Stored Procedures

Database Programmability

SoftUni Team

Technical Trainers

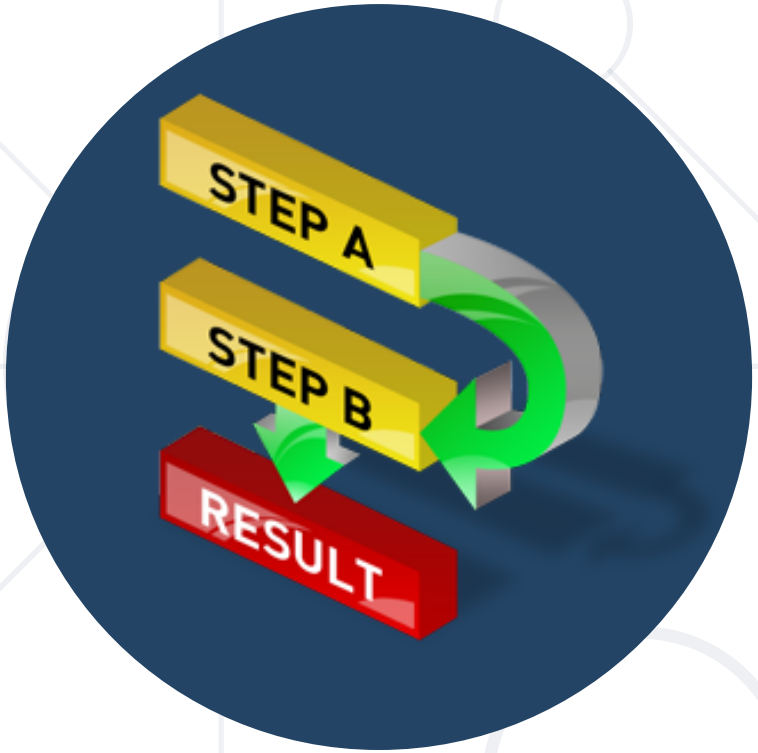
Software University

SoftUni

Software University

https://about.softuni.bg/

Table of Contents



Stored Procedures

What Are Stored Procedures?

- **Stored procedures** are **named sequences** of **T-SQL statements**.
  - **Encapsulate** repetitive program **logic**
  - Can **accept input parameters**
  - Can **return output results**
- **Benefits** of stored procedures
  - **Share** application logic
  - Improved **performance**
  - **Reduced** network **traffic**
  - They can be used as a **security** mechanism

Types of Stored Procedures

Software University

- **User-defined**
  - Can be created in a **user-defined database** or in all system databases except the **Resource** database
  - Can be developed in either **Transact-SQL** or as a reference to a **Microsoft .NET Framework** method
- **Temporary**
  - A form of user-defined procedures stored in **tempdb**

Creating Stored Procedures

- Syntax: **CREATE PROCEDURE … AS …**
- Example:

```
USE SoftUni
GO

CREATE PROC dbo.usp_SelectEmployeesBySeniority
AS
   SELECT *
   FROM Employees
   WHERE DATEDIFF(Year, HireDate, GETDATE()) > 20
GO
```

Procedure Name

Procedure Logic

Executing Stored Procedures

- Executing a stored procedure by **EXEC**

```
EXEC usp_SelectEmployeesBySeniority
```

- Executing a stored procedure within an INSERT statement

```
INSERT INTO Customers
EXEC usp_SelectEmployeesBySeniority
```

Altering Stored Procedures

- Use the ALTER PROCEDURE statement

```
USE SoftUni
GO
ALTER PROC usp_SelectEmployeesBySeniority
AS
  SELECT FirstName, LastName, HireDate,
    DATEDIFF(Year, HireDate, GETDATE()) as Years
  FROM Employees
  WHERE DATEDIFF(Year, HireDate, GETDATE()) > 20
  ORDER BY HireDate
GO
```

