

Load and Search MySQL Data Using VB.NET 2005 in Windows Applications

Written by
Dr. Ernest Bonat, Ph.D.
Visual WWW, Inc.



Visual WWW, Inc.

'Created by Developers for Developers'

www.evisualwww.com | info@evisualwww.com

Copyright @ 2000 - 2007 Visual WWW, Inc. All right reserved

Table of Content

Table of Content.....	2
Introduction	2
Required Software	2
MySQL Data Load Using DataGridView Control	2
MySQL Data Search Using Stored Procedures.....	10
MySQL Data Search Using DataView Object.....	15
Conclusions.....	16
Visual WWW Legal Notice	17

Introduction

MySQL data load and search are very important business requirements in any Windows or Internet web application development. In general, any application needs to show a result set of data and/or a single record to the end-users. In Windows applications it is very popular to show a result set of data by using the DataGridView, ListView or TreeView controls. A single record can be shown by the simple combination of the following controls: TextBox, ComboBox, ListBox, CheckBox, RadioButton, etc. MySQL data search is provided by using the required ADO.NET data objects and by refreshing the controls if necessary. These two processes, data load and search, should be fast and should be done with the proper code which depends on the controls in the Windows Form or Web Page. In this article I will show you how load and sort MySQL data using the DataGridView control. To search MySQL data the LIKE SQL operator will be used. Both programming implementations are done by using stored procedures for MySQL 5.0 database engine.

Required Software

- [MySQL Database Server 5.0.45](#)
- [MySQL Connector/NET 5.0.8](#)
- [Toad for MySQL Freeware 3.0.0](#)
- [Microsoft Visual Basic 2005 Express Edition](#)

MySQL Data Load Using DataGridView Control

I have received a lot of questions from many VB.NET developers around the world about how to load MySQL data into the DataGridView Windows control. These questions have been sent to me by e-mail (ebonat@evisualwww.com) or have been posted at www.vbmysql.com website forum. In fact, I would like to invite you to register and participate in our discuss forum. At this time, I'm in charge of keeping this website alive and responding all your MySQL/VB (VB 6.0 and VB.NET 2005) related questions. In general all of the developers knew how to load the data by using the DataSet or DataView ADO.NET objects. Most of the questions were about the DataGridView format and data search after the grid was loaded. For some reason(s) all of them were using dynamic SQL instead of stored procedures or functions. It's very hard for me to believe that we still have developers up there doing dynamic (embedding) or/and parameterized SQL programming in their applications after MySQL AB Corporation released MySQL server version 5.0 with stored procedures, functions, triggers and views database objects development. I have made the decision to use stored procedures in this article to show them how easy is to develop and call them in VB.NET 2005. I developed these stored procedures using [Toad for MySQL Freeware 3.0.0](#) freeware version from [Quest Software, Inc.](#) In my experience, Toad for MySQL is a powerful database management tool used to design and develop MySQL database objects like stored procedures, functions, triggers and views. In my previous article "[MySQL Data Loading with Lookup Tables](#)", I show how to execute stored procedures in MySQL 5.0/VB.NET 2005 environment. Feel free to read the article and download the source code to apply in your real production applications.

In Windows database applications the data gets loaded in the load event of the form. Just in case, you need to be careful when to use this event to load the data as you don't want to load a lot of data that it may slow down

your form. I created a simple solution project in VB.NET 2005 with a form represented in Figure 1. This form named SearchForm contains a DataGridView with the contact info and a Contact Name TextBox entry to search for it.

Contact Name	Birth Date	No Of Children	Married	PC	Laptop	Monthly Salary	Comment
Ana Dean	5/14/1988	2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$50,000.00	Computer Trainer
Ernest Bonat	5/29/1976	4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$70,000.00	Windows Developer
John Sams	5/8/1986	0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	\$48,000.00	Technical Writer
Jonathan Jones	2/28/2006	2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$65,000.00	Senior Business Analyst
Michael Rivas	3/8/2001	3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	\$55,000.00	Senior Web Design
Mike Black	5/3/1964	2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$75,000.00	Database Manager
Sherri Jones	6/4/1975	1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	\$35,000.00	Technical Writer

Figure 1: MySQL data load and search form

Listing 1 shows the load event of the Form SearchForm where the procedure SearchFormLoad() is called.

```
Private Sub SearchForm_Load(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) _  
    Handles MyBase.Load  
    Call SearchFormLoad()  
End Sub
```

