Module Summary

In this module, **Managing Views**, you learnt about:

- Viewing Information
- Modifying and Dropping
- Working with Stored Procedures

Click on each link for a summary of the lesson.

Working with Stored Procedures

Stored procedures can be nested; that is, a stored procedure can be called from within another stored procedure. In stored procedures, error handling is done using the `TRY...CATCH` construct.

Module Overview

Welcome to the module, **More About Stored Procedures**. Stored procedures are created to access and manipulate database objects in an efficient manner. The definition of a stored procedure as well as its dependencies can be viewed using certain system stored procedures. Values can be passed between a stored procedure and the calling program using input and output parameters.

In this module, you will learn about:

- Viewing Information
- Modifying and Dropping Stored Procedures
- Working with Stored Procedures

Towards the end of the module, there are demonstrations and/ or simulations for reinforcing the theoretical concepts.

Lesson Overview

In this first lesson, **Viewing Information**, you will learn to:

- Describe how to view definitions of stored procedures.
- State how to view dependencies of stored procedures.

Viewing Definitions

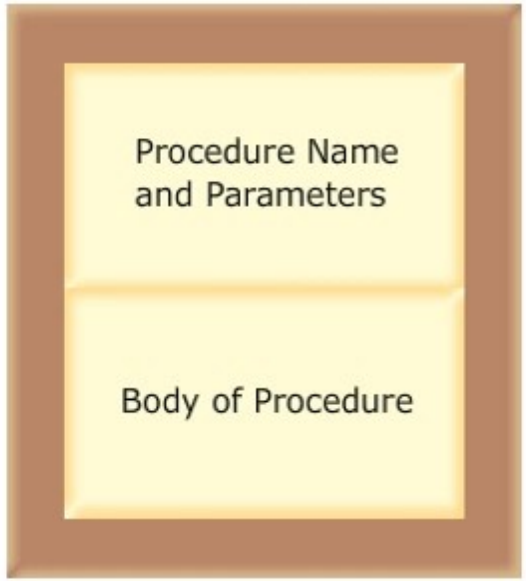
A stored procedure is defined using a set of Transact-SQL statements. This definition consists of two parts:

- Name, input and output parameters of the stored procedures.
- Body of the stored procedure.

You can view the definition of a stored procedure using one of the following:

- `sp_helptext` system stored procedure
- `sys.sql_modules` system view
- `OBJECT_DEFINITION` function

Stored Procedure Definition



Viewing Definitions Using "sp_helptext"

The definition of a stored procedure can be viewed using the `sp_helptext` system stored procedure. To view the definition, you must specify the name of the stored procedure as the parameter when executing `sp_helptext`. This definition is in the form of Transact-SQL statements.

The Transact-SQL statements of the procedure definition include the `CREATE PROCEDURE` statement as well as the SQL statements that define the body of the procedure.

```
1  sp_helptext 'Display_Customers'
```

Results | Messages

	Text
1	CREATE PROCEDURE Display_Customers
2	@firstname varchar(30) = '%'
3	AS
4	SELECT FirstName, LastName, DateofBirth, City
5	FROM dbo.Employee_Details
6	WHERE FirstName LIKE @firstname

Syntax

Snippet

More

Viewing Definitions Using "sp_helptext"

The definition of a stored procedure can be viewed using the `sp_helptext` system stored procedure. To view the definition, you must specify the name of the stored procedure as the parameter when executing `sp_helptext`. This definition is in the form of

The following syntax is used to view the definition of a stored procedure.

```
sp_helptext '<procedure_name>'
```

where,
`procedure_name`: Specifies the name of the procedure.

Syntax

```
1  sp_helptext 'Display_Customers'
```

Results | Messages

	Text
1	CREATE PROCEDURE Display_Customers
2	@firstname varchar(30) = '%'
3	AS
4	SELECT FirstName, LastName, DateofBirth, City
5	FROM dbo.Employee_Details
6	WHERE FirstName LIKE @firstname

Syntax | Snippet | More

Viewing Definitions Using "sp_helptext"

The definition of a stored procedure can be viewed using the `sp_helptext` system stored procedure. To view the definition, you must specify the name of the stored procedure as the parameter when executing `sp_helptext`. This definition is in the form of

The following code displays the definition of the stored procedure named `Display_Customers`.

```
sp_helptext 'Display_Customers'
```

Snippet

```
1  sp_helptext 'Display_Customers'
```

Results | Messages

	Text
1	CREATE PROCEDURE Display_Customers
2	@firstname varchar(30) = '%'
3	AS
4	SELECT FirstName, LastName, DateofBirth, City
5	FROM dbo.Employee_Details
6	WHERE FirstName LIKE @firstname

Syntax | Snippet

More

Viewing Definitions Using "sp_helptext"

The definition of a stored procedure can be viewed using the `sp_helptext` system stored procedure. To view the definition, you must specify the name of the stored procedure as the parameter when executing `sp_helptext`. This definition is in the form of Transact-SQL statements.

The Transact-SQL statements of the procedure definition include the `CREATE PROCEDURE` statement as well as the SQL statements that define the body of the procedure.

```
1  sp_helptext 'Display_Customers'
```

Results | Messages

	Text
1	CREATE PROCEDURE Display_Customers
2	@firstname varchar(30) = '%'
3	AS
4	SELECT FirstName, LastName, DateofBirth, City
5	FROM dbo.Employee_Details
6	WHERE FirstName LIKE @firstname

Syntax

Snippet

More

More

The definition of stored procedures can be encrypted to prevent it from being viewed.

Viewing Definitions using "sys.sql_modules" and "OBJECT_DEFINITION"

The definition of a stored procedure can also be viewed using the `sys.sql_modules` system view and `OBJECT_DEFINITION` system function. When the ID of the procedure is specified as a parameter when calling the `OBJECT_DEFINITION` function, the procedure definition is displayed.

```
1  SELECT OBJECT_DEFINITION (OBJECT_ID('Display_Customers'))
2  AS Definition
```

Results | Messages

	Definition
1	CREATE PROCEDURE Display_Customers @firstname varchar(30) = '%' AS SELECT FirstName, ...

Syntax

Snippet

Viewing Definitions using "sys.sql_modules" and "OBJECT_DEFINITION"

The definition of a stored procedure can also be viewed using the `sys.sql_modules` system view and `OBJECT_DEFINITION` system function. When the ID of the procedure is specified as a parameter when calling the `OBJECT_DEFINITION`

```
1  SELECT OBJECT_DEFINITION (OBJECT_ID('Display_Customers'))
2  AS Definition
```

Results | Messages

Definition

...ROCEDURE Display_Customers @firstname varchar(30) = '%' AS SELECT FirstName, ...

The following syntax is used to view the definition of a stored procedure using the `OBJECT_DEFINITION` system function.

```
OBJECT_DEFINITION (<object_id>)
```

where,
`object_id`: Specifies the ID of the stored procedure.

Syntax

Syntax | Snippet

Viewing Definitions using "sys.sql_modules" and "OBJECT_DEFINITION"

The definition of a stored procedure can also be viewed using the `sys.sql_modules` system view and `OBJECT_DEFINITION` system function. When the ID of the procedure is specified as a parameter when calling the `OBJECT_DEFINITION` function, the procedure definition is displayed.

Syntax

Snippet

The following code displays definition of the `Display_Customers` stored procedure:

```
SELECT OBJECT_DEFINITION (OBJECT_ID('Display_Customers'))
AS Definition
```

where,

`OBJECT_ID`: Is the system function that returns ID of the specified object (in this case, `Display_Customers` stored procedure).

Also, the definition of the stored procedure can be viewed using the `sys.sql_modules` system view as shown in the following code:

```
SELECT definition FROM sys.sql_modules
WHERE object_id = OBJECT_ID('Display_Customers')
```

where,

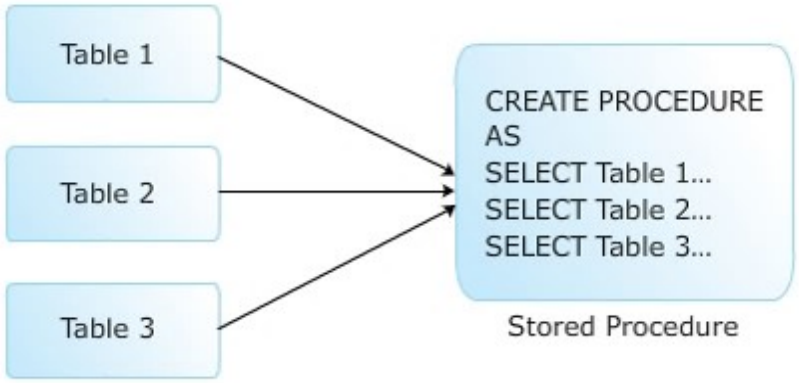
`definition`: Is the column in the `sys.sql_modules` system view that displays the definition of the specified object (in this case, `Display_Customers` stored procedure).

Snippet

Viewing Dependencies

A stored procedure can have reference to other database objects during its execution. Similarly, other database objects can have reference to the stored procedure. Both these sets of database objects are referred to as dependencies of the stored procedure.

When changes are made to a stored procedure, appropriate changes need to be made to the dependencies to ensure proper execution of queries. For example, if you change the name of a stored procedure, the dependent objects may fail during execution. This happens because these objects are still referencing the procedure by using the old name.

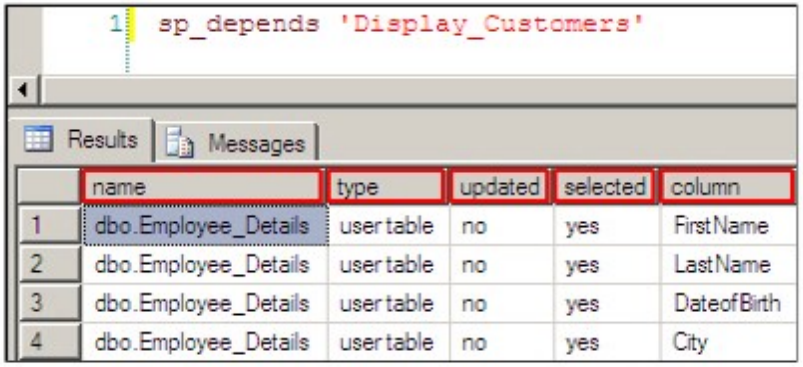


"sp_depends" Stored Procedure

In SQL Server 2005, the dependencies of a stored procedure can be viewed using the `sp_depends` system stored procedure. To view the dependencies, the stored procedure name has to be specified as a parameter when executing `sp_depends`.

When `sp_depends` is executed, it returns a list of all objects that reference the specified procedure as well as a list of all objects that the procedure references.

