Day 16- Facilitation Guide

Index

For (1.5 hrs) ILT

I. Introduction to PlSql

Though SQL is the natural language of the DBA,it suffers from various inherent disadvantages,when used as a conventional programming language.

- SQL does not have any **procedural capabilities** i.e. SQL does not provide the programming techniques of common checking,looping and branching that is vital for data testing before its permanent storage.
- SQL statements are passed to the SQL engine **one** at a time,Each time an SQL statement is executed,a call is made to the engine's resources.This adds to the traffic on the network,thereby decreasing the speed of data processing,especially in a multi-user environment.
- While processing an SQL sentence if an error occurs,the sql engine displays its own error messages.SQL has no facility for programmed handling of errors that arise during the manipulation of data.

Although SQL is a very powerful tool,iys sets of disadvantages prevent, it from being a fully structured programming language.For fully structured programming language,Oracle provides PL/SQL.

PL/SQL is a super set of SQL.PL/SQL is a block -structured language that enables developers to combine the power of SQL with procedural statement.PL/SQL bridge the gap between database technology and procedural programming languages.

Advantages of PL/SQL

1. PL/SQL is a development tool that not only supports SQL data manipulation but also provides facilities of conditional checking,branching and looping.
2. PL/SQL sends an entire block of SQL statements to the Oracle engine all in one go.There is a definite improvement in the performance time of Oracle engine.
3. PL/SQL also permits dealing with errors as required and facilitates displaying user-friendly messages,when errors are encountered.
4. PL/SQL allows declaration and use of variables in blocks of code.These variables can be used to store intermediate results of a query for later processing,or calculate values and insert them into an Oracle table later.

You might be curious about the Oracle Database. Let's begin by addressing that question.

What is an Oracle database?

An Oracle database is a relational database management system (RDBMS) developed by Oracle Corporation. It is one of the most widely used and respected database systems in the world. Oracle databases are known for their robustness, scalability, and comprehensive features, making them a popular choice for businesses and enterprises of all sizes.

Key characteristics and features of Oracle databases include:

Relational Database: Oracle uses a relational model, organizing data into tables with rows and columns. This structured approach allows for efficient data storage and retrieval.

ACID Compliance: Oracle databases adhere to ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring data integrity and transaction reliability.

Security: Oracle places a strong emphasis on data security, providing robust authentication, authorization, and encryption features.

Performance: Oracle databases are optimized for performance, with features like query optimization, indexing, and caching.

PL/SQL: Oracle's proprietary procedural language, PL/SQL, allows developers to create stored procedures, functions, and triggers for advanced data manipulation and control.

Support for Various Data Types: Oracle supports a wide range of data types, including text, numbers, dates, and more, to accommodate various application requirements.

Oracle databases are commonly used in a variety of industries, including finance, healthcare, government, and e-commerce, where data reliability, performance, and security are critical. The Oracle Database Management System (DBMS) comes in different editions, including Oracle Database Standard Edition, Oracle Database Enterprise Edition, and others, each tailored to different business needs and budgets.

Let's discuss about various key components of PL/SQL:

1. **The generic PL/SQL block**

Every programming environment allows the creation of structured,logical blocks of code that describe process,which have to be applied to data.A single PL/SQL block consists of a set of SQL statements,clubbed together and passed to the oracle engine entirely.This block has to be logically grouped together for the engine to recognize it as a singular code block.A PL/SQL block has a definite structure,which can be divided into sections.The sections of a PL/SQL block are:

1. Declare section
2. The master Begin and End section that also(optionally) contains an Exception section.

Each of these is explained below:

The Declare Section

Code block start with a declaration section,in which,memory variables and other Oracle objects can be declared and if required initialized.Once declared,they can be used in SQL statements for data manipulation.

The Begin Section

It consists of a set of SQL and PL/SQL statements which describe processes that have to be applied to table data.Actual data manipulation,retrieval,looping and branching constructs are specified in this section.

The Exceptions Section

This Section deals with handling of errors that arise during execution of the data manipulation statements,which make up the PL/SQL code block.Errors can arise due to syntax,logic,and /or validation rule violation.