Procedure Name

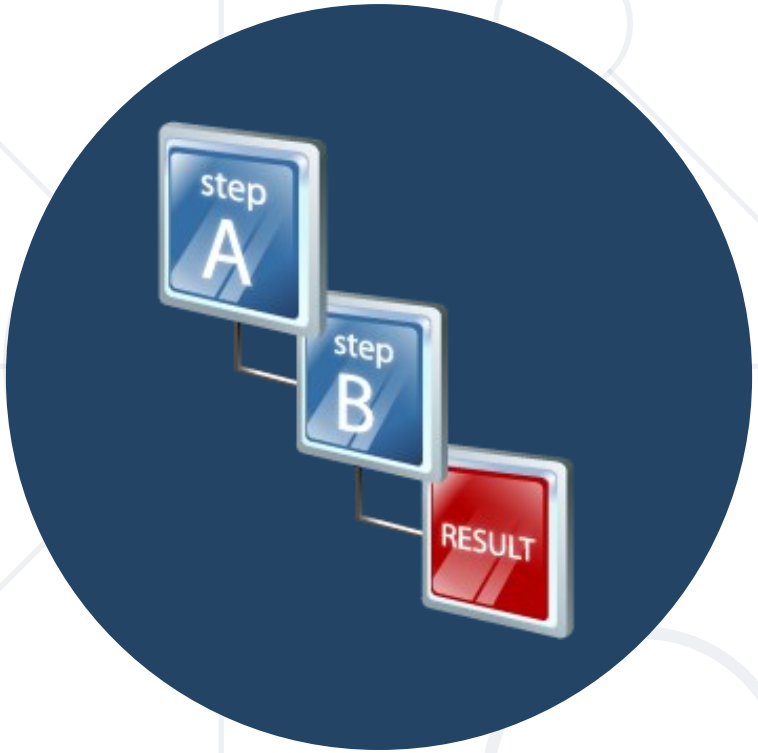
Dropping Stored Procedures

- **DROP PROCEDURE**

```
DROP PROC usp_SelectEmployeesBySeniority
```

- You could **check** if any objects **depend** on the stored procedure by executing **the system stored procedure sp_depends**

```
EXEC sp_depends 'usp_SelectEmployeesBySeniority'
```



Stored Procedures with Parameters

Defining Parameterized Procedures

- To define a **parameterized procedure** use the syntax:

```
CREATE PROCEDURE usp_ProcedureName
(@parameter1Name parameterType,
  @parameter2Name parameterType,…) AS
```

- Choose the parameter types carefully and provide an **appropriate default values**

```
CREATE PROC
usp_SelectEmployeesBySeniority(
  @minYearsAtWork int = 5) AS …
```

Parameterized Stored Procedures – Example


```
CREATE PROC usp_SelectEmployeesBySeniority
      (@minYearsAtWork int = 5)
AS
  SELECT FirstName, LastName, HireDate,
         DATEDIFF(Year, HireDate, GETDATE()) as Years
    FROM Employees
   WHERE DATEDIFF(Year, HireDate, GETDATE()) > @minYearsAtWork
   ORDER BY HireDate
GO

EXEC usp_SelectEmployeesBySeniority 10
```

Procedure Name

Procedure Logic

Usage

Passing Parameter Values

Software University

- Passing values **by parameter name**

```
EXEC usp_AddCustomer
  @customerID = 'ALFKI',
  @companyName = 'Alfreds Futterkiste',
  @address = 'Obere Str. 57',
  @city = 'Berlin',
  @phone = '030-0074321'
```

- Passing values **by position**

```
EXEC usp_AddCustomer 'ALFKI2', 'Alfreds
Futterkiste', 'Obere Str. 57', 'Berlin',
'030-0074321'
```

Returning Values Using OUTPUT Parameters

Software University

```
CREATE PROCEDURE dbo.usp_AddNumbers
    @firstNumber SMALLINT,
    @secondNumber SMALLINT,
    @result INT OUTPUT
AS
    SET @result = @firstNumber + @secondNumber
GO

DECLARE @answer smallint
EXECUTE usp_AddNumbers 5, 6, @answer OUTPUT
SELECT 'The result is: ', @answer

-- The result is: 11
```