Click on each highlighted area on the image to learn more.



	name	type	updated	selected	column
1	dbo.Employee_Details	user table	no	yes	FirstName
2	dbo.Employee_Details	user table	no	yes	LastName
3	dbo.Employee_Details	user table	no	yes	DateofBirth
4	dbo.Employee_Details	user table	no	yes	City

Syntax

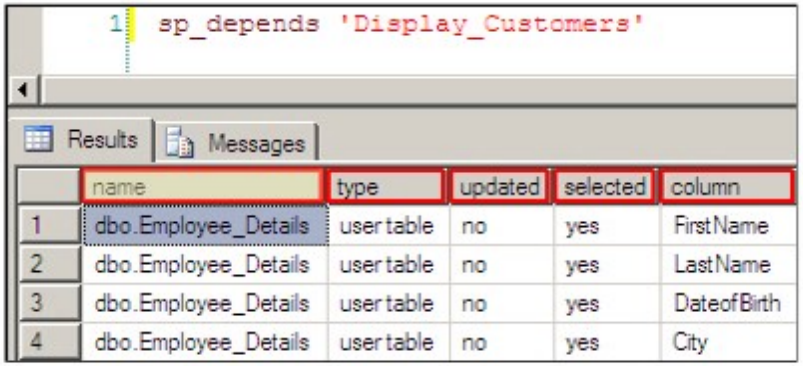
Snippet

"sp_depends" Stored Procedure

In SQL Server 2005, the dependencies of a stored procedure can be viewed using the `sp_depends` system stored procedure. To view the dependencies, the stored procedure name has to be specified as a parameter when executing `sp_depends`.

When `sp_depends` is executed, it returns a list of all objects that reference the specified procedure as well as a list of all objects that the procedure references.

Click on each highlighted area on the image to learn more.



	name	type	updated	selected	column
1	dbo.Employee_Details	user table	no	yes	FirstName
2	dbo.Employee_Details	user table	no	yes	LastName
3	dbo.Employee_Details	user table	no	yes	DateofBirth
4	dbo.Employee_Details	user table	no	yes	City

name

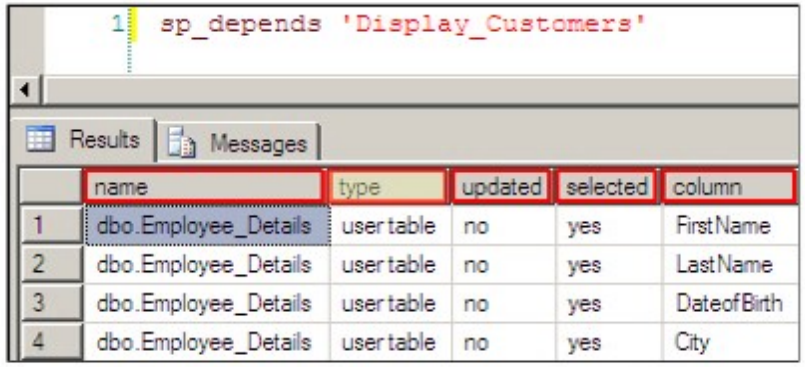
Specifies the name of the entity for which a dependency exists.

"sp_depends" Stored Procedure

In SQL Server 2005, the dependencies of a stored procedure can be viewed using the `sp_depends` system stored procedure. To view the dependencies, the stored procedure name has to be specified as a parameter when executing `sp_depends`.

When `sp_depends` is executed, it returns a list of all objects that reference the specified procedure as well as a list of all objects that the procedure references.

Click on each highlighted area on the image to learn more.



	name	type	updated	selected	column
1	dbo.Employee_Details	user table	no	yes	FirstName
2	dbo.Employee_Details	user table	no	yes	LastName
3	dbo.Employee_Details	user table	no	yes	DateofBirth
4	dbo.Employee_Details	user table	no	yes	City

type

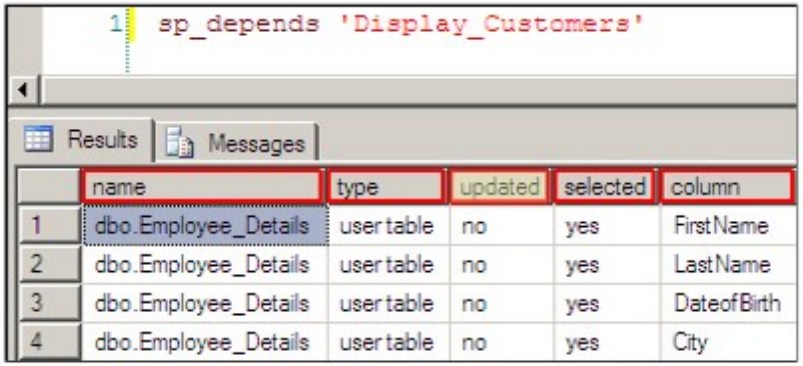
Specifies the type of the entity.

"sp_depends" Stored Procedure

In SQL Server 2005, the dependencies of a stored procedure can be viewed using the `sp_depends` system stored procedure. To view the dependencies, the stored procedure name has to be specified as a parameter when executing `sp_depends`.

When `sp_depends` is executed, it returns a list of all objects that reference the specified procedure as well as a list of all objects that the procedure references.

Click on each highlighted area on the image to learn more.



```
1  sp_depends 'Display_Customers'
```

Results | Messages

	name	type	updated	selected	column
1	dbo.Employee_Details	user table	no	yes	FirstName
2	dbo.Employee_Details	user table	no	yes	LastName
3	dbo.Employee_Details	user table	no	yes	DateofBirth
4	dbo.Employee_Details	user table	no	yes	City

updated

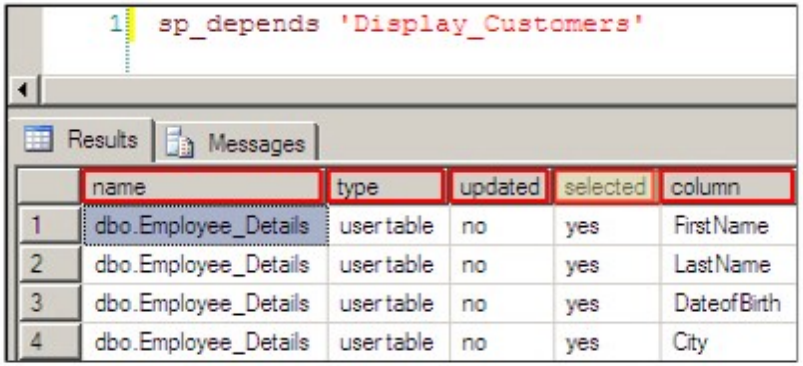
Specifies whether or not the entity has been updated.

"sp_depends" Stored Procedure

In SQL Server 2005, the dependencies of a stored procedure can be viewed using the `sp_depends` system stored procedure. To view the dependencies, the stored procedure name has to be specified as a parameter when executing `sp_depends`.

When `sp_depends` is executed, it returns a list of all objects that reference the specified procedure as well as a list of all objects that the procedure references.

Click on each highlighted area on the image to learn more.



```
1  sp_depends 'Display_Customers'
```

Results | Messages

	name	type	updated	selected	column
1	dbo.Employee_Details	user table	no	yes	FirstName
2	dbo.Employee_Details	user table	no	yes	LastName
3	dbo.Employee_Details	user table	no	yes	DateofBirth
4	dbo.Employee_Details	user table	no	yes	City

selected

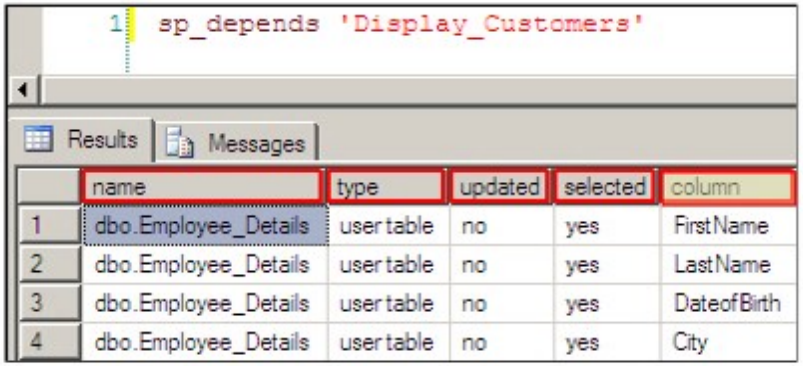
Specifies whether the entity has been used in the `SELECT` statement.

"sp_depends" Stored Procedure

In SQL Server 2005, the dependencies of a stored procedure can be viewed using the `sp_depends` system stored procedure. To view the dependencies, the stored procedure name has to be specified as a parameter when executing `sp_depends`.

When `sp_depends` is executed, it returns a list of all objects that reference the specified procedure as well as a list of all objects that the procedure references.

Click on each highlighted area on the image to learn more.



	name	type	updated	selected	column
1	dbo.Employee_Details	user table	no	yes	FirstName
2	dbo.Employee_Details	user table	no	yes	LastName
3	dbo.Employee_Details	user table	no	yes	DateofBirth
4	dbo.Employee_Details	user table	no	yes	City

column

Specifies the column or parameter on which the dependency exists.

"sp_depends" Stored Procedure

In SQL Server 2005, the dependencies of a stored procedure can be viewed using the `sp_depends` system stored procedure. To view the dependencies, the stored procedure name has to be specified as a parameter when executing `sp_depends`.

When `sp_depends` is executed, it returns a list of all objects that reference the specified procedure as well as a list of all objects that the procedure references.

The following syntax is used to view the dependencies of a stored procedure.

```
sp_depends '<procedure_name>'
```

Syntax

Syntax | Snippet

```
1  sp_depends 'Display_Customers'
```

Results | Messages

	name	type	updated	selected	column
1	dbo.Employee_Details	user table	no	yes	FirstName
2	dbo.Employee_Details	user table	no	yes	LastName
3	dbo.Employee_Details	user table	no	yes	DateofBirth
4	dbo.Employee_Details	user table	no	yes	City

"sp_depends" Stored Procedure

In SQL Server 2005, the dependencies of a stored procedure can be viewed using the `sp_depends` system stored procedure. To view the dependencies, the stored procedure name has to be specified as a parameter when executing `sp_depends`.

When `sp_depends` is executed, it returns a list of all objects that reference the specified procedure as well as a list of all objects that the procedure references.

The following code displays the dependencies of the stored procedure `Display_Customers`.

```
sp_depends 'Display_Customers'
```

Snippet

```
1  sp_depends 'Display_Customers'
```

Results | Messages

	name	type	updated	selected	column
1	dbo.Employee_Details	user table	no	yes	FirstName
2	dbo.Employee_Details	user table	no	yes	LastName
3	dbo.Employee_Details	user table	no	yes	DateofBirth
4	dbo.Employee_Details	user table	no	yes	City

Syntax | Snippet

Knowledge Check

Can you match the stored procedure terms against their corresponding description?

Select an option from each drop-down list box and then click on **Submit**.

	Description	Terms
(A)	System stored procedure used to display the definition of a stored procedure.	Select Step...
(B)	System function used to display the definition of a stored procedure.	Select Step...
(C)	System stored procedure used to display the dependencies of a stored procedure.	Select Step...
(D)	System view used to display definition of a stored procedure.	Select Step...
(E)	System function used to display the definition of a stored procedure by specifying the object ID of the procedure.	Select Step...

