

Module 9: Implementing Stored Procedures

Overview

- Introduction to Stored Procedures
 - Creating Executing Modifying Dropping
- Using Parameters in Stored Procedures
- Executing Extended Stored Procedures
- Handling Error Messages

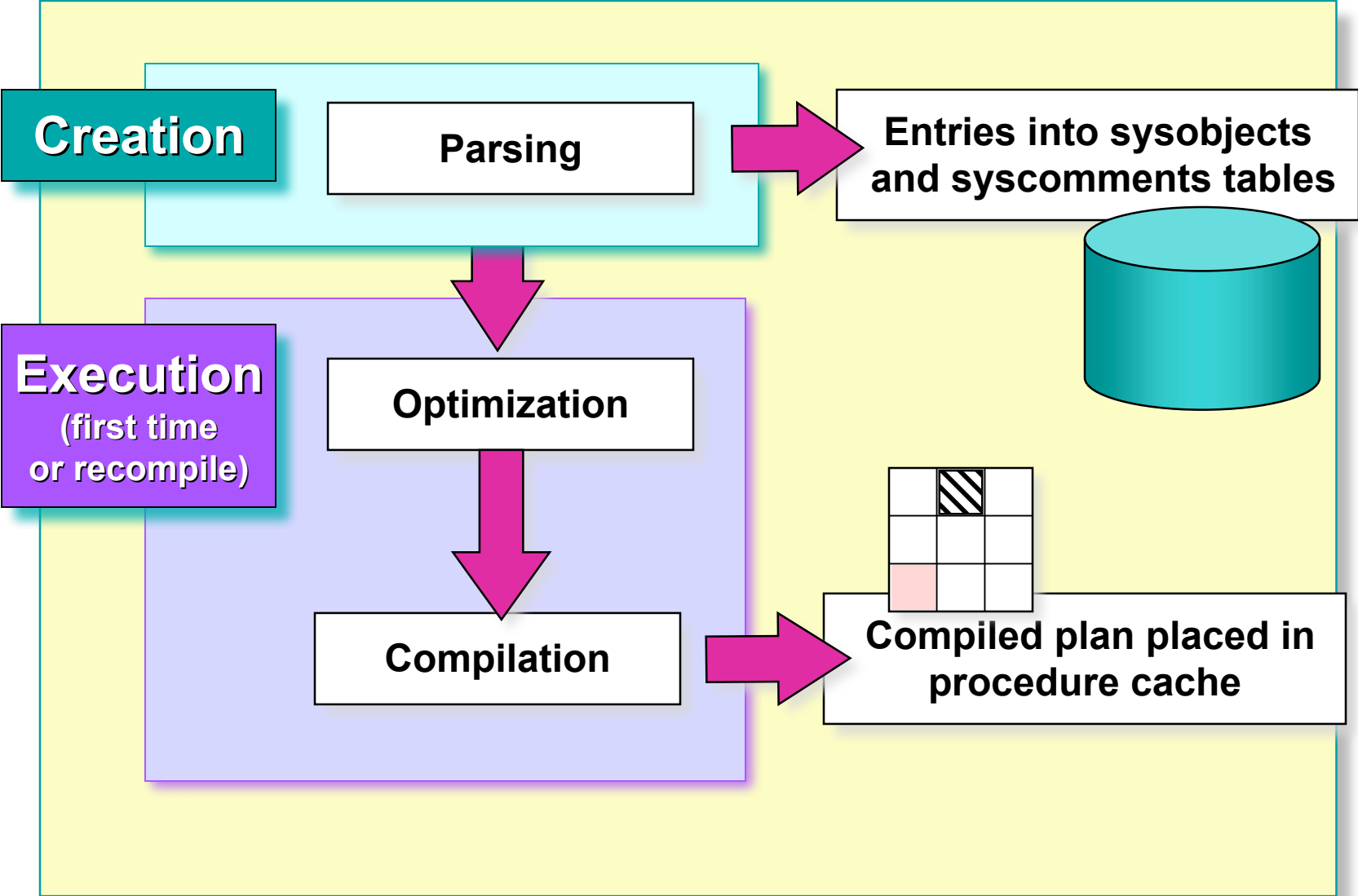
◆ Introduction to Stored Procedures

- Defining
- Initial Processing
- Subsequent Processing
- Advantages

Introduction: Defining Stored Procedures

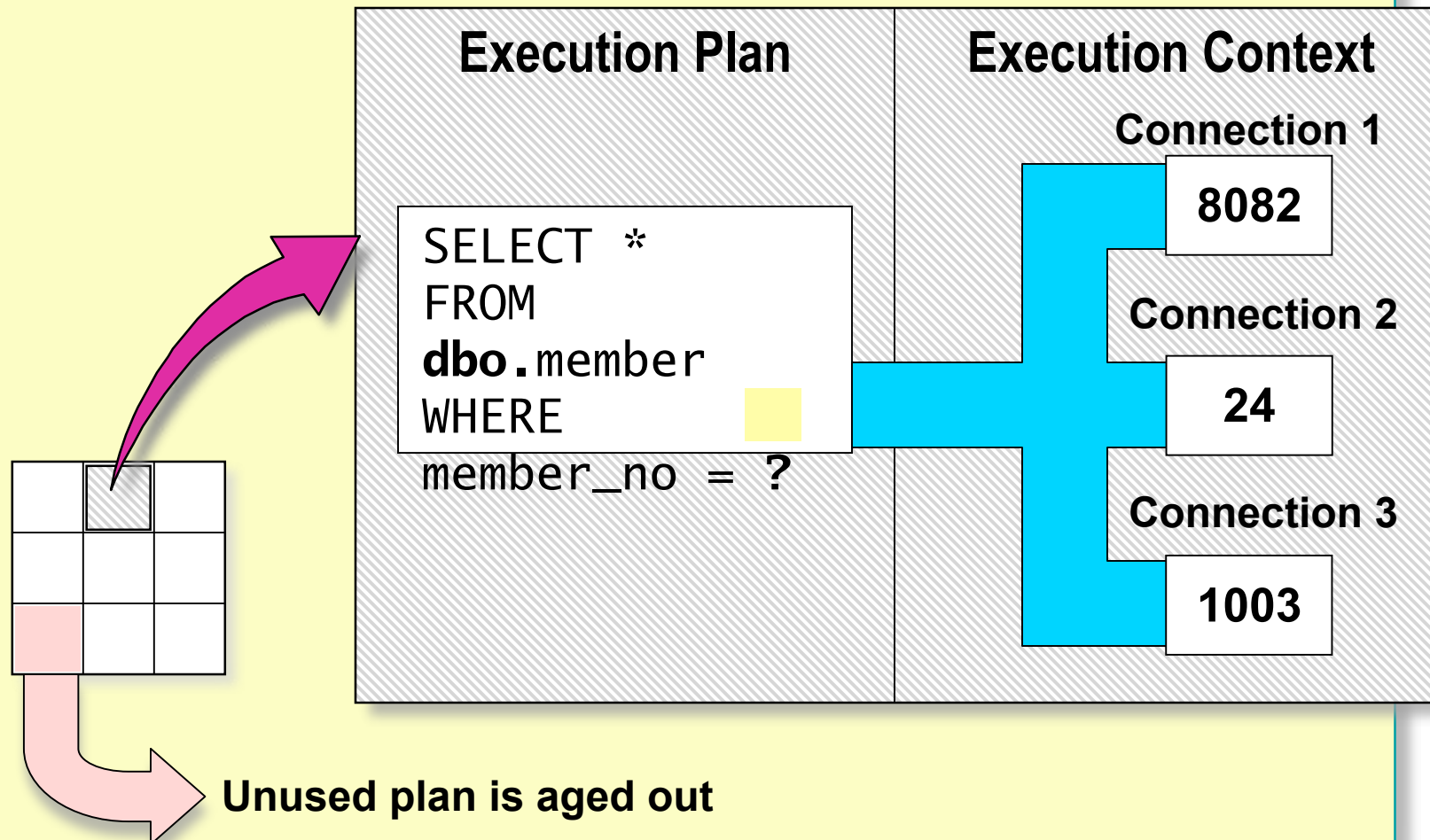
- **Named Collections of Transact-SQL Statements**
- **Accept Input Parameters and Return Values**
- **Return Status Value to Indicate Success or Failure**
- **Encapsulate Repetitive Tasks**
- **Five Types**
 - System, Local, Temporary, Remote, Extended

Introduction: Initial Processing of Stored Procedures



Introduction: Subsequent Processing of Stored Procedures

Execution Plan Retrieved



Introduction: Advantages of Stored Procedures

- **Share Application Logic**
- **Shield Database Schema Details**
- **Provide Security Mechanisms**
- **Improve Performance**
- **Reduce Network Traffic**

◆ **Creating, Executing, Modifying, and Dropping Stored Procedures**

- **Creating**
- **Guidelines for Creating**
- **Executing**
- **Altering and Dropping**

Creating Stored Procedures

- Create in Current Database Using the **CREATE PROCEDURE** (or **CREATE PROC**) Statement

```
USE Northwind
GO
CREATE PROC dbo.OverdueOrders
AS
    SELECT *
    FROM dbo.Orders
    WHERE RequiredDate < GETDATE() AND ShippedDate IS Null
GO
```

- Can Nest to 32 Levels
- Use **sp_help** to Display Information
 - **sp_help** <procedure name>

Guidelines for Creating Stored Procedures

- **dbo User Should Own All Stored Procedures**
 - E.g., `dbo.OverdueOrders`
- **One Stored Procedure \Leftrightarrow for \Rightarrow One Task**
- **Create, Test, and Troubleshoot**
- **Avoid `sp_` Prefix in Stored Procedure Names**
 - Used for system store procedures
- **Use Same Connection Settings for All Stored Procedures**
- **Minimize Use of Temporary Stored Procedures**
- **Never Delete Entries Directly From Syscomments**