Creating procedure

Executing procedure

Display results

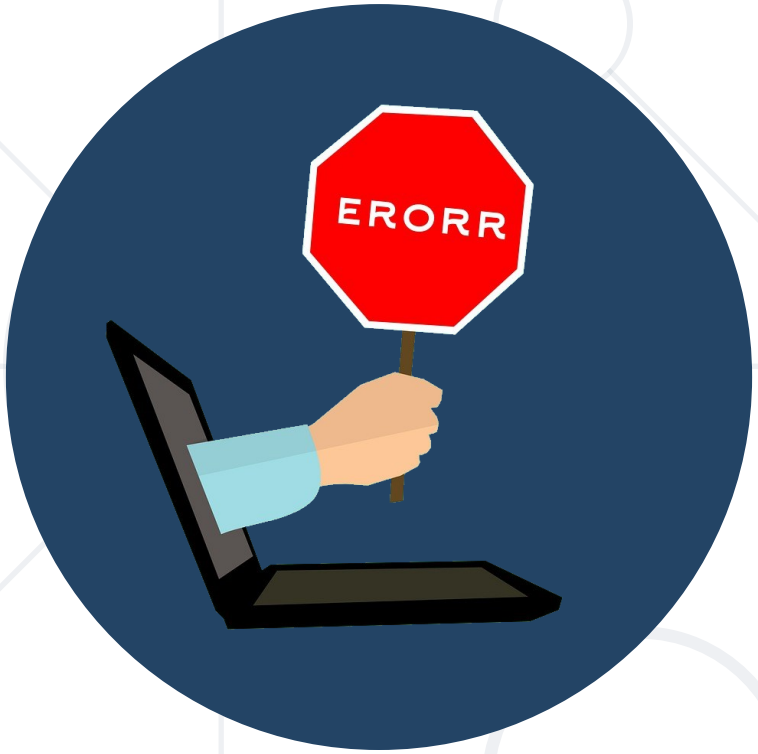
Returning Multiple Results

Checks if procedure exists and then Creates or Alters it

Multiple SELECT statements

```
CREATE OR ALTER PROC usp_MultipleResults
AS
SELECT FirstName, LastName FROM Employees
SELECT FirstName, LastName, d.[Name] AS Department
FROM Employees AS e
JOIN Departments AS d ON e.DepartmentID = d.DepartmentID;
GO

EXEC usp_MultipleResults
```



Error Handling

Error Throwing

- **THROW**
  - Raises an exception and transfers execution to a CATCH block
  - Arguments:
    - error_number - INT (between **50000** and **2147483647**)
    - message - **NVARCHAR(2048)**
    - state - **TINYINT** (between 0 and 255)

```
IF(@candidateAge < @minimalCandidateAge)
BEGIN
  THROW 50001, 'The candidate is too young!', 1;
END
```

Error Handling

- **TRY...CATCH**
  - SQL Statements can be enclosed in a **TRY** block.
  - If an error occurs in the **TRY** block, control is passed to another group of statements that is enclosed in a **CATCH** block

```
BEGIN TRY
    { sql_statement | statement_block }
END TRY
BEGIN CATCH
    [ { sql_statement | statement_block } ]
END CATCH
[ ; ]
```

Error Handling

```
BEGIN TRY
    -- Generate a divide-by-zero error.
    SELECT 1/0
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber
        ,ERROR_SEVERITY() AS ErrorSeverity
        ,ERROR_STATE() AS ErrorState
        ,ERROR_PROCEDURE() AS ErrorProcedure
        ,ERROR_LINE() AS ErrorLine
        ,ERROR_MESSAGE() AS ErrorMessage;
END CATCH
GO
```

Error Handling

- **@@ERROR**
  - Returns 0 if the previous Transact-SQL statement encountered no errors
  - Returns an error number if the previous statement encountered an error
  - **@@ERROR** is cleared and reset on each statement executed, check it immediately following the statement being verified, or save it to a local variable that can be checked later

Problem: Employees with Three Projects

- Create a procedure that assigns projects to an employee
  - If the employee has more than 3 projects, throw an exception

EmployeeID	ProjectID
1	5
1	6
2	6
2	7
2	8

Solution: Employees with Three Projects (1)

Software University

Procedure Name

```
CREATE PROCEDURE udp_AssignProject
(@EmployeeID INT, @ProjectID INT)
AS
BEGIN
DECLARE @maxEmployeeProjectsCount INT = 3
DECLARE @employeeProjectsCount INT
SET @employeeProjectsCount =
(SELECT COUNT(*)
   FROM [dbo].[EmployeesProjects] AS ep
   WHERE ep.EmployeeId = @EmployeeID)
   --INSERT NEW DATA
END
```

Parameters

Declare Variables

Solution: Employees with Three Projects (2)

Software University

```
IF(@employeeProjectsCount >= @maxEmployeeProjectsCount)
BEGIN
   THROW 50001, 'The employee has too many projects!', 1;
END

INSERT INTO [dbo].[EmployeesProjects]
  (EmployeeID, ProjectID)
VALUES (@EmployeeID, @ProjectID)
```

Throw Exception

Summary

- **Stored Procedures** allow us to save time by
  - Shortening code
  - Simplifying complex tasks

```
CREATE PROC usp_ProcedureName
AS …
```

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://softuni.org
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg