iX Database Access

Built: 10.05.2010
K. Spindler (KSR)

Content

1 Introduction

Reading and writing of databases can quite easily be done using C# scripting.

All actual databases are able to handle SQL-statements – so the difference concerning the handling between the 4 databases in the iX databases project is quite marginal.

Microsoft Office Access, previously known as **Microsoft Access**, is a pseudo relational database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software development tools. It is a member of the Microsoft Office suite of applications, included in the Professional and higher editions or sold separately. The current versions are Microsoft Office Access 2007 for Windows. In late 2009, Microsoft released the beta version of Microsoft Access 2010.

Microsoft SQL Server Compact (**SQL CE**) is a compact relational database produced by Microsoft for applications that run on mobile devices and desktops. Prior to the introduction of the desktop platform, it was known as *SQL Server for Windows CE* and *SQL Server Mobile Edition*. The latest release is the SQL Server Compact 3.5 SP2 supporting .NET Framework 3.5 as well as Windows Mobile 2003, 5.0, 6.0, and 6.5. It includes both 32-bit and 64-bit native support. SQL CE targets occasionally-connected applications and applications with an embedded database. It is free to download as well as redistribute.

Microsoft SQL Server is a relational model database server produced by Microsoft. Its primary query languages are T-SQL and ANSI SQL.

MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases.

SQL, often referred to as **Structured Query Language**, is a database computer language designed for managing data in relational database management systems (RDBMS), and originally based upon relational algebra. Its scope includes data query and update, schema creation and modification, and data access control. SQL was one of the first languages for Edgar F. Codd's relational model in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks" and became the most widely used language for relational databases.

2 General description of the iX database example project

All 4 parts of the demo project are very similar – as well the graphics as the C# scripting part.

When starting the project or navigating to another screen the preconfigured database is opened and loaded to the DataGridView on the lower left. In the upper left some details concerning the connected database table are displayed.

Other databases and database tables can be opened using the browse ("..") button beside the database table name and / or editing the connection string (SQL Server, MySQL).

When reading a database table the actual state of the database is read into a DataTable which is displayed by the DataGridView. So when you edit the DataGridView you edit the DataTable NOT! the database. In order to get your changes back to the database you have to update the database.

So writing / editing a database can typically be done in two different ways – either writing directly to the database using SQL – statements like "INSERT", "UPDATE", "DELETE" etc. or editing the DataTable shown by the DataGridView directly and updating the database later on.

In the example the Buttons have the following function:

"Write actual values to DB": inserts the actual process data directly to the database using the "INSERT INTO TABLENAME.." statement and afterwards reading the database again.

"Write table to DB": Updates the changes made to the DataGridView / the data table to the database using the "CommandBuilder" and the "DataAdapter.Update" method.

"Delete Records":	Deletes the records / rows of the database depending on the delete command using the "DELETE FROM TABLENAME WHERE ..." After deleting the number of deleted records is displayed and the database is read again.

3 Frequently used databases

3.1 MS Access:

Connection the MS Access is typically established using OleDB:

OLE DB (*Object Linking and Embedding, Database*, sometimes written as **OLEDB** or **OLE-DB**) is an API designed by Microsoft for accessing data from a variety of sources in a uniform manner. It is a set of interfaces implemented using the Component Object Model (COM); it is otherwise unrelated to OLE. It was designed as a higher-level replacement for, and successor to, ODBC, extending its feature set to support a wider variety of non-relational databases, such as object databases and spreadsheets that do not necessarily implement SQL.

Requirements working with Access using OleDB:

Namespaces: using System.Data; // needed for DataTables, DataSets...
using System.Data.OleDb; // for OleDB

