Creating Our First Stored Procedure

Jared Westover
SQL ARCHITECT

@WestoverJared

Module Overview

Elements making up a stored procedure

- New create or alter syntax
- Dropping them
- Easily script them

Working with parameters

- Input and output
- Proper usage

Using the debug option in SSMS

- Stepping through the code

Implementing best practices

- Design your own

```
CREATE PROCEDURE Sales.SelectSalesPerson
AS
SELECT FirstName, LastName FROM Sales.SalesPerson;
GO
```

Syntax for Creating Stored Procedures

Don’t start the name with sp_

```
CREATE OR ALTER PROCEDURE Sales.SelectSalesPerson
AS
BEGIN
    SELECT FirstName, LastName FROM Sales.SalesPerson;
END
GO
```

New Syntax to Create or Alter

No longer need to check if the sproc exists

```
EXECUTE Sales.SelectSalesPerson;
GO;
EXEC Sales.SelectSalesPerson;
GO;
```

Executing Stored Procedures

Make sure to include the schema name

Naming Stored Procedures

Please don't start the name with sp_

- Reserved for system

Popular to use usp_ as the prefix

Typically naming is verb then noun

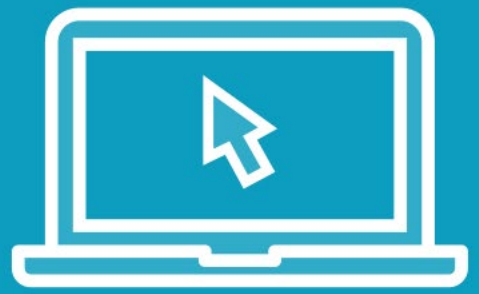
- InsertCustomer
- UpdateClient

Sometimes you will see primary table as prefix

- SalesPerson_InsertPerson

Most important thing is consistency

Demo

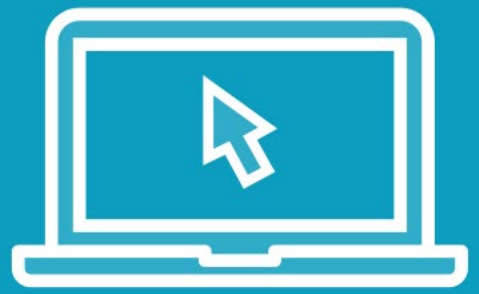


Creating our first stored procedure

- Selecting a sales person

Executing our stored procedure

Demo



Scripting our stored procedure

- Apply to another environment
- Save in git

Parameters

Passes data to our stored procedure

- Can have multiple values

Return data to our application

- Output
- Determine success
- Pass between stored procedures

Create local parameters for input

Name them properly

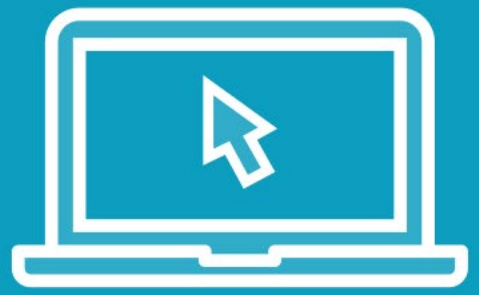
- @parameter1 or @a are bad

```
CREATE OR ALTER PROCEDURE Sales.SelectSalesPerson
@SalesPersonEmail nvarchar(250)
AS
SELECT FirstName, LastName FROM Sales.SalesPerson
WHERE Email = @SalesPersonEmail;
GO;
```

Adding Parameters

It's generally a good idea to use proper data typing on parameters

Demo



Passing in parameters to stored procedures

- Single and multiple values

Returning a single value from a stored procedure

Using Debug in SSMS

Similar to debug in Visual Studio

Ability to add break points

- Condition

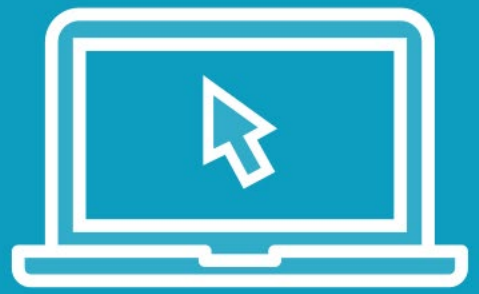
The local window

- Variable values

Only run on a test instance

- Sysadmin role required

Demo



Stepping through the debug

- Setting a break point

“Adapt what is useful, reject what is useless, and add what is specifically your own.”

Bruce Lee

Common Stored Procedure Best Practices

Set nocount on

Reduce network traffic

Sp_ prefix

Reserved for system sprocs

Schema name

Without schema checks master db

Select *

Pulling unnecessary columns into data set

Action name

Easily identify the purpose

What We Covered

How to create a stored procedure

- Create or alter statement
- Scripting stored procedures
- Dropping if exists

Implemented parameters

- Input and output

Using debug in SSMS

- Added break points
- Walked through code

Creating best practices

- Make them your own

Next Module: Setting up Queries in Stored Procedures