# Advanced SQL Server

Authored by : Sushant Banerjee
Email : sushantba@cybage.com

Presented by : Sushant Banerjee
Extn. 7221

# Agenda

- Stored Procedures
  - Introduction to SPs
  - Types of SPs
  - Parameters in SPs
- User-defined Functions
  - Introduction to UDFs
  - Types of UDFs
- Triggers
- Transactions

# Stored Procedures

# What is a Stored Procedure

- A stored procedure is similar to procedures in any programming language.
- A stored procedure contains a set of T-SQL programming statements that is stored as a permanent object in the database.
- A stored procedure accepts input parameters and returns multiple values in the form of output parameters.

# Benefits of Stored Procedures

- Allow Modular Programming
- Allow Faster Execution
- Reduce Network Traffic
- Apply Security

# Stored Procedure Types

- System Stored Procedures
- User-defined Stored Procedures
- Extended Stored Procedures

# Creating Procedures

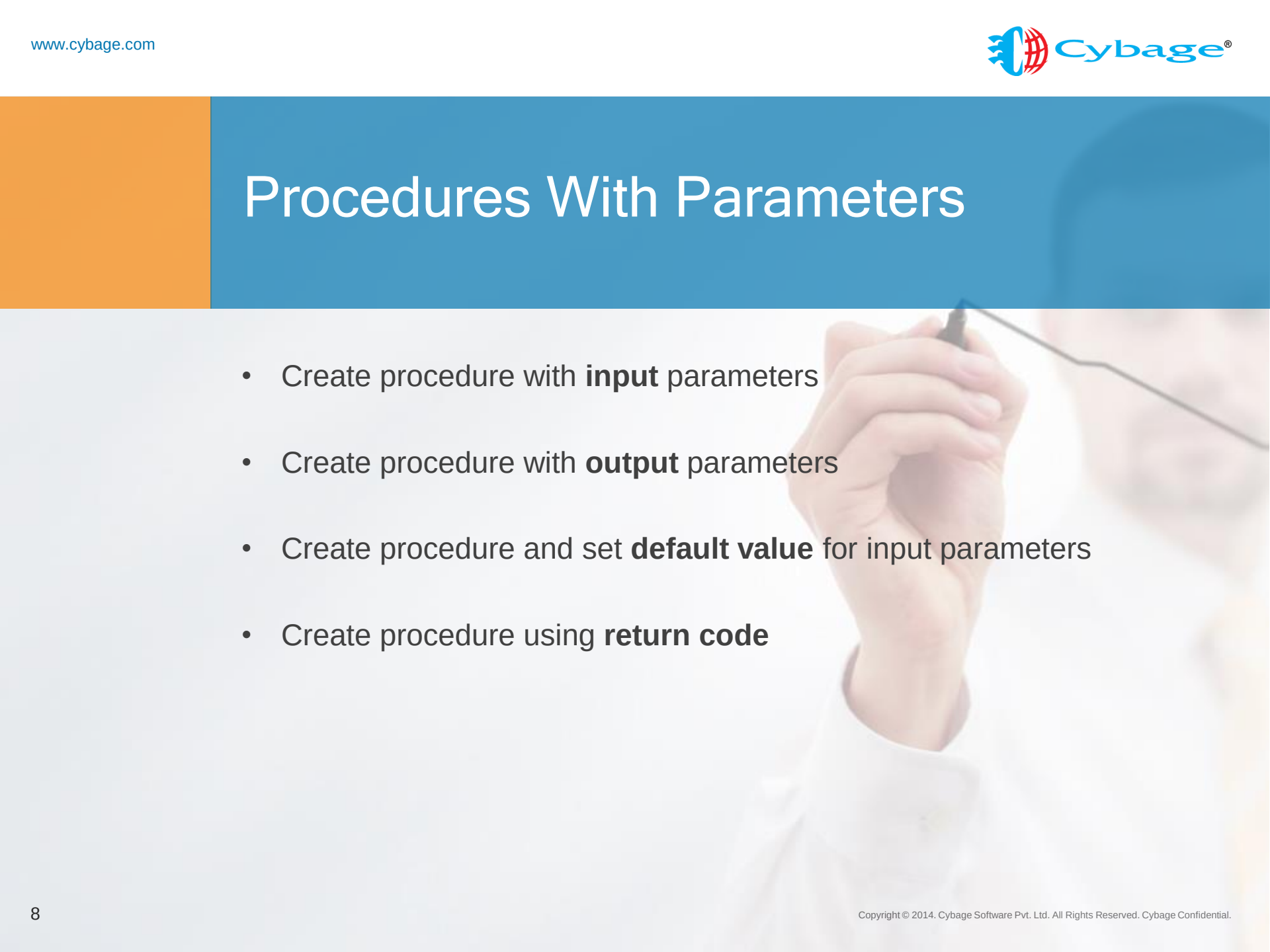
Syntax :

```
--Creating simple stored procedure
CREATE PROCEDURE procedureName
AS
--Write your T-SQL Statements here…

--Calling stored procedures
EXECUTE procedureName
```