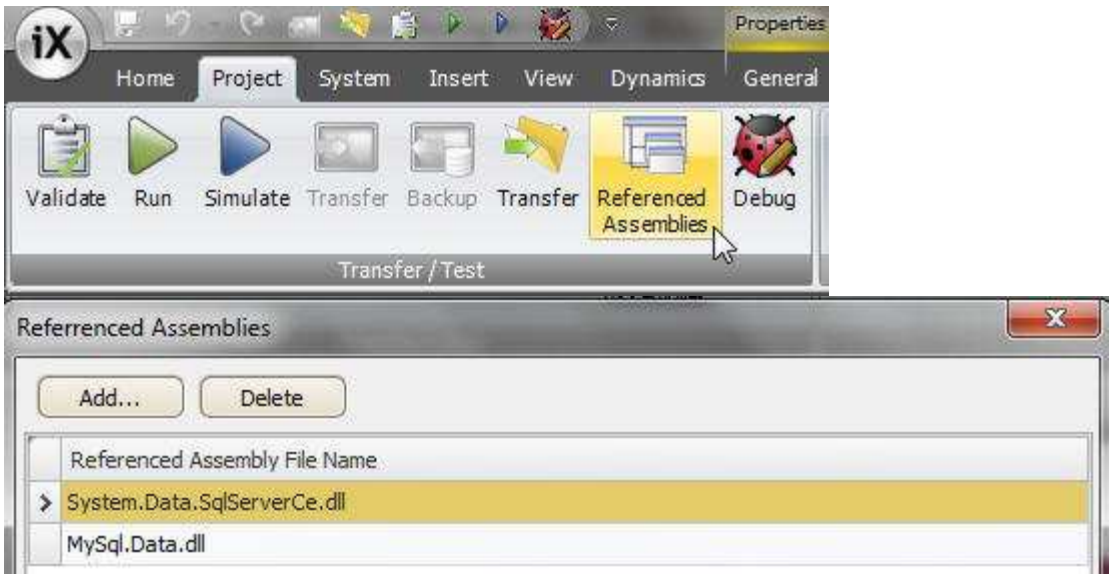
In the iX databases project by default a MS Access database "iX_DB.mdb" with DataTable "DATA" located in the project files folder is addressed.

3.2 SQL compact:

Requirements working with SQL compact:

Namespaces: using System.Data; // needed for DataTables, DataSets...
using System.Data.SqlServerCe; // needed for SQLCompact Database IO

Referenced Assemblies: to be able to work with the "System.Data.SqlServerCe" Namespace the "System.Data.SqlServerCe.dll" has to be entered in Project → Referenced Assemblies



In the iX databases project by default the "database.sdf" is addressed using DataTable „recipe1".

3.3 SQL Server:

Requirements working with SQL Server:

Namespaces: using System.Data; // needed for DataTables, DataSets...
using System.Data.SqlClient; // for SQL Server

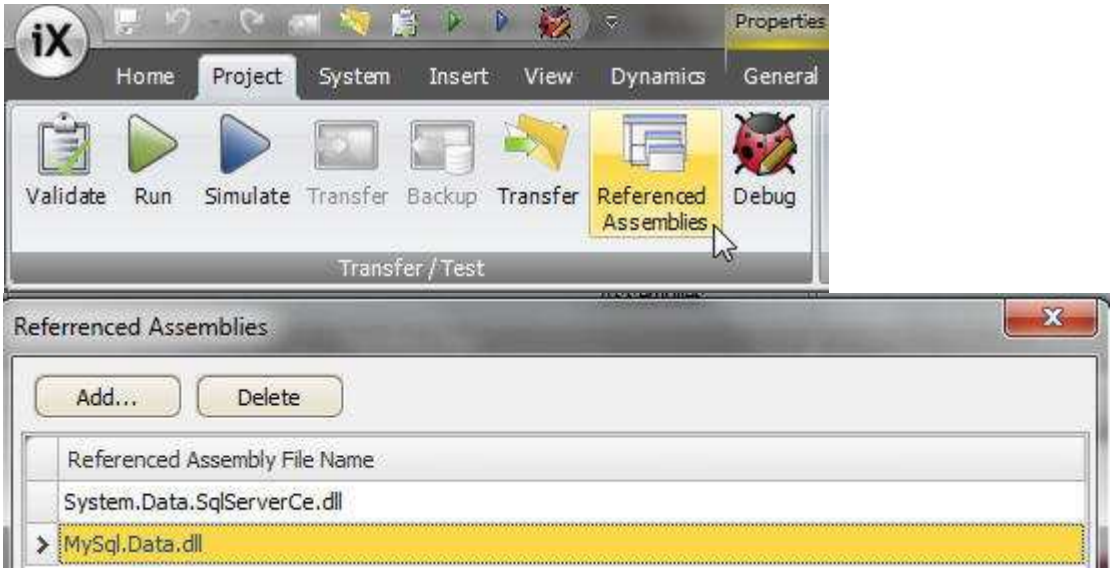
In the iX databases project by default a SQL Server database „KSR_DB“ with DataTable “DATA” on Server „192.168.247.128\KSR_SQLSERVER“ is addressed.