Listing 1: Load event of the form SearchForm

The code of the procedure SearchFormLoad() is shown in Listing 2. In this procedure the ADO.NET Connection String (Server=xxx;Database=xxx;Uid=xxx;Pwd=xxx;) is passed as a constructor to the class SearchClass (Listing 3). Just a reminder that this Connection String value is defined and stored in the app.config file as I explained in the article "[Define and Store MySQL ADO Connection String in VB.NET 2005](#)". The Using statement creates and destroys the instance of this class SearchObject. The procedure DataLoadingSearching() (Listing 4) is calling by passing the DataGridView and the record count variable by reference. If an error does not occurred the record count value is shown in the TextBox CountTextBox.

```
Private Sub SearchFormLoad()  
    Cursor = Cursors.WaitCursor  
    Try  
        Using SearchObject As New SearchClass(mMySQLConnectionString)  
            Call SearchObject.DataLoadingSearching(SearchDataGridView, _  
                mRecordCountInt32, _  
                mErrorMessageString)  
            If IsNothing(mErrorMessageString) Then  
                CountTextBox.Text = FormatNumber(mRecordCountInt32.ToString, 0)  
            Else  
                MessageBox.Show(mErrorMessageString, _  
                    Me.Text, _  
                    MessageBoxButtons.OK, _  
                    MessageBoxIcon.Error)  
            End If  
        End Using  
    End Try  
End Sub
```

```
        TextBoxName.Focus()
    End Using
    Catch exError As Exception
        MessageBox.Show(exError.Message)
    End Try
    Cursor = Cursors.Default
End Sub
```

Listing 2: Procedure SearchFormLoad() for loading the data into the DataGridView SearchDataGridView

Listing 3 shows the classes SearchClass and ObjectDisposeClass. Both classes are defined in the Namespace SearchLibrary with MySqlConnection imported library from [Connector/NET 5.0.8](#). As you can see the class ObjectDisposeClass implements the IDisposable interface to release unmanaged resources created by class SearchClass. This implementation needs to be inherited by the class SearchClass so the Using statement can be used properly. The parameterized contractor has been included to store the ADO.NET Connection String mMySQLConnectionString as I explained before.

```
Imports MySql.Data.MySqlClient
Namespace SearchLibrary
    Public Class SearchClass
        Inherits ObjectDisposeClass
        Private mDataView As DataView
        Private mMySQLConnectionString As String
        Public Sub New(ByVal pMySQLConnectionString As String)
            mMySQLConnectionString = pMySQLConnectionString
        End Sub
    End Class
    Public Class ObjectDisposeClass
        Implements IDisposable
        Private disposedValue As Boolean = False
        Protected Overridable Sub Dispose(ByVal disposing As Boolean)
            If Not Me.disposedValue Then
                If disposing Then
                    ' TODO: free managed resources when explicitly called
                End If
                ' TODO: free shared unmanaged resources
            End If
            Me.disposedValue = True
        End Sub
        Public Sub Dispose() Implements IDisposable.Dispose
            Dispose(True)
            GC.SuppressFinalize(Me)
        End Sub
    End Class
End Namespace
```

Listing 3: SearchClass and ObjectDisposeClass classes code

Listing 4 shows the overloaded procedure DataLoadingSearching() code. This is the main procedure that formats and loads the contact name data into the DataGridView. The Using statement is used for the following ADO.NET objects: Connection object MySqlConnection, Command object MySqlCommand, DataAdapter object MySqlDataAdapter and DataSet object DataSet. The command text property of the Command object is defined by the user stored procedure `usp_select_all_data` (Listing 5). The Command object MySqlCommand is passed to the DataAdapter object MySqlDataAdapter as a contractor that fills the DataSet object DataSet. The DataView object mDataView was created by using the Tables property of the DataSet. This DataView mDataView is the required object to load the DataGridView with data by assigned it to the DataSource property. As you can see I did not use any dynamic or parameterized SQL statement to set the command text property.

In Microsoft SQL Server 2005 and Oracle 10g, the stored procedures are precompiled SQL statements in the database engine as a single unit of code. They are parsed and optimized when they are first executed, and a compiled version of the stored procedure (execution plan) remains in memory cache for later use. This results in tremendous application performance boosts. In MySQL 5.0 database engine the stored procedures are recompiled each time when a connection is made to the database. There is no global stored procedure compile cache (execution plan) on the database server. I believe in future releases of MySQL we will see this important implementation. In my sixteen years of experience developing database business applications for many clients, I have found the following three main benefits of stored procedures (functions and triggers) implementation:

- Improve application performance by reducing network traffic
- Provide a single tier of business rules development and maintenance
- Enhance application security by reducing SQL injection attacks