Submit

View Answer

Knowledge Check

37 | 16

Score

Can you match the stored procedure terms against their corresponding description?

Select an option from each drop-down list box and then click on **Submit**.

	Description	Terms
(A)	System stored procedure used to display the definition of a stored procedure.	sp_helptext
(B)	System function used to display the definition of a stored procedure.	OBJECT_DEFINITION()
(C)	System stored procedure used to display the dependencies of a stored procedure.	sp_depends
(D)	System view used to display definition of a stored procedure.	sys.sql_modules
(E)	System function used to display the definition of a stored procedure by specifying the object ID of the procedure.	OBJECT_DEFINITION()

Correct

The correct answers are displayed.

Submit

View Answer

Lesson Review

In this first lesson, **Viewing Information**, you learnt to:

- Describe how to view definitions of stored procedures.
- State how to view dependencies of stored procedures.

Lesson Overview

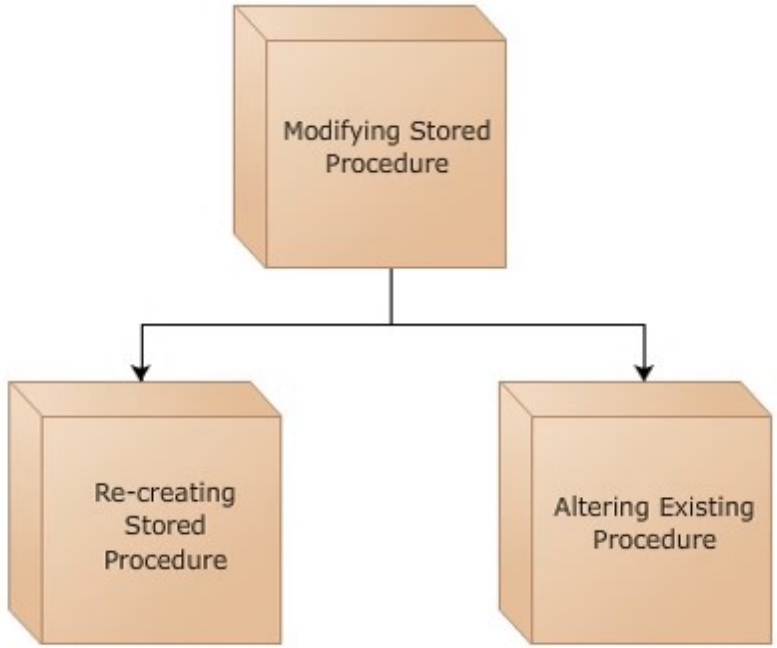
In this second lesson, **Modifying and Dropping Stored Procedures**, you will learn to:

- Describe how to modify stored procedures.
- Describe how to drop stored procedures.

Stored Procedures

A stored procedure can be modified to change the procedure name as well as the procedure definition. You can change the procedure statements or parameters by either re-creating the stored procedure or by altering the existing procedure.

To modify or rename a procedure, you must either own the procedure or have the required permissions to carry out the modifications.

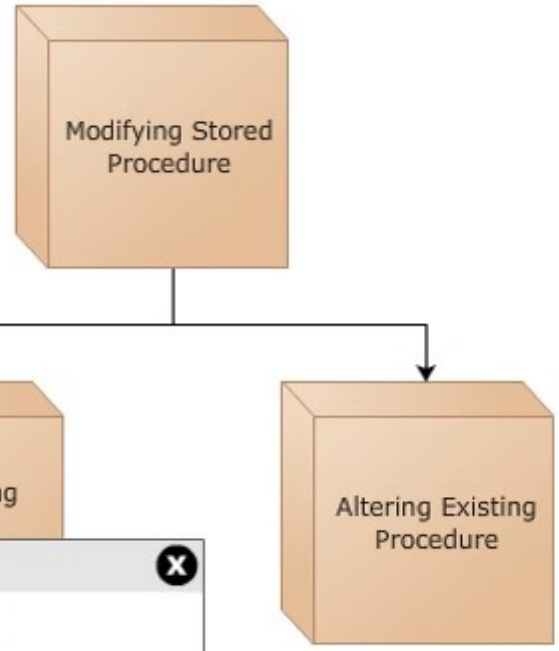


More

Stored Procedures

A stored procedure can be modified to change the procedure name as well as the procedure definition. You can change the procedure statements or parameters by either re-creating the stored procedure or by altering the existing procedure.

To modify or rename a procedure, you must either own the procedure or have the required permissions to carry out the modifications.



More

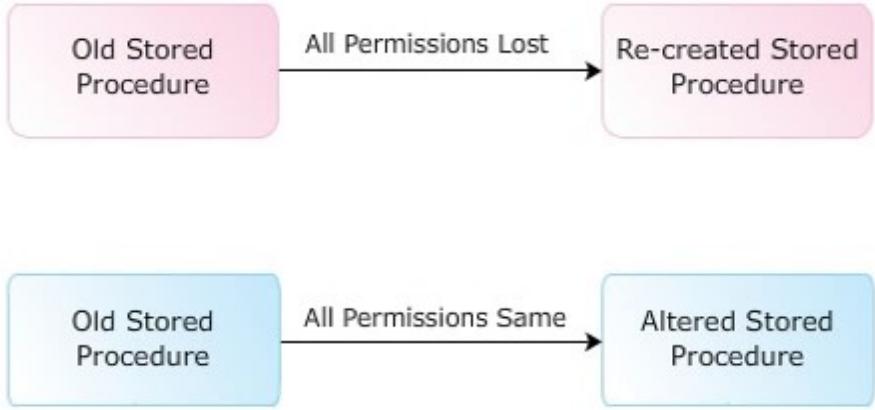
The database owner can modify or rename all procedures.

More

Modifying Stored Procedures

The permissions associated with the stored procedure are lost when a stored procedure is re-created. However, when a stored procedure is altered, the permissions defined for the stored procedure remain the same even though the procedure definition is changed.

A procedure can be altered using the `ALTER PROCEDURE` statement.



Syntax

Snippet

More

Modifying Stored Procedures

The permissions associated with the stored procedure are lost when a stored procedure is re-created. However, when a stored procedure is altered, the permissions defined for the stored procedure remain the same even though the procedure definition is changed.

A procedure can be altered using the `ALTER PROCEDURE` statement.

The following syntax is used to modify a stored procedure.

```
ALTER PROCEDURE <procedure_name>
     @parameter <data_type> [ OUTPUT ]
[ WITH { ENCRYPTION | RECOMPILE } ]
AS <sql_statement>
```

where,

`ENCRYPTION`: Encrypts the stored procedure definition.

`RECOMPILE`: Indicates that the procedure is compiled at run time.

`sql_statement`: Specifies the Transact-SQL statements to be included in the body of the procedure.

Syntax

Syntax

Snippet

More

Modifying Stored Procedures

The permissions associated with the stored procedure are lost when a stored procedure is re-created. However, when a stored procedure is altered, the permissions defined for the stored procedure remain the same even though the procedure definition is changed.

A procedure can be altered using the `ALTER PROCEDURE` statement.

Syntax

Snippet

More

The following code modifies the definition of the stored procedure named `Display_Customers`.

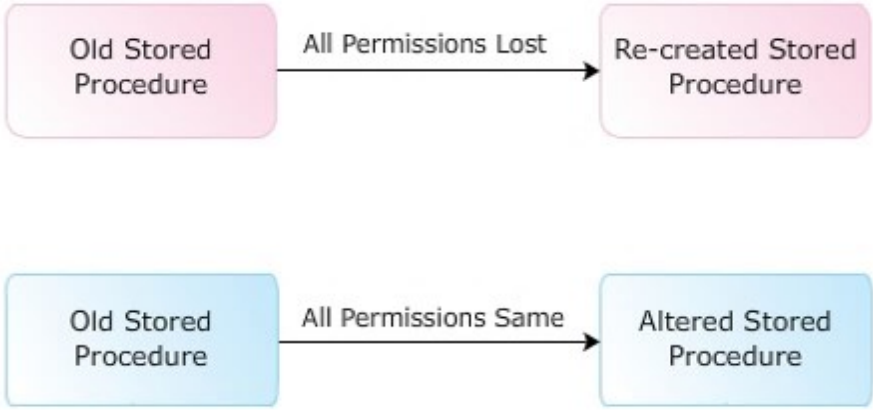
```
ALTER PROCEDURE Display_Customers
AS
   SELECT CustID, AccNo, AccName, City, State, Country
   FROM Customer_Details
    WHERE City='New York'
```

Snippet

Modifying Stored Procedures

The permissions associated with the stored procedure are lost when a stored procedure is re-created. However, when a stored procedure is altered, the permissions defined for the stored procedure remain the same even though the procedure definition is changed.

A procedure can be altered using the `ALTER PROCEDURE` statement.



Syntax

Snippet

More

More

When you change the definition of a stored procedure, the dependent objects may fail when executed. This happens if the dependent objects are not updated to reflect the changes made to the stored procedure.

Guidelines

Stored procedures are altered using the `ALTER PROCEDURE` statement. The following facts have to be considered while using the `ALTER PROCEDURE` statement:

- When a stored procedure is created using options such as the `WITH ENCRYPTION` option, these options should also be included in the `ALTER PROCEDURE` statement.
- The `ALTER PROCEDURE` statement alters a single procedure. When a stored procedure calls other stored procedures, the nested stored procedures are not affected by altering the calling procedure.
- The creators of the stored procedure, members of the `sysadmin` server role and members of the `db_owner` and `db_ddladmin` fixed database roles have the permission to execute the `ALTER PROCEDURE` statement.

More

Guidelines

Stored procedures are altered using the `ALTER PROCEDURE` statement. The following facts have to be considered while using the `ALTER PROCEDURE` statement:

- When a stored procedure is created using options such as the `WITH ENCRYPTION` option, these options should also be included in the `ALTER PROCEDURE` statement.
- The `ALTER PROCEDURE` statement alters a single procedure. When a stored procedure calls other stored procedures, the nested stored procedures are not affected by altering the calling procedure.
- The creators of the stored procedure, members of the `sysadmin` server role and members of the `db_owner` and `db_ddladmin` fixed database roles have the permission to execute the `ALTER PROCEDURE` statement.

More

More

It is recommended that you do not modify system stored procedures. If you need to change the functionality of a system stored procedure, then create a user-defined system stored procedure by copying the statements from an existing stored procedure and modify this user-defined procedure.