The End Section

This marks the end of a PL/SQL block.

2. PL/SQL Datatype

PL/SQL (Procedural Language/Structured Query Language) supports various data types for declaring variables and defining the structure of data in your programs. Here's a tutorial on some commonly used PL/SQL data types:

Numeric Data Types:

NUMBER: Used to store numeric values. It can have optional precision and scale. For example:

```
emp_salary NUMBER(10, 2);
```

Character Data Types:

CHAR / VARCHAR2: Used to store character strings. CHAR has a fixed length, while VARCHAR2 has a variable length. For example:

```
emp_name VARCHAR2(50);
```

Date and Time Data Types:

DATE: Used to store date and time values.

```
hire_date DATE;
```

Boolean Data Type:

BOOLEAN: Used to store Boolean values (TRUE, FALSE, or NULL).

Collection Data Types:

PL/SQL Tables: An ordered set of elements, similar to an array or list in other programming languages.

```
TYPE emp_ids IS TABLE OF NUMBER;
emp_id_list emp_ids;
```

Record Data Type:

RECORD: Used to create a composite data structure, similar to a structure or object in other languages.

```
TYPE emp_record IS RECORD (
  emp_id NUMBER,
  emp_name VARCHAR2(50)
);
emp_data emp_record;
```

Cursor Data Type:

CURSOR: Used to fetch rows from a result set in a SQL query.

```
CURSOR emp_cursor IS
  SELECT emp_id, emp_name FROM employees;
```

LOB Data Types:

- CLOB / BLOB: Used to store large binary or character data, such as text or binary files.

```
emp_resume CLOB;
```

3. **Variables Declarations**

In PL/SQL (Procedural Language/Structured Query Language), you can declare and use variables to store data temporarily within your program. PL/SQL variables are typically used to hold values that you want to manipulate or use in various operations. Here's how you declare and use PL/SQL variables:

```
DECLARE
  -- Declare variables
 StudentID NUMBER := 101;  -- Numeric variable with an initial value
 StudentName VARCHAR2(50);  -- String variable (VARCHAR2) with a maximum length of 50 characters

BEGIN
  -- Assign values to variables
 StudentName := 'John Doe';


  -- Display variables
  DBMS_OUTPUT.PUT_LINE('Student Id: ' ||   StudentID);
  DBMS_OUTPUT.PUT_LINE(Student Name: ' ||    StudentName);
```

```
END;
```

Food for thought..

The trainer can ask the students the following question to engage them in a discussion:
How do PL/SQL data types differ from standard SQL data types, and why is this difference important in database programming?

II. Stored Procedures

A stored procedure in PL/SQL (Procedural Language/Structured Query Language) is a precompiled block of code that can be stored in a database and executed as a single unit. Stored procedures allow you to encapsulate complex SQL and business logic, making it easier to manage and maintain database operations. Here's how to create and use stored procedures in PL/SQL:

Creating a Stored Procedure:

To create a stored procedure in PL/SQL, you typically use the CREATE PROCEDURE statement. Here's a basic example of a stored procedure that retrieves employee information based on an employee ID:

```
CREATE OR REPLACE PROCEDURE get_student_info(
  p_StudentID NUMBER,
  o_student_name OUT VARCHAR2,
) AS
BEGIN
  SELECT emp_name, emp_salary
  INTO o_employee_name, o_employee_salary
  FROM employees
  WHERE emp_id = p_employee_id;
END get_employee_info;
/
```

In this example:

CREATE OR REPLACE PROCEDURE is used to create or update a stored procedure.

p_employee_id, o_employee_name, and o_employee_salary are parameters. p_employee_id is an input parameter, while o_employee_name and o_employee_salary are output parameters.
The procedure retrieves employee information based on the input employee ID and stores the results in output parameters.

