Functions and Stored Procedures

Database Programmability

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://about.softuni.bg/

Questions

Software University

sli.do

#csharp-db

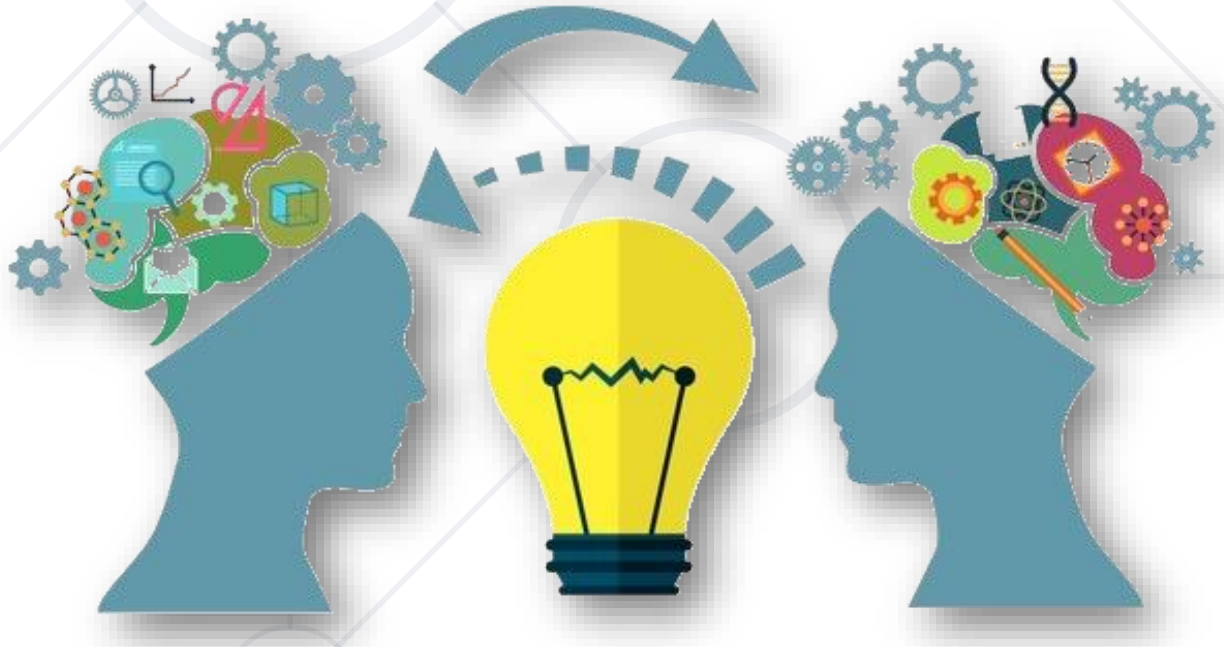
Table of Contents

User-Defined Functions

Definition, Usage, Syntax

Functions: Basic Definition

- At its core, a function **receives an input** and **produces an output**



Types of User-Defined Functions

- **Scalar functions**
  - **Similar** to the **built-in functions**
  - Returns a **single value**
- **Table-valued functions**
  - Similar to **a view with parameters**
  - **Returns a table** as a result of a single **SELECT** statement
    - Inline table-valued function (TVF)
    - Multi-statement table-valued function (MSTVF)

Functions: Limitations

- User-defined functions **cannot** be used to perform actions that **modify** the database state
- User-defined functions **cannot** return **multiple** result sets
- User-defined functions cannot make use of **dynamic SQL** or **temp tables**. Table variables are allowed.
- User-defined functions can be nested up to 32 levels
- Error handling is restricted in a user-defined function UDF does not support **`TRY...CATCH`**, **`@ERROR`** or **`RAISERROR`**

Create Functions (Scalar)

```
CREATE FUNCTION udf_ProjectDurationWeeks (@StartDate DATETIME,
@EndDate DATETIME)
RETURNS INT
AS
BEGIN
    DECLARE @projectWeeks INT;
    IF(@EndDate IS NULL)
    BEGIN
        SET @EndDate = GETDATE()
    END
    SET @projectWeeks = DATEDIFF(WEEK, @StartDate, @EndDate)
    RETURN @projectWeeks;
END
```

Function Name

Parameters

Return Type

Variable

IF Statement

Return Value

Create Functions (Table-Valued Function)