I understand that development MySQL stored procedures, functions and triggers requires a good knowledge of the SQL:2003 programming language. As a developer, it's your responsibility to learn this language and provide for your clients with the best of business applications development. I can't image being hired today as a IT Application Developer and don't know how to develop stored procedures, functions and triggers in MySQL 5.0 (SQL:2003), Oracle 10g (PL/SQL) or MS SQL Server 2005 (T-SQL). Remember every business application that you build for a client your name is behind it. So, always do the best programming practice you can provide, even if you need to spend a lot time leaning out site the client environment. In my particular case, in 2002 I learned VB.NET from 4:00 am to 8:00 am – just a friendly thought. As I said in my article "[MySQL Data Loading with Lookup Tables](#)", a good starting point could be reading a book published in 2006 by [O'Reilly Media, Inc.](#), ISBN: 0-596-10089-2, "[MySQL Stored Procedure Programming](#)" by Guy Harrison and Steven Feuerstein. I highly recommend reading this book for any Windows or Internet web application development with MySQL 5.0 database engine.

```
Public Overloads Sub DataLoadingSearching(ByRef pDataGridViewControl As DataGridView, _
                                         ByRef pRecordCountInt32 As Int32, _
                                         ByRef pErrorMessageString As String)
    Try
        Using mMySQLConnection As New MySqlConnection(mMySQLConnectionString)
            mMySQLConnection.Open()
            Using mMySQLCommand As New MySqlCommand
                With mMySQLCommand
                    .Connection = mMySQLConnection
                    .CommandType = CommandType.StoredProcedure
                    .CommandText = "usp_select_all_data"
                End With
                Using mMySQLDataAdapter As New MySqlDataAdapter(mMySQLCommand)
                    Using mDataSet As New DataSet
                        mDataSet.Clear()
                        mMySQLDataAdapter.Fill(mDataSet, "data")
                        mDataView = New DataView
                        mDataView.Table = mDataSet.Tables("data")
                        pRecordCountInt32 = mDataView.Count
                    End Using
                End Using
            End Using
        End Using
        Call DataGridViewFormat(pDataGridViewControl)
        With pDataGridViewControl
            .DataSource = mDataView
            .AutoSizeRows()
        End With
    Catch exError As Exception
        pErrorMessageString = exError.Message
    End Try
End Sub
```

```
End Try  
End Sub
```

Listing 4: Overloaded procedure DataLoadingSearching() (data load) with user stored procedure `usp_select_all_data`

```
DROP PROCEDURE IF EXISTS `vwww`.`usp_select_all_data`;  
CREATE PROCEDURE `usp_select_all_data`()  
BEGIN  
    SELECT `data`.`id`, `data`.`name`, `data`.`birthdate`, `data`.`numberofchildren`, `data`.`married`,  
        `data`.`computerpc`, `data`.`computerlaptop`, `data`.`salary`, `data`.`comment`  
    FROM `data`  
    ORDER BY `data`.`name`;  
END;
```

Listing 5: User stored procedure `usp_select_all_data` to select all columns and data from table `data`

Listing 6 shows the procedure DataLoading_DynamicSQL() with dynamic SQL statement SQL "SELECT data.id, data.name, data.birthdate, data.numberofchildren, data.married, data.computerpc, data.computerlaptop, data.salary, data.comment FROM data ORDER BY data.name". As you can see the command type property of the Command object mMySQLCommand is set to Text. I have provided this code so you can see the real difference between the procedures DataLoadingSearching() and DataLoading_DynamicSQL() (Listing 4 and 6).

```
Public Sub DataLoading_DynamicSQL(ByRef pDataGridViewControl As DataGridView, _  
    ByRef pRecordCountInt32 As Int32, _  
    ByRef pErrorMessageString As String)  
    Dim SQLString As String  
    Try  
        SQLString = "SELECT data.id, data.name, data.birthdate, data.numberofchildren,  
            data.married, data.computerpc, data.computerlaptop, data.salary,  
            data.comment FROM data ORDER BY data.name"  
        Using mMySQLConnection As New MySqlConnection(mMySQLConnectionString)  
            mMySQLConnection.Open()  
            Using mMySQLCommand As New MySqlCommand  
                With mMySQLCommand  
                    .Connection = mMySQLConnection  
                    .CommandType = CommandType.Text  
                    .CommandText = SQLString  
                End With  
                Using mMySQLDataAdapter As New MySqlDataAdapter(mMySQLCommand)  
                    Using mDataSet As New DataSet  
                        mDataSet.Clear()  
                        mMySQLDataAdapter.Fill(mDataSet, "data")  
                        mDataView = New DataView  
                        mDataView.Table = mDataSet.Tables("data")  
                        pRecordCountInt32 = mDataView.Count  
                    End Using  
                End Using  
            End Using  
            Call DataGridViewFormat(pDataGridViewControl)  
            With pDataGridViewControl  
                .DataSource = mDataView  
                .AutoResizeRows()  
            End With  
        Catch exError As Exception  
            pErrorMessageString = exError.Message  
        End Try  
    End Try
```

