Stored procedures

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

SQL

Meghan Kwartler

IT Consultant

What is a stored procedure?

What?

Routines that

- Accept input parameters

- Perform actions (`EXECUTE` `SELECT` , `INSERT` , `UPDATE` , `DELETE` , and other SP statements)

- Return status (success or failure)

- Return output parameters

Why use stored procedures?

Why?

- Can reduce execution time

- Can reduce network traffic

- Allow for Modular Programming

- Improved Security

What's the difference?

UDFs

- Must return value
  - Table-valued allowed

- Embedded `SELECT` execute allowed

- No output parameters

- No `INSERT` , `UPDATE` , `DELETE`

- Cannot execute SPs

- No Error Handling

SPs

- Return value optional
  - No table-valued

- Cannot embed in `SELECT` to execute

- Return output parameters & status

- `INSERT` , `UPDATE` , `DELETE` allowed

- Can execute functions & SPs

- Error Handling with `TRY...CATCH`

CREATE PROCEDURE with OUTPUT parameter

```sql
-- First four lines of code

-- SP name must be unique

CREATE PROCEDURE dbo.cuspGetRideHrsOneDay
    @DateParm date,
    @RideHrsOut numeric OUTPUT
AS

.......
```

CREATE PROCEDURE with OUTPUT parameter

```sql
CREATE PROCEDURE dbo.cuspGetRideHrsOneDay
    @DateParm date,
    @RideHrsOut numeric OUTPUT
AS
SET NOCOUNT ON
BEGIN
SELECT
  @RideHrsOut = SUM(
    DATEDIFF(second, PickupDate, DropoffDate)
  )/ 3600
FROM YellowTripData
WHERE CONVERT(date, PickupDate) = @DateParm
RETURN
END;
```

Output parameters vs. return values

Output parameters

- Can be any data type

- Can declare multiple per SP

- Cannot be table-valued parameters

Return value

- Used to indicate success or failure

- Integer data type only

- 0 indicates success and non zero indicates failure

You're ready to CREATE PROCEDUREs!

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

datacamp

Oh CRUD!

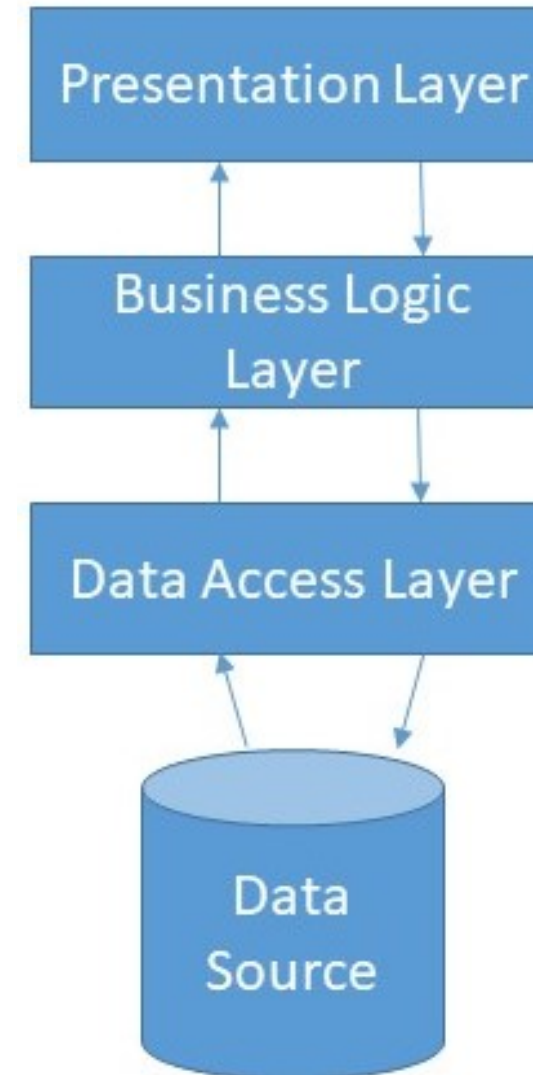## WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

SQL

Meghan Kwartler
IT Consultant

Why stored procedures for CRUD?

