**What are Stored Procedures?**

A stored procedure is a pre-written SQL query that can be executed with a single command and is stored directly within the database. It is commonly used for tasks that need to be performed repeatedly. Think of it as a mini-program that you can call upon to execute specific actions on your data.

To illustrate, imagine you are a chef in a busy kitchen who needs to prepare the same complex dish for multiple customers every day. Instead of writing out the recipe each time, you create a "stored recipe" that outlines all the necessary steps. This is essentially what a stored procedure is in the context of databases.

**Importance of Stored Procedures**

Stored procedures in SQL are indispensable tools for data analysts, developers, database administrators, and anyone involved with SQL who aims to optimize database operations and maintain data integrity. They offer a transformative way to streamline database interactions, boost efficiency, enhance security, and foster better code organization and maintainability. Here are some compelling benefits of using stored procedures:

**Reusability**: One of the standout advantages of stored procedures is their reusability. By eliminating the need to rewrite complex SQL queries repeatedly, they save time and reduce errors. Once you define a stored procedure, you can effortlessly invoke it by name whenever necessary, executing the embedded query with ease. For instance, in an online store database, if you frequently need to update customer information upon order placement, creating a stored procedure called “ProcessOrder” allows you to encapsulate and simplify those steps, enhancing workflow and accuracy.

**Parameter Acceptance**: Stored procedures excel in their ability to accept parameters, which act as customizable inputs. This feature enables you to tailor the procedure’s behavior according to specific values you provide, making stored procedures incredibly adaptable for various scenarios while maintaining consistent logic.

**Logic and Control Flow**: Beyond simple data retrieval, stored procedures can incorporate advanced control flow statements such as IF-ELSE conditions and loops. This capability allows you to undertake more sophisticated tasks, facilitating complex data manipulation and processing. By leveraging stored procedures, you not only ensure efficiency but also significantly enhance the overall robustness of your database operations.

**Creating a stored procedure:**

To create a stored procedure, we use the command “CREATE PROCEDURE”. Let’s create a simple stored procedure that retrieves the Suffix, FirstName, Middle Name, LastName columns from the Person.Person table in the AdventureWorks database.

| **CREATE PROCEDURE Employee\_name** |  |
| --- | --- |
|  | **AS** |
|  |  |
|  | **BEGIN** |
|  | **SELECT Suffix, FirstName, MiddleName, LastName** |
|  | **FROM Person.Person** |
|  | **END;** |

**Steps:**

* CREATE PROCEDURE is the command used to initiate the creation of a stored procedure.
* Following the create command is the name of the stored procedure “employee\_name”.
* The “AS” keyword indicates the start of the procedure body.
* The “BEGIN” block signifies the beginning of the executable code within the stored procedure.
* The “SELECT” statement retrieves data from the database. Here, it selects four columns Suffix, FirstName, MiddleName, LastName
* The “FROM” clause specifies the table from which to retrieve data. In this case, it’s Person.Person, which likely refers to a table named Person within a schema named Person.
* The “END” block marks the end of the executable code within the stored procedure.

**Executing Sql Stored procedures:**

To execute a stored procedure we use the command EXEC or EXECUTE to call the name of the procedure written which was in our case “Employee\_name”.

EXECUTE Employee\_name;

**Modifying stored procedures**

In order to modify a set of procedures we make use of a command ALTER PROCEDURE command.

Using our case study we want to add another column “Title”

| ALTER PROCEDURE Employee\_name |  |
| --- | --- |
|  | AS |
|  | BEGIN |
|  | SELECT Suffix, FirstName, MiddleName, LastName, Title |
|  | FROM Person.Person; |
|  | END; |

**Using Parameter In SQL Stored Procedures**

Parameters are special variables used within stored procedures to provide dynamic input values. They act like placeholders that you can fill with specific data when you execute the procedure. It offers several advantages over hardcoding values directly into your SQL statements. Parameters ensures and improves the flexibility, reusability, maintainability of your stored procedure.

To use a parameter, it must be defined and declared when creating the stored procedure. This includes specifying the parameter name and the data type it can accept. Occasionally you can specify whether it’s an input parameter (providing data to the procedure), an output parameter (returning data from the procedure), or both. Within the query in the stored procedure, the parameter can be referenced by its name, as needed.

| **ALTER PROCEDURE Employee\_name** |  |
| --- | --- |
|  | **@suffix varchar(5) — Specify data type and size** |
|  |  |
|  | **AS** |
|  | **BEGIN** |
|  | **SELECT Suffix, FirstName, MiddleName, LastName, Title** |
|  | **FROM Person.Person** |
|  | **WHERE Suffix = @suffix; — Filter by the provided suffix** |
|  | **END;** |

From the above query we altered the employee\_name to add a parameter named @Suffix. To execute a stored procedure with a parameter, we must include the parameter input in our query. Let’s include an input to get records of people with the suffix ‘Sr.’, using the query below.

|  | EXEC Employee\_name @suffix = 'Sr.'; |
| --- | --- |

**Renaming Stored Procedures**

You can easily rename a stored procedure by using the command “SP\_RENAME”. Let’s rename the employee\_name procedure to employee\_names, using the query below

|  | **sp\_rename 'Employee\_name','Employee\_names'** |
| --- | --- |

N/B: When renaming the first name is the old initial stored procedure name and the second name is the new stored procedure name.

**Dropping Stored Procedures**

Dropping a SQL stored procedure is a process of removing its definition from the database. The primary method for dropping stored procedures involves using the “DROP PROCEDURE” statement. Let’s drop the employee\_names using the “DROP PROCEDURE” statement:

|  | **DROP PROCEDURE dbo.employee\_names** |
| --- | --- |

**Best Stored Procedure Practices**

* Write clear and well-commented SQL code within your stored procedures to ensure they are maintainable.
* Utilize consistent indentation to improve code structure and readability.
* Avoid putting too much logic into stored procedures as this can make them difficult to debug and test.
* Employ parameters for user input instead of hardcoding values within the stored procedure.
* Avoid using “SELECT \*” as it retrieves all columns, potentially impacting performance. Specify the required columns instead.
* Always utilize efficient “WHERE” clauses to filter data precisely.
* Utilize temporary tables strategically for complex data manipulation within the stored procedure. However, use them judiciously as they can consume memory.
* Regularly review and optimize your stored procedures for performance improvements.