End Sub

Listing 6: Procedure DataLoading_DynamicSQL() with dynamic SQL "SELECT data.id, data.name, data.birthdate, data.numberofchildren, data.married, data.computerpc,data.computerlaptop, data.salary, data.comment FROM data ORDER BY data.name"

Before the data is loaded into the DataGridView, it must be formatted properly according to the column names and data types defined in the SQL SELECT statement in the user stored procedure `usp_select_all_data` (Listing 5). This statement selects all columns from the table `data` defined in Listing 7.

```
DROP TABLE IF EXISTS `vwww`.`data`
CREATE TABLE `data` (
  `id` int(20) NOT NULL auto_increment,
  `name` varchar(50) NOT NULL,
  `birthdate` date NOT NULL,
  `numberofchildren` smallint(20) default NULL,
  `married` tinyint(1) default '0',
  `computerpc` tinyint(1) default '0',
  `computerlaptop` tinyint(1) default '0',
  `salary` double(10,2) default NULL,
  `comment` varchar(300) default NULL,
  PRIMARY KEY (`id`),
  KEY `ix1_data` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Listing 7: Table `data` definition

The procedure DataGridViewFormat() formats the DataGridView as shown in Listing 8. One particular thing to consider is that the first formatted column represents the `id` field and it should be hidden from the end-user. We can do that by setting the Visible property of the column to False after the column is formatted.

```
Private Sub DataGridViewFormat(ByRef pDataGridViewControl As DataGridView)
  With pDataGridViewControl
    .AlternatingRowsDefaultCellStyle.BackColor = Color.LightCyan
    .SelectionMode = DataGridViewSelectionMode.FullRowSelect
    .AllowUserToOrderColumns = True
    .AllowUserToDeleteRows = False
    .AllowUserToAddRows = False
    .ReadOnly = True
    .MultiSelect = False
    .Columns.Clear()

    Dim ColumnContactID As New DataGridViewTextBoxColumn
    With ColumnContactID
      .DataPropertyName = "id"
      .Name = "id"
      .HeaderText = "ID"
      .AutoSizeMode = DataGridViewAutoSizeColumnMode.ColumnHeader
      .DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft
    End With
    .Columns.Add(ColumnContactID)
    ColumnContactID.Visible = False

    Dim ColumnContactName As New DataGridViewTextBoxColumn
    With ColumnContactName
      .DataPropertyName = "name"
      .Name = "name"
      .HeaderText = "Contact Name"
      .AutoSizeMode = DataGridViewAutoSizeColumnMode.ColumnHeader
    End With
  End With
End Sub
```

```
.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft
End With
.Columns.Add(ColumnContactName)

Dim ColumnBirthDate As New DataGridViewTextBoxColumn
With ColumnBirthDate
    .DataPropertyName = "birthdate"
    .Name = "birthdate"
    .HeaderText = "Birth Date"
    .DefaultCellStyle.Format = "d"
    .AutoSizeMode = DataGridViewAutoSizeColumnMode.ColumnHeader
    .DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter
End With
.Columns.Add(ColumnBirthDate)

Dim ColumnNoOfChildren As New DataGridViewTextBoxColumn
With ColumnNoOfChildren
    .DataPropertyName = "numberofchildren"
    .Name = "numberofchildren"
    .HeaderText = "No Of Children"
    .AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells
    .DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter
End With
.Columns.Add(ColumnNoOfChildren)

Dim ColumnMarried As New DataGridViewCheckBoxColumn
With ColumnMarried
    .DataPropertyName = "married"
    .Name = "married"
    .HeaderText = "Married"
    .ThreeState = True
    .TrueValue = 1
    .FalseValue = 0
    .AutoSizeMode = DataGridViewAutoSizeColumnMode.ColumnHeader
    .DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter
End With
.Columns.Add(ColumnMarried)

Dim ColumnComputerPC As New DataGridViewCheckBoxColumn
With ColumnComputerPC
    .DataPropertyName = "computerpc"
    .Name = "computerpc"
    .HeaderText = "PC"
    .ThreeState = True
    .TrueValue = 1
    .FalseValue = 0
    .AutoSizeMode = DataGridViewAutoSizeColumnMode.ColumnHeader
    .DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter
End With
.Columns.Add(ColumnComputerPC)

Dim ColumnComputerLaptop As New DataGridViewCheckBoxColumn
With ColumnComputerLaptop
    .DataPropertyName = "computerlaptop"
    .Name = "computerlaptop"
    .HeaderText = "Laptop"
    .ThreeState = True
```

```
.TrueValue = 1
.FalseValue = 0
.AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells
.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter
End With
.Columns.Add(ColumnComputerLaptop)