- Decouples SQL code from other application layers

N-Tier Architecture

Presentation Layer

Business Logic Layer

Data Access Layer

Data Source
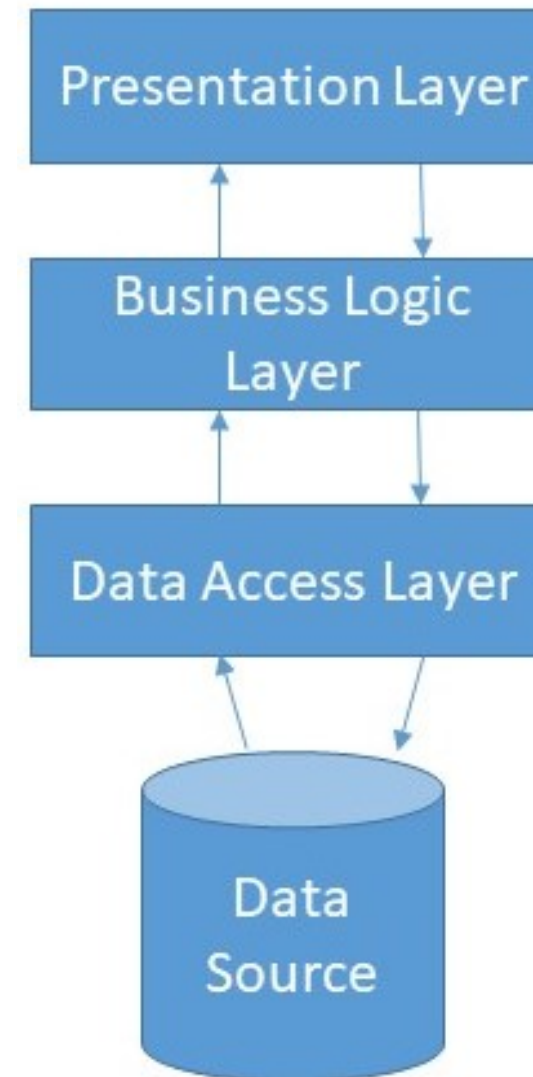
Why stored procedures for CRUD?

- Decouples SQL code from other application layers

- **Improved security**

N-Tier Architecture

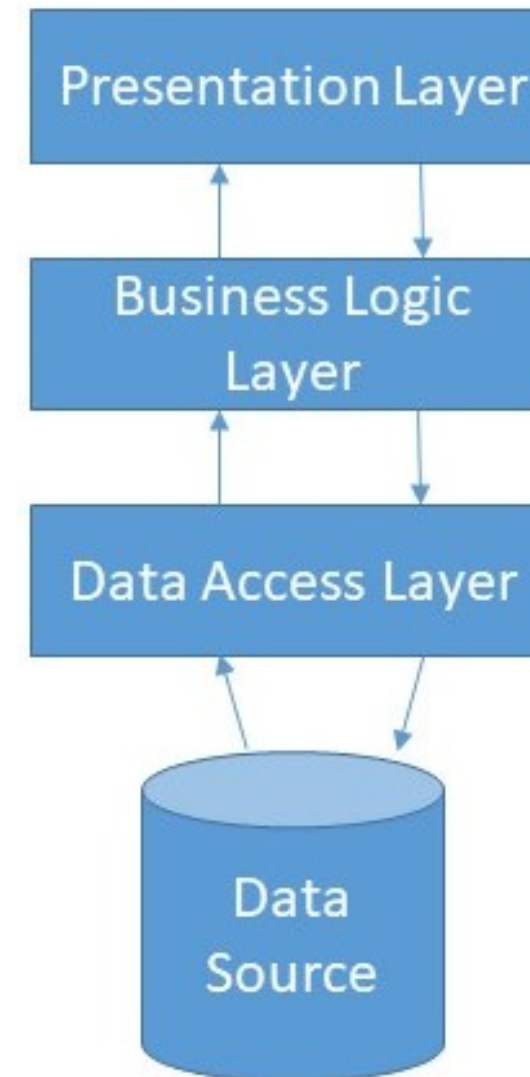Presentation Layer

Business Logic Layer

Data Access Layer

Data Source

Why stored procedures for CRUD?

- Decouples SQL code from other application layers

- Improved security

- **Performance**

N-Tier Architecture

C for CREATE

```sql
CREATE PROCEDURE dbo.cusp_TripSummaryCreate (
  @TripDate as date,
  @TripHours as numeric(18, 0)
) AS BEGIN INSERT INTO dbo.TripSummary(Date, TripHours)
VALUES
  (@TripDate, @TripHours)
SELECT Date, TripHours
FROM dbo.TripSummary
WHERE Date = @TripDate END
```

R for READ

```sql
CREATE PROCEDURE cusp_TripSummaryRead
    (@TripDate as date)
AS
BEGIN
SELECT Date, TripHours
FROM TripSummary
WHERE Date = @TripDate
END;
```

U for UPDATE

```sql
CREATE PROCEDURE dbo.cusp_TripSummaryUpdate
    (@TripDate as date,
    @TripHours as numeric(18,0))
AS
BEGIN
UPDATE dbo.TripSummary
SET Date = @TripDate,
    TripHours = @TripHours
WHERE Date = @TripDate
END;
```

D for DELETE

```sql
CREATE PROCEDURE cusp_TripSummaryDelete
    (@TripDate as date,
     @RowCountOut int OUTPUT)
AS
BEGIN
DELETE
FROM TripSummary
WHERE Date = @TripDate

SET @RowCountOut = @@ROWCOUNT
END;
```

Your turn for CRUD!

