

SAMPLE CODE: DATABASE ACCESS AND UPDATING

EXAMPLE 1: USING A DATAGRIDVIEW BOUND TO A JOINED TABLE (MOST OF THE WORK WAS DONE IN THE DESIGNER)

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

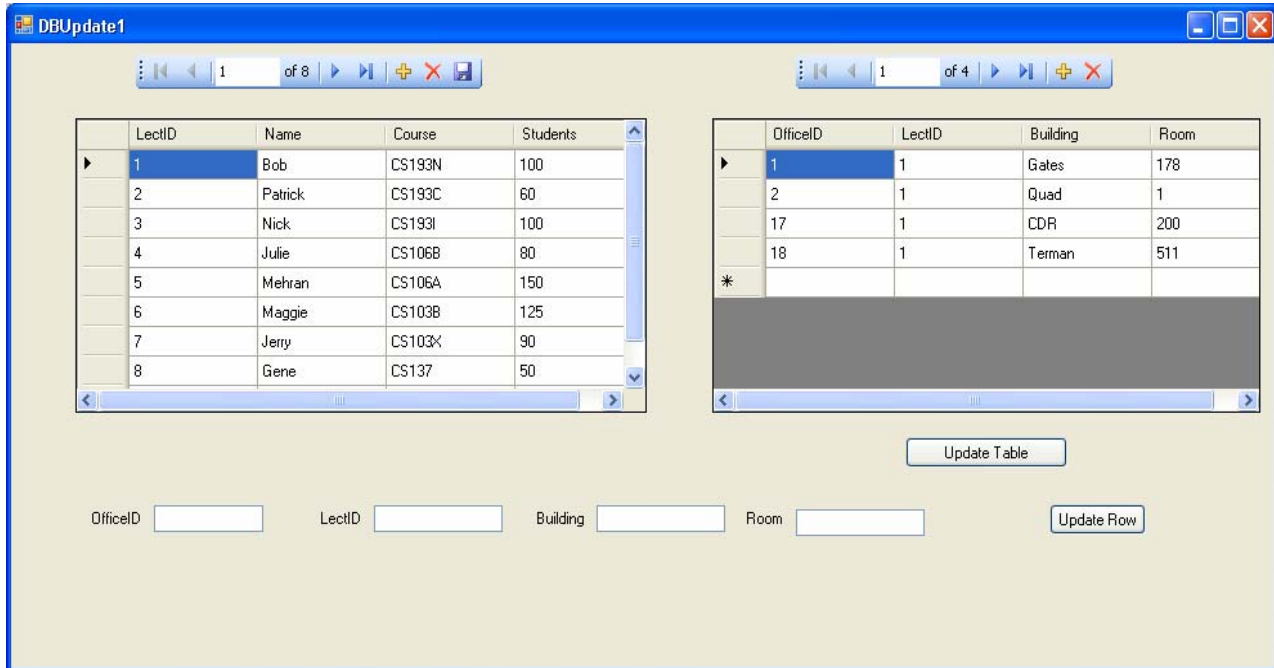
    private void Form1_Load(object sender, EventArgs e)
    {
        this.lecturersTableAdapter1.Fill(testSQL2DataSet.Lecturers);

        // TODO: This line of code loads data into the
        'testSQL2DataSet.LecturersRooms' table. You can move, or remove it, as needed.

        this.lecturersRoomsTableAdapter.Fill(this.testSQL2DataSet.LecturersRooms);
    }

    private void lecturersRoomsBindingSource_CurrentChanged(object sender,
                                                             EventArgs e)
    {
        DataRowView drv = (DataRowView) lecturersRoomsBindingSource.Current;
        TestSQL2DataSet.LecturersRoomsRow row =
            (TestSQL2DataSet.LecturersRoomsRow) drv.Row;
        TestSQL2DataSet.LecturersRow lectRow =
            testSQL2DataSet.Lecturers.FindByLectID(row.LectID);
        textBox1.Text = lectRow.Course;
    }
}
```

EXAMPLE 2: DATABASE UPDATES



```

private void Form1_Load(object sender, EventArgs e)
{
    // TODO: This line of code loads data into the
    'testSQL2DataSet.Offices' table. You can move, or remove it, as needed.
    this.officesTableAdapter.Fill(this.testSQL2DataSet.Offices);
    // TODO: This line of code loads data into the
    'testSQL2DataSet.Lecturers' table. You can move, or remove it, as needed.
    this.lecturersTableAdapter.Fill(this.testSQL2DataSet.Lecturers);
}

//This is the code for the Update Row button, which takes values from
textboxes.
//It looks up the row and uses its values as the original values in the
call to
//NewUpdateQuery, which pushes the new row out to the database. Then we
call Fill
//to reload the table. No need to clear it first here, since
ClearBeforeFill is set.
//NewUpdateQuery actually does the same thing as the Update method, but
it's an example
//of adding an additional query to the command collection.

private void btnUpdate_Click(object sender, EventArgs e)
{
    int rowsAffected;
    int officeID = Int32.Parse(tbOfficeID.Text);
    int lectID = Int32.Parse(tbLectID.Text);
    TestSQL2DataSet.OfficesRow row =
testSQL2DataSet.Offices.FindByOfficeID(officeID);
    if (row == null)
    {
        rowsAffected = 0;
    }
}

```