3.4 MySQL

Requirements working with MySQL:

Namespaces: using System.Data; // needed for DataTables, DataSets...
using MySql.Data.MySqlClient; // for MySQL

Referenced Assemblies: to be able to work with the “MySql.Data.MySqlClient” Namespace the “MySql.Data.MySqlClient.dll” has to be entered in Project → Referenced Assemblies



In the iX databases project by default a MySQL Server database „MySQL“ with DataTable “MySQL_Tbl” on Server „192.168.247.128“ is addressed.

4 Code snippets used in the iX database project

Using the example of SQL compact:

Declarations:

```
DataTable dt = new DataTable();
SqlCeConnection conn = new SqlCeConnection();
```

4.1 Establish connection to the database / Connection string

```
private SqlCeConnection Connect2SqlCe ()
    {
    string m_ConnectionsString = "Data Source=Database.sdf;Default Lock Timeout=60000";
        try
        {
            conn.ConnectionString = m_ConnectionsString;
            conn.Open();
            return conn;
        }
        catch (Exception ex)
        {
            conn.Close();
            MessageBox.Show(ex.Message, "Alert");
            return null;
        }
    }
```

4.2 Read DataTable

```
private void TableRead ()
    {
        conn = Connect2SqlCe ();
        try
        {
            string Commandstring = "select * FROM Recipe1";
            SqlCeCommand cmd = new SqlCeCommand(Commandstring,conn);
            SqlCeDataAdapter da = new SqlCeDataAdapter(cmd);
            SqlCeCommandBuilder cb = new SqlCeCommandBuilder(da);
            dt.Rows.Clear();
            dt.Columns.Clear();
            da.Fill (dt);
            DataGridView1.DataSource = dt;
            DataGridView1.AutoResizeColumns();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: Failed to retrieve the required data from the
                DataBase.\n{" + ex.Message + "}", "Attention!");
            return;
        }
        finally
        {
            conn.Close();
        }
    }
```

4.3 Browse database for existing tables and display them in a listbox

```
private void SQLCE_DB_Read()
        {
                conn = Connect2SqlCe ();
                try
                {
                        string Commandstring = "select Table_Name from
                                INFORMATION_SCHEMA.TABLES";
                        SqlCeCommand cmd = new SqlCeCommand(Commandstring,conn);
                        SqlCeDataReader reader = cmd.ExecuteReader();

                        int i = 0;
                        while (reader.Read())
                        {
                                // filter the displayed tables
                                string strTable = string.Format("{0}",reader[0]);

                                // do not display „NEO_SYSTEMTABLE“
                                if (!(strTable == "NEO_SYSTEMTABLE"))
                                {
                                        ListBox_Tables.Items.Add(strTable);
                                }
                                ++i;
                        }

                        // Call Close when done reading.
                        reader.Close();
                }
                catch (Exception ex)
                {
                        MessageBox.Show("Error: Failed to retrieve the required data from the
                                DataBase.\n{" + ex.Message + "}", "Attention!");
                }
                finally
                {
                        conn.Close();
                }
        }
```