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

Let's EXEC!

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

SQL

Meghan Kwartler
IT Consultant

Ways to EXECute

- No output parameter or return value

- Store return value

- With output parameter

- With output parameter & store return value

- Store result set

No output parameter or return value

```
EXEC dbo.cusp_TripSummaryUpdate
    @TripDate = '1/5/2017'
    @TripHours = '300'
```

With output parameter

```
DECLARE @RideHrs as numeric(18,0)

EXEC dbo.cuspSumRideHrsOneDay
    @DateParm = '1/5/2017',
    @RideHrsOut = @RideHrs OUTPUT
```

```
SELECT @RideHrs as TotalRideHrs
```

```
+---------------+
| TotalRideHrs  |
|---------------+
| 77733         |
+---------------+
```

With return value

```
Declare @ReturnValue as int

EXEC @ReturnValue =
     dbo.cusp_TripSummaryUpdate
     @TripDate = '1/5/2017',
     @TripHours = 300

Select @ReturnValue as ReturnValue
```

```
+-------------+
| ReturnValue |
|-------------+
| 0           |
+-------------+
```

With return value & output parameter

```
Declare @ReturnValue as int
Declare @RowCount as int

EXEC @ReturnValue =
     dbo.cusp_TripSummaryDelete
     @TripDate = '1/5/2017',
     @RowCountOut = @RowCount OUTPUT

Select @ReturnValue as ReturnValue,
       @RowCount as RowCount
```

```
+-------------+----------+
| ReturnValue | RowCount |
|-------------+----------|
| 0           | 1        |
+-------------+----------|
```

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

EXEC & store result set

```sql
DECLARE @TripSummaryResultSet as TABLE(
        TripDate date,
        TripHours numeric(18,0))
INSERT INTO @TripSummaryResultSet
EXEC cusp_TripSummaryRead @TripDate = '1/5/2017'
SELECT * FROM @TripSummaryResultSet
```

```
+------------+-----------+
| TripDate   | TripHours |
|------------+-----------|
| 2017-01-05 | 200       |
+------------+-----------|
```

Time to EXEC your SPs!

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

TRY & CATCH those errors!

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

SQL

Meghan Kwartler
IT Consultant

To handle errors or not

What is error handling?

- Anticipation, detection and resolution of errors

- Maintains normal flow of execution

- Integrated into initial design

What happens without error handling?

- Sudden shut down or halts execution

- Generic error messages without helpful context are provided

Let's TRY

```sql
ALTER PROCEDURE dbo.cusp_TripSummaryCreate
  @TripDate nvarchar(30),
  @RideHrs numeric,
  @ErrorMsg nvarchar(max) = null OUTPUT
AS
BEGIN
  BEGIN TRY
      INSERT INTO TripSummary (Date, TripHours)
      VALUES (@TripDate, @RideHrs)
  END TRY
.........
```

Time to CATCH

```sql
ALTER PROCEDURE dbo.cusp_TripSummaryCreate
  @TripDate nvarchar(30),
  @RideHrs numeric,
  @ErrorMsg nvarchar(max) = null OUTPUT
AS
BEGIN
  BEGIN TRY
      INSERT INTO TripSummary (Date, TripHours)
      VALUES (@TripDate, @RideHrs)
  END TRY
  BEGIN CATCH
      SET @ErrorMsg = 'Error_Num: ' +
      CAST (ERROR_NUMBER() AS varchar) +
      ' Error_Sev: ' +
      CAST(ERROR_SEVERITY() AS varchar) +
      ' Error_Msg: ' + ERROR_MESSAGE()
  END CATCH
END
```

Show me the ERROR...

```sql
DECLARE @ErrorMsgOut as nvarchar(max)

EXECUTE dbo.cusp_TripSummaryCreate
        @TripDate = '1/32/2018',
        @RideHrs = 100,
        @ErrorMsg = @ErrorMsgOUT OUTPUT

SELECT @ErrorMsgOut as ErrorMessag
```

```
ErrorMessage
--------------------------------------------------

Error_Num: 241 Error_Sev: 16
Error_Msg: Conversion failed when converting date and/or time from
        character string
```

THROW vs RAISERROR

THROW

- Introduced in SQL Server 2012

- Simple & easy to use

- Statements following will **NOT** be executed

RAISERROR

- Introduced in SQL Server 7.0

- Generates new error and cannot access details of original error (e.g. line number where error originally occurred)

- Statements following can be executed

Your turn to CATCH!

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

datacamp