Dim ColumnSalary As New DataGridViewTextBoxColumn
With ColumnSalary
.DataPropertyName = "salary"
.Name = "salary"
.HeaderText = "Monthly Salary"
.DefaultCellStyle.Format = "c"
.AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells
.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight
End With
.Columns.Add(ColumnSalary)

Dim ColumnComment As New DataGridViewTextBoxColumn
With ColumnComment
.DataPropertyName = "comment"
.Name = "comment"
.HeaderText = "Comment"
.AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells
.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft
.DefaultCellStyle.NullValue = "None"
End With
.Columns.Add(ColumnComment)
End With
End Sub
```

Listing 8: Procedure DataGridViewFormat() for formatting the DataGridView SearchDataGridView

MySQL Data Search Using Stored Procedures

Searching for data in any Windows or Internet web application is a must. I haven't found any business application without this requirement. You must find out the best way to implement this requirement for that specific application. A common request from clients is for search functions to be flexible and fast. Because of that, the first idea that comes to my head is stored procedures implementation. As I explained before stored procedures improve application performance by reducing network traffic. To show you this approach the Search button was created (Figure 2). As you can see, after the letter c was entered in the Contact Name TextBox, the end-user clicks on Search button and two records were found. Let's look at the code implementation.

Contact Name	Birth Date	No Of Children	Married	PC	Laptop	Monthly Salary	Comment
Michael Rivas	3/8/2001	3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	\$55,000.00	Senior Web Design
Mike Black	5/3/1964	2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$75,000.00	Database Manager

Figure 2: MySQL data load and search form

Listing 9 shows the code of the click event of the Search button. First of all, before searching for any Contact Name, the end-user needs to enter a string value. To check for it the Length property was used. If the Length of the Contact Name TextBox is equal to zero, a message will show requiring an entry. As you can see from Listing 9 the overloaded procedure DataLoadingSearching() (Listing 10) provides the searching code by passing by value a Contact Name ContactNameString. Let's look closer at this procedure.

```
Private Sub SearchButton_Click(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs)  
    Handles SearchButton.Click  
  
    Cursor = Cursors.WaitCursor  
    Dim ContactNameString As String  
    Try  
        ContactNameString = TextBoxName.Text.Trim  
        If ContactNameString.Length = 0 Then  
            Cursor = Cursors.Default  
            MessageBox.Show("Contact Name is required.", _  
                Me.Text, _  
                MessageBoxButtons.OK, _  
                MessageBoxIcon.Stop)  
            TextBoxName.Focus()  
            Exit Sub  
        End If  
        Using SearchObject As New SearchClass(mMySQLConnectionString)  
            Call SearchObject.DataLoadingSearching(SearchDataGridView, _
```

```
                mRecordCountInt32, _
                ContactNameString, _
                mErrorMessageString)
    If IsNothing(mErrorMessageString) Then
        CountTextBox.Text = FormatNumber(mRecordCountInt32.ToString, 0)
        If mRecordCountInt32 > 0 Then
            SearchDataGridView.Focus()
        Else
            Cursor = Cursors.Default
            MessageBox.Show("Contact Name " & ContactNameString & _
                " was not found. Try again.", _
                Me.Text, _
                MessageBoxButtons.OK, _
                MessageBoxIcon.Error)
            TextBoxName.Focus()
            TextBoxName.SelectAll()
        End If
    Else
        MessageBox.Show(mErrorMessageString, _
            Me.Text, _
            MessageBoxButtons.OK, _
            MessageBoxIcon.Error)
        TextBoxName.Focus()
    End If
End Using
Catch exError As Exception
    MessageBox.Show(exError.Message, _
        Me.Text, _
        MessageBoxButtons.OK, _
        MessageBoxIcon.Error)
End Try
Cursor = Cursors.Default
End Sub
```

Listing 9: Click event of the Search button

The overloaded procedure `DataLoadingSearching()` shown in Listing 10 provides the search by Contact Name. The main idea of this procedure, compare to the first one, lies in the execution of the user stored procedure ``usp_find_name`` with an input parameter `par_contact_name` (Listing 11). In the WHERE clause of the SELECT statement the LIKE comparison with the CONCAT() string function have been included. The combination of both with a wildcard character '%' at the beginning and the end of the parameter `par_contact_name` allows us to search for any Contact Name containing the passing string parameter `pContactNameString`. In VB.NET 2005 to pass a parameter to the stored procedure a `MySQLParameter` object `NameMySQLParameter` needs to be created and added to the parameters collection. Five properties of this parameter object need to be setup properly: `ParameterName`, `Direction`, `MySQLDbType`, `Size` and `Value`. The data type `MySQLDbType` and the size need to be defined according with the column definition in the table ``data``. This is common mistake many developers make when these parameter settings do not correspond with the column definition (data type and size). After this parameter has been added to the parameter collection and the Command object `mMySQLCommand` is passed as a constructor to the `DataAdapter` object `mMySQLDataAdapter`, the `DataGridView` object `mDataGridView` is created in the same way that we explained before in Listing 4.