```

        MessageBox.Show("0 rows affected. Couldn't find row with
OfficeID " + officeID);
    }
    else
    {
        rowsAffected = officesTableAdapter.NewUpdateQuery(lectID,
tbBuilding.Text, tbRoom.Text, officeID, lectID, row.Building, row.Room, officeID);
        if (rowsAffected > 0)
        {
            MessageBox.Show(rowsAffected + " rows affected");
            officesTableAdapter.Fill(this.testSQL2DataSet.Offices);
        }
        else
        {
            MessageBox.Show("0 rows affected. Couldn't find row: " +
row.OfficeID + " " + row.LectID + " " + row.Building + " " + row.Room);
        }
    }
}

//This is the code for the Update Table button.

private void btnUpdateTable_Click(object sender, EventArgs e)
{
    officesTableAdapter.Update(testSQL2DataSet.Offices);
}
}

```

AND HERE, IN ALL ITS GLORY, IS THE TYPED DATASET

```

//-----
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:2.0.50727.832
//
// Changes to this file may cause incorrect behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//-----

#pragma warning disable 1591

namespace DBUpdate1 {
    using System;

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGe
nerator", "2.0.0.0")]
[Serializable()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.ComponentModel.ToolboxItem(true)]
[System.Xml.Serialization.XmlSchemaProviderAttribute("GetTypedDataSetSchema")]
[System.Xml.Serialization.XmlRootAttribute("TestSQL2DataSet")]
[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.DataSet")]
public partial class TestSQL2DataSet : System.Data.DataSet {

    private LecturersDataTable tableLecturers;

    private OfficesDataTable tableOffices;

```

```

private System.Data.DataRelation relationFK_Offices_Lecturers;

private System.Data.SchemaSerializationMode _schemaSerializationMode =
System.Data.SchemaSerializationMode.IncludeSchema;

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public TestSQL2DataSet() {
    this.BeginInit();
    this.InitClass();
    System.ComponentModel.CollectionChangeEventHandler
schemaChangedHandler = new
System.ComponentModel.CollectionChangeEventHandler(this.SchemaChanged);
    base.Tables.CollectionChanged += schemaChangedHandler;
    base.Relations.CollectionChanged += schemaChangedHandler;
    this.EndInit();
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
protected TestSQL2DataSet(System.Runtime.Serialization.SerializationInfo
info, System.Runtime.Serialization.StreamingContext context) :
    base(info, context, false) {
    if ((this.IsBinarySerialized(info, context) == true)) {
        this.InitVars(false);
        System.ComponentModel.CollectionChangeEventHandler
schemaChangedHandler1 = new
System.ComponentModel.CollectionChangeEventHandler(this.SchemaChanged);
        this.Tables.CollectionChanged += schemaChangedHandler1;
        this.Relations.CollectionChanged += schemaChangedHandler1;
        return;
    }
    string strSchema = ((string)(info.GetValue("XmlSchema",
typeof(string))));
    if ((this.DetermineSchemaSerializationMode(info, context) ==
System.Data.SchemaSerializationMode.IncludeSchema)) {
        System.Data.DataSet ds = new System.Data.DataSet();
        ds.ReadXmlSchema(new System.Xml.XmlTextReader(new
System.IO.StringReader(strSchema)));
        if ((ds.Tables["Lecturers"] != null)) {
            base.Tables.Add(new
LecturersDataTable(ds.Tables["Lecturers"]));
        }
        if ((ds.Tables["Offices"] != null)) {
            base.Tables.Add(new OfficesDataTable(ds.Tables["Offices"]));
        }
        this.DataSetName = ds.DataSetName;
        this.Prefix = ds.Prefix;
        this.Namespace = ds.Namespace;
        this.Locale = ds.Locale;
        this.CaseSensitive = ds.CaseSensitive;
        this.EnforceConstraints = ds.EnforceConstraints;
        this.Merge(ds, false, System.Data.MissingSchemaAction.Add);
        this.InitVars();
    }
    else {
        this.ReadXmlSchema(new System.Xml.XmlTextReader(new
System.IO.StringReader(strSchema)));
    }
    this.GetSerializationData(info, context);
    System.ComponentModel.CollectionChangeEventHandler
schemaChangedHandler = new
System.ComponentModel.CollectionChangeEventHandler(this.SchemaChanged);
    base.Tables.CollectionChanged += schemaChangedHandler;
    this.Relations.CollectionChanged += schemaChangedHandler;
}

```

```

    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
[System.ComponentModel.Browsable(false)]

[System.ComponentModel.DesignerSerializationVisibility(System.ComponentModel.Desig
nerSerializationVisibility.Content)]
    public LecturersDataTable Lecturers {
        get {
            return this.tableLecturers;
        }
    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
[System.ComponentModel.Browsable(false)]

[System.ComponentModel.DesignerSerializationVisibility(System.ComponentModel.Desig
nerSerializationVisibility.Content)]
    public OfficesDataTable Offices {
        get {
            return this.tableOffices;
        }
    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
[System.ComponentModel.BrowsableAttribute(true)]

[System.ComponentModel.DesignerSerializationVisibilityAttribute(System.ComponentModel
del.DesignerSerializationVisibility.Visible)]
    public override System.Data.SchemaSerializationMode
SchemaSerializationMode {
        get {
            return this._schemaSerializationMode;
        }
        set {
            this._schemaSerializationMode = value;
        }
    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.DesignerSerializationVisibilityAttribute(System.ComponentModel
del.DesignerSerializationVisibility.Hidden)]
    public new System.Data.DataTableCollection Tables {
        get {
            return base.Tables;
        }
    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.DesignerSerializationVisibilityAttribute(System.ComponentModel
del.DesignerSerializationVisibility.Hidden)]
    public new System.Data.DataRelationCollection Relations {
        get {
            return base.Relations;
        }
    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
protected override void InitializeDerivedDataSet() {
    this.BeginInit();
    this.InitClass();
    this.EndInit();
}

```