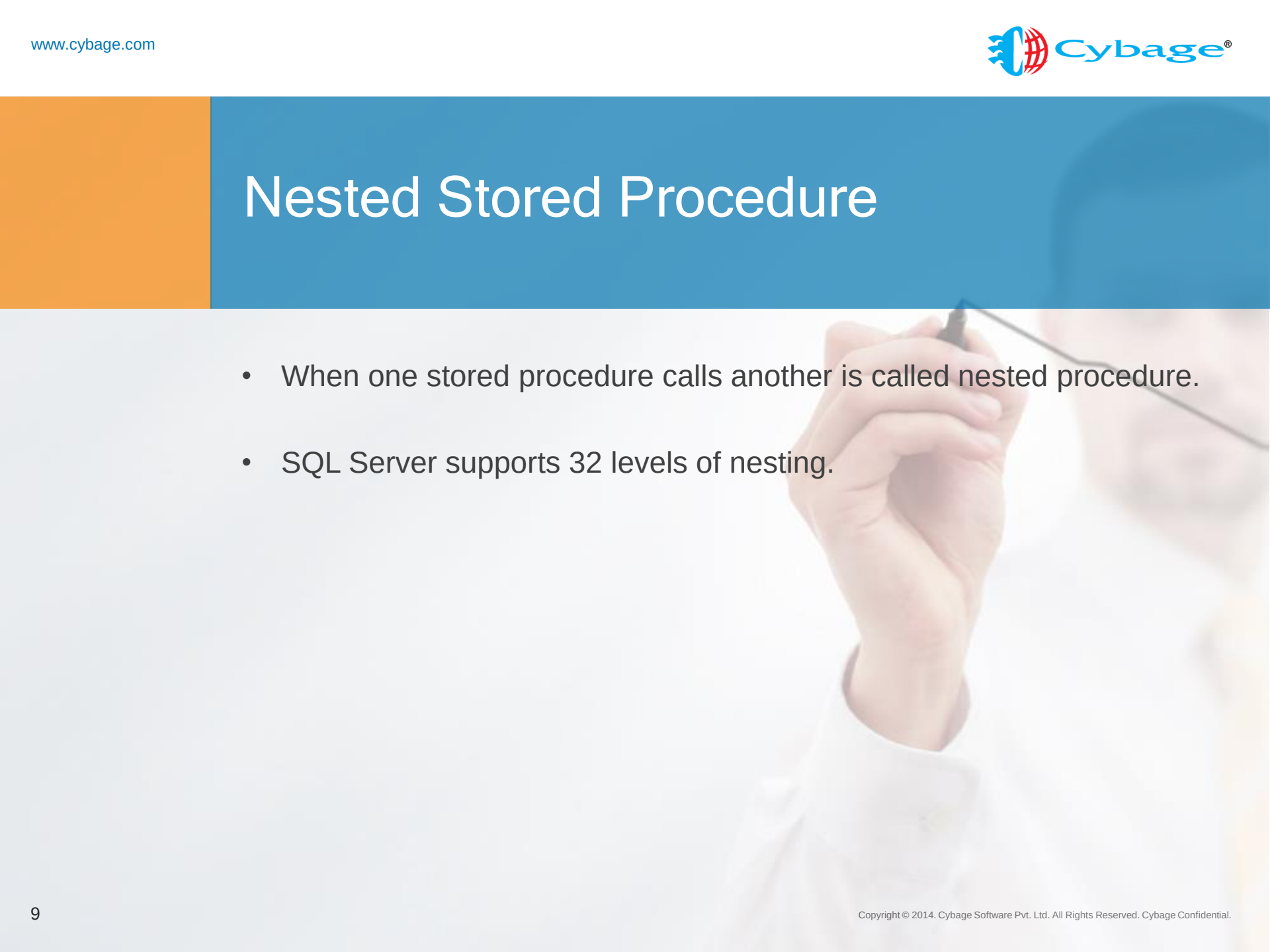
# Procedures With Parameters

- Create procedure with **input** parameters
- Create procedure with **output** parameters
- Create procedure and set **default value** for input parameters
- Create procedure using **return code**



# Nested Stored Procedure

- When one stored procedure calls another is called nested procedure.
- SQL Server supports 32 levels of nesting.



# Modify and Delete Procedure

- ALTER PROCEDURE
- DROP PROCEDURE

# Error Handling

TRY…CATCH

**BEGIN TRY**

--Write code that may raise error

**END TRY**

**BEGIN CATCH**

--Handle Error raised in the TRY block

**END CATCH**

# Error Functions

- ERROR_NUMBER()
- ERROR_MESSAGE()
- ERROR_SEVERITY()
- ERROR_STATE()
- ERROR_LINE()
- ERROR_PROCEDURE()

# Summary

- Why stored procedure
- Creating stored procedure