```
Public Overloads Sub DataLoadingSearching(ByRef pDataGridViewControl As DataGridView, _
    ByRef pRecordCountInt32 As Int32, _
    ByVal pContactNameString As String, _
    ByRef pErrorMessageString As String)
    Dim NameMySQLParameter As New MySQLParameter
    Try
        Using mMySQLConnection As New MySqlConnection(mMySQLConnectionString)
```

```
mMySqlConnection.Open()
Using mMySqlCommand As New MySqlCommand
    With mMySqlCommand
        .Connection = mMySqlConnection
        .CommandType = CommandType.StoredProcedure
        .CommandText = "usp_find_name"
        With NameMySqlParameter
            .ParameterName = "?par_contact_name"
            .Direction = ParameterDirection.Input
            .MySqlDbType = MySqlDbType.VarChar
            .Size = 50
            .Value = pContactNameString
        End With
        .Parameters.Add(NameMySqlParameter)
    End With
    Using mMySqlDataAdapter As New MySqlDataAdapter(mMySqlCommand)
        Using mDataSet As New DataSet
            mDataSet.Clear()
            mMySqlDataAdapter.Fill(mDataSet, "data")
            mDataView = New DataView
            mDataView.Table = mDataSet.Tables("data")
            pRecordCountInt32 = mDataView.Count
        End Using
    End Using
End Using
Call DataGridViewFormat(pDataGridViewControl)
With pDataGridViewControl
    .DataSource = mDataView
    .AutoSizeRows()
End With
End Using
Catch exError As Exception
    pErrorMessageString = exError.Message
Finally
    If Not IsNothing(NameMySqlParameter) Then
        NameMySqlParameter = Nothing
    End If
End Try
End Sub
```

Listing 10: Overloaded procedure DataLoadingSearching() (data search) with stored procedure `usp_find_name`

```
DROP PROCEDURE IF EXISTS `vwww`.`usp_find_name`;
CREATE PROCEDURE `usp_find_name`(
    IN par_contact_name VARCHAR(50)
)
BEGIN
    SELECT `data`.`id`, `data`.`name`, `data`.`birthdate`, `data`.`numberofchildren`, `data`.`married`,
        `data`.`computerpc`, `data`.`computerlaptop`, `data`.`salary`, `data`.`comment`
    FROM `data`
    WHERE `data`.`name` LIKE CONCAT('%', par_contact_name, '%')
    ORDER BY `data`.`name`;
END;
```

Listing 11: User stored procedure `usp_find_name` to search for Contact Name using LIKE operator and string function CONCAT()

Listing 12 shows the procedure `DataSearching_ DynamicSQL()` using dynamic SQL statement "SELECT data.id, data.name, data.birthdate, data.numberofchildren, data.married, data.computerpc, data.computerlaptop, data.salary, data.comment FROM data WHERE data.name LIKE CONCAT('%,' & "" & pContactNameString & "" & ",%') ORDER BY data.name". Comparing with procedure `DataLoadingSearching()` in Listing 10, the parameter object `NameMySqlParameter` is not required because the Contact Name string value `pContactNameString` is included in the dynamic SQL statement. Between procedures `DataLoadingSearching()` and `DataSearching_ DynamicSQL()` you can see a quite big difference. The procedure `DataSearching_ DynamicSQL()` looks more simple than procedure `DataLoadingSearching()`. Whenever the program calls the procedure `DataSearching_ DynamicSQL()` the dynamic SQL statement needs to be compiled by the MySQL database engine decreasing application performance by increasing network traffic. By the way, a very "good malicious" SQL injection attack can look for your Contact Name data and I believe you don't want that to happen. So, dynamic SQL implementation in your business database applications is not a very good programming practice today. Here is what [Wikipedia, The Free Encyclopedia](#), says about SQL injection: "SQL injection is a technique that exploits a security vulnerability occurring in the database layer of an application. The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed. It is in fact an instance of a more general class of vulnerabilities that can occur whenever one programming or scripting language is embedded inside another". Because of that I want you think "twice" if you really need dynamic SQL implementation in your client's business applications.