```

}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public override System.Data.DataSet Clone() {
    TestSQL2DataSet cln = ((TestSQL2DataSet)(base.Clone()));
    cln.InitVars();
    cln.SchemaSerializationMode = this.SchemaSerializationMode;
    return cln;
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
protected override bool ShouldSerializeTables() {
    return false;
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
protected override bool ShouldSerializeRelations() {
    return false;
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
protected override void ReadXmlSerializable(System.Xml.XmlReader reader) {
    if ((this.DetermineSchemaSerializationMode(reader) ==
System.Data.SchemaSerializationMode.IncludeSchema)) {
        this.Reset();
        System.Data.DataSet ds = new System.Data.DataSet();
        ds.ReadXml(reader);
        if ((ds.Tables["Lecturers"] != null)) {
            base.Tables.Add(new
LecturersDataTable(ds.Tables["Lecturers"]));
        }
        if ((ds.Tables["Offices"] != null)) {
            base.Tables.Add(new OfficesDataTable(ds.Tables["Offices"]));
        }
        this.DataSetName = ds.DataSetName;
        this.Prefix = ds.Prefix;
        this.Namespace = ds.Namespace;
        this.Locale = ds.Locale;
        this.CaseSensitive = ds.CaseSensitive;
        this.EnforceConstraints = ds.EnforceConstraints;
        this.Merge(ds, false, System.Data.MissingSchemaAction.Add);
        this.InitVars();
    }
    else {
        this.ReadXml(reader);
        this.InitVars();
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
protected override System.Xml.Schema.XmlSchema GetSchemaSerializable() {
    System.IO.MemoryStream stream = new System.IO.MemoryStream();
    this.WriteXmlSchema(new System.Xml.XmlTextWriter(stream, null));
    stream.Position = 0;
    return System.Xml.Schema.XmlSchema.Read(new
System.Xml.XmlTextReader(stream), null);
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
internal void InitVars() {
    this.InitVars(true);
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

```

```

        internal void InitVars(bool initTable) {
            this.tableLecturers =
                ((LecturersDataTable)(base.Tables["Lecturers"]));
            if ((initTable == true)) {
                if ((this.tableLecturers != null)) {
                    this.tableLecturers.InitVars();
                }
            }
            this.tableOffices = ((OfficesDataTable)(base.Tables["Offices"]));
            if ((initTable == true)) {
                if ((this.tableOffices != null)) {
                    this.tableOffices.InitVars();
                }
            }
            this.relationFK_Offices_Lecturers =
                this.Relations["FK_Offices_Lecturers"];
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        private void InitClass() {
            this.DataSetName = "TestSQL2DataSet";
            this.Prefix = "";
            this.Namespace = "http://tempuri.org/TestSQL2DataSet.xsd";
            this.EnforceConstraints = true;
            this.SchemaSerializationMode =
                System.Data.SchemaSerializationMode.IncludeSchema;
            this.tableLecturers = new LecturersDataTable();
            base.Tables.Add(this.tableLecturers);
            this.tableOffices = new OfficesDataTable();
            base.Tables.Add(this.tableOffices);
            this.relationFK_Offices_Lecturers = new
                System.Data.DataRelation("FK_Offices_Lecturers", new System.Data.DataColumn[] {
                    this.tableLecturers.LectIDColumn}, new
                    System.Data.DataColumn[] {
                        this.tableOffices.LectIDColumn}, false);
            this.Relations.Add(this.relationFK_Offices_Lecturers);
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        private bool ShouldSerializeLecturers() {
            return false;
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        private bool ShouldSerializeOffices() {
            return false;
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        private void SchemaChanged(object sender,
            System.ComponentModel.CollectionChangeEventArgs e) {
            if ((e.Action == System.ComponentModel.CollectionChangeAction.Remove))
            {
                this.InitVars();
            }
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        public static System.Xml.Schema.XmlSchemaComplexType
            GetTypedDataSetSchema(System.Xml.Schema.XmlSchemaSet xs) {
            TestSQL2DataSet ds = new TestSQL2DataSet();
            System.Xml.Schema.XmlSchemaComplexType type = new
                System.Xml.Schema.XmlSchemaComplexType();
        }

```

```

        System.Xml.Schema.XmlSchemaSequence sequence = new
System.Xml.Schema.XmlSchemaSequence();
        xs.Add(ds.GetSchemaSerializable());
        System.Xml.Schema.XmlSchemaAny any = new
System.Xml.Schema.XmlSchemaAny();
        any.Namespace = ds.Namespace;
        sequence.Items.Add(any);
        type.Particle = sequence;
        return type;
    }

    public delegate void LecturersRowChangeEventHandler(object sender,
LecturersRowChangeEvent e);

    public delegate void OfficesRowChangeEventHandler(object sender,
OfficesRowChangeEvent e);

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGe
nerator", "2.0.0.0")]
[System.Serializable()]

[System.Xml.Serialization.XmlSchemaProviderAttribute("GetTypedTableSchema")]
public partial class LecturersDataTable : System.Data.DataTable,
System.Collections.IEnumerable {

    private System.Data.DataColumn columnLectID;

    private System.Data.DataColumn columnName;

    private System.Data.DataColumn columnCourse;

    private System.Data.DataColumn columnStudents;