- Passing parameters
- Handling errors

# User-Defined Functions

# What is User-Defined Functions

- User-Defined Functions
  - Can accept parameters
  - Process the request
  - Return a result
- The return value may be
  - A single scalar value or
  - A result set

# Benefits of UDF

- Allow Modular Programming
- Allow Faster Execution
- Reduce Network Traffic

# Types of UDF

- Scalar Functions :
  - Returns a single data value of the type defined in the RETURNS clause.
- Table-Valued Functions :
  - Returns a table data type, where the table is the result set of a SELECT statement.

# Creating a Scalar Function

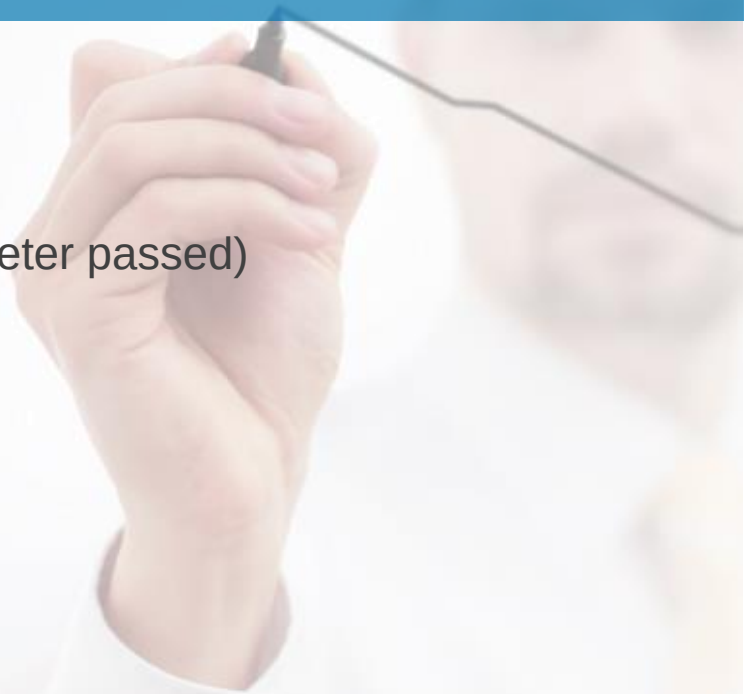
Syntax :

```
CREATE FUNCTION Schema.FunctionName (@parameter data type)
RETURNS return data type
AS
BEGIN
--Write logic here
 RETURN returnValue
END
```