```
Public Sub DataSearching_ DynamicSQL (ByRef pDataGridViewControl As DataGridView, _
                                     ByRef pRecordCountInt32 As Int32, _
                                     ByVal pContactNameString As String, _
                                     ByRef pErrorMessageString As String)

    Dim SQLString As String
    Try
        SQLString = "SELECT data.id, data.name, data.birthdate, data.numberofchildren,
data.married, data.computerpc, data.computerlaptop, data.salary, data.comment
FROM data WHERE data.name LIKE CONCAT('%,' & "" &
pContactNameString & "" & ",%') ORDER BY data.name"
        Using mMySQLConnection As New MySqlConnection(mMySQLConnectionString)
            mMySQLConnection.Open()
            Using mMySQLCommand As New MySqlCommand
                With mMySQLCommand
                    .Connection = mMySQLConnection
                    .CommandType = CommandType.Text
                    .CommandText = SQLString
                End With
            Using mMySQLDataAdapter As New MySqlDataAdapter(mMySQLCommand)
                Using mDataSet As New DataSet
                    mDataSet.Clear()
                    mMySQLDataAdapter.Fill(mDataSet, "data")
                    mDataView = New DataView
                    mDataView.Table = mDataSet.Tables("data")
                    pRecordCountInt32 = mDataView.Count
                End Using
            End Using
        End Using
        Call DataGridViewFormat(pDataGridViewControl)
        With pDataGridViewControl
            .DataSource = mDataView
            .AutoSizeRows()
        End With
    End Using
    Catch exError As Exception
        pErrorMessageString = exError.Message
    End Try
```

End Sub

Listing 12: Procedure DataSearching_DynamicSQL() with dynamic SQL "SELECT data.id, data.name, data.birthdate, data.numberofchildren, data.married, data.computerpc, data.computerlaptop, data.salary, data.comment FROM data WHERE data.name LIKE CONCAT('%,' & "" & pContactNameString & "" & ',%') ORDER BY data.name"

Before finishing this topic I want to explain why I find implementing stored procedures, functions and triggers in my client's business applications to be very useful. The first thing that comes to my mind is where and how to implement the client's business rules so that I don't need to compile and deploy my application again and again. The answer to this is, code location and organization, and of course, application upgrade and maintenance. I need to find an effective and unique location to develop and maintain these rules easily. How about in the database engine? In general, the database engine is a separated block (tier) of any normal Windows business application. So if, in the future, I implement the business rules in the database engine tier, then any possible upgrades will not require application compilation and deployment to the client's PC. This is possible if the upgrade does not require changing the public interface of the stored procedures (parameters name and data types definition). This covers the third benefit of stored procedures, functions and triggers development. Referring to the code, let's look at a simple example. As we saw before, Listing 11 shows the user stored procedure `usp_find_name` with LIKE operator and string function CONCAT(). So far this stored procedure can find any Contact Name record(s) if the input parameter par_contact_name value is anywhere inside the `data`.`name` column. A possible upgrade could be useful to finding any Contact Name record(s) that starts with any parameter par_contact_name value. The implementation of this upgrade is very simple. It is done by removing the first wildcard character '%' in the CONCAT() string function as show in Listing 13. At this point the upgrade was made to the stored procedure code, not to the application code. So, because of that the application compilation and deployment back to the client's PC is not necessary. The point I'm trying to make is that implementing client's business rules at the database tier allows you to locate, upgrade and maintain them easily. By the way, I have no previous information of any database engine slow down process or resource problems because stored procedures overhead implementation on Oracle, SQL Server or IBM DB2 servers.

```
DROP PROCEDURE IF EXISTS `www`.`usp_find_name`;
CREATE PROCEDURE `usp_find_name`(
    IN par_contact_name VARCHAR(50)
)
BEGIN
    SELECT `data`.`id`, `data`.`name`, `data`.`birthdate`, `data`.`numberofchildren`, `data`.`married`,
        `data`.`computerpc`, `data`.`computerlaptop`, `data`.`salary`, `data`.`comment`
    FROM `data`
    WHERE `data`.`name` LIKE CONCAT(par_contact_name, '%')
    ORDER BY `data`.`name`;
END;
```

Listing 13: User stored procedure to search for Contact Name using LIKE operator and string function CONCAT()

An example of MySQL data searching using the stored procedure `usp_find_name` (Listing 13) is shown in Figure 3. As you can see two records were found searching by letter m at the beginning of the Contact Name data column.

Contact Name	Birth Date	No Of Children	Married	PC	Laptop	Monthly Salary	Comment
Michael Rivas	3/8/2001	3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	\$55,000.00	Senior Web Design
Mike Black	5/3/1964	2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$75,000.00	Database Manager

Figure 3: MySQL data load and search form

MySQL Data Search Using DataView Object

Another way to search for MySQL data is by using Microsoft DataView ADO.NET object. Two main properties of the DataView object are required for data searching: RowFilter() and RowStateFilter(). The property RowFilter() gets or sets the expression used to filter which rows are viewed in the DataView. The other one property RowStateFilter() gets or sets the row state filter used in the DataView. Listing 14 shows the procedure DataSearching_DataView() using the stored procedure `usp_select_all_data` for data selection and the DataView object for searching. As you can see the filter string value FilterSQLString = "name LIKE " & "%" & pContactNameString & "%" does not include the CONCAT() MySQL string function. This function is not part of the Microsoft DataView ADO.NET object. if you include the the CONCAT() function, an error will occurs saying: "The expression contains undefined function call CONCAT()". The property RowStateFilter() is set to the filter string and the RowStateFilter() is set to return current rows state. I did not notice any speed differences between stored procedures and DataView object implementation for MySQL data searching. As you may have already found out I'm a hardcode programmer. Here one of my programming principles: "I control the code and objects, the code and objects can't control me". Considering that I prefer to use stored procedures approach for MySQL data searching, because I have full control of the application code.