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public LecturersDataTable() {
    this.TableName = "Lecturers";
    this.BeginInit();
    this.InitClass();
    this.EndInit();
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
internal LecturersDataTable(System.Data.DataTable table) {
    this.TableName = table.TableName;
    if ((table.CaseSensitive != table.DataSet.CaseSensitive)) {
        this.CaseSensitive = table.CaseSensitive;
    }
    if ((table.Locale.ToString() != table.DataSet.Locale.ToString()))
{
        this.Locale = table.Locale;
    }
    if ((table.Namespace != table.DataSet.Namespace)) {
        this.Namespace = table.Namespace;
    }
    this.Prefix = table.Prefix;
    this.MinimumCapacity = table.MinimumCapacity;
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
protected
LecturersDataTable(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context) :
    base(info, context) {

```

```

        this.InitVars();
    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public System.Data.DataColumn LectIDColumn {
    get {
        return this.columnLectID;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public System.Data.DataColumn NameColumn {
    get {
        return this.columnName;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public System.Data.DataColumn CourseColumn {
    get {
        return this.columnCourse;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public System.Data.DataColumn StudentsColumn {
    get {
        return this.columnStudents;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
[System.ComponentModel.Browsable(false)]
public int Count {
    get {
        return this.Rows.Count;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public LecturersRow this[int index] {
    get {
        return ((LecturersRow)(this.Rows[index]));
    }
}

public event LecturersRowChangeEventHandler LecturersRowChanging;
public event LecturersRowChangeEventHandler LecturersRowChanged;
public event LecturersRowChangeEventHandler LecturersRowDeleting;
public event LecturersRowChangeEventHandler LecturersRowDeleted;

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public void AddLecturersRow(LecturersRow row) {
    this.Rows.Add(row);
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public LecturersRow AddLecturersRow(string Name, string Course, int
Students) {
    LecturersRow rowLecturersRow = ((LecturersRow)(this.NewRow()));
    rowLecturersRow.ItemArray = new object[] {

```

```

        null,
        Name,
        Course,
        Students};
    this.Rows.Add(rowLecturersRow);
    return rowLecturersRow;
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public LecturersRow FindByLectID(int LectID) {
    return ((LecturersRow)(this.Rows.Find(new object[] {
        LectID})));
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public virtual System.Collections.IEnumerator GetEnumerator() {
    return this.Rows.GetEnumerator();
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public override System.Data.DataTable Clone() {
    LecturersDataTable cln = ((LecturersDataTable)(base.Clone()));
    cln.InitVars();
    return cln;
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
protected override System.Data.DataTable CreateInstance() {
    return new LecturersDataTable();
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
internal void InitVars() {
    this.columnLectID = base.Columns["LectID"];
    this.columnName = base.Columns["Name"];
    this.columnCourse = base.Columns["Course"];
    this.columnStudents = base.Columns["Students"];
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
private void InitClass() {
    this.columnLectID = new System.Data.DataColumn("LectID",
typeof(int), null, System.Data.MappingType.Element);
    base.Columns.Add(this.columnLectID);
    this.columnName = new System.Data.DataColumn("Name",
typeof(string), null, System.Data.MappingType.Element);
    base.Columns.Add(this.columnName);
    this.columnCourse = new System.Data.DataColumn("Course",
typeof(string), null, System.Data.MappingType.Element);
    base.Columns.Add(this.columnCourse);
    this.columnStudents = new System.Data.DataColumn("Students",
typeof(int), null, System.Data.MappingType.Element);
    base.Columns.Add(this.columnStudents);
    this.Constraints.Add(new
System.Data.UniqueConstraint("Constraint1", new System.Data.DataColumn[] {
        this.columnLectID}, true));
    this.columnLectID.AutoIncrement = true;
    this.columnLectID.AllowDBNull = false;
    this.columnLectID.ReadOnly = true;
    this.columnLectID.Unique = true;
    this.columnName.AllowDBNull = false;
    this.columnName.MaxLength = 10;
    this.columnCourse.AllowDBNull = false;
    this.columnCourse.MaxLength = 10;
}

```

```

    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    public LecturersRow NewLecturersRow() {
        return ((LecturersRow)(this.NewRow()));
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    protected override System.Data.DataRow
NewRowFromBuilder(System.Data.DataRowBuilder builder) {
        return new LecturersRow(builder);
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    protected override System.Type GetRowType() {
        return typeof(LecturersRow);
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    protected override void
OnRowChanged(System.Data.DataRowChangeEventArgs e) {
        base.OnRowChanged(e);
        if ((this.LecturersRowChanged != null)) {
            this.LecturersRowChanged(this, new
LecturersRowChangeEvent(((LecturersRow)(e.Row)), e.Action));
        }
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    protected override void
OnRowChanging(System.Data.DataRowChangeEventArgs e) {
        base.OnRowChanging(e);
        if ((this.LecturersRowChanging != null)) {
            this.LecturersRowChanging(this, new
LecturersRowChangeEvent(((LecturersRow)(e.Row)), e.Action));
        }
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    protected override void
OnRowDeleted(System.Data.DataRowChangeEventArgs e) {
        base.OnRowDeleted(e);
        if ((this.LecturersRowDeleted != null)) {
            this.LecturersRowDeleted(this, new
LecturersRowChangeEvent(((LecturersRow)(e.Row)), e.Action));
        }
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    protected override void
OnRowDeleting(System.Data.DataRowChangeEventArgs e) {
        base.OnRowDeleting(e);
        if ((this.LecturersRowDeleting != null)) {
            this.LecturersRowDeleting(this, new
LecturersRowChangeEvent(((LecturersRow)(e.Row)), e.Action));
        }
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    public void RemoveLecturersRow(LecturersRow row) {
        this.Rows.Remove(row);
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]

```

