1 Declaring PL/SQL Variables

✅Recognize valid and invalid identifiers

Indentifiers: 2.2.2

Reserved words and keywords: D

References to identifiers: 2.4

✅List the uses of variables, declare and initialize variables, use bind variables

Declarations: 2.3 - 2.3.4

Scalar variable declaration syntax

Constant declaration syntax

Name Resolution: B

Assigning values to variables: 2.6

VARIABLE command for bind variables

Using bind variables: sqpug 5.12

✅List and describe various data types using the %TYPE and %ROWTYPE attributes

PL/SQL datatypes: 3

Declaring items using %TYPE: 2.3.5

%TYPE

Declaring items using %ROWTYPE: 5.12.4

%ROWTYPE

2 Writing Executable Statements

✅Identify lexical units in a PL/SQL block

Lexical units: 2.2

✅Use built-in SQL functions in PL/SQL and sequences in PL/SQL expressions

Expressions, SQL Functions: 2.7

Sequence currval and nextval in PL/SQL: 7.1.2.1

✅Describe when implicit conversions take place and when explicit conversions have to be dealt with

Datatype conversion 10g

Datatype conversion oreilly

✅Write nested blocks and qualify variables with labels

Block

Scope and visibility: 2.5

✅Write readable code with appropriate indentation

Coding style

Indentation standards

3 Writing SQL in PL/SQL

✅Create PL/SQL executable blocks using DML and transaction control statements

DML, static SQL: 6 - 6.1.1

INSERT

UPDATE

DELETE

Transaction processing and control: 6.6 - 6.6.6.1

✅Make use of the INTO clause to hold the values returned by a SQL statement

Processing query result with SELECT INTO: 6.3.1

SELECT INTO syntax

4 Writing Control Structures

✅Identify the uses and types of control structures (IF, CASE statements and expressions)

Types of control statemnets, IF, CASE: 4 - 4.1.5

✅Construct and identify loop statements

Basic loop: 4.2 - 4.2.1

For loop: 4.2.6

While loop: 4.2.7

✅Use EXIT and CONTINUE statements inside loops

EXIT: 4.2.2 - 4.2.3

CONTINUE: 4.2.4 - 4.2.5

GOTO: 4.3 - 4.3.1

NULL: 4.3.2

5 Working with Composite Data Types

Composite data types: 5

✅Create user-defined PL/SQL records

Record variables: 5.12 - 5.12.3

Working with records: 5.13 - 5.17

✅Create a record with the %ROWTYPE attribute

Declaring records using %ROWTYPE: 5.12.4

✅Create an INDEX BY table and INDEX BY table of records

Collection types: 5.1

Associative arrays: 5.2

✅Describe the differences among records, collections, and collections of records

Varray: 5.3

Nested table: 5.4

✅Initialize collections and records

Collection constructors: 5.5

Qualified expressions: 5.6

Assign values to collection variables: 5.7

Collection comparison: 5.9

Collection methods: 5.10

Collection types defined in package spec: 5.11

6 Using Explicit Cursors

✅Distinguish between implicit and explicit cursors and use SQL cursor attributes

Cursors overview and implicit: 6.2 - 6.2.1.4

✅Declare and control explicit cursors, use simple loops and cursor FOR loops to fetch data

Explicit cursors: 6.2.2 - 6.2.2.5, 6.2.2.7

Control cursors and cursor FOR LOOP: 6.3.2 - 6.3.4

✅Declare and use cursors with parameters

Explicit cursors accept parameters: 6.2.2.6

Cursor variables: 6.4

Cursor expression: 6.5

✅Lock rows with the FOR UPDATE clause and reference the current row with the WHERE CURRENT OF clause

cursors FOR UPDATE, WHERE CURRENT OF: 6.6.6.2 - 6.6.6.3

7 Handling Exceptions

PL/SQL has compile-time warnings and runtime errors. The latter are called exceptions.

✅Define PL/SQL exceptions

Exception handling overview: 11.2

✅Recognize unhandled exceptions

Unhandled exceptions: 11.9

✅Handle different types of exceptions (internally defined exceptions, predefined exceptions and user-defined exceptions)

Internally defined: 11.3

Predefined: 11.4

User-defined: 11.5

Redeclared predefined: 11.6

Raising exceptions: 11.7

Error codes and messages: 11.10

After exception handling: 11.11 - 11.12

✅Propagate exceptions

Exception propagation: 11.8

8 Using PL/SQL Subprograms

✅Differentiate between anonymous blocks and subprograms

Subprogram overview: 8

Types of subprogram: 8.2

Subprogram properties: 8.4

Subprogram parts: 8.5

Side effects: 8.11

✅Create a simple procedure and invoke it from an anonymous block

Subprogram invocation: 8.3

Subprogram invocation resolution: 8.8

✅Identify benefits of subprograms

Reasons to use subprograms: 8.1

Overloading: 8.9

Recursion: 8.10

External subprograms: 8.15

9 Creating Procedures and Using Parameters

✅Create a procedure with parameters

Subprogram parameters: 8.7 - 8.7.5

✅Use named notation

Positional, named and mixed notation: 8.7.6

✅Work with procedures (create, invoke and remove procedures)