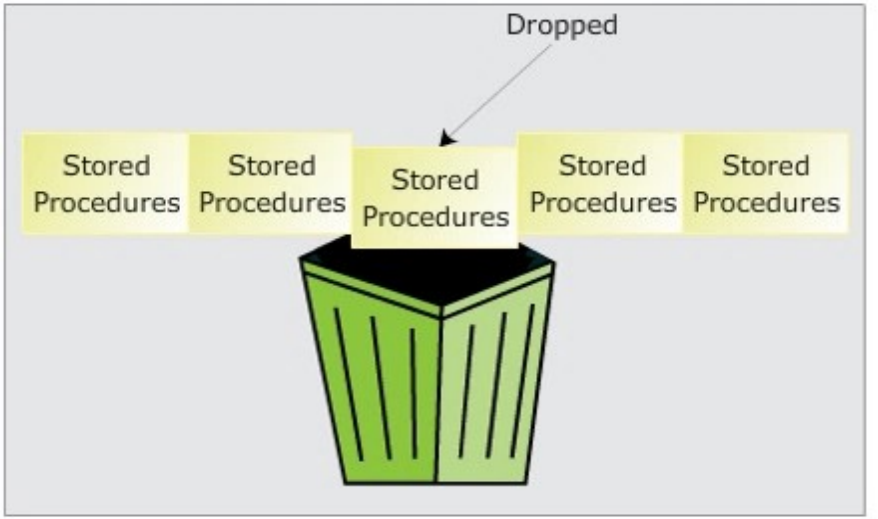
Dropping Stored Procedures

Stored procedures can be dropped if they are no longer needed. If another stored procedure calls a deleted procedure, an error message is displayed.

If a new procedure is created using the same name as well as the same parameters as the dropped procedure, all calls to the dropped procedure will be executed successfully. This is because they will now refer to the new procedure, which has the same name and parameters as the deleted procedure.

Before dropping a stored procedure, execute the `sp_depends` system stored procedure to determine which objects depend on the procedure.

A procedure is dropped using the `DROP PROCEDURE` statement.



Syntax

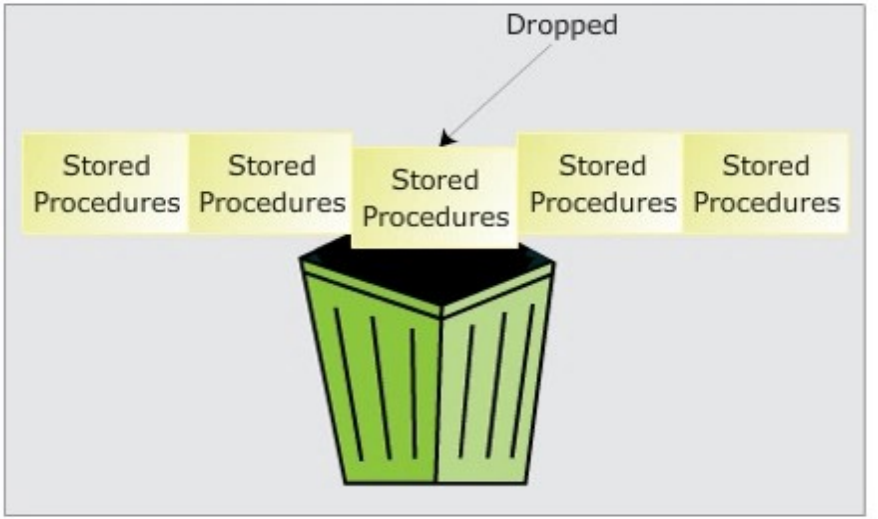
Snippet

Dropping Stored Procedures

Stored procedures can be dropped if they are no longer needed. If another stored procedure calls a deleted procedure, an error message is displayed.

If a new procedure is created using the same name as well as the same parameters as the dropped procedure, all calls to the dropped procedure will be executed successfully. This is because they will now refer to the new procedure, which has the same name and parameters as the deleted procedure.

Before dropping a stored procedure, execute the `sp_depends` system stored procedure to determine which objects depend on the procedure.



A procedure is dropped using the `DROP PROCEDURE` statement.

Syntax | Snippet

The following syntax is used to drop a stored procedure.

```
DROP PROCEDURE <procedure_name>
```

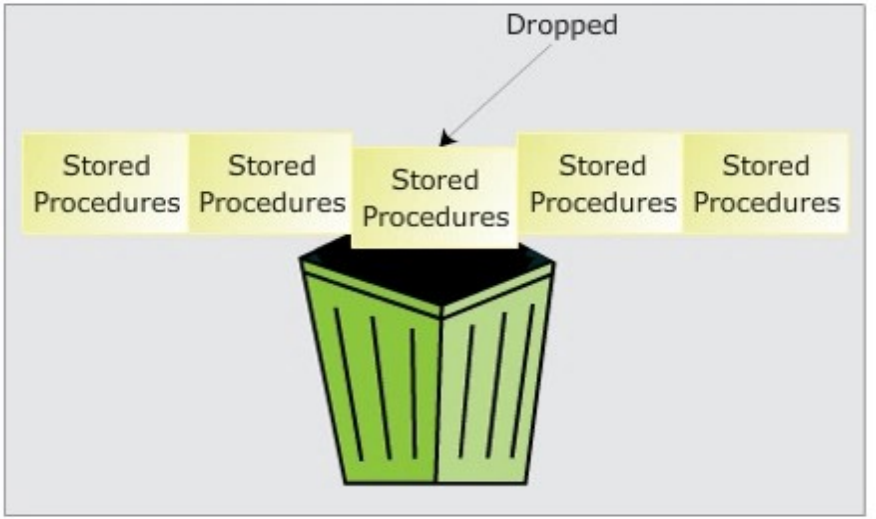
Syntax

Dropping Stored Procedures

Stored procedures can be dropped if they are no longer needed. If another stored procedure calls a deleted procedure, an error message is displayed.

If a new procedure is created using the same name as well as the same parameters as the dropped procedure, all calls to the dropped procedure will be executed successfully. This is because they will now refer to the new procedure, which has the same name and parameters as the deleted procedure.

Before dropping a stored procedure, execute the `sp_depends` system stored procedure to determine which objects depend on the procedure.



A procedure is dropped using the `DROP PROCEDURE` statement.

Syntax

Snippet

The following code drops the stored procedure, `Display_Customers`.

```
DROP PROCEDURE Display_Customers
```

Snippet

37 | 16

Score

Knowledge Check

Which of these statements about stored procedures of SQL Server 2005 are true and which statements are false?

Select an option for each statement and then click on **Submit**.

	Statements	True	False
(A)	When a stored procedure is created using options, these options should be included in the `ALTER PROCEDURE` statement to retain their functionality.	○	○
(B)	When a stored procedure is altered, the procedure definition and the permissions defined for the stored procedure change.	○	○
(C)	When a call is made to a deleted stored procedure, the calling procedure is executed successfully.	○	○
(D)	When a stored procedure is re-created, all permissions associated with the stored procedure are lost.	○	○
(E)	When a procedure is dropped, a new procedure can be created by the same name.	○	○

Submit

38 | 16

Score

Knowledge Check

Which of these statements about stored procedures of SQL Server 2005 are true and which statements are false?

Select an option for each statement and then click on **Submit**.

	Statements	True	False
(A)	When a stored procedure is created using options, these options should be included in the `ALTER PROCEDURE` statement to retain their functionality.	◉	○
(B)	When a stored procedure is altered, the procedure definition and the permissions defined for the stored procedure change.	○	◉
(C)	When a call is made to a deleted stored procedure, the calling procedure is executed successfully.	○	◉
(D)	When a stored procedure is re-created, all permissions associated with the stored procedure are lost.	◉	○
(E)	When a procedure is dropped, a new procedure can be created by the same name.	◉	○

Correct

The correct answers are displayed.

Submit

Lesson Review

In this second lesson, **Modifying and Dropping Stored Procedures**, you learnt to:

- Describe how to modify stored procedures.
- Describe how to drop stored procedures.

Lesson Overview

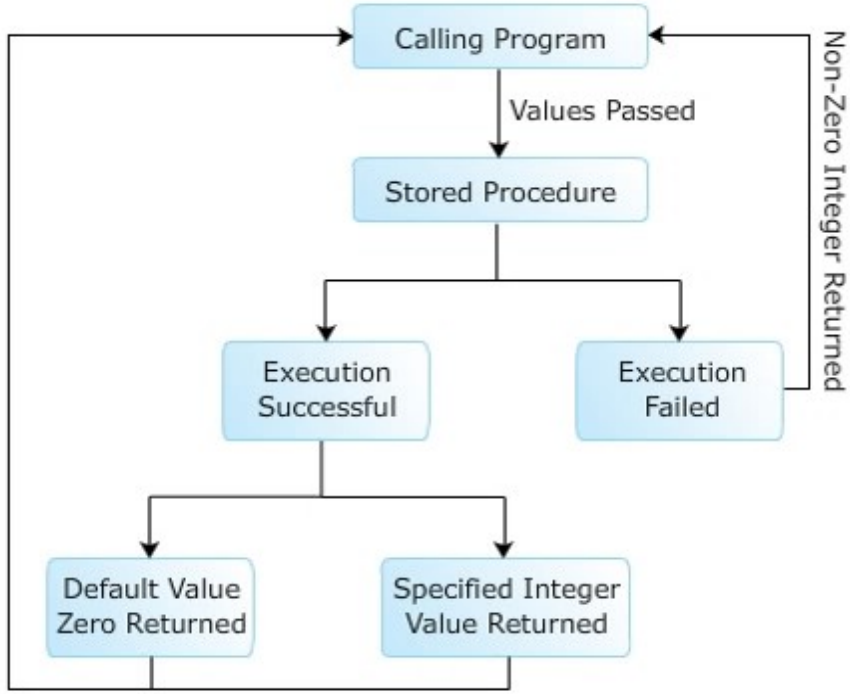
In this third lesson, **Working with Stored Procedures**, you will learn to:

- Explain how to return values in stored procedures.
- Explain how to use parameters in stored procedures.
- Describe nested stored procedures.
- Explain handling of error messages in stored procedures.

Returning Values

Values are passed to a stored procedure by the calling program when the stored procedure is executed. The procedure performs the required task using these values and, by default, returns a zero or non-zero integer. This returned value is referred to as a return code. The return code indicates whether or not the procedure was successfully executed.

Instead of returning the default return code, SQL Server 2005 allows you to explicitly specify an integer value to be returned through the stored procedure. This value is returned using the `RETURN` statement.