Calling a Stored Procedure:

You can call a stored procedure from PL/SQL code or from an external application using JDBC, as mentioned earlier. Here's an example of calling the get_employee_info procedure:

```
DECLARE
  emp_name VARCHAR2(50);
  emp_salary NUMBER;
BEGIN
  get_employee_info(101, emp_name, emp_salary);
  DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_name);
  DBMS_OUTPUT.PUT_LINE('Employee Salary: ' || emp_salary);
END;
/
```

Modifying a Stored Procedure:

If you need to modify a stored procedure, you can use the ALTER PROCEDURE statement to make changes to the code. For example:

```
ALTER PROCEDURE get_employee_info(
  p_employee_id NUMBER,
  o_employee_name OUT VARCHAR2,
  o_employee_salary OUT NUMBER
) AS
BEGIN
  SELECT emp_name, emp_salary
  INTO o_employee_name, o_employee_salary
  FROM employees
  WHERE emp_id = p_employee_id;

  -- Additional logic here
END get_employee_info;
/
```

Dropping a Stored Procedure:

To remove a stored procedure from the database, you can use the DROP PROCEDURE statement:
DROP PROCEDURE get_employee_info;

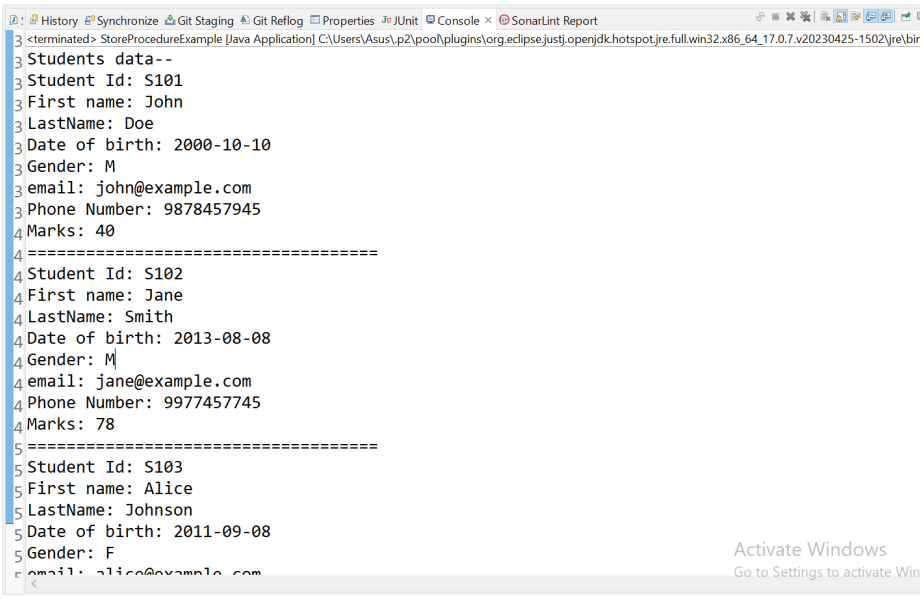
Stored procedures are useful for encapsulating complex business logic, improving code reusability, and enhancing database security. They allow you to centralize database operations, making it easier to manage and maintain your database application.

Let's explore a comprehensive example using JDBC:

```
package jdbcConnectivity;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
class RetrieveData
{
	//Create a method for query execution
	public static ResultSet getStudentRecords(Statement stmt) throws SQLException
	{
		//System.out.println(stmt);
		String sql="select * from Student";

		//Executing the query
		ResultSet rset=stmt.executeQuery(sql);
		return rset;
	}
}
public class StoreProcedureExample {
	public static void main(String[] args) {

		//creating Connection object and Statement object using try-with-resource
		try(Connection con=ConnectDB.dbConnect();
				Statement stmt=con.createStatement();){

			//invoking getStudentRecords() method to run the query
		ResultSet rs=RetrieveData.getStudentRecords(stmt);
```

```
                System.out.println("Students data--");
                //checking data present or not
                while(rs.next())
                {
                        //displaying all data one by one
                        System.out.println("Student Id: "+rs.getString("StudentID"));
                        System.out.println("First name: "+rs.getString("FirstName"));
                        System.out.println("LastName: "+rs.getString("LastName"));
                        System.out.println("Date of birth: "+rs.getDate("DateOfBirth"));
                        System.out.println("Gender: "+rs.getString("Gender"));
                        System.out.println("email: "+rs.getString("Email"));
                        System.out.println("Phone Number: "+rs.getString("Phone"));
                        System.out.println("Marks: "+rs.getInt("marks"));

System.out.println("====================================");
                }
                }catch (Exception e) {
                        System.out.println(e);
                }
        }
}
```

Output:



Question: What are the benefits of using stored procedures to encapsulate complex database operations?

III. Triggers

In SQL, a trigger is a database object that is associated with a specific table and is automatically executed in response to certain events or actions that occur on that table. Triggers are used to enforce business rules, maintain data integrity, or perform automated actions when data is modified in a database. The events that can trigger the execution of a trigger typically include:

- INSERT: Triggered when a new row is inserted into the table.
- UPDATE: Triggered when one or more rows in the table are updated.
- DELETE: Triggered when one or more rows are deleted from the table.

Triggers are defined using SQL statements and are stored in the database along with other objects.

They consist of two main components:

- Trigger Event: This specifies the event or action that causes the trigger to execute. It can be one of the events mentioned above (INSERT, UPDATE, DELETE).

- Trigger Action: This is the set of SQL statements that are executed when the trigger is fired. The trigger action can include SQL queries, data manipulation, or other database operations.

Triggers can be useful for a variety of purposes, such as:

- Enforcing data validation rules.
- Logging changes to the database.
- Maintaining data consistency (e.g., updating related records when a record is modified).
- Implementing auditing and security measures.
- Automatically generating or updating derived data.
- It's important to use triggers judiciously because they can introduce complexity and potential performance overhead in a database system. Care should be taken to ensure that triggers are well-designed and do not lead to unintended consequences or performance bottlenecks.

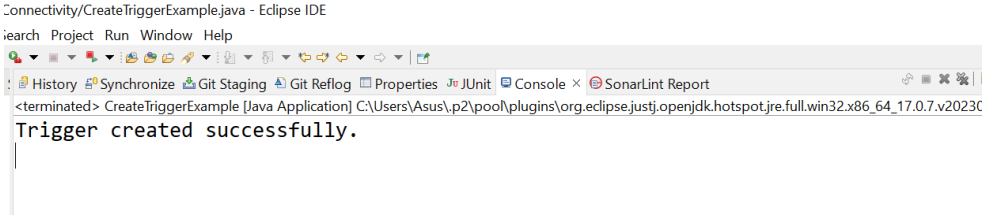
Here we will create Trigger for User Table:

```
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| uid       | int         | NO   | PRI | NULL    |       |
| name      | varchar(30) | YES  |     | NULL    |       |
| address   | varchar(50) | YES  |     | NULL    |       |
| timestamp | datetime    | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

Let's explore a comprehensive example using JDBC:

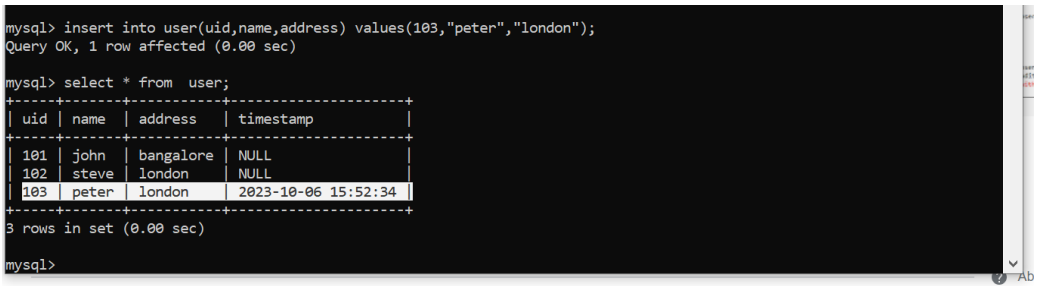
```
package jdbcConnectivity;
import java.sql.Connection;
import java.sql.Statement;
public class CreateTriggerExample {
        public static void main(String[] args) {
         try (Connection connection = ConnectDB.dbConnect();
         Statement statement = connection.createStatement()) {
         // Define the SQL statement to create a trigger
         String createTriggerSQL= "CREATE TRIGGER update_timestamp " +
"BEFORE INSERT ON user " +
"FOR EACH ROW " +
"BEGIN " +
" SET NEW.timestamp = NOW(); " +
"END;";