4.4 INSERT records INTO a database

```
void Btn_DB_Write_Click(System.Object sender, System.EventArgs e)
        {
                conn = Connect2SqlCe ();
                try
                {
                        // insert actual process data
                        string Commandstring = "INSERT INTO Recipe1
                                (FieldName,Voltage,Temperature,Pressure,Comment)" +
                                String.Format("VALUES('{0}','{1}','{2}','{3}','{4}')",
                                RandomString(),
                                Globals.Tags.Voltage.Value,
                                Globals.Tags.Temperature.Value,
                                Globals.Tags.Pressure.Value,
                                Globals.Tags.Comment.Value
                                );
                        SqlCeCommand cmd = new SqlCeCommand(Commandstring, conn );
                        cmd.ExecuteNonQuery();
                }
                catch (Exception ex)
                {
                        MessageBox.Show("Error: Failed to retrieve the required data from the
                                DataBase.\n{" + ex.Message +"}", "Attention!");
                        return;
                }
                finally
                {
                        conn.Close();
                }

                TableRead();
        }


// generate a "random" recipe name
private string RandomString()
        {
                int rnd_val;
                Random random = new Random();
                rnd_val= Convert.ToInt32(Math.Floor(10000* random.NextDouble()));
                return ("RZP_" + rnd_val.ToString());
        }
```

4.5 Update database

```
void Btn_Update_DB_Click(System.Object sender, System.EventArgs e)
        {
                conn = Connect2SqlCe ();
                try
                {
                        string Commandstring = "select * FROM Recipe1“;
                        SqlCeCommand cmd = new SqlCeCommand(Commandstring,conn);
                        SqlCeDataAdapter da = new SqlCeDataAdapter(cmd);
                        SqlCeCommandBuilder cb = new SqlCeCommandBuilder(da);

                        dt.PrimaryKey = null;
                        //important if rows are changed but the row is "not left"
                        foreach (DataRow dr in dt.Rows)
                        {
                                dr.EndEdit();
                        }
                        da.Update (dt);
                }
                catch (Exception ex)
                {
                        MessageBox.Show("Error: Failed to retrieve the required data from the
                                DataBase.\n{" + ex.Message +"}", "Attention!");
                        return;
                }
                finally
                {
                        conn.Close();
                }
        }
```

4.6 DELETE records FROM database

```
void Btn_Record_DEL_Click(System.Object sender, System.EventArgs e)
		{
		conn = Connect2SqlCe ();
		try
		{
			string Commandstring = "DELETE FROM Recipe1 WHERE comment = 'test';
			SqlCeCommand cmd = new SqlCeCommand(Commandstring, conn );
			Globals.Tags.Rows_Deleted.Value = cmd.ExecuteNonQuery();
		}
		catch (Exception ex)
		{
			MessageBox.Show("Error: Failed to retrieve the required data from the
				DataBase.\n{" + ex.Message +"}", "Attention!");
			return;
		}
		finally
		{
			conn.Close();
		}
		TableRead();
	}
```

4.7 File-Dialog to open another database

```
void Btn_DB_Input_Click(System.Object sender, System.EventArgs e)
	{
		try
		{
			OpenFileDialog dlgOpen = new OpenFileDialog();
			dlgOpen.InitialDirectory = @"C:\";
			dlgOpen.Filter = "SQL compact database (*.sdf)|*.sdf";
			dlgOpen.Title = "Choose Database";
			dlgOpen.ShowDialog();
			if (! (dlgOpen.FileName == string.Empty) )
			{
				Globals.Tags.SQLCE_DataSource.Value = dlgOpen.FileName;
				ListBox_Tables.Items.Clear();
				SQLCE_DB_Read();
				dt = new DataTable();
				DataGridView1.DataSource = dt;
				Globals.Tags.IsTableSelected.Value = 0;
			}
		}
		catch (Exception ex)
		{
			MessageBox.Show(ex.Message, "Alert");
		}
	}
```

5 Other database related Code snippets

Additional Namespaces:

```
using System.Text; //needed for StringBuilder
using System.Data.Odbc; //needed for reading CSV via MS Text Driver
using System.IO; //needed for Streamwriter
using System.Data;      // needed for DataTables, DataSets...
using System.Data.SqlServerCe;          // needed for SQLCompact Database IO
```

Declarations:

```
DataTable dt = new DataTable();
DataTable dt_tmp = new DataTable();
SqlCeConnection conn = new SqlCeConnection();
```