# Calling a Scalar Function

Function Call Syntax :

SELECT Schema.FunctionName(parameter passed)



# Creating a Table-Valued Function

Syntax :

```
CREATE FUNCTION schema.FunctionName(@parameter data type)
RETURNS TABLE
AS
RETURN
(
SELECT statements…
);
```

# Calling a Table-Valued Function

Syntax :

SELECT * FROM schema.FunctionName(parameter value)

# Stored Procedure Vs. UDF

| Stored Procedure | User-Defined Functions |
|---|---|
| Have input and output parameters | Only input parameters |
| Can have 0 or more parameters | At least 1 parameter mandatory |
| Cannot be called from a UDF | Can be called from a SP |
| May or may not return values | Must return a value |
| Allows DML statements | DML statements not allowed |
| Allows TRY..CATCH | Doesn't allow TRY..CATCH |

# Summary

- Benefits of using UDF
- Types of UDF
- Difference between SP and UDF

# Triggers

# What is a Trigger

- There are two primary mechanism to enforce data integrity in SQL Server databases :
- Constraints
- Triggers

- A Trigger is special type of stored procedure
- A Trigger executes automatically
- Execution depends on a language event, for example an DML statements

# Types of Trigger

- DML Triggers
- DDL Triggers
- Logon Triggers

# DML Triggers

- DML Triggers are invoked automatically when a DML statement such as INSERT, UPDATE or DELETE is executed on a table or view.
- DML triggers can work just like constraint to enforce data integrity.
- DML triggers can be used to prevent invalid INSERT, UPDATE and DELETE operations.

# Creating DML Triggers

Syntax :

```
CREATE TRIGGER TriggerName
ON TableName
FOR INSERT, UPDATE, DELETE
AS
--Write your logic here
ROLLBACK
```

# Transactions

# Why Transactions

- Modifying multiple tables
- Executing multiple commands
- Part of the operation may fail or succeed
- Transaction helps us to control the operations.

# ACID Properties

- A Transaction is a sequence of operations performed as a single logical unit of work.
- This logical unit of work must have following four properties (ACID) :

- **Atomicity**
  - Enforced by Transaction Management Features
- **Consistency**
  - Enforced by Transaction Management Features
- **Isolation**
  - Enforced by Locking Facility
- **Durability**
  - Enforced by Logging Facility.

# Transaction Modes

- Auto commit Transactions
  - By default transactions are committed in SQL Server
- Explicit Transactions
  - Begin and Commit or Rollback transactions.

# Transaction Statements

- Begin Transaction
- Commit Transaction
- Rollback Transaction
- Save Transaction.

# Error Handling in Transactions

Syntax :

```
BEGIN TRY
   BEGIN TRANSACTION TransactionName
   --Write multiple INSERT, UPDATE, DELETE statements
   COMMIT TRANSACTION TransactionName
END TRY
BEGIN CATCH
   ROLLBACK TRANSACTION TransactionName
END CATCH
```

# Bibliography, Important Links

- http://msdn.microsoft.com/en-us/library/hh230827(v=sql.110).aspx
- https://msdn.microsoft.com/en-us/library/ms178110(v=sql.110).aspx

# Any Questions?

# Thank you!