Software University

Function Name

```
CREATE FUNCTION udf_AverageSalaryByDepartment()
RETURNS TABLE AS
RETURN
(
      SELECT d.[Name] AS Department, AVG(e.Salary) AS AverageSalary
      FROM Departments AS d
      JOIN Employees AS e ON d.DepartmentID = e.DepartmentID
      GROUP BY d.DepartmentID, d.[Name]
)
```

No Parameters

Return Type

Return Value

Create Functions (Multi-statement TVF)

Software University

```
CREATE FUNCTION udf_EmployeeListByDepartment(@DepName nvarchar(20))
RETURNS @result TABLE(
    FirstName nvarchar(50) NOT NULL,
    LastName nvarchar(50) NOT NULL,
    DepartmentName nvarchar(20) NOT NULL) AS
BEGIN
    WITH Employees_CTE (FirstName, LastName, DepartmentName)
    AS(
        SELECT e.FirstName, e.LastName, d.[Name]
        FROM Employees AS e
        LEFT JOIN Departments AS d ON d.DepartmentID = e.DepartmentID)

    INSERT INTO @result SELECT FirstName, LastName, DepartmentName
      FROM Employees_CTE WHERE DepartmentName = @DepName
    RETURN
END
```

Execute Functions

- Functions are called using `schemaName.functionName`

```
SELECT [ProjectID],
       [StartDate],
       [EndDate],
       dbo.udf_ProjectDurationWeeks([StartDate],[EndDate])
    AS ProjectWeeks
  FROM [SoftUni].[dbo].[Projects]
```

Call the Function

ProjectID	StartDate	EndDate	ProjectWeeks
1	2016-09-01	2016-10-07	5
2	2016-10-01	2016-10-07	1
3	2015-10-07	NULL	52

Problem: Salary Level Function

- Write a function `ufn_GetSalaryLevel(@Salary MONEY)` that receives salary of an employee and returns the level of the salary
  - If salary is < 30000 return "Low"
  - If salary is between 30000 and 50000 (inclusive) returns "Average"
  - If salary is > 50000 return "High"

	FirstName	LastName	Salary	SalaryLevel
1	Guy	Gilbert	12500.00	Low
2	Kevin	Brown	13500.00	Low
3	Roberto	Tamburello	43300.00	Average
4	Rob	Walters	29800.00	Low
5	Thierry	D'Hers	25000.00	Low

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/1025#4

Solution: Salary Level Function

Function Name

```
CREATE FUNCTION
ufn_GetSalaryLevel(@Salary MONEY)
RETURNS NVARCHAR(10)
AS
BEGIN
  -- Function logic here
END;
```

Input Parameters

Return Type

Function Body

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/1025#4

Solution: Salary Level Function

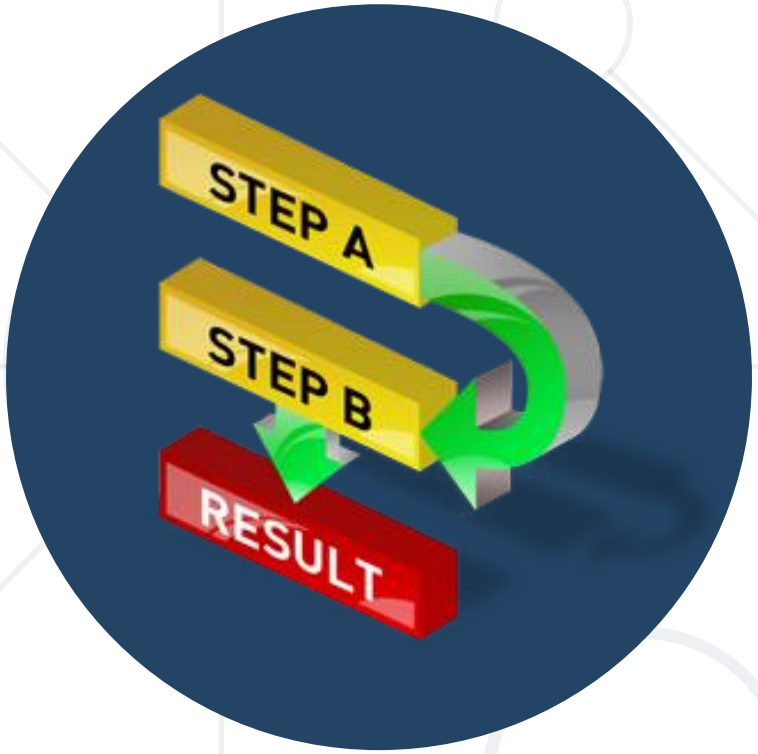
```
DECLARE @salaryLevel VARCHAR(10)

IF (@Salary < 30000)
    SET @salaryLevel = 'Low'
ELSE IF(@Salary <= 50000)
    SET @salaryLevel = 'Average'
ELSE
    SET @salaryLevel = 'High'

RETURN @salaryLevel
```

