# Chapter 1 Review – Introduction to PL/SQ>

|  |  |  |
| --- | --- | --- |
| Ch | Pg | Question |
| 1 | 2 | A **programming** **language** converts user’s actions into instructions a computer can understand. |
| 1 | 2 | Programmers use a **procedural** **language** to code a logical sequence of steps for making decisions and instructing a computer to perform tasks. For that you need a procedural language like Oracle PL/SQL. |
| 1 | 2 | What is PL/SQL? What role does it play in application programming? Why does a programmer need to learn it?  An **application** **program** is software application used to support a business. |
| 1 | 3 | A procedural language makes it possible for developers to perform **decision**-**making** **logic**, such as IF-THEN conditions. |
| 1 | 4 | PL/SQL Advantages:   * ***Tight integration with SQL.*** * ***Improved******performance***   + It allows sending blocks of statements to Oracle in a single statement.   + PL/Code modules are stored in executable form, which makes procedure calls efficient.   + Executable code is automatically cached in memory and shared among users.   + A PL/SQL engine is embedded in Oracle developer tools so that PL/SQL code can be processed on the client machine which reduces network traffic. * ***Increased******productivity*** because PL/SQL code can used in many Oracle tools, and the coding is the same in all. You can use VB or Java to develop an application but still harness the power of PL/SQL with Oracle. * ***Portability***- PL/SQL can run on any platform that Oracle can run. * ***Tighter******security***– Database security with stored **program units**, which give users access without having to be granted specific privileges. |
| 1 | 4 | PL/SQL program modules stored in the database are referred to as **stored** **procedures**. |
| 1 | 4 | An application model has three main components:   * ***User*** ***interface***- the windows displayed to users to enter information or take actions. * ***Program*** ***logic*** – The programming code that provides logic of what the application does. * ***Database***- the database management system providing the physical storage structure for data and mechanisms to retrieve, add, change, and remove data. The Oracle server provides this component. |
| 1 | 5 | In a ***client***/***server*** **model** an executable program or application is loaded on the user’s computer. Some processing can take place on the client side. Other processing might require transmitting requests, such as an SQL statement to query requested data, to the database server. |
| 1 | 6 | A **named** **program unit** is simply a block of PL/SQL code that has been named so that it can be saved (stored) and reused. |
| 1 | 6 | Stored Program Unit Type   |  |  | | --- | --- | | Stored Program Unit Type | Description | | Procedure | Performs a task. Can receive and return multiple values | | Function | Performs a task and typically returns only one value. With certain parameters, it can be used in SQL statements. | | Database trigger | Performs a task automatically when a data manipulation language (DML) action occurs on the associated table or system event. | | Package | Groups related procedures and functions, which makes additional programming features available. | |
| 1 | 6 | The **three-tier model** has become more widely because it attempts to reduce application maintenance and allows supporting more users. Unlike the two-tier model, application code isn’t loaded on the client machine: it’s stored on the application server. This model’s three tiers are the user interface, the application server, and the database server. The Oracle application server allows deploying Oracle Forms applications via the Web and contains the user interface and processing logic, which together respond to user actions and send code to the database server for processing. |
| 1 | 8 | **SQL\*Plus** is the console or command-line interface which is a basic tool included with the Oracle server that allows users to send SQL and PL/SQL statements directly to the Oracle database server for processing. |
| 1 | 9 | Oracle SQL Developer has a GUI for developers to explore database objects, review and edit code quickly and create and debug program units. |
| 1 | 13 | To run a script file click **File > Open** from the menu. Select the file and then click Open. Click the **Run Script** button. |
| 1 | 15 | SELECT <columns> FROM <table\_views> WHERE <conditions> GROUP BY <columns> HAVING <aggregate conditions> ORDER BY <columns> |

# Chapter 2: Basic PL/SQL Block Structure

|  |  |  |
| --- | --- | --- |
| Ch | Pg | Question |
| 2 | 37 | **Anonymous** **blocks** are blocks of code that aren’t stored for reuse and, as far as the Oracle server is concerned, no longer exist after they run. |
| 2 | 38 | PL/SQL Block Structure   |  |  | | --- | --- | | Block Section | Section Content | | DECLARE | Creates variables, cursors, and types | | BEGIN | Contains SQL statements, conditional logic processing, loops, and assignment statements | | EXCEPTION | Contains error handlers | | END | Closes the block | |
| 2 | 38 | The **DECLARE** sections contains code that creates variables, cursors, and data types used to hold data for processing and manipulation. |
| 2 | 38 | The **BEGIN** section, the heart of the PL/SQL block, contains all the processing action or programming logic.   * Conditional logic is used to make decisions on what action to take * Loops are used to repeat code * Assignment statements are used to put or change values in variables. |
| 2 | 38 | The **EXCEPTION** section contains handlers that enable you to control what the application does if an error occurs during executable statements. |
| 2 | 39 | **Variables** are named memory areas that hold values so that they can be retrieved and manipulated in programs.  A **record** (which is a composite variable) is needed if multiple values need to be stored.  Create a **cursor**, which is a structure specifically suited to processing a group of rows, if you intend to process a number of rows retrieved with a select statement. |
| 2 | 41 | DECLARE  lv\_ord\_date DATE; // DATE is data type  lv\_last\_txt VARCHAR2(25); // values are NULL when begin section starts execution  lv\_qty\_num NUMBER(2);  lv\_ship\_flag\_bln BOOLEAN;  BEGIN  -- - pl/sql executable statemetns---  END; |
| 2 | 41 | PL/SQL assignments statements are constructed as follows: ***variable\_name := value or expression*;** |
| 2 | 41 | One simple way to display values onscreen in SQL Developer is to use the **DBMS\_OUTPUT.PUT\_LINE** procedure. The PUT\_LINE procedure is available in the DBMS\_OUTPUT Oracle-supplies package. |
| 2 | 44 | Do not use double quotes around a variable name. Doing so makes the variable name case sensitive, which can cause confusion. |
| 2 | 48 | The **DEFAULT** keyword can be used in placed of the: = to get the same result.  You can set other controls on variable values with the NOT NULL and CONSTANT options.  The **NOT** **NULL** option requires that the variable **always contains a value**  The **CONSTANT** option can be added to the variable declaration to ensure that the variable **always contains a particular value in the block**.  DECLARE  lv\_shipcntry\_txt VARCHAR(15) NOT NULL := ‘US’;  lv\_taxrate\_num CONSTANT NUMBER(2,2) := 0.06;  BEGIN |
| 2 | 53 | The statements used to control the flow of logic processing in programs are commonly called **control structures**. |
| 2 | 53 | An **IF statement** is a mechanism for checking a condition to determine whether statements should or shouldn’t be processed |
| 2 | 56 | IF *condition* THEN statement END IF; |
| 2 | 56 | IF *condition* THEN statement  ELSIF *statement* IS NULL(IS NOT NULL, NVL, AND NVL2) THEN  ELSE *statement*  END IF; |
| 2 | 64 | CASE lv\_state\_txt  WHEN ‘VA’ THEN lv\_tax\_num :- lv\_sub\_num \* 0;06;  WHEN condition THEN statement;  ELSE statement;  END CASE;  If no TRUE conditions are found in the WHEN clauses, and an ELSE clause isn’t included, the CASE statement includes an implicit ELSE clause that raises an Oracle error. |
| 2 | 67 | A **CASE expression** evaluates conditions and returns a value in an assignment statement.  BEGIN // use if the goal is to assign a value to a variable  lv\_tax\_num := CASE lv\_state\_txt  WHEN ‘VA’ THEN lv\_sub\_num \* 0.06 //WHEN clauses don’t end in semicolons  ELSE lv\_sub\_num \* 0.04 // If no WHEN condition is met and no ELSE clause the result is NULL rather than an error  END; // not END CASE  END; |
| 2 | 69 | **Looping constructs** make it possible to repeat processing a portion of code. |
| 2 | 69 | Three types of PL/SQL loops are covered in the following sections: **basic, WHILE, and FOR** |
| 2 | 69 | A **basic loop** uses the LOOP and END LOOP markers to begin and end the lop code, which includes any statements to be repeated.  LOOP  *Statements*  EXIT WHEN *condition*; // if the EXIT WHEN clause isn’t included the result is an **infinite loop OR IF statement THEN EXIT**;  END LOOP; |
| 2 | 72 | A **WHILE loop** includes a condition to check at the top of the loop in the LOOP clause.  WHILE condition LOOP  s*tatements*  END LOOP; |
| 2 | 73 | A **FOR loop** indicates how many times to loop by including a range in the statement  FOR counter IN lower\_bound…upper\_bound LOOP  Statements  END LOOP; |
| 2 | 75 | The **CONTINUE** or **CONTINUE WHEN** statement provides a mechanism for existing a loop’s current iteration. It moves loop processing to the next iteration. CONTINUE WHEN MOD(i,5) <> 0; |