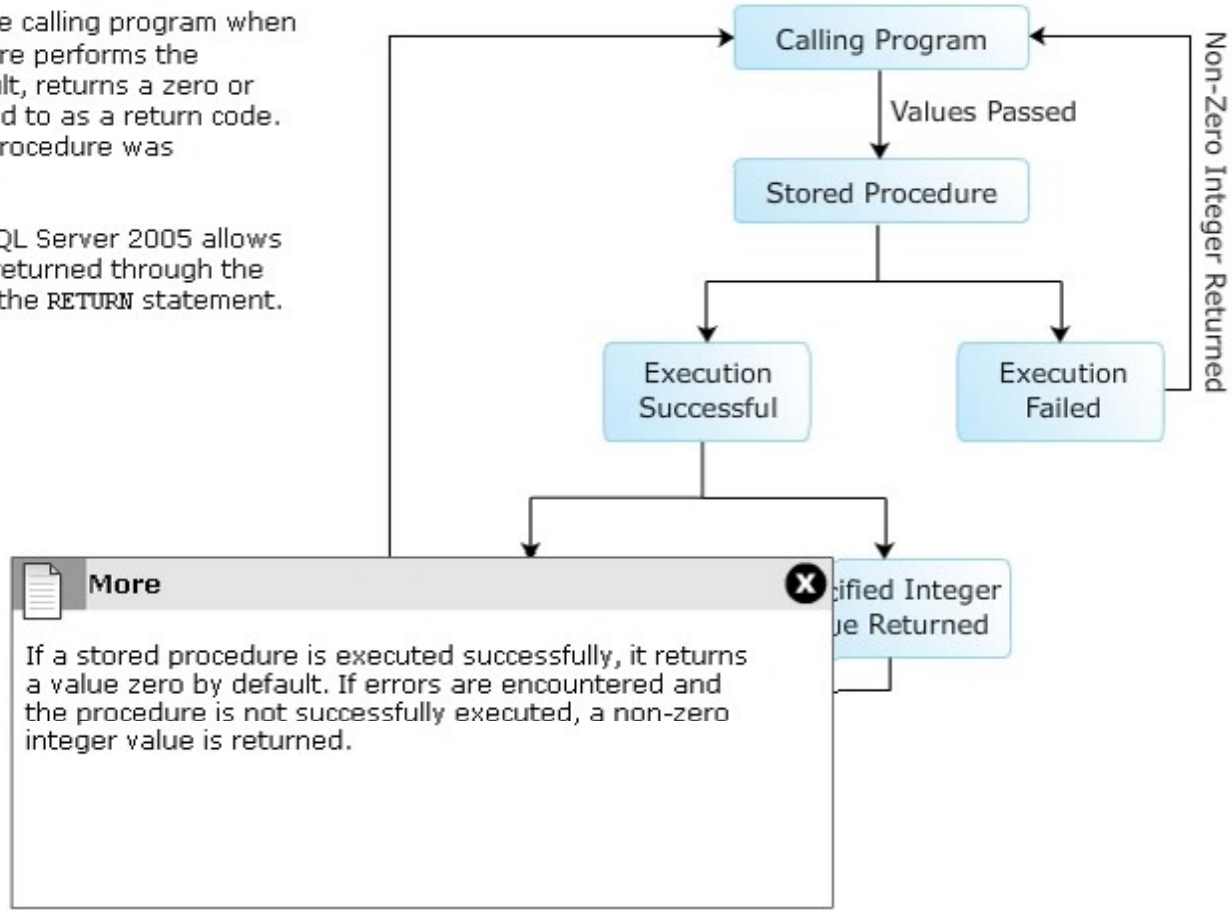


More

Returning Values

Values are passed to a stored procedure by the calling program when the stored procedure is executed. The procedure performs the required task using these values and, by default, returns a zero or non-zero integer. This returned value is referred to as a return code. The return code indicates whether or not the procedure was successfully executed.

Instead of returning the default return code, SQL Server 2005 allows you to explicitly specify an integer value to be returned through the stored procedure. This value is returned using the `RETURN` statement.



More

"RETURN" Statement

The `RETURN` statement passes control back to the calling program. Any Transact-SQL statements following the `RETURN` statement are not executed.

When the `RETURN` statement is used in a stored procedure, it cannot return a null value. If a procedure tries to return a null value, a warning message is generated and the value zero is returned.

Syntax

Snippet

"RETURN" Statement

The `RETURN` statement passes control back to the calling program. Any Transact-SQL statements following the `RETURN` statement are not executed.

When the `RETURN` statement is used in a stored procedure, it cannot return a null value. If a procedure tries to return a null value, a warning message is generated and the value zero is returned.

The following code is used to pass the control back to the calling program.

```
RETURN [<integer_expression>]
```

where,
`integer_expression`: Is the integer value that is returned.

Syntax

Syntax | Snippet

"RETURN" Statement

The RETURN statement passes control back to the calli
Transact-SQL statements following the RETURN statem
executed.

When the RETURN statement is used in a stored proce
return a null value. If a procedure tries to return a nul
warning message is generated and the value zero is

Syntax

Snippet

The following code creates a stored procedure Cal_Square. The RETURN statement returns the square of the number supplied as a parameter by the calling program.

```
CREATE PROCEDURE Cal_Square @num int = 0
AS
     BEGIN
          RETURN (@num * @num);
     END
```

The following code declares an integer variable, square that accepts the return value from the Cal_Square procedure.

```
DECLARE @square int;
EXECUTE @square = Cal_Square 10;
PRINT @square;
```

In the above code, the value 10 is passed to the Cal_Square stored procedure. The procedure calculates the square of 10 and passes this value to the calling program. The calling program accepts the value in the variable, square and prints the value, 100.

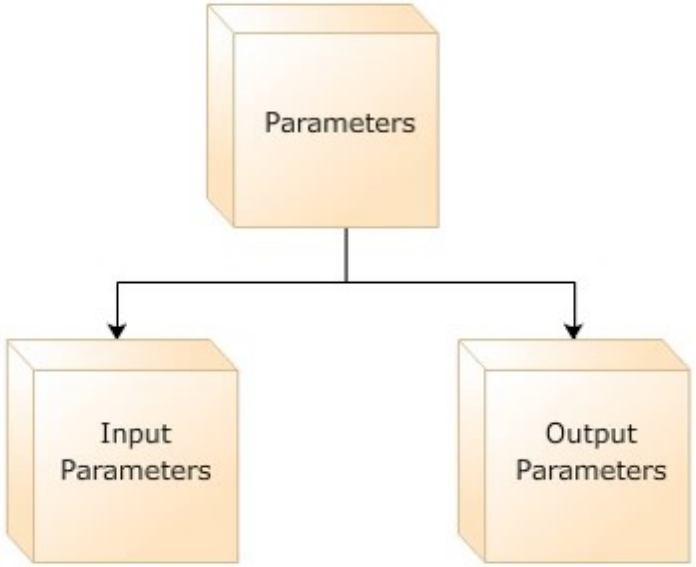
Snippet

Using Parameters

Data is passed between the stored procedure and the calling program when a call is made to a stored procedure. This data transfer is done using parameters. Parameters are of two types:

- Input Parameters
- Output Parameters

Click on each link to learn more.

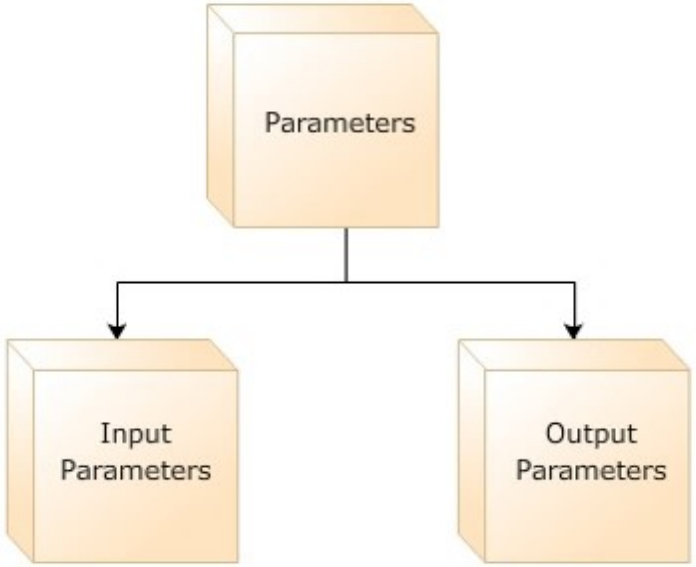


Using Parameters

Data is passed between the stored procedure and the calling program when a call is made to a stored procedure. This data transfer is done using parameters. Parameters are of two types:

- Input Parameters
- Output Parameters

Click on each link to learn more.



Input Parameters

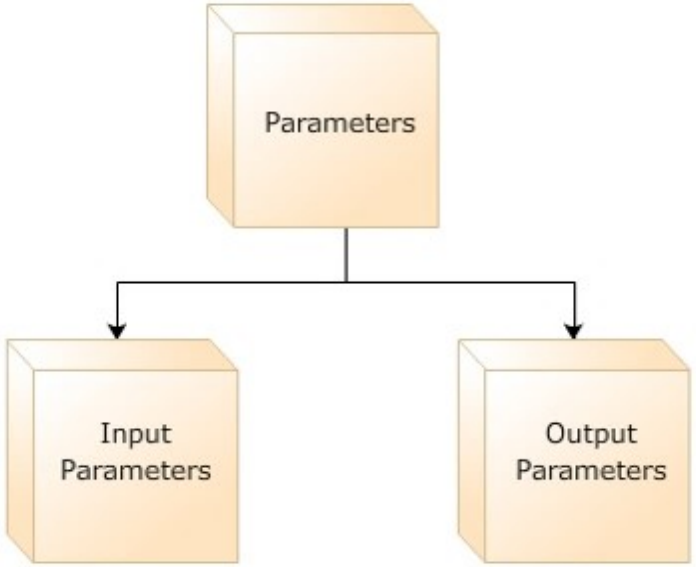
Input parameters allow the calling program to pass values to a stored procedure. These values are accepted into variables defined in the stored procedure.

Using Parameters

Data is passed between the stored procedure and the calling program when a call is made to a stored procedure. This data transfer is done using parameters. Parameters are of two types:

- Input Parameters
- Output Parameters

Click on each link to learn more.



Output Parameters

Output parameters allow a stored procedure to pass values back to the calling program. These values are accepted into variables by the calling program.

Input Parameters

Values are passed from the calling program to the stored procedure and these values are accepted into the input parameters of the stored procedure. The input parameters are defined at the time of creation of the stored procedure.

The values passed to input parameters can be either constants or variables. These values are passed to the procedure at the time of calling the procedure. The stored procedure performs the specified tasks using these values.

```
1  EXECUTE StateWise_Customers 'California'
```

Results | Messages

	AccNo	AccName	City	State
1	1	Joanne	Hollywood	California
2	2	Sandra	Los Angeles	California
3	3	Susan	Van Nuys	California

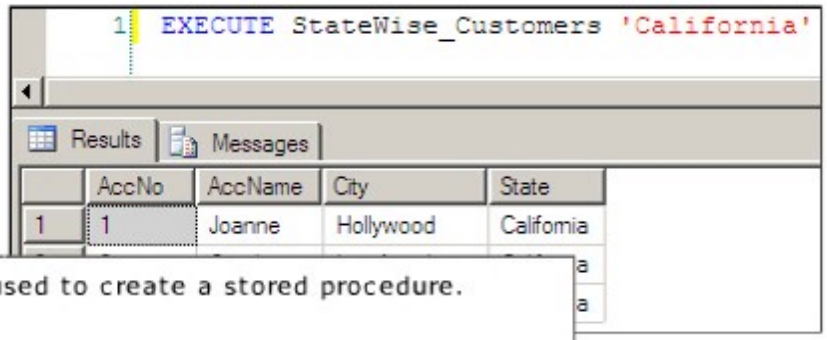
Syntax

Snippet

Input Parameters

Values are passed from the calling program to the stored procedure and these values are accepted into the input parameters of the stored procedure. The input parameters are defined at the time of creation of the stored procedure.

