
Technical Note

Executing SQL Server Stored Procedures with REP++

Author: R&D Department

Publication date: January 10, 2007



© 2007 Consyst SQL Inc. All rights reserved.

Executing SQL Server Stored Procedures with REP++

Overview

A stored procedure is a subroutine (or procedure) that is physically stored within a database. A stored procedure is modular and runs directly on the database engine, which is generally faster at processing database requests.

The exact implementation of stored procedures varies from one database system to another. Most major database vendors support them in some form. They are usually written in a proprietary database language like T-SQL for Microsoft SQL Server, PL/SQL for Oracle database or PL/PgSQL for PostgreSQL.

This article describes how to execute **SQL Server** stored procedures with REP++ and how to get scalar or result set return values.

Executing stored procedures that do not return any value

In this section, you will create a simple stored procedure that attempts to create a new client. Then, you will write the code that uses REP++ to call your stored procedure. If the client already exists, you will display a message box containing the returned error message.

1. Open the Contact Management Demo database and create the **CreateClient** stored procedure:

```
CREATE procedure CreateClient (
    @ClientFirstName    varchar(40),
    @ClientLastName    varchar(40)
)
as
declare    @Code    varchar(16)

if len(@ClientFirstName) > 4
    Set @Code = substring(@ClientFirstName, 1, 4)
else
    Set @Code = @ClientFirstName

if len(@ClientLastName) > 4
    Set @Code = upper(@Code + substring(@ClientLastName, 1, 4))
else
    Set @Code = upper(@Code + @ClientLastName)

insert Demo_Client (Code,
                    ClientFirstName,
                    ClientLastName,
                    ClientType,
                    ClientSalesToDate,
                    CreationDate,
                    ModificationDate)

values (@Code,
        @ClientFirstName,
        @ClientLastName,
        6,
```

Technical Note

```
0,  
getdate(),  
getdate())  
GO
```

2. Create a new Windows® application project. Make sure that you add a reference to **sddnet.dll** (The REP++ Class Library for .NET).
3. Add a button to the default form and handle its **Click** event to call the **CreateClient** stored procedure as follows:

```
private void button1_Click(object sender, EventArgs e) {  
    RepPP.Application app;  
    Connection        connection;  
    SqlCommand        sqlCommand;  
    ErrorCode         errCodeResult;  
    string            strSqlCmd;  
    string            strMessage;  
    bool              bSuccess = false;  
  
    using (app = RepPP.Application.CreateFromRes()) {  
        connection = app.DataConnection;  
        strSqlCmd = @"execute CreateClient @ClientFirstName=:P1,  
                                @ClientLastName=:P2";  
  
        sqlCommand = connection.SqlCommands.Open(strSqlCmd);  
        sqlCommand.SetParameterValue("P1", "Santa", FieldType.sdFieldString);  
        sqlCommand.SetParameterValue("P2", "Clause", FieldType.sdFieldString);  
        try {  
            errCodeResult = (ErrorCode)sqlCmd.Execute();  
            if (errCodeResult != ErrorCode.sdNoErr) {  
                strMessage = "Cannot execute the stored procedure";  
                strMessage += "\nRep++ Error Code: " + errCodeResult;  
                strMessage += "\nDB Error Code:    " + connection.ErrorCode;  
                strMessage += "\nDB Error Message: " + connection.ErrorMessage;  
                MessageBox.Show(strMessage);  
            } else {  
                MessageBox.Show("The stored procedure executed successfully!");  
                bSuccess = true;  
            }  
        } finally {  
            if (bSuccess) {  
                connection.Commit();  
            } else {  
                connection.Rollback();  
            }  
        }  
    }  
}
```

4. Build your project and run the application.

The first time you click the button, the client will be created successfully. The second time, however, you will get an error message stating that the client exists already!

Executing stored procedures that return scalar output parameters

You will now write a simple stored procedure that creates a new address for a given client and returns the auto-generated ID of the newly added address. Then, you will write the code that uses REP++ to call your stored procedure and retrieve the value of the address ID.