Forward declaration: 8.6

Procedure declaration and definition: 13.50

CREATE PROCEDURE

ALTER PROCEDURE

DROP PROCEDURE

✅Handle exceptions in procedures and display a procedure's information

DESCRIBE

*_OBJECTS views

*_PROCEDURES views

*_ARGUMENTS views

*_SOURCE views

*_ERRORS views

DBA_OBJECT_SIZE view

10 Creating Functions

✅Differentiate between a procedure and a function

Function difference: 8.5.1 - 8.5.2

✅Describe the uses of functions

Function result cache: 8.12

Functions that SQL can invoke: 8.13

✅Work with functions (create, invoke and remove functions)

Function declaration and definition: 13.36

CREATE FUNCTION

ALTER FUNCTION

DROP FUNCTION

11 Creating Packages

✅Identify the benefits and the components of packages

What is package?: 10.1

Reasons to use packages: 10.2

✅Work with packages (create package specification and body, invoke package subprograms, remove a package and display package information)

Package specification: 10.3

Package body: 10.4

Instantiation and initialization: 10.5

Package state: 10.6

Serially reusable: 10.7

Writing guidelines: 10.8

CREATE PACKAGE

CREATE PACKAGE BODY

ALTER PACKAGE

DROP PACKAGE

✅Overload package subprograms and use forward declarations

Forward declaration: 8.6

Overloaded subprograms: 8.9

12 Working with Packages

✅Use package types and variables

Package example: 10.9

✅Use packaged constants and functions in SQL

STANDARD package: 10.10

✅Use ACCESSIBLE BY to restrict access to package subprograms

ACCESSIBLE BY: 10.1

13 Using Dynamic SQL

✅Describe the execution flow of SQL statements

SQL Processing 19c

DBMS_SQL execution flow

SQL Processing Oracle7: READ ONLY

SQL execution Burleson: READ ONLY

✅Use Native Dynamic SQL (NDS)

Dynamic sql overview: 7 - 7.1

NDS: 7.2

DBMS_SQL package: 7.3

DBMS_SQL package reference: READ ONLY

SQL injection: 7.4 READ ONLY

✅Bind PL/SQL types in SQL statements

see NDS in section above

14 Design Considerations

✅Create standard constants and exceptions

✅Write and call local subprograms

✅Control the run-time privileges of a subprogram

IR, DR, AUTHID: 8.14

✅Perform autonomous transactions

Autonomous transactions: 6.7

Pragma AUTONOMOUS_TRANSACTION: 13.4

✅Use NOCOPY hint, PARALLEL ENABLE hint and DETERMINISTIC clause

Subprogram parameter aliasing: 8.7.4.1

Formal parameter NOCOPY: 13.35

PARALLEL_ENABLE: 13.47

DETERMINISTIC: 13.23

✅Use bulk binding and the RETURNING clause with DML

Bulk binding, bulk sql: 12.4

RETURNING INTO: 13.55

15 Creating Compound, DDL, and Event Database Triggers

✅Describe different types of triggers and their uses

Trigger overview: 9.1

Reasons to use triggers: 9.2

DML triggers: 9.3

Correlation names and pseudorecords: 9.4

Subprograms invoked by triggers: 9.6

Trigger compilation, invalidation, recompilation: 9.7

Trigger exception handling: 9.8

Trigger design: 9.9

Trigger restrictions: 9.10

Trigger firing order: 9.11

Trigger enabling, disabling: 9.12

Trigger changing and debugging: 9.13

Oracle db data transfer and triggers: 9.14

Views for trigger info: 9.16

✅Create triggers on DDL statements

System triggers: 9.5

✅Create triggers on system events

CREATE TRIGGER statement

ALTER TRIGGER statement

DROP TRIGGER statement

16 Using the PL/SQL Compiler

✅Describe the PL/SQL compiler and features

PL/SQL optimizer: 12.1

✅Use the PL/SQL compiler initialization parameters

PL/SQL compilation parameters: 1.3.2

PLSCOPE_SETTINGS

Using PL/Scope: READ ONLY

PLSQL_CCFLAGS

PLSQL_CODE_TYPE

PLSQL_OPTIMIZE_LEVEL

Compiling units for native execution: 12.10

✅Use the PL/SQL compile time warnings

Compile-time warnings: 11.1

DBMS_WARNING

PLSQL_WARNINGS

17 Managing PL/SQL Code

Source text wrapping: A: READ ONLY

✅Describe and use conditional compilation

Conditional compilation: 2.9

✅Code-based access control: granting roles to program units

Security for DR and IR: READ ONLY

Using code based access control

CBAC granting roles: oracle-base: READ ONLY

PL/SQL security blog: Feuerstein: READ ONLY

✅Whitelist code access with the ACCESSIBLE BY clause

ACCESSIBLE BY: 13.1

☑ Mark code as deprecated

DEPRECATE pragma: 13.22

18 Managing Dependencies

Understanding Schema Object Dependency - Oracle 19c Database Development Guide

*_DEPENDENCIES: READ ONLY

DEPTREE: READ ONLY

IDEPTREE: READ ONLY

*_OBJECTS: READ ONLY