5.1 Write DataTable to CSV-format

```
private void WriteDataTable(string path, DataTable datatable, char seperator)
        {
                using (StreamWriter sw = new StreamWriter(path, false,
                System.Text.Encoding.Default))
                {
                        int numberOfColumns = datatable.Columns.Count;
                        for (int i = 0; i < numberOfColumns; i++)
                        {
                                sw.Write(datatable.Columns[i]);
                                if (i < numberOfColumns - 1)
                                        sw.Write(seperator);
                        }
                        sw.Write(sw.NewLine);
                        foreach (DataRow dr in datatable.Rows)
                        {
                                for (int i = 0; i < numberOfColumns; i++)
                                {
                                        sw.Write(dr[i].ToString());
                                        if (i < numberOfColumns - 1)
                                                sw.Write(seperator);
                                }
                                sw.Write(sw.NewLine);
                        }
                }
        }
```

5.2 Read DataTable from CSV-format (via ODBC)

```
// reads the csv format using Odbc - seperator must be ','
private DataTable ReadDataTable(string path)
	{
		try
		{
			FileInfo fileInfo = new FileInfo(path);
			DataTable dataTable = new DataTable();
			string connectionString = String.Format("Driver={{Microsoft Text Driver
				(*.txt; *.csv)}};Dbq={0};", fileInfo.DirectoryName);
			OdbcConnection connection = new OdbcConnection(connectionString);
			OdbcDataAdapter ODBCda = new OdbcDataAdapter(String.Format("select *
				from [{0}]",fileInfo.Name), connection);
			ODBCda.Fill(dataTable);
			return dataTable;
		}
		catch (Exception ex)
		{
			MessageBox.Show(ex.Message, "Alert");
			return null;
		}
	}
```

5.3 Compare the columns - structure of 2 DataTables

```
// compares two datatables - returns 0 when structure matches
private int CompareTables (DataTable dt1,DataTable dt2)
	{
		// different amount of columns
		if(!(dt1.Columns.Count == dt2.Columns.Count))
		{
			return -1;
		}

		// create a string from the column names of dt1
		StringBuilder dc1_Info = new StringBuilder();
		foreach (DataColumn dc1 in dt1.Columns)
		{
			dc1_Info.AppendFormat("{0},",dc1.ColumnName);
		}

		// create a string from the column names of dt2
		StringBuilder dc2_Info = new StringBuilder();
		foreach (DataColumn dc2 in dt2.Columns)
		{
			dc2_Info.AppendFormat("{0},",dc2.ColumnName);
		}

		// do the strings match?
		if (!(dc1_Info.ToString() == dc2_Info.ToString()))
		{
			return -1;
		}
		// OK
		return 0;
	}
```

5.4 Check if a DataTable has changes / UNDO changes

Are there changes ?

```
Globals.Tags.DataTable_Has_Changes.Value = (dt.GetChanges()!= null);
```

```
void Btn_Undo_Click(System.Object sender, System.EventArgs e)
    {
        dt.RejectChanges();
        // dt.AcceptChanges() would takes over all changes made to the dt BUT after that
        // updating  the database via da.Update() is no longer possible, because "there are
        // no changes"
    }
```

5.5 Import selected rows from one DataTable to another DataTable

```
// import the selected rows to the selected table - existing recipes are overwritten

void Btn_Import_RzpRows_Click(System.Object sender, System.EventArgs e)
    {
        int selectedRowCount =
            DataGridView2.Rows.GetRowCount(DataGridViewElementStates.Selected);

        // check whether the recipe structure matches
        int status = CompareTables (dt,dt_tmp);
        if (status != 0)
        {
            MessageBox.Show("Structure of the imported data does not match !"
                + status.ToString(), "Alert");
            return;
        }

        if (selectedRowCount > 0)
        {
            // the LoadDataRow method browses the columns of a DataTable for
            // Primary Keys ...  so  there has to be at least one PK...
            dt.PrimaryKey = new DataColumn[] {dt.Columns["FieldName"]};

            dt.BeginLoadData();
            int i = 0;
            for (i = 0; i < selectedRowCount; i++)
            {
                dt.LoadDataRow(dt_tmp.Rows
                    [DataGridView2.SelectedRows[i].Index].ItemArray, false);
            }
            dt.EndLoadData();
            DataGridView1.DataSource = dt;
        }
    }
```