1. Open the Contact Management Demo database and create the **CreateAddress** stored procedure:

Technical Note

```
CREATE procedure CreateAddress (
    @ClientCode      varchar(16),
    @AddressLine1    varchar(80),
    @City            varchar(40),
    @PostalCode      varchar(6),
    @AddressCode     int          OUT
)
as
begin
    Insert Demo_Address(ClientCode, Address_Line1, City, PostalCode)
    values (@ClientCode, @AddressLine1, @City, @PostalCode)
    -- Get Autoincrement Value
    Set @AddressCode = SCOPE_IDENTITY()
end
GO
```

2. Add a second button to the default form and handle its **Click** event to call the **CreateAddress** stored procedure as follows:

```
private void button2_Click(object sender, EventArgs e) {
    RepPP.Application app;
    Connection        connection;
    SqlCommand        sqlCommand;
    ErrorCode         errorCodeResult;
    string            strSQLCmd;
    string            strMessage;
    bool              bSuccess = false;

    using (app = RepPP.Application.CreateFromRes()) {
        connection = app.DataConnection;
        strSQLCmd  = @"execute CreateAddress @ClientCode=:P1,
                                @AddressLine1=:P2,
                                @City=:P3,
                                @PostalCode=:P4,
                                @AddressCode=:P5 OUTPUT";

        sqlCommand = connection.SqlCommands.Open(strSQLCmd);
        sqlCommand.SetParameterValue("P1", "SANTCLAU", FieldType.sdFieldString);
        sqlCommand.SetParameterValue("P2", "Christmas St.", FieldType.sdFieldString);
        sqlCommand.SetParameterValue("P3", "North Pole", FieldType.sdFieldString);
        sqlCommand.SetParameterValue("P4", "HOHOHO", FieldType.sdFieldString);
        sqlCommand.SetParameterValue("P5",
                                    int.MaxValue.ToString(),
                                    FieldType.sdFieldNumeric);

        try {
            errorCodeResult = (ErrorCode)sqlCmd.Execute();
            if (errorCodeResult != ErrorCode.sdNoErr) {
                strMessage = "Cannot execute the stored procedure";
                strMessage += "\nRep++ Error Code: " + errorCodeResult;
                strMessage += "\nDB Error Code: " + connection.ErrorCode;
                strMessage += "\nDB Error Message: " + connection.ErrorMessage;
                MessageBox.Show(strMessage);
            } else {
                strMessage = "The stored procedure executed successfully.";
                strMessage += "\nNew address code: " + sqlCommand.GetParameterValue("P5");
                MessageBox.Show(strMessage);
                bSuccess = true;
            }
        } finally {
            if (bSuccess) {
                connection.Commit();
            } else {
                connection.Rollback();
            }
        }
    }
}
```

Technical Note



In order to get the value of a stored procedure's scalar output parameter, you must allocate a buffer that is sufficient for its type. For instance:

- If the return value is an integer, your buffer must be able to hold the string representation of the largest integer.
- If the return value is a string (e.g. `varchar(40)`), your buffer must be able to hold the maximum number of characters allowed (i.e. 40).

Use the **`SqlCommand.SetParameterValue`** method to allocate a sufficient buffer for a scalar output parameter.

3. Build your project and run the application.

Every time you execute the **`CreateAddress`** stored procedure, an address is created and the auto-generated ID of the address is displayed.

Executing stored procedures that return result sets

In this section, you will create a simple stored procedure that returns the client codes of all the clients. Then, you will write the code that uses REP++ to call your stored procedure and retrieve the result set.

1. Open the Contact Management Demo database and create the **`GetClients`** stored procedure:

```
CREATE PROCEDURE GetClients
AS
BEGIN
    SELECT CLIENTCODE
    FROM DEMO_CLIENT;
END
GO
```

2. Add a third button to the default form and handle its **`Click`** event to call the **`GetClients`** stored procedure and fetch the client codes as follows:

```
private void button3_Click(object sender, EventArgs e) {
    RepPP.Application app;
    SqlCursor sqlCur = null;
    ErrorCode eErr;
    string strMessage;

    using (app = RepPP.Application.CreateFromRes()) {
        eErr = (ErrorCode)app.DataConnection.Execute("execute GetClients", ref sqlCur);
        if (eErr == ErrorCode.sdNoErr) {
            strMessage = "Clients:";
            while (sqlCur.Next() == (int)ErrorCode.sdNoErr) {
                strMessage += "\n" + sqlCur.GetColumnValue("CLIENTCODE");
            }
            sqlCur.Close();
            MessageBox.Show(strMessage);
        }
    }
}
```

3. Build your project and run the application.

Every time you execute the **`GetClients`** stored procedure, a message box containing the list of client codes is displayed.