**What is a Stored Procedure?**

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

**Stored Procedure Syntax**

CREATE PROCEDURE *procedure\_name*  
AS  
*sql\_statement*  
GO;

CREATE PROCEDURE `new\_procedure` ()

BEGIN

END

**Execute a Stored Procedure**

EXEC *procedure\_name*;

**Demo Database**

Below is a selection from the "Customers" table in the Northwind sample database:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CustomerID | CustomerName | ContactName | Address | City | PostalCode | Country |
| 1 | Alfreds Futterkiste | Maria Anders | Obere Str. 57 | Berlin | 12209 | Germany |
| 2 | Ana Trujillo Emparedados y helados | Ana Trujillo | Avda. de la Constitución 2222 | México D.F. | 05021 | Mexico |
| 3 | Antonio Moreno Taquería | Antonio Moreno | Mataderos 2312 | México D.F. | 05023 | Mexico |
| 4 | Around the Horn | Thomas Hardy | 120 Hanover Sq. | London | WA1 1DP | UK |
| 5 | Berglunds snabbköp | Christina Berglund | Berguvsvägen 8 | Luleå | S-958 22 | Sweden |

**Stored Procedure Example**

The following SQL statement creates a stored procedure named "SelectAllCustomers" that selects all records from the "Customers" table:

**Example**

CREATE PROCEDURE SelectAllCustomers  
AS  
SELECT \* FROM Customers  
GO;

Execute the stored procedure above as follows:

**Example**

EXEC SelectAllCustomers;

**Stored Procedure With One Parameter**

The following SQL statement creates a stored procedure that selects Customers from a particular City from the "Customers" table:

**Example**

CREATE PROCEDURE SelectAllCustomers @c varchar(30)  
AS  
SELECT \* FROM Customers WHERE City = @c  
GO;

Execute the stored procedure above as follows:

**Example**

EXEC SelectAllCustomers @c = "London";

**Stored Procedure With Multiple Parameters**

Setting up multiple parameters is very easy. Just list each parameter and the data type separated by a comma as shown below.

The following SQL statement creates a stored procedure that selects Customers from a particular City with a particular PostalCode from the "Customers" table:

**Example**

CREATE PROCEDURE SelectAllCustomers @City nvarchar(30), @PostalCode nvarchar(10)  
AS  
SELECT \* FROM Customers WHERE City = @City AND PostalCode = @PostalCode  
GO;

Execute the stored procedure above as follows:

**Example**

EXEC SelectAllCustomers @City = "London", @PostalCode = "WA1 1DP";

**The SQL BACKUP DATABASE Statement**

The BACKUP DATABASE statement is used in SQL Server to create a full back up of an existing SQL database.

**Syntax**

BACKUP DATABASE *databasename*  
TO DISK = '*filepath*';

**The SQL BACKUP WITH DIFFERENTIAL Statement**

A differential back up only backs up the parts of the database that have changed since the last full database backup.

**Syntax**

BACKUP DATABASE *databasename*  
TO DISK = '*filepath*'  
WITH DIFFERENTIAL;

**BACKUP DATABASE Example**

The following SQL statement creates a full back up of the existing database "testDB" to the D disk:

**Example**

BACKUP DATABASE testDB  
TO DISK = 'D:\backups\testDB.bak';

**Tip:** Always back up the database to a different drive than the actual database. Then, if you get a disk crash, you will not lose your backup file along with the database.

**BACKUP WITH DIFFERENTIAL Example**

The following SQL statement creates a differential back up of the database "testDB":

**Example**

BACKUP DATABASE testDB  
TO DISK = 'D:\backups\testDB.bak'  
WITH DIFFERENTIAL;

**Tip:** A differential back up reduces the back up time (since only the changes are backed up).