Executing Stored Procedures

- Executing a Stored Procedure by Itself

```
EXEC OverdueOrders
```

- Executing a Stored Procedure Within an INSERT Statement

```
INSERT INTO Customers  
EXEC EmployeeCustomer
```

**Inserts the results
from the Query in
EmployeeCustomer**

Altering and Dropping Stored Procedures

■ Altering Stored Procedures

- Include any options in `ALTER PROCEDURE` (or `ALTER PROC`)
- Does not affect nested stored procedures

```
USE Northwind
GO
ALTER PROC dbo.OverdueOrders
AS
SELECT CONVERT(char(8), RequiredDate, 1) RequiredDate,
       CONVERT(char(8), OrderDate, 1) OrderDate,
       OrderID, CustomerID, EmployeeID
FROM Orders
WHERE RequiredDate < GETDATE() AND ShippedDate IS Null
ORDER BY RequiredDate
GO
```

- **Dropping** stored procedures
- Execute the `sp_depends` stored procedure to determine whether objects depend on the stored procedure

◆ Using Parameters in Stored Procedures

- Using Input Parameters
- Executing Using Input Parameters
- Returning Values Using Output Parameters
- Explicitly Recompiling

Using Input Parameters

- **Validate All Incoming Parameter Values First**
 - Highly recommended since testing and fixing is harder
- **Provide Appropriate Default Values and Include Null Checks**

```
CREATE PROCEDURE dbo.[Year to Year Sales]
    @BeginningDate DateTime, @EndingDate DateTime
AS
IF @BeginningDate IS NULL OR @EndingDate IS NULL
BEGIN
    RAISERROR('NULL values are not allowed', 14, 1)
    RETURN
END
SELECT O.ShippedDate,
       O.OrderID,
       OS.Subtotal,
       DATENAME(yy,ShippedDate) AS Year
FROM ORDERS O INNER JOIN [Order Subtotals] OS
    ON O.OrderID = OS.OrderID
WHERE O.ShippedDate BETWEEN @BeginningDate AND @EndingDate
GO
```

Executing Stored Procedures Using Input Parameters

■ Passing Values by Parameter Name

More robust but requires parameter names and tighter coordination between developers.

```
EXEC AddCustomer
  @CustomerID = 'ALFKI',
  @ContactName = 'Maria Anders',
  @CompanyName = 'Alfreds Futterkiste',
  @ContactTitle = 'Sales Representative',
  @Address = 'Obere Str. 57',
  @City = 'Berlin',
  @PostalCode = '12209',
  @Country = 'Germany',
  @Phone = '030-0074321'
```

■ Passing Values by Position

Less robust but supports "programming to interfaces."

```
EXEC AddCustomer 'ALFKI2', 'Alfreds
Futterkiste', 'Maria Anders', 'Sales
Representative', 'Obere Str. 57', 'Berlin',
NULL, '12209', 'Germany', '030-0074321'
```

Returning Values Using Output Parameters

Creating Stored Procedure

```
CREATE PROCEDURE dbo.MathTutor
    @m1 smallint,
    @m2 smallint,
    @result smallint OUTPUT
AS
    SET @result = @m1* @m2
GO
```

Executing Stored Procedure

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

Results of Stored Procedure

```
The result is: 30
```


Explicitly Recompiling Stored Procedures

■ Recompile When the Execution Plan Changes

- Stored procedure returns widely varying result sets
- A new index is added to an underlying table
- The parameter value is atypical

■ Recompile by Using

- CREATE PROCEDURE [WITH RECOMPILE]
- EXECUTE [WITH RECOMPILE]
- sp_recompile <procedure name>

Executing Extended Stored Procedures

- Are Programmed Using Open Data Services API
- Can Include C, C++, Java Features
- Can Contain Multiple Functions
- Can Be Called from a Client or SQL Server
- Can Be Added to the master Database Only

E.g., Execute a command in cmdshell.

```
EXEC master..xp_cmdshell 'dir c:\'
```

Handling Error Messages

- **RETURN Statement Exits Query or Procedure Unconditionally**
- **sp_addmessage Creates Custom Error Messages**
- **@@error Contains Error Number for Last Executed Statement**
- **RAISERROR Statement**
 - Returns user-defined or system error message
 - Sets system flag to record error

Performance Considerations






- **Windows 2000 System Monitor**

- Object: SQL Server: Cache Manager
- Object: SQL Statistics

- **SQL Profiler**

- Can monitor events
- Can test each statement in a stored procedure

Recommended Practices

-  **Verify Input Parameters**
-  **Design Each Stored Procedure to Accomplish a Single Task**
-  **Validate Data Before You Begin Transactions**
-  **Use the Same Connection Settings for All Stored Procedures**
-  **Use WITH ENCRYPTION to Hide Text of Stored Procedures**

Review

- Introduction to Stored Procedures
 - Creating Executing Modifying Dropping
- Using Parameters in Stored Procedures
- Executing Extended Stored Procedures
- Handling Error Messages