Variable

IF Statement

Return Result

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/1025#4



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

- **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

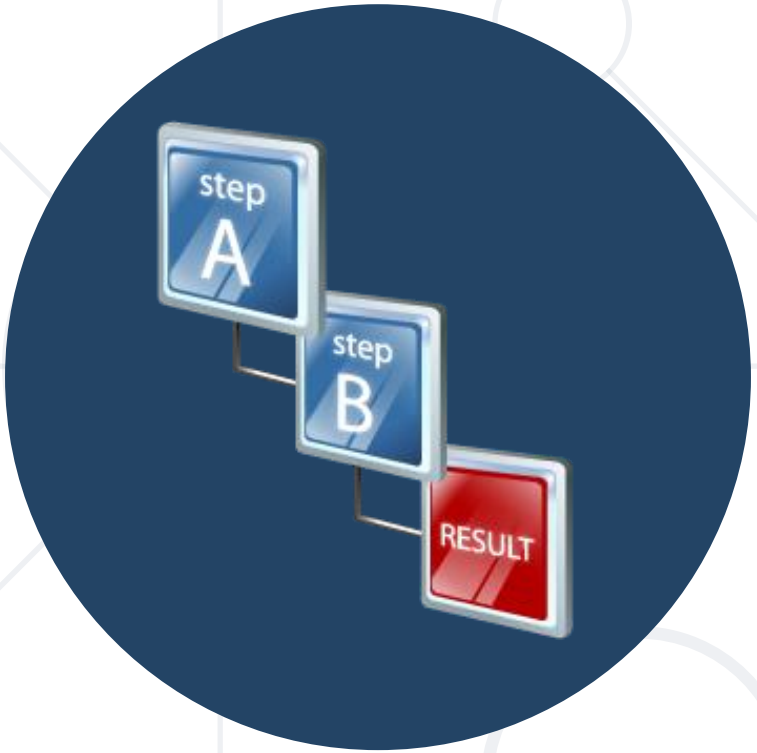
Dropping Stored Procedures

- 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 **appropriate default values**

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

Parameterized Stored Procedures - Example

Software University

```
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

- 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

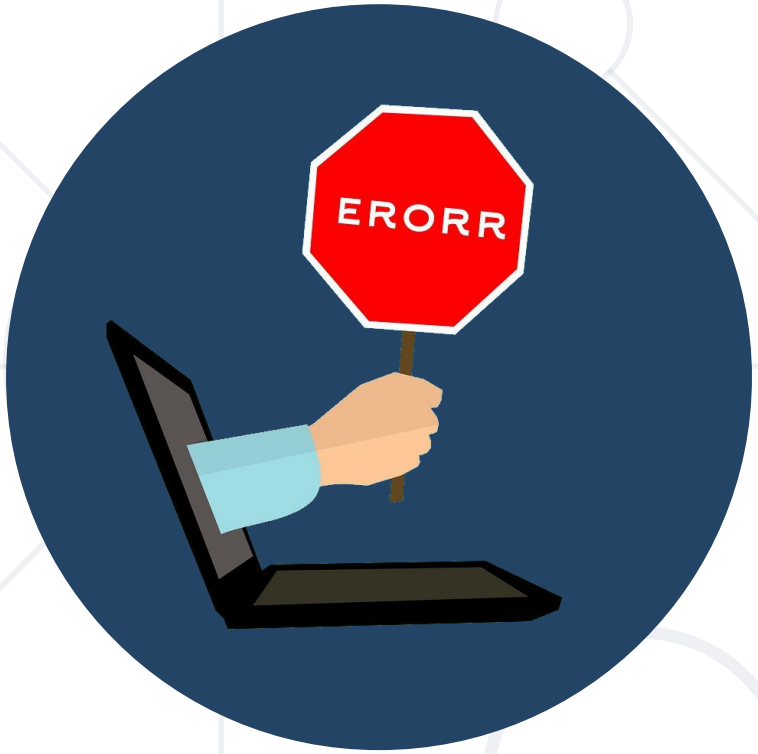
Display results

Returning Multiple Results

Checks if procedure exists and then Creates or Alters it

