SQL STORED PROCEDURES

SQL Stored Procedures are a powerful feature in **database management systems** (DBMS) that allow developers to encapsulate SQL code and business logic. Stored procedures are precompiled SQL statements that are stored in the **database** and can be executed as a **single unit**.

In this article, we will explain SQL stored procedures, explaining their **purpose**, **benefits**, **syntax**, **types**, and **practical examples**. We will also explore why they are a critical part of SQL development and how they can be used for better **database performance** and **security**.

What is a SQL Stored Procedure?

A SQL Stored Procedure is a collection of SQL statements **bundled together** to perform a specific task. These procedures are stored in the database and can be called upon by users, applications, or other procedures. Stored procedures are essential for **automating database tasks**, improving efficiency, and **reducing redundancy**. By encapsulating logic within **stored procedures**, developers can streamline their **workflow** and enforce **consistent business rules** across multiple applications and systems.

Syntax:

```
CREATE PROCEDURE procedure_name
(parameter1 data_type, parameter2 data_type, ...)
AS
BEGIN
  — SQL statements to be executed
END
```

Key Terms

- **CREATE PROCEDURE:** This keyword creates the stored procedure with the given name.
- **@parameter1, @parameter2:** These are input parameters that allow you to pass values into the stored procedure.
- **BEGIN...END**: These keywords define the block of SQL statements that make up the procedure body.

Why Use SQL Stored Procedures?

There are several key reasons why SQL Stored Procedures are widely used in database management:

1. **Performance Optimization**: Since stored procedures are precompiled, they execute faster than running ad-hoc SQL queries. The database engine can reuse the execution plan, eliminating the need for repeated query parsing and optimization.
2. **Security**: By using stored procedures, developers can restrict direct access to sensitive data. Users can execute procedures without accessing the underlying tables, helping to protect critical information.
3. **Code Reusability**: SQL stored procedures can be reused in multiple applications or different parts of an application. This reduces the need to rewrite complex queries repeatedly.
4. **Reduced Network Traffic**: Instead of sending multiple individual queries to the database server, stored procedures allow you to execute multiple operations in one go, reducing network load.
5. **Maintainability**: Stored procedures simplify code maintenance. Changes made to the procedure are automatically reflected wherever the procedure is used, making it easier to manage complex logic.

Example of Creating a Stored Procedure

In this example, we create a stored procedure called **GetCustomersByCountry**, which accepts a Country parameter and returns the **CustomerName** and **ContactName** for all customers from

that country. The procedure is designed to query the **Customers** table, which contains customer information, including their **names**, **contact details**, and **country**.

CustomerID	CustomerName	ContactName	Country
1	Shubham	Thakur	India
2	Aman	Chopra	Australia
3	Naveen	Tulasi	Sri Lanka
4	Aditya	Arpan	Austria
5	Nishant	Jain	Spain

Customers Table

By passing a country as a parameter, the stored procedure dynamically fetches the relevant customer details from the table

Query:

```
-- Create a stored procedure named "GetCustomersByCountry"
CREATE PROCEDURE GetCustomersByCountry
  @Country VARCHAR(50)
AS
BEGIN
  SELECT CustomerName, ContactName
  FROM Customers
  WHERE Country = @Country;
END;


-- Execute the stored procedure with parameter "Sri lanka"
EXEC GetCustomersByCountry @Country = 'Sri lanka';
```

Output

CustomerName	Contact Name
Naveen	Tulasi

Note: We will need to make sure that the user account has the **necessary privileges** to create a **database**. We can try logging in as a **different user** with administrative privileges or contact the database administrator to grant the necessary privileges to our user account. If we are using a cloud-based database service, make sure that we have correctly configured the user account and its permissions.

Types of SQL Stored Procedures

SQL stored procedures are categorized into different types based on their use case and functionality:

1. **System Stored Procedures**: These are predefined stored procedures provided by the SQL Server for performing administrative tasks. Examples include sp_help for viewing database object information and sp_rename for renaming database objects.
2. **User-Defined Stored Procedures**: These are custom stored procedures created by the user to perform specific operations. For example, creating a procedure that calculates the total sales for a particular product category.
3. **Extended Stored Procedures**: These allow for the execution of external functions, which might be implemented in other languages such as C or C++. Extended procedures are typically used for integrating external tools into SQL Server.
4. **CLR Stored Procedures**: These are stored procedures written in .NET languages (like C#) and executed within SQL Server. CLR stored procedures are useful when advanced functionality is needed that isn't easily achievable with T-SQL alone

Advantages of Using SQL Stored Procedures

1. **Improved Performance**: Stored procedures are precompiled, meaning they execute faster than running multiple individual queries.
2. **Enhanced Security**: Users can be granted permission to execute stored procedures without directly accessing the underlying tables.
3. **Code Reusability**: Stored procedures allow for reusability, making it easier to maintain and update code.
4. **Reduced Network Traffic**: By bundling multiple SQL statements into one call, stored procedures reduce network load and improve application performance.
5. **Better Error Handling**: SQL stored procedures provide a structured way to manage errors using TRY...CATCH blocks.

Conclusion

SQL stored procedures are an essential part of SQL development, offering benefits such as improved **performance**, **security**, and **maintainability**. By encapsulating SQL queries into reusable units, stored procedures simplify database management, enhance efficiency, and ensure consistent business logic execution. By using stored procedures, we can **automate tasks**, minimize the risk of SQL injection, and ensure consistent execution of complex SQL logic. Stored procedures are integral to modern database management and an important component in building scalable, efficient, and secure database systems.