The values passed to input parameters can be either constants or variables. These values are passed to the procedure at the time of calling the procedure. The stored procedure performs the specified tasks using these values.



Syntax

Snippet

The following syntax is used to create a stored procedure.

```
CREATE PROCEDURE <procedure_name>
    @parameter <data_type>
AS <sql_statement>
```

where,
`data_type`: Specifies the system defined data type.

The following syntax is used to execute a stored procedure and pass values as input parameters:

```
EXECUTE <procedure_name> <parameters>
```

Syntax

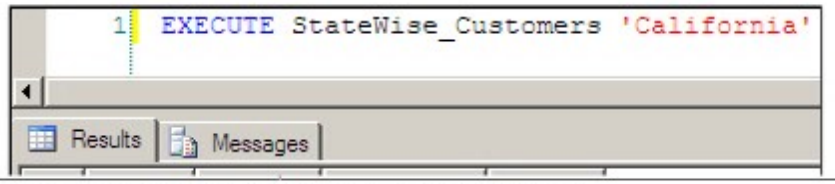
Input Parameters

Values are passed from the calling program to the stored procedure and these values are accepted into the input parameters of the stored procedure. The input parameters are defined at the time of creation of the stored procedure.

The values passed to input parameters can be either constants or variables. These values are passed to the procedure at the time of calling the procedure. The stored procedure performs the specified tasks using these values.

Syntax

Snippet



The following code creates a stored procedure, `StateWise_Customers`, with a parameter `statename` to accept the name of a state and display records of customers for that state.

```
CREATE PROCEDURE StateWise_Customers
    @statename varchar(40)
AS
    SELECT AccNo, AccName, City, State FROM
Customer_Details WHERE State = @statename;
```

The following code executes the `Statewise_Customers` stored procedure with California being passed as the input parameter:

```
EXECUTE StateWise_Customers 'California'
```

Snippet

Output Parameters

Stored procedures occasionally need to return output back to the calling program. This transfer of data from the stored procedure to the calling program is performed using output parameters.

Output parameters are defined at the time of creation of the procedure. To specify an output parameter, the OUTPUT keyword is used while declaring the parameter. Also, the calling statement has to have a variable specified with the OUTPUT keyword to accept the output from the called procedure.

```
1  CREATE PROCEDURE Total_Customers
2      @statename varchar(40), @no_of_cust int OUTPUT
3  AS
4      SELECT @no_of_cust = COUNT(*) FROM Customer_Details
5      WHERE State = @statename
6
7  DECLARE @tot_customers int
8  EXECUTE Total_Customers 'California', @tot_customers OUTPUT
9  PRINT @tot_customers
```

Messages

5 ← Total number of customers in California

Syntax

Snippet

Output Parameters

Stored procedures occasionally need to return output back to the calling program. This transfer of data from the stored procedure to the calling program is performed using output parameters.

Output parameters are defined at the time of creation of the procedure. To specify an output parameter, the OUTPUT keyword is used while declaring the parameter. Also, the calling statement has to have a variable specified with the OUTPUT keyword to accept the output from the called procedure.

Syntax | Snippet

```
1  CREATE PROCEDURE Total_Customers
2      @statename varchar(40), @no_of_cust int OUTPUT
3  AS
4      SELECT @no_of_cust = COUNT(*) FROM Customer_Details
5      WHERE State = @statename
6
```

The following syntax is used to pass output parameters in a stored procedure.

```
CREATE PROCEDURE <procedure_name>
     @parameter <data_type>,
     @parameter <data_type> OUTPUT
AS <sql_statement>
```

The following syntax is used to execute a stored procedure with the OUTPUT parameter specified.

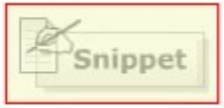
```
EXECUTE <procedure_name> <parameters> [OUTPUT]
```

Syntax

Output Parameters

Stored procedures occasionally need to return output back to the calling program. This transfer of data from the stored procedure to the calling program is performed using output parameters.

Output parameters are defined at the time of creation of the procedure. To specify an output parameter, the OUTPUT keyword is used while declaring the parameter. Also, the calling statement has to have a variable specified with the OUTPUT keyword to accept the output from the called procedure.



The following code creates a stored procedure, Total_Customers with input parameter statename to accept the name of a state and output parameter no_of_cust to display the number of customers in that state.

```
CREATE PROCEDURE Total_Customers
     @statename varchar(40),
     @no_of_cust int OUTPUT
AS
     SELECT @no_of_cust = COUNT(*) FROM Customer_Details WHERE
State = @statename;
```

The following code declares a variable tot_customers to accept the output of the procedure Total_Customers.

```
DECLARE @tot_customers int;
EXECUTE Total_Customers 'California', @tot_customers OUTPUT;
PRINT @tot_customers;
```

The above code passes California as the input to the Total_Customers stored procedure and accepts the output in the variable tot_customers. The output is printed using the PRINT command.

Snippet

Characteristics of "Output" Parameters

Stored procedures return information to the calling procedure with output parameters. To use an output parameter, specify the `OUTPUT` keyword in both the `CREATE PROCEDURE` and `EXECUTE` statements. If the `OUTPUT` keyword is omitted, the procedure is still executed but it does not return a value.

The `OUTPUT` parameters have the following characteristics:

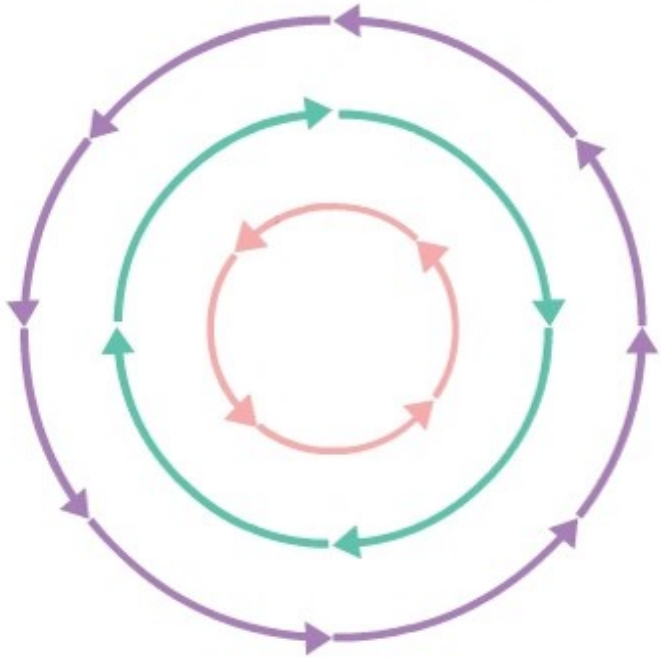
- The parameter cannot be of `text` and `image` data type.
- The calling statement must contain a variable to receive the return value.
- The variable can be used in subsequent Transact-SQL statements in the batch or the calling procedure.
- Output parameters can be cursor placeholders.

Nested Stored Procedures

In SQL Server 2005, you can call stored procedures from inside other stored procedures. The called procedures can in turn call other procedures. This architecture of calling one procedure from another procedure is referred to as nested stored procedure architecture.

When a stored procedure calls another stored procedure, the level of nesting is said to be increased by one. Similarly, when a called procedure completes its execution and passes control back to the calling procedure, the level of nesting is said to be decreased by one. The maximum level of nesting supported by SQL Server 2005 is 32. If the level of nesting exceeds 32, the calling process fails.

Nested Stored Procedures



Snippet

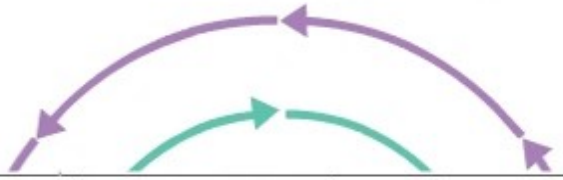
More

Nested Stored Procedures

In SQL Server 2005, you can call stored procedures from inside other stored procedures. The called procedures can in turn call other procedures. This architecture of calling one procedure from another procedure is referred to as nested stored procedure architecture.

When a stored procedure calls another stored procedure, the level of nesting is said to be increased by one. Similarly, when a called procedure completes its execution and passes control back to the calling procedure, the level of nesting is said to be decreased by one. The maximum level of nesting supported by SQL Server 2005 is 32. If the level of nesting exceeds 32, the calling process fails.

Nested Stored Procedures



The following code is used to create a stored procedure `NestedProcedure` that calls two other stored procedures, namely `Display_Customers` and `CityWise_Customers`.

```
CREATE PROCEDURE NestedProcedure
AS
    EXECUTE Display_Customers
    EXECUTE CityWise_Customers 'New York'
```

When the procedure `NestedProcedure` is executed, this procedure in turn invokes the `Display_Customers` and `CityWise_Customers` stored procedures and passes the value "New York" as the input parameter to the `CityWise_Customers` stored procedure.

Snippet

Snippet

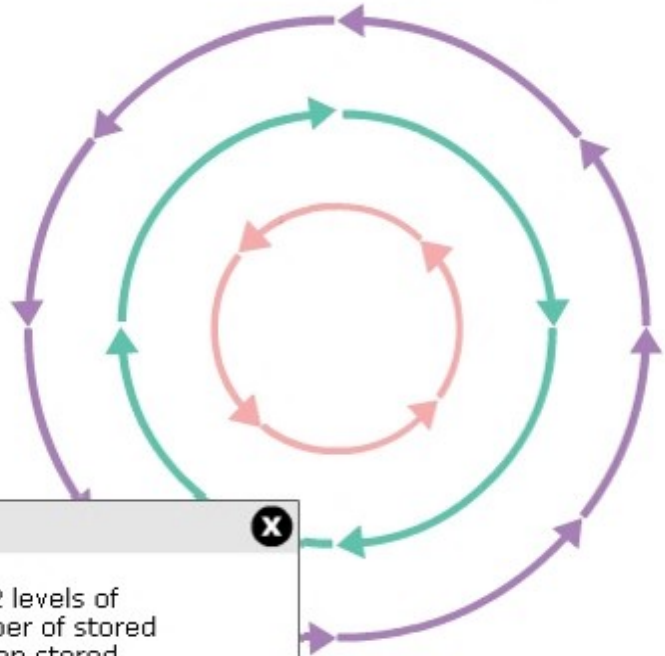
More

Nested Stored Procedures

In SQL Server 2005, you can call stored procedures from inside other stored procedures. The called procedures can in turn call other procedures. This architecture of calling one procedure from another procedure is referred to as nested stored procedure architecture.

When a stored procedure calls another stored procedure, the level of nesting is said to be increased by one. Similarly, when a called procedure completes its execution and passes control back to the calling procedure, the level of nesting is said to be decreased by one. The maximum level of nesting supported by SQL Server 2005 is 32. If the level of nesting exceeds 32, the calling process fails.