```
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
```

Multiple SELECT statements



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

Summary

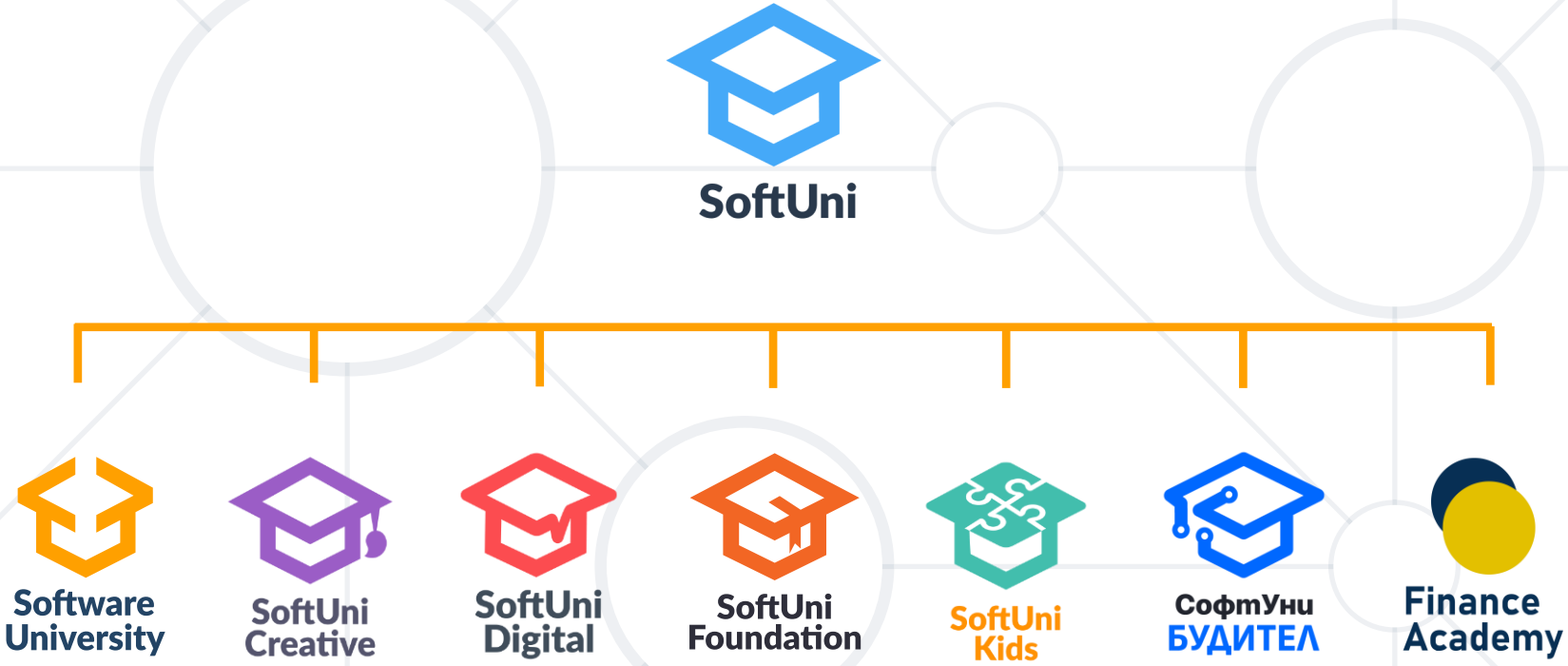
- **Functions** allow for complex calculations
  - Usually return a scalar value

```
CREATE FUNCTION f_ProcedureName
RETURNS …
AS
…
```

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

```
CREATE PROC usp_ProcedureName
AS …
```

Questions?



SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

top EMPLOYER България Bulgaria 2024 FOR A BETTER WORLD OF WORK

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

AMBITIONED

PHAR VISION

INDEAVR

Serving the high achievers

Trainings @ Software University (SoftUni)

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

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://about.softuni.bg
- © Software University – https://softuni.bg