```
Public Sub DataSearching_DataView(ByRef pDataGridViewControl As DataGridView, _
                                   ByRef pRecordCountInt32 As Int32, _
                                   ByVal pContactNameString As String, _
                                   ByRef pErrorMessageString As String)
    Dim FilterSQLString As String
    Try
        Using mMySQLConnection As New MySqlConnection(mMySQLConnectionString)
            mMySQLConnection.Open()
            Using mMySQLCommand As New MySqlCommand
                With mMySQLCommand
                    .Connection = mMySQLConnection
                    .CommandType = CommandType.StoredProcedure
                    .CommandText = "usp_select_all_data"
                End With
                Using mMySQLDataAdapter As New MySqlDataAdapter(mMySQLCommand)
                    Using mDataSet As New DataSet
                        mDataSet.Clear()
                        mMySQLDataAdapter.Fill(mDataSet, "data")
                    End Using
                End Using
            End Using
        End Using
    Catch ex As Exception
        pErrorMessageString = ex.Message
    End Try
End Sub
```

```
        mDataView = New DataView
        mDataView.Table = mDataSet.Tables("data")
        'An error occurred: The expression contains undefined function call CONCAT()."
        'FilterSQLString = "name LIKE CONCAT('%', & "" & pContactNameString & "" &
";%'")
        FilterSQLString = "name LIKE " & "" & pContactNameString & ""
        With mDataView
            .RowFilter = FilterSQLString
            .RowStateFilter = DataViewRowState.CurrentRows
            .Sort = "name"
            pRecordCountInt32 = .Count
        End With
    End Using
End Using
End Using
Call DataGridViewFormat(pDataGridViewControl)
With pDataGridViewControl
    .DataSource = mDataView
    .AutoSizeRows()
End With
End Using
Catch exError As Exception
    pErrorMessageString = exError.Message
End Try
End Sub
```

Listing 14: Procedure DataSearching_DataView() for searching with DataView object mDataView

Listing 15 shows the code of the click event of the Refresh button. The procedure SearchFormLoad() is called again to refresh the data from the table `data` and the Contact Name TextBox is clear to start a new search if necessary.

```
Private Sub RefreshButton_Click(ByVal sender As System.Object, _
                                ByVal e As System.EventArgs) _
    Handles RefreshButton.Click
    Call SearchFormLoad()
    With TextBoxName
        .Clear()
        .Focus()
    End With
End Sub
```

Listing 15: Click event of the Refresh button

Conclusions

Based on the presented here info the following conclusions could be made:

- MySQL stored procedures, functions and triggers objects development is one of the best programming practices for Windows database application implementation today. These objects provide the following three main benefits:
 - Improve application performance by reducing network traffic
 - Provide a single tier of business rules development and maintenance
 - Enhance application security by reducing SQL injection attacks
- MySQL data load using the DataGridView control requires a properly columns format based on table definition.
- MySQL data search using stored procedures in the database tier provides a flexible code location and organization for application upgrades and maintenance.

- The ADO.NET DataView object can provide a complete implementation of MySQL data load and search for Windows database applications development.

To download the source codes and a PDF format for this article go to [Visual WWW Downloads](#).

Visual WWW Legal Notice

This document contains freeware information. The information described in this document may be used or copied or transmitted for computer programming development purposes only without the written permission of Visual WWW, Inc. No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying and recording for any purpose.

The information contained in this document is subject to change without notice. Visual WWW makes no warranty of any kind with respect to this information. VISUAL WWW SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTY OF THE MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Visual WWW shall not be liable for any direct, indirect, incidental, consequential, or other damage alleged in connection with the furnishing or use of this information.

Visual WWW, Inc. and Visual WWW logo are registered trademarks of Visual WWW, Inc. Other trademarks and registered trademarks used in this document are property of their respective owners.