Nested Stored Procedures



Snippet

More

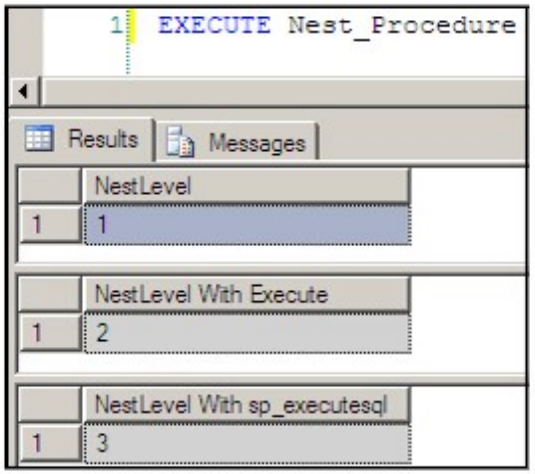
More

Although there can be a maximum of 32 levels of nesting, there is no limit as to the number of stored procedure that can be called from a given stored procedure.

"@@NESTLEVEL" Function

The level of nesting of the current procedure can be determined using the `@@NESTLEVEL function`. When the `@@NESTLEVEL function` is executed within a Transact-SQL string, the value returned is the current nesting level + 1.

If you use `sp_executesql` to execute the `@@NESTLEVEL` function, the value returned is the current nesting level + 2 (as another stored procedure, namely `sp_executesql`, gets added to the nesting chain).



Syntax

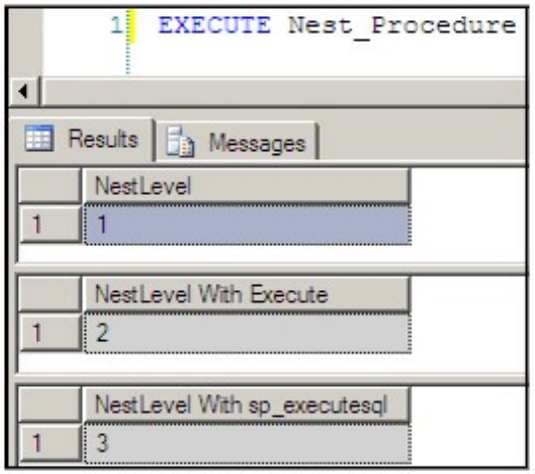
Snippet

More

"@@NESTLEVEL" Function

The level of nesting of the current procedure can be determined using the `@@NESTLEVEL function`. When the `@@NESTLEVEL function` is executed within a Transact-SQL string, the value returned is the current nesting level + 1.

If you use `sp_executesql` to execute the `@@NESTLEVEL` function, the value returned is the current nesting level + 2 (as another stored procedure, namely `sp_executesql`, gets added to the nesting chain).



Syntax

Snippet

More

```
@@NESTLEVEL

where,
@@NESTLEVEL: Is a function that returns an integer value
specifying the level of nesting.
```

Syntax

"@@NESTLEVEL" Function

The level of nesting of the current procedure can be
the `@@NESTLEVEL function`. When the `@@NESTLEVEL f`
executed within a Transact-SQL string, the value ret
current nesting level + 1.

If you use `sp_executesql` to execute the `@@NESTLEVE`
value returned is the current nesting level + 2 (as ar
procedure, namely `sp_executesql`, gets added to the

The following code creates a procedure `Nest_Procedure` that executes the `@@NESTLEVEL` function to determine the level of nesting in three different scenarios:

```
CREATE PROCEDURE Nest_Procedure
AS
    SELECT @@NESTLEVEL AS NestLevel;
    EXECUTE ('SELECT @@NESTLEVEL AS [NestLevel With Execute]');
    EXECUTE sp_executesql N'SELECT @@NESTLEVEL AS [NestLevel
With sp_executesql]';
```

The following code executes the `Nest_Procedure` stored procedure:

```
EXECUTE Nest_Procedure
```

Three outputs are displayed for the three different methods used to call the `@@NESTLEVEL` function.

Syntax

Snippet

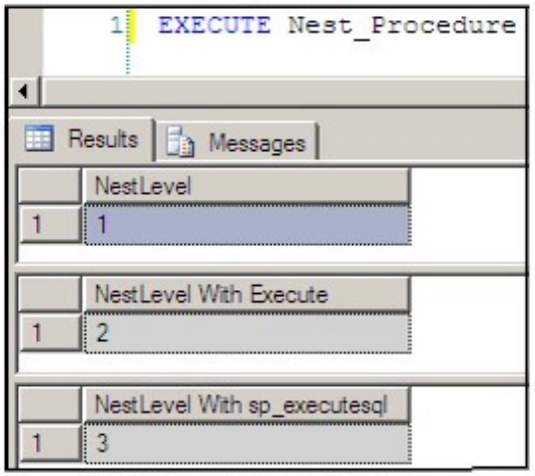
More

Snippet

"@@NESTLEVEL" Function

The level of nesting of the current procedure can be determined using the `@@NESTLEVEL function`. When the `@@NESTLEVEL function` is executed within a Transact-SQL string, the value returned is the current nesting level + 1.

If you use `sp_executesql` to execute the `@@NESTLEVEL` function, the value returned is the current nesting level + 2 (as another stored procedure, namely `sp_executesql`, gets added to the nesting chain).



Syntax

Snippet

More

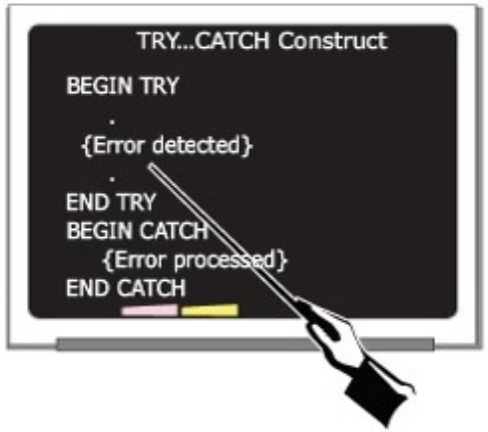
More

The `sp_executesql` stored procedure can also be used to execute Transact-SQL statements.

Handling Error Messages

SQL Server 2005 introduces the `TRY...CATCH` construct for handling errors in stored procedures. The `TRY...CATCH` construct consists of two blocks of code, the `TRY` block and the `CATCH` block.

When an error is detected in an SQL statement inside a `TRY` block, control is passed to the `CATCH` block, where the error can be processed. After the error is processed in the `CATCH` block, the control is transferred to the SQL statement that is written after the `END CATCH` statement. The SQL statements in the `TRY` block written after the statement that generates the error are not executed.

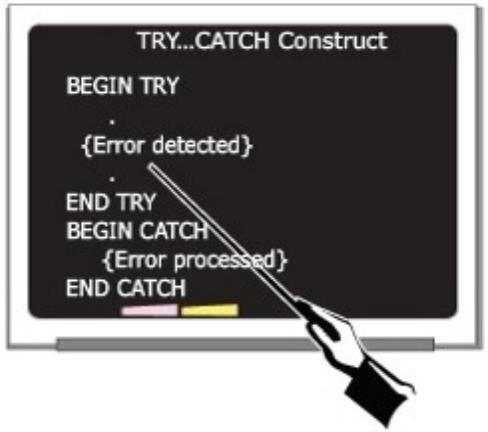


More

Handling Error Messages

SQL Server 2005 introduces the `TRY...CATCH` construct for handling errors in stored procedures. The `TRY...CATCH` construct consists of two blocks of code, the `TRY` block and the `CATCH` block.

When an error is detected in an SQL statement inside a `TRY` block, control is passed to the `CATCH` block, where the error can be processed. After the error is processed in the `CATCH` block, the control is transferred to the SQL statement that is written after the `END CATCH` statement. The SQL statements in the `TRY` block written after the statement that generates the error are not executed.



More

More

If there are no errors inside the `TRY` block, control passes to the statement written after the `END CATCH` statement of the corresponding `CATCH` block.

"TRY...CATCH" Construct

The `ERROR_MESSAGE()` is a system function used to display error messages. In SQL Server 2005, all errors have an associated error number. Each error number corresponds to a default error message. The `sys.messages` system view contains all the default error messages corresponding to the various error numbers.

The `ERROR_MESSAGE()` system function can be used in the `CATCH` block to display the default error message for the error occurring in the `TRY` block.

```
1  CREATE PROCEDURE Error_Procedure
2  AS
3  BEGIN
4      DECLARE @result int
5      SELECT 'This statement will be executed'
6      SELECT @result = 'Hello'
7      SELECT 'This statement will not be executed'
8  END
9
10 BEGIN TRY
11     EXECUTE Error_Procedure
12 END TRY
13 BEGIN CATCH
14     SELECT ERROR_MESSAGE() as ErrorMessage
15 END CATCH
```

Results | Messages

	(No column name)
1	This statement will be executed

	ErrorMessage
1	Conversion failed when converting the varchar value 'Hello' to data type int.

Syntax

Snippet

More

"TRY...CATCH" Construct

The `ERROR_MESSAGE()` is a system function used to display error messages. In SQL Server 2005, all errors have an associated error number. Each error number corresponds to a default error message. The `sys.messages` system view contains all the default error messages corresponding to the various error numbers.

The `ERROR_MESSAGE()` system function can be used in the `CATCH` block to display the default error message for the error occurring in the `TRY` block.

```
1  CREATE PROCEDURE Error_Procedure
2  AS
3  BEGIN
4      DECLARE @result int
5      SELECT 'This statement will be executed'
6      SELECT @result = 'Hello'
7      SELECT 'This statement will not be executed'
8  END
9
10 BEGIN TRY
```

The following syntax is used to handle errors in a stored procedure using `TRY…CATCH` construct.

```
BEGIN TRY
    { <sql_statement> | <statement_block> }
END TRY
BEGIN CATCH
    { <sql_statement> | <statement_block> }
END CATCH
```

where,

`statement_block`: Specifies a group of Transact-SQL statements in a batch or enclosed in a `BEGIN…END` block.

Syntax

Syntax | Snippet | More

"TRY...CATCH" Construct

The `ERROR_MESSAGE()` is a system function used to display error messages. In SQL Server 2005, all errors have an associated error number. Each error number corresponds to a default error message. The `sys.messages` system view contains all the default error messages corresponding to the various error numbers.

The `ERROR_MESSAGE()` system function can be used in the `CATCH` block to display the default error message for the error occurring in the `TRY` block.

```
1  CREATE PROCEDURE Error_Procedure
2  AS
3  BEGIN
4      DECLARE @result int
5      SELECT 'This statement will be executed'
6      SELECT @result = 'Hello'
7      SELECT 'This statement will not be executed'
8  END
```

The following code creates a procedure `Error_Procedure` where one of the statements generates an error.

```
CREATE PROCEDURE Error_Procedure
AS
BEGIN
    DECLARE @result int;
    SELECT 'This statement will be executed'
    SELECT @result = 'Hello'
    SELECT 'This statement will not be executed'
END
```

Continued...

Snippet

Syntax

Snippet

More

"TRY...CATCH" Construct

The ERROR_MESSAGE() is a system function used to display error messages. In SQL Server 2005, all errors have an associated error number. Each error number corresponds to a default error message. The sys.messages system view contains all the default error messages corresponding to the various error numbers.