5.6 Get the datatype of a column and „translate“ it to the SQL compact datatype

```
// for creating SQLCE DataTable
        private string getDataType (System.Type ColumnTyp)
        {
                if (ColumnTyp == typeof(System.String))
                {
                        return "nvarchar(50)";
                }
                if (ColumnTyp == typeof(System.Int16))
                {
                        return "smallint";
                }
                if (ColumnTyp == typeof(System.Byte))
                {
                        return "tinyint";
                }
                if (ColumnTyp == typeof(System.Int32))
                {
                        return "int";
                }
                if (ColumnTyp == typeof(System.DateTime))
                {
                        return "real";
                }
                if (ColumnTyp == typeof(System.Decimal))
                {
                        return "float";
                }
                if (ColumnTyp == typeof(System.Double))
                {
                        return "real";
                }
                return "shit";
        }
```

5.7 Creating a Clone of the selected DataTable

```
// creates a new recipe table from the selected recipe
        void Btn_CreateTable_Click(System.Object sender, System.EventArgs e)
        {
                if (Globals.Tags.NewTableName.Value != string.Empty)
                {
                        conn = Connect2SqlCe (m_ConnectionsString);
                        try
                        {
                                SqlCeCommand cmd = conn.CreateCommand();
                                cmd = conn.CreateCommand();

                                StringBuilder SQL_CMD = new StringBuilder();

                                SQL_CMD.AppendFormat("CREATE TABLE {0}
                                        (",Globals.Tags.NewTableName.Value);

                                for (int i = 0; i < dt.Columns.Count; i++)
                                {
                                        SQL_CMD.AppendFormat("{0}
                                                {1}",dt.Columns[i].ColumnName,
                                                getDataType(dt.Columns[i].DataType));
                                        if (i == (dt.Columns.Count - 1))
                                        {
                                                SQL_CMD.AppendFormat(")");
                                        }
                                        else
                                        {
                                                SQL_CMD.AppendFormat(",");
                                        }
                                }
                                cmd.CommandText = SQL_CMD.ToString();
                                SqlCeDataReader reader = cmd.ExecuteReader();

                                // Call Close when done reading.
                                reader.Close();
                        }
                        catch (Exception ex)
                        {
                                MessageBox.Show("Error: Failed to retrieve the required data from
                                        the DataBase.\n{" + ex.Message +"}", "Attention!");
                                return;
                        }
                        finally
                        {
                                conn.Close();
                        }

                        conn = Connect2SqlCe (m_ConnectionsString);

                        try
                        {
                                SqlCeCommand cmd = conn.CreateCommand();
                                cmd.CommandText = "select * FROM " +
                                        Globals.Tags.NewTableName.Value;
                                SqlCeDataAdapter da = new SqlCeDataAdapter(cmd);

                                SqlCeCommandBuilder cb = new SqlCeCommandBuilder(da);
```

```
                        //important if rows are changed but the row is "not left"
                        foreach (DataRow dr in dt.Rows)
                        {
                                dr.EndEdit();
                                dr.SetAdded();
                        }
                        da.Update (dt);
                }
                catch (Exception ex)
                {
                        MessageBox.Show("Error: Failed to retrieve the required data from
                                the DataBase.\n{" + ex.Message +"}", "Attention!");
                        return;
                }
                finally
                {
                        conn.Close();
                }
                ListBox_Tables.Items.Clear();
                DatabaseRead();
        }
}
```

5.8 Deleting a DataTable

```
void Btn_Delete_Table_Click(System.Object sender, System.EventArgs e)
        {
                conn = Connect2SqlCe (m_ConnectionsString);
                try
                {
                        SqlCeCommand cmd = conn.CreateCommand();
                        cmd.CommandText = "DROP TABLE " + Globals.Tags.selected_table.Value;

                        SqlCeDataReader reader = cmd.ExecuteReader();

                        // Call Close when done reading.
                        reader.Close();
                }
                catch (Exception ex)
                {
                        MessageBox.Show("Error: Failed to retrieve the required data from the
                                DataBase.\n{" + ex.Message +"}", "Attention!");
                        return;
                }
                finally
                {
                        conn.Close();
                }
                Globals.Tags.selected_table.Value = string.Empty;
                dt = new DataTable();
                DataGridView1.DataSource = dt;
                ListBox_Tables.Items.Clear();
                DatabaseRead();
        }
```