```

        public static System.Xml.Schema.XmlSchemaComplexType
GetTypedTableSchema(System.Xml.Schema.XmlSchemaSet xs) {
    System.Xml.Schema.XmlSchemaComplexType type = new
System.Xml.Schema.XmlSchemaComplexType();
    System.Xml.Schema.XmlSchemaSequence sequence = new
System.Xml.Schema.XmlSchemaSequence();
    TestSQL2DataSet ds = new TestSQL2DataSet();
    xs.Add(ds.GetSchemaSerializable());
    System.Xml.Schema.XmlSchemaAny any1 = new
System.Xml.Schema.XmlSchemaAny();
    any1.Namespace = "http://www.w3.org/2001/XMLSchema";
    any1.MinOccurs = new decimal(0);
    any1.MaxOccurs = decimal.MaxValue;
    any1.ProcessContents =
System.Xml.Schema.XmlSchemaContentProcessing.Lax;
    sequence.Items.Add(any1);
    System.Xml.Schema.XmlSchemaAny any2 = new
System.Xml.Schema.XmlSchemaAny();
    any2.Namespace = "urn:schemas-microsoft-com:xml-diffgram-v1";
    any2.MinOccurs = new decimal(1);
    any2.ProcessContents =
System.Xml.Schema.XmlSchemaContentProcessing.Lax;
    sequence.Items.Add(any2);
    System.Xml.Schema.XmlSchemaAttribute attribute1 = new
System.Xml.Schema.XmlSchemaAttribute();
    attribute1.Name = "namespace";
    attribute1.FixedValue = ds.Namespace;
    type.Attributes.Add(attribute1);
    System.Xml.Schema.XmlSchemaAttribute attribute2 = new
System.Xml.Schema.XmlSchemaAttribute();
    attribute2.Name = "tableName";
    attribute2.FixedValue = "LecturersDataTable";
    type.Attributes.Add(attribute2);
    type.Particle = sequence;
    return type;
}
}

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGe
nerator", "2.0.0.0")]
[System.Serializable()]

[System.Xml.Serialization.XmlSchemaProviderAttribute("GetTypedTableSchema")]
public partial class OfficesDataTable : System.Data.DataTable,
System.Collections.IEnumerable {

    private System.Data.DataColumn columnOfficeID;

    private System.Data.DataColumn columnLectID;

    private System.Data.DataColumn columnBuilding;

    private System.Data.DataColumn columnRoom;

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    public OfficesDataTable() {
        this.TableName = "Offices";
        this.BeginInit();
        this.InitClass();
        this.EndInit();
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]

```

```

internal OfficesDataTable(System.Data.DataTable table) {
    this.TableName = table.TableName;
    if ((table.CaseSensitive != table.DataSet.CaseSensitive)) {
        this.CaseSensitive = table.CaseSensitive;
    }
    if ((table.Locale.ToString() != table.DataSet.Locale.ToString()))
    {
        this.Locale = table.Locale;
    }
    if ((table.Namespace != table.DataSet.Namespace)) {
        this.Namespace = table.Namespace;
    }
    this.Prefix = table.Prefix;
    this.MinimumCapacity = table.MinimumCapacity;
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
protected
OfficesDataTable(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context) :
    base(info, context) {
    this.InitVars();
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public System.Data.DataColumn OfficeIDColumn {
    get {
        return this.columnOfficeID;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public System.Data.DataColumn LectIDColumn {
    get {
        return this.columnLectID;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public System.Data.DataColumn BuildingColumn {
    get {
        return this.columnBuilding;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public System.Data.DataColumn RoomColumn {
    get {
        return this.columnRoom;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
[System.ComponentModel.Browsable(false)]
public int Count {
    get {
        return this.Rows.Count;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public OfficesRow this[int index] {
    get {
        return ((OfficesRow)(this.Rows[index]));
    }
}

```

```

    }
}

public event OfficesRowChangeEventHandler OfficesRowChanging;
public event OfficesRowChangeEventHandler OfficesRowChanged;
public event OfficesRowChangeEventHandler OfficesRowDeleting;
public event OfficesRowChangeEventHandler OfficesRowDeleted;

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public void AddOfficesRow(OfficesRow row) {
    this.Rows.Add(row);
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public OfficesRow AddOfficesRow(LecturersRow
parentLecturersRowByFK_Offices_Lecturers, string Building, string Room) {
    OfficesRow rowOfficesRow = ((OfficesRow)(this.NewRow()));
    rowOfficesRow.ItemArray = new object[] {
        null,
        parentLecturersRowByFK_Offices_Lecturers[0],
        Building,
        Room};
    this.Rows.Add(rowOfficesRow);
    return rowOfficesRow;
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public OfficesRow FindByOfficeID(int OfficeID) {
    return ((OfficesRow)(this.Rows.Find(new object[] {
        OfficeID})));
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public virtual System.Collections.IEnumerator GetEnumerator() {
    return this.Rows.GetEnumerator();
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public override System.Data.DataTable Clone() {
    OfficesDataTable cln = ((OfficesDataTable)(base.Clone()));
    cln.InitVars();
    return cln;
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
protected override System.Data.DataTable CreateInstance() {
    return new OfficesDataTable();
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
internal void InitVars() {
    this.columnOfficeID = base.Columns["OfficeID"];
    this.columnLectID = base.Columns["LectID"];
    this.columnBuilding = base.Columns["Building"];
    this.columnRoom = base.Columns["Room"];
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
private void InitClass() {
    this.columnOfficeID = new System.Data.DataColumn("OfficeID",
typeof(int), null, System.Data.MappingType.Element);
}

```

```

        base.Columns.Add(this.columnOfficeID);
        this.columnLectID = new System.Data.DataColumn("LectID",
typeof(int), null, System.Data.MappingType.Element);
        base.Columns.Add(this.columnLectID);
        this.columnBuilding = new System.Data.DataColumn("Building",
typeof(string), null, System.Data.MappingType.Element);
        base.Columns.Add(this.columnBuilding);
        this.columnRoom = new System.Data.DataColumn("Room",
typeof(string), null, System.Data.MappingType.Element);
        base.Columns.Add(this.columnRoom);
        this.Constraints.Add(new
System.Data.UniqueConstraint("Constraint1", new System.Data.DataColumn[] {
            this.columnOfficeID}, true));
        this.columnOfficeID.AutoIncrement = true;
        this.columnOfficeID.AllowDBNull = false;
        this.columnOfficeID.ReadOnly = true;
        this.columnOfficeID.Unique = true;
        this.columnLectID.AllowDBNull = false;
        this.columnBuilding.AllowDBNull = false;
        this.columnBuilding.MaxLength = 10;
        this.columnRoom.AllowDBNull = false;
        this.columnRoom.MaxLength = 10;
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    public OfficesRow NewOfficesRow() {
        return ((OfficesRow)(this.NewRow()));
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    protected override System.Data.DataRow
NewRowFromBuilder(System.Data.DataRowBuilder builder) {
        return new OfficesRow(builder);
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    protected override System.Type GetRowType() {
        return typeof(OfficesRow);
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    protected override void
OnRowChanged(System.Data.DataRowChangeEventArgs e) {
        base.OnRowChanged(e);
        if ((this.OfficesRowChanged != null)) {
            this.OfficesRowChanged(this, new
OfficesRowChangeEvent(((OfficesRow)(e.Row)), e.Action));
        }
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    protected override void
OnRowChanging(System.Data.DataRowChangeEventArgs e) {
        base.OnRowChanging(e);
        if ((this.OfficesRowChanging != null)) {
            this.OfficesRowChanging(this, new
OfficesRowChangeEvent(((OfficesRow)(e.Row)), e.Action));
        }
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    protected override void
OnRowDeleted(System.Data.DataRowChangeEventArgs e) {
        base.OnRowDeleted(e);
    }

```

```

        if ((this.OfficesRowDeleted != null)) {
            this.OfficesRowDeleted(this, new
OfficesRowChangeEvent(((OfficesRow)(e.Row)), e.Action));
        }
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    protected override void
OnRowDeleting(System.Data.DataRowChangeEventArgs e) {
        base.OnRowDeleting(e);
        if ((this.OfficesRowDeleting != null)) {
            this.OfficesRowDeleting(this, new
OfficesRowChangeEvent(((OfficesRow)(e.Row)), e.Action));
        }
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    public void RemoveOfficesRow(OfficesRow row) {
        this.Rows.Remove(row);
    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    public static System.Xml.Schema.XmlSchemaComplexType
GetTypedTableSchema(System.Xml.Schema.XmlSchemaSet xs) {
        System.Xml.Schema.XmlSchemaComplexType type = new
System.Xml.Schema.XmlSchemaComplexType();
        System.Xml.Schema.XmlSchemaSequence sequence = new
System.Xml.Schema.XmlSchemaSequence();
        TestSQL2DataSet ds = new TestSQL2DataSet();
        xs.Add(ds.GetSchemaSerializable());
        System.Xml.Schema.XmlSchemaAny any1 = new
System.Xml.Schema.XmlSchemaAny();
        any1.Namespace = "http://www.w3.org/2001/XMLSchema";
        any1.MinOccurs = new decimal(0);
        any1.MaxOccurs = decimal.MaxValue;
        any1.ProcessContents =
System.Xml.Schema.XmlSchemaContentProcessing.Lax;
        sequence.Items.Add(any1);
        System.Xml.Schema.XmlSchemaAny any2 = new
System.Xml.Schema.XmlSchemaAny();
        any2.Namespace = "urn:schemas-microsoft-com:xml-diffgram-v1";
        any2.MinOccurs = new decimal(1);
        any2.ProcessContents =
System.Xml.Schema.XmlSchemaContentProcessing.Lax;
        sequence.Items.Add(any2);
        System.Xml.Schema.XmlSchemaAttribute attribute1 = new
System.Xml.Schema.XmlSchemaAttribute();
        attribute1.Name = "namespace";
        attribute1.FixedValue = ds.Namespace;
        type.Attributes.Add(attribute1);
        System.Xml.Schema.XmlSchemaAttribute attribute2 = new
System.Xml.Schema.XmlSchemaAttribute();
        attribute2.Name = "tableName";
        attribute2.FixedValue = "OfficesDataTable";
        type.Attributes.Add(attribute2);
        type.Particle = sequence;
        return type;
    }
}

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGe
nerator", "2.0.0.0")]
public partial class LecturersRow : System.Data.DataRow {

```

```

private LecturersDataTable tableLecturers;

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
internal LecturersRow(System.Data.DataRowBuilder rb) :
    base(rb) {
    this.tableLecturers = ((LecturersDataTable)(this.Table));
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public int LectID {
    get {
        return ((int)(this[this.tableLecturers.LectIDColumn]));
    }
    set {
        this[this.tableLecturers.LectIDColumn] = value;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public string Name {
    get {
        return ((string)(this[this.tableLecturers.NameColumn]));
    }
    set {
        this[this.tableLecturers.NameColumn] = value;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public string Course {
    get {
        return ((string)(this[this.tableLecturers.CourseColumn]));
    }
    set {
        this[this.tableLecturers.CourseColumn] = value;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public int Students {
    get {
        try {
            return ((int)(this[this.tableLecturers.StudentsColumn]));
        }
        catch (System.InvalidCastException e) {
            throw new System.Data.StrongTypingException("The value for
column \'Students\' in table \'Lecturers\' is DBNull.", e);
        }
    }
    set {
        this[this.tableLecturers.StudentsColumn] = value;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public bool IsStudentsNull() {
    return this.IsNull(this.tableLecturers.StudentsColumn);
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public void SetStudentsNull() {
    this[this.tableLecturers.StudentsColumn] = System.Convert.DBNull;
}

```

```

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public OfficesRow[] GetOfficesRows() {
    return
(((OfficesRow[]) (base.GetChildRows(this.Table.ChildRelations["FK_Offices_Lecturers"])))));
}

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "2.0.0.0")]
public partial class OfficesRow : System.Data.DataRow {

    private OfficesDataTable tableOffices;

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
internal OfficesRow(System.Data.DataRowBuilder rb) :
    base(rb) {
        this.tableOffices = ((OfficesDataTable)(this.Table));
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public int OfficeID {
    get {
        return ((int)(this[this.tableOffices.OfficeIDColumn]));
    }
    set {
        this[this.tableOffices.OfficeIDColumn] = value;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public int LectID {
    get {
        return ((int)(this[this.tableOffices.LectIDColumn]));
    }
    set {
        this[this.tableOffices.LectIDColumn] = value;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public string Building {
    get {
        return ((string)(this[this.tableOffices.BuildingColumn]));
    }
    set {
        this[this.tableOffices.BuildingColumn] = value;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public string Room {
    get {
        return ((string)(this[this.tableOffices.RoomColumn]));
    }
    set {
        this[this.tableOffices.RoomColumn] = value;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public LecturersRow LecturersRow {

```

```

        get {
            return
            ((LecturersRow)(this.GetParentRow(this.Table.ParentRelations["FK_Offices_Lecturers
"])));
        }
        set {
            this.SetParentRow(value,
this.Table.ParentRelations["FK_Offices_Lecturers"]);
        }
    }
}

```

```

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGe
nerator", "2.0.0.0")]

```

```

    public class LecturersRowChangeEvent : System.EventArgs {

        private LecturersRow eventRow;

        private System.Data.DataRowAction eventAction;

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        public LecturersRowChangeEvent(LecturersRow row,
System.Data.DataRowAction action) {
            this.eventRow = row;
            this.eventAction = action;
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        public LecturersRow Row {
            get {
                return this.eventRow;
            }
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        public System.Data.DataRowAction Action {
            get {
                return this.eventAction;
            }
        }
    }
}

```

```

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGe
nerator", "2.0.0.0")]

```

```

    public class OfficesRowChangeEvent : System.EventArgs {

        private OfficesRow eventRow;

        private System.Data.DataRowAction eventAction;

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        public OfficesRowChangeEvent(OfficesRow row, System.Data.DataRowAction
action) {
            this.eventRow = row;
            this.eventAction = action;
        }

        [System.Diagnostics.DebuggerNonUserCodeAttribute()]
        public OfficesRow Row {
            get {
                return this.eventRow;
            }
        }
    }
}

```

```

    }

    [System.Diagnostics.DebuggerNonUserCodeAttribute()]
    public System.Data.DataRowAction Action {
        get {
            return this.eventAction;
        }
    }
}
}
}
namespace DBUpdate1.TestSQL2DataSetTableAdapters {

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "2.0.0.0")]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.ComponentModel.ToolboxItem(true)]
[System.ComponentModel.DataObjectAttribute(true)]

[System.ComponentModel.DesignerAttribute("Microsoft.VisualStudio.Data.Design.TableAdapterDesigner, Microsoft.VisualStudio",
    ", Version=8.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a")]
[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]
public partial class LecturersTableAdapter : System.ComponentModel.Component {

    private System.Data.SqlClient.SqlDataAdapter _adapter;

    private System.Data.SqlClient.SqlConnection _connection;

    private System.Data.SqlClient.SqlCommand[] _commandCollection;

    private bool _clearBeforeFill;

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public LecturersTableAdapter() {
    this.ClearBeforeFill = true;
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
private System.Data.SqlClient.SqlDataAdapter Adapter {
    get {
        if ((this._adapter == null)) {
            this.InitAdapter();
        }
        return this._adapter;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
internal System.Data.SqlClient.SqlConnection Connection {
    get {
        if ((this._connection == null)) {
            this.InitConnection();
        }
        return this._connection;
    }
    set {
        this._connection = value;
        if ((this.Adapter.InsertCommand != null)) {
            this.Adapter.InsertCommand.Connection = value;
        }
        if ((this.Adapter.DeleteCommand != null)) {

```

```

        this.Adapter.DeleteCommand.Connection = value;
    }
    if ((this.Adapter.UpdateCommand != null)) {
        this.Adapter.UpdateCommand.Connection = value;
    }
    for (int i = 0; (i < this.CommandCollection.Length); i = (i + 1))
    {
        if ((this.CommandCollection[i] != null)) {
            ((System.Data.SqlClient.SqlCommand)(this.CommandCollection[i])).Connection =
            value;
        }
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
protected System.Data.SqlClient.SqlCommand[] CommandCollection {
    get {
        if ((this._commandCollection == null)) {
            this.InitCommandCollection();
        }
        return this._commandCollection;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public bool ClearBeforeFill {
    get {
        return this._clearBeforeFill;
    }
    set {
        this._clearBeforeFill = value;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
private void InitAdapter() {
    this._adapter = new System.Data.SqlClient.SqlDataAdapter();
    System.Data.Common.DataTableMapping tableMapping = new
System.Data.Common.DataTableMapping();
    tableMapping.SourceTable = "Table";
    tableMapping.DataSetTable = "Lecturers";
    tableMapping.ColumnMappings.Add("LectID", "LectID");
    tableMapping.ColumnMappings.Add("Name", "Name");
    tableMapping.ColumnMappings.Add("Course", "Course");
    tableMapping.ColumnMappings.Add("Students", "Students");
    this._adapter.TableMappings.Add(tableMapping);
    this._adapter.DeleteCommand = new System.Data.SqlClient.SqlCommand();
    this._adapter.DeleteCommand.Connection = this.Connection;
    this._adapter.DeleteCommand.CommandText = "DELETE FROM
[dbo].[Lecturers] WHERE (([LectID] = @Original_LectID) AND ([Name] = " +
        "@Original_Name) AND ([Course] = @Original_Course) AND
        (@IsNull_Students = 1 AND " +
        " [Students] IS NULL) OR ([Students] = @Original_Students))";
    this._adapter.DeleteCommand.CommandType =
System.Data.CommandType.Text;
    this._adapter.DeleteCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_LectID", System.Data.SqlDbType.Int,
0, System.Data.ParameterDirection.Input, 0, 0, "LectID",
System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._adapter.DeleteCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_Name", System.Data.SqlDbType.Char,

```

```

0, System.Data.ParameterDirection.Input, 0, 0, "Name",
System.Data.DataRowVersion.Original, false, null, "", "", "");
    this._adapter.DeleteCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_Course", System.Data.SqlDbType.Char,
0, System.Data.ParameterDirection.Input, 0, 0, "Course",
System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._adapter.DeleteCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@IsNull_Students", System.Data.SqlDbType.Int,
0, System.Data.ParameterDirection.Input, 0, 0, "Students",
System.Data.DataRowVersion.Original, true, null, "", "", ""));
    this._adapter.DeleteCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_Students",
System.Data.SqlDbType.Int, 0, System.Data.ParameterDirection.Input, 0, 0,
"Students", System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._adapter.InsertCommand = new System.Data.SqlClient.SqlCommand();
    this._adapter.InsertCommand.Connection = this.Connection;
    this._adapter.InsertCommand.CommandText = "INSERT INTO
[dbo].[Lecturers] ([Name], [Course], [Students]) VALUES (@Name, @Cour" +
        "se, @Students);\r\nSELECT LectID, Name, Course, Students FROM
Lecturers WHERE (Lec" +
        "tID = SCOPE_IDENTITY());";
    this._adapter.InsertCommand.CommandType =
System.Data.CommandType.Text;
    this._adapter.InsertCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Name", System.Data.SqlDbType.Char, 0,
System.Data.ParameterDirection.Input, 0, 0, "Name",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._adapter.InsertCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Course", System.Data.SqlDbType.Char, 0,
System.Data.ParameterDirection.Input, 0, 0, "Course",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._adapter.InsertCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Students", System.Data.SqlDbType.Int, 0,
System.Data.ParameterDirection.Input, 0, 0, "Students",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._adapter.UpdateCommand = new System.Data.SqlClient.SqlCommand();
    this._adapter.UpdateCommand.Connection = this.Connection;
    this._adapter.UpdateCommand.CommandText = @"UPDATE [dbo].[Lecturers]
SET [Name] = @Name, [Course] = @Course, [Students] = @Students WHERE (([LectID] =
@Original_LectID) AND ([Name] = @Original_Name) AND ([Course] = @Original_Course)
AND ((@IsNull_Students = 1 AND [Students] IS NULL) OR ([Students] =
@Original_Students)));";
    this._adapter.UpdateCommand.CommandText = "SELECT LectID, Name, Course, Students FROM Lecturers WHERE (LectID = @LectID)";
    this._adapter.UpdateCommand.CommandType =
System.Data.CommandType.Text;
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Name", System.Data.SqlDbType.Char, 0,
System.Data.ParameterDirection.Input, 0, 0, "Name",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Course", System.Data.SqlDbType.Char, 0,
System.Data.ParameterDirection.Input, 0, 0, "Course",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Students", System.Data.SqlDbType.Int, 0,
System.Data.ParameterDirection.Input, 0, 0, "Students",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_LectID", System.Data.SqlDbType.Int,
0, System.Data.ParameterDirection.Input, 0, 0, "LectID",
System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_Name", System.Data.SqlDbType.Char,

```

```

0, System.Data.ParameterDirection.Input, 0, 0, "Name",
System.Data.DataRowVersion.Original, false, null, "", "", "");
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_Course", System.Data.SqlDbType.Char,
0, System.Data.ParameterDirection.Input, 0, 0, "Course",
System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@IsNull_Students", System.Data.SqlDbType.Int,
0, System.Data.ParameterDirection.Input, 0, 0, "Students",
System.Data.DataRowVersion.Original, true, null, "", "", ""));
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_Students",
System.Data.SqlDbType.Int, 0, System.