The ERROR_MESSAGE() system function can be used in the CATCH block to display the default error message for the error occurring in the TRY block.

```
1  CREATE PROCEDURE Error_Procedure
2  AS
3  BEGIN
4      DECLARE @result int
5      SELECT 'This statement will be executed'
6      SELECT @result = 'Hello'
7      SELECT 'This statement will not be executed'
8  END
```

The following TRY…CATCH construct is used to handle the error raised when executing the second SELECT statement in the Error_Procedure stored procedure due to invalid data type conversion. The error is processed in the CATCH block.

```
BEGIN TRY
     EXECUTE Error_Procedure
END TRY
BEGIN CATCH
     SELECT ERROR_MESSAGE() as ErrorMessage;
END CATCH;
```

The statement after the statement that generated the error will not be executed.

Snippet

Syntax

Snippet

More

"TRY...CATCH" Construct

The `ERROR_MESSAGE()` is a system function used to display error messages. In SQL Server 2005, all errors have an associated error number. Each error number corresponds to a default error message. The `sys.messages` system view contains all the default error messages corresponding to the various error numbers.

The `ERROR_MESSAGE()` system function can be used in the `CATCH` block to display the default error message for the error occurring in the `TRY` block.

```
1  CREATE PROCEDURE Error_Procedure
2  AS
3  BEGIN
4      DECLARE @result int
5      SELECT 'This statement will be executed'
6      SELECT @result = 'Hello'
7      SELECT 'This statement will not be executed'
8  END
9
10 BEGIN TRY
11     EXECUTE Error_Procedure
12 END TRY
13 BEGIN CATCH
14     SELECT ERROR_MESSAGE() as ErrorMessage
15 END CATCH
```

Results | Messages

...har value 'Hello' to data type int.

More

It is possible to nest the `TRY` and `CATCH` blocks. This means that a `TRY...CATCH` construct can be placed inside other `TRY` and `CATCH` blocks. If an error occurs within a nested `TRY` block, the program control is transferred to the `CATCH` block that corresponds to the nested `TRY` block.

Syntax | Snippet | More

"@@ERROR Function"

Error handling in stored procedures can also be assisted by the `@@ERROR` function. The `@@ERROR` function returns the error number for the error occurring in the last Transact-SQL statement. If the last Transact-SQL statement has encountered no errors, the `@@ERROR` function returns 0. The `@@ERROR` function cannot check for errors occurring in any statements other than the last one executed.

Apart from the `ERROR_MESSAGE()` and the `@@ERROR` functions, there are other system functions that provide further information about errors.

Syntax

Snippet

Error Function	Description
`ERROR_NUMBER()`	Returns the error number.
`ERROR_SEVERITY()`	Returns the severity of the error.
`ERROR_STATE()`	Returns the error state number.
`ERROR_PROCEDURE()`	Returns the name of the stored procedure in which the error occurred.
`ERROR_LINE()`	Returns the line number that caused the error.

```
1  USE ABC_Bank
2  GO
3  INSERT INTO Employee_Details
4  VALUES(16,'David','Johnson','1981/12/29','M','Miami','12000')
5  IF @@ERROR <> 0
6       PRINT 'An error occurred while inserting the record.'
7  GO
```

Messages

```
Msg 2627, Level 14, State 1, Line 1
Violation of PRIMARY KEY constraint 'PK_Employee_Details'. Cannot insert duplic
The statement has been terminated.
An error occurred while inserting the record.
```

"@@ERROR Function"

Error handling in stored procedures can also be assisted by the `@@ERROR` function. The `@@ERROR` function returns the error number for the error occurring in the last Transact-SQL statement. If the last Transact-SQL statement has encountered no errors, the `@@ERROR` function returns 0.

Error Function	Description
`ERROR_NUMBER()`	Returns the error number.
`ERROR_SEVERITY()`	Returns the severity of the error.
`ERROR_STATE()`	Returns the error state number.
`ERROR_PROCEDURE()`	Returns the name of the stored procedure in which the error occurred.
`ERROR_LINE()`	Returns the line number that caused the error.

The following syntax is used to view the error number.

```
@@ERROR
```

Syntax

Syntax | Snippet

```
ABC_Bank

RT INTO Employee_Details
ES(16,'David','Johnson','1981/12/29','M','Miami','12000')
@ERROR <> 0
 PRINT 'An error occurred while inserting the record.'
GO
```

Messages

```
Msg 2627, Level 14, State 1, Line 1
Violation of PRIMARY KEY constraint 'PK_Employee_Details'. Cannot insert duplic
The statement has been terminated.
An error occurred while inserting the record.
```

"@@ERROR Function"

Error handling in stored procedures can also be assisted by the `@@ERROR` function. The `@@ERROR` function returns the error number for the error occurring in the last Transact-SQL statement. If the last Transact-SQL statement has encountered no errors, the `@@ERROR` function returns 0. The `@@ERROR` function cannot check for errors occurring in any statements other than the last one executed.

Apart from the `ERROR_MESSAGE()` and the `@@ERROR` functions, there are other system functions that provide further information about errors.

Syntax

Snippet

In the following code, while inserting a record in the table, an error occurred because of primary key violation. The error number is returned by the `@@ERROR` function.

```
USE ABC_Bank
GO
INSERT INTO Employee_Details
VALUES
(16,'David','Johnson','1981/12/29','M','Miami','12000');
IF @@ERROR <> 0
     PRINT 'An error occurred while inserting the record.';
GO
```

In the above code, an error has occurred in the last statement before the execution of the `@@ERROR` function, an error number (which is a non-zero number) is returned. This non-zero number causes the `PRINT` statement to be executed. Also, the default error message for the `PRIMARY KEY` violation is displayed.

Snippet

Knowledge Check

Can you match the keywords in SQL Server 2005 against their corresponding descriptions?

Click an option on the left and its matching option on the right, and then click on **Submit**.

	Description		Keywords
(A)	Displays the default error message for an error.	(1)	`RETURN`
(B)	Specifies an integer value to be returned through the stored procedure.	(2)	`@@ERROR`
(C)	Specifies the level of nesting of the current procedure.	(3)	`ERROR_MESSAGE()`
(D)	Returns the line number that caused the error.	(4)	`@@NESTLEVEL`
(E)	Returns the error number for the error in the last Transact-SQL statement.	(5)	`@@ERROR_LINE`

Submit

39 | 16

Score

Knowledge Check

Can you match the keywords in SQL Server 2005 against their corresponding descriptions?

Click an option on the left and its matching option on the right, and then click on **Submit**.

	Description		Keywords
(A)	Displays the default error message for an error.	(1)	RETURN
(B)	Specifies an integer value to be returned through the stored procedure.	(2)	@@ERROR
(C)	Specifies the level of nesting of the current procedure.	(3)	ERROR_MESSAGE()
(D)	Returns the line number that caused the error.	(4)	@@NESTLEVEL
(E)	Returns the error number for the error in the last Transact-SQL statement.	(5)	@@ERROR_LINE

Correct

The correct answers are displayed.

Submit

39 | 16

Score

Knowledge Check

Which of these statements about returning values and parameters in SQL Server 2005 are true and which statements are false?

Select an option for each statement and then click on **Submit**.

	Statements	True	False
(A)	The return code indicates the execution status of the stored procedure.	○	○
(B)	The `RETURN` function specifies the return value for a stored procedure.	○	○
(C)	Input parameters pass values to a stored procedure.	○	○
(D)	The return code returns 1 if the procedure is successfully executed.	○	○
(E)	The `OUTPUT` keyword specifies that the variables are involved in passing values from the called procedure to the calling program.	○	○

Submit

40 | 16

Score

Knowledge Check

Which of these statements about returning values and parameters in SQL Server 2005 are true and which statements are false?

Select an option for each statement and then click on **Submit**.

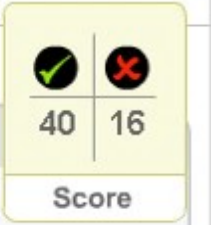
	Statements	True	False
(A)	The return code indicates the execution status of the stored procedure.	◉	○
(B)	The `RETURN` function specifies the return value for a stored procedure.	○	◉
(C)	Input parameters pass values to a stored procedure.	◉	○
(D)	The return code returns 1 if the procedure is successfully executed.	○	◉
(E)	The `OUTPUT` keyword specifies that the variables are involved in passing values from the called procedure to the calling program.	◉	○

Correct

The correct answers are displayed.

Submit

Knowledge Check



Which of these statements about nested procedures and error handling in SQL Server 2005 are true and which statements are false?

Select an option for each statement and then click on **Submit**.

	Statements	True	False
(A)	When a stored procedure is executed from another stored procedure, the procedures are said to be nested.	○	○
(B)	When the level of nesting exceeds 24, the calling chain fails.	○	○
(C)	When an error is detected inside a `TRY` block, control is passed to the corresponding `CATCH` block.	○	○
(D)	When an error occurs in the `TRY` block, the statements following the statement that caused the error are executed only after the `CATCH` block is executed.	○	○
(E)	When the `@@ERROR` function is called, an error message is returned for the error occurring in the last executed statement.	○	○

Submit

41 | 16

Score

Knowledge Check

Which of these statements about nested procedures and error handling in SQL Server 2005 are true and which statements are false?

Select an option for each statement and then click on **Submit**.

	Statements	True	False
(A)	When a stored procedure is executed from another stored procedure, the procedures are said to be nested.	◉	○
(B)	When the level of nesting exceeds 24, the calling chain fails.	○	◉
(C)	When an error is detected inside a `TRY` block, control is passed to the corresponding `CATCH` block.	◉	○
(D)	When an error occurs in the `TRY` block, the statements following the statement that caused the error are executed only after the `CATCH` block is executed.	○	◉
(E)	When the `@@ERROR` function is called, an error message is returned for the error occurring in the last executed statement.	○	◉

Correct

The correct answers are displayed.

Submit

Lesson Review

In this last lesson, **Working with Stored Procedures**, you learnt to:

- Explain how to return values in stored procedures.
- Explain how to use Parameters in stored procedures.
- Describe nested stored procedures.
- Explain handling of error messages in stored procedures.

Module Summary

In this module, **Managing Views**, you learnt about:

- Viewing Information
- Modifying and Dropping
- Working with Stored Procedures

Click on each link for a summary of the lesson.

Module Summary

In this module, **Managing Views**, you learnt about:

- Viewing Information
- Modifying and Dropping
- Working with Stored Procedures

Click on each link for a summary of the lesson.

Viewing Information

The definition of the stored procedure refers to the procedure name, the input and output parameters as well as the body of the procedure. The `sp_helptext` procedure can be used to display the stored procedure definition. An object that references another object is considered dependent on that object. The `sp_depends` procedure is used to display the information about database object dependencies.

Module Summary

In this module, **Managing Views**, you learnt about:

- Viewing Information
- Modifying and Dropping
- Working with Stored Procedures

Click on each link for a summary of the lesson.

Modifying and Dropping Stored Procedures

In a stored procedure, statements or parameters can be changed by either re-creating the stored procedure or by altering an existing stored procedure. A stored procedure can be dropped if it is not needed anymore.