# 3: Handling Data in PL/SQL Blocks

|  |  |  |
| --- | --- | --- |
| Ch | Pg | Question |
| 3 | 88 | A SELECT statement in a PL/SQL block must include an INTO clause. This clause follows the SELECT clause and indicates which variables should hold the values retrieved from the database. |
| 3 | 94 | The **%TYPE** attribute tells the system to look up a database column’s data type and use if for the declared variable.   * The programmer doesn’t have to look up the data type to make sure the correct type and size are declared. * If changes are made to the database structure, such as increasing a column’s size, the programmer doesn’t have to be concerned with modifying all the variable declarations in any code referencing the column. * It causes a slight performance hit because the database server must look up the data type from the data dictionary. |
| 3 | 94 | Specifying the data type using the %TYPE attribute is called using an **anchored data type.** |
| 3 | 101 | A **record** can store and handle multiple values or fields, each having its own name and data type, as one unit. This structure is similar to a table row. |
| 3 | 101 | A **collection** can store an ordered group of elements of **the same type**, with each element having a unique subscript or index. This structure is similar to arrays found in other languages. A difference between the **record** and collection is that record variables hold a variety of data types, whereas a **collection** holds a group of elements of the same type |
| 3 | 101 | Creating a record data type with the TYPE statement is called declaring a programmer defined record  TYPE type\_basket IS RECORD {  Idbasket bb\_basket.idbasket%TYPE,  Dtcreated bb\_basket.dtcreated%TYPE);  rec\_basket type\_basket;  BEGIN  SELECT idbasket,dtcreated INTO rec\_basket FROM bb\_basket WHERE idshopper = lv\_shopper\_num;  END; |
| 3 | 103 | A **%ROWTYPE** attribute determines the needed data type by referencing a database table structure. It reviews the table structure for all column names and data types and creates the record data type based on this information.  DECLARE  rec\_shopper bb\_shopper%ROWTYPE;  BEGIN  SELECT \* INTO rec\_shopper FROM bb\_shopper WHER idshopper = 25;  END; |
| 3 | 103 | If table columns are altered, you don’t need to modify data-handling variables because the columns and data types are retrieved automatically from the data dictionary and reflect the current table structure. |
| 3 | 108 | There are two collection types: an associative array (index by table) and a table of records. |
| 3 | 108 | An **associative array** is a variable that can handle many rows of data but only one field. This structure is a set of key-value pairs in which each key uniquely identifies a value in the array. The index can be an integer or a string variable. |
| 3 | 108 | |  |  | | --- | --- | | Characteristic | Description | | One-dimensional | Can have only one column. | | Unconstrained | Rows added dynamically as needed | | Sparse | A row exists only when a value is assigned. Rows don’t have to be assigned sequentially. | | Homogenous | All elements have the same data type | | Indexed | Integer index serves as the table’s primary key. | |
| 3 | 108 | |  |  | | --- | --- | | Attribute name | Description | | COUNT | Number of rows in the table | | DELETE | Removes a row from the table | | EXISTS | TRUE if the specified row does exist | | FIRST and LAST | Smaller and largest index value in the table | | PRIOR and NEXT | Index for the previous and next row in the table, compared with the specified value | |
| 3 | 111 | A **table of records** is another type of collection. It can handle more than one record or row of data. |
| 3 | 111 | A group of initialized variables is declared to represent data from the application page and a number to indicate the row number or index of the table of records that should be added.  DECLARE  TYPE type\_basketitem IS TABLE OF bb\_basketitem%ROWTYPE  INDEX BY BINARY\_INTEGER;  table\_items type\_basketitmes |
| 3 | 113 | Bulk processing is a mechanism Oracle uses to reduce performance issues. When the **BULK COLLECT** clause and **FORALL** statements are used in PL/SQL programs, they provide instructions to the processing engine to group SQL actions for processing, which reduces context switching. |
| 3 | 115 | A GOTO statement instructs a program to jump to a specific area of code. It’s used to branch logic so that only certain portions of code are processed based on some condition. It interrupts the low of execution, which makes understanding and maintaining code more difficult. |