applications. First, we will explore the advanced features of the language that allow powerful and adaptable database applications to be built. Next, we will discuss performance tuning techniques that allow these applications to run efficiently. Finally, we will consider critical security measures that should be implemented to counter hacker attacks and other security threats.

The target audience for this textbook is senior application developers and developers who will be building, debugging, and tuning PL/SQL program units.

At Course Completion

After completing this course, students will be able to:

- Learn external Java classes using the JDBC interface and external C programs contained within DLL.
- Learn to work with libraries.
- Learn to use dynamic SQL to extend the functionality and flexibility of database programs.
- Learn to work with the DBMS_SQL() system-supplied package for maximum flexibility.
- Learn to identify SQL injection attack vulnerabilities within an application.
- Learn countermeasures to address security risks and protect against hacking.
- Learn to incorporate collections and other advanced types into application logic to increase efficiency.
- Learn to work with LOBs, including piece-wise data manipulation and dynamic modification of SecureFiles storage options.
- Learn to expand database application functionality with advanced system-supplied database utility.
- Learn to create packages, integrating one's applications with external mail systems, database internals, and other facilities.
- Learn to tune with the DBMS_PROFILER() system-supplied package and debug with the.dBMS_TRACE() system-supplied package.
- Write efficient PL/SQL code and avoid common coding mistakes.
- Learn to enable native compilation and execute all database-resident program units.
- Learn to control and manage PL/SQL compilation for high-efficiency execution.
- Learn to analyze PL/SQL code structure by means of the PL/Scope facility.
- Learn to analyze PL/SQL application performance and tune bottlenecks using the PL/SQL Hierarchical.
- Learn to implement fine-grained security mechanisms as part of an advanced security model using application contexts and the Oracle virtual private database (VPD).
- Learn partitioning and DML parallelization using the system-supplied package dBMS_PARALLEL_EXECUTE().
- Learn to use the wrap utility to hide the source code of database-resident programs.

Prerequisites

Experience in the following *is required* for this Oracle class:

- Oracle SQL and PL/SQL

COURSE OUTLINE

Advanced Programming: Why Needed and PL/SQL Execution: Internals

Why Advanced Programming?
SQL and PL/SQL Execution Internals
SQL and PL/SQL PGA Internals

Advanced Programming: Dynamic SQL

Advantages Of Dynamic SQL
Native Dynamic SQL
Dynamic SQL Using DBMS_SQL()

Advanced Programming: Using Collections

Bulk Bind Using Collections
About SQL%BULK_ROWCOUNT()
About SQL%BULK_EXCEPTIONS()
Collection Methods
More About RETURNING Clause
Advanced Collection Features
IN INDICES OF Clause
IN VALUES OF Clause

Advanced Programming: Java and C Interface Methods

Advanced Program Interfaces
Calling Java Classes
Calling C Programs
System-Supplied Packages: DBMS_METADATA() – Part I
Why Retrieve Object Definitions?
Retrieving Default Metadata
Retrieving Customized Metadata
Using SET_COUNT()
Using ADD_TRANSFORM()
Using FETCH DDL()
Calling FETCH_DDL()

System-Supplied Packages: DBMS_METADATA() – Part II

SET_TRANSFORM_PARAM()
GET_QUERY()

System-Supplied Packages: DBMS_METADATA() – Part III

FETCH CLOB()
SET_FILTER() Dependent Objects
SET_PARSE_ITEM()
Primary and Dependent Object DDL

System-Supplied Packages: DBMS_REDEFINITION()

About Table Redefinition
Using DBMS_REDEFINITION()
DBA_REDEFINITION_ERRORS
CAN_REDEF_TABLE()
START_REDEF_TABLE()
FINISH_REDEF_TABLE()
ABORT_REDEF_TABLE()
COPY_TABLE_DEPENDENTS()
SYNC_INTERIM_TABLE()

System-Supplied Packages: DBMS_LOB()

Working With External BFILES
Working With Internal LOBS
SUBSTR()
INSTR()
Dynamic SECUREFILE Options

High-Performance: Advanced System-Supplied Packages

COMPRESSION and UTL_COMPRESS()
LZ_COMPRESS()
LZ_UNCOMPRESS()
DBMS_DESCRIBE()
UTL_MAIL()
DBMS_UTILITY()
COMPILE_SCHEMA()
DB_VERSION()
GET_PARAMETER_VALUE()
WAIT_ON_PENDING_DML()
GET_TIME()
GET_ENDIANNESS()
DBMS_FILE_TRANSFER()

High Performance: Programming and Coding Techniques

- Autonomous Transactions
- Using NOCOPY FOR Parameters
- Choosing The Optimum Data Type
- About NOT NULL
- Useful PL/SQL Coding Techniques
- Handling String Literals
- User-Defined SQL Functions

High Performance: Influencing Oracle Pl/Sql Compilation

- PL/SQL Compiler Optimization
- PLSQL_OPTIMIZE_LEVEL
- Controlling Compilation Messages
- PL/SQL Native Execution
- Wrapping Source Code

High Performance: Dynamic Partitioning and Parallelization

- Dynamic Partitioning (Chunks)
- Creating and Processing Chunks
- CREATE_TASK()
- CREATE_CHUNKS_BY_ROWID()
- CREATE_CHUNKS_BY_NUMBER_COL()
- EXECUTE_RUN_TASK()
- TASK_STATUS()
- DROP_TASK()
- Monitoring Chunk Processing

High Performance: Using PL/SCOPE For Code Analysis

- Configuring PL/SCOPE
- PLSCOPE_SETTINGS
- Using PL/SCOPE Data

High Performance: Tuning With The Hierarchical Profiler

- What Is The Hierarchical Profiler?
- Configuring The Profiler
- Managing Profiler Runs
- Analyzing Profiler Data
- Interpreting The Results
- DBMSHP_RUNS
- DBMSHP_FUNCTION_INFO
- DBMSHP_PARENT_CHILD_INFO

High Performance: Debugging With DBMS_TRACE()

- Using The Trace Facility
- DBMS_TRACE() To Manage Runs
- Examining The Trace Data
- EVENT_KIND Values

Application Security: SQL Injection Attacks

- Understanding The Threat
- Applying Countermeasures

Application Security: Virtual Private Databases

- Understanding VPDS
- Preparing For A VPD
- Configuring A VPD
- Managing Application Contexts
- Managing Policies and Security Rules