         // Execute the SQL statement to create the trigger
         statement.executeUpdate(createTriggerSQL);
         System.out.println("Trigger created successfully.");
         } catch (Exception e) {
         e.printStackTrace();
         }
        }
}
```

Output:



In this example Customize the createTriggerSQL string to define the trigger. This trigger is named update_timestamp, and it is executed BEFORE INSERT on the your_table table. It sets the timestamp column to the current timestamp using the NOW() function whenever a new row is inserted.

When you insert a new row into your_table, the trigger will automatically update the timestamp column with the current date and time before the insertion occurs.Here is the output:



IV. MySql Administration

- MySQL administration involves managing and maintaining a MySQL database system to ensure its proper functioning, security, and performance. Here are some key aspects of MySQL administration:

- Installation and Setup: The first step in MySQL administration is to install the MySQL database server software on the desired host. After installation, you need to configure server settings, such as setting the root password and network parameters, to make it ready for use.

- User and Privilege Management: MySQL allows you to create multiple users and assign specific privileges to them. Administrators need to manage user accounts, grant or revoke privileges, and ensure secure access control to the database.

- Backup and Recovery: Regularly backing up your MySQL databases is crucial to protect against data loss due to hardware failures, user errors, or other unforeseen events. MySQL offers various backup methods, including logical backups (using mysqldump) and physical backups (using tools like MySQL Enterprise Backup or file-level backups).
-

- Monitoring and Performance Tuning: Monitoring the performance of your MySQL server is essential to identify and address issues related to query optimization, resource usage, and server health. Tools like MySQL Performance Schema and Query Analyzer can help in diagnosing performance bottlenecks.

- Security: Ensuring the security of your MySQL database is critical. This involves implementing security best practices, applying security patches, using firewalls, and configuring authentication and authorization mechanisms.

- Database Maintenance: Routine maintenance tasks include optimizing database tables, managing indexes, checking for and repairing data corruption, and monitoring disk space usage.

- High Availability and Replication: MySQL provides features like replication, clustering, and failover solutions to ensure high availability and reliability of your database system. Setting up replication or clustering may be necessary for applications that require 24/7 uptime.

- Scaling: As your database grows, you might need to scale it horizontally (adding more servers) or vertically (upgrading server hardware) to handle increased load and maintain performance.

- Logs and Auditing: MySQL maintains various logs (error log, slow query log, general query log) that can be useful for troubleshooting and auditing. Administrators should review these logs regularly.

- Data Security and Compliance: If your organization handles sensitive data, you may need to ensure compliance with data protection regulations such as GDPR, HIPAA, or PCI-DSS. This includes encrypting data, implementing access controls, and auditing data access.

- Backup and Restore Strategies: Develop and document a comprehensive backup and restore strategy, including regular testing of backups to ensure they can be successfully restored.

- Upgrade and Patch Management: Keeping MySQL up-to-date with the latest security patches and upgrades is essential for security and compatibility with new features.

- Automation: Consider using automation tools and scripts to streamline routine administrative tasks and reduce human errors.

- MySQL administration can be a complex and specialized field, but there are various resources available, including official MySQL documentation, online tutorials, and community forums, to help you learn and master these skills. It's important to stay informed about best practices and evolving technologies in the world of MySQL to effectively manage and maintain your database systems.

Exercise:

Use ChatGPT to explore more about MySQL Administration
Put the below problem statement in the message box and see what ChatGPT says.

I am interested in learning how to establish a new MySQL account. Could you kindly provide a step-by-step guide for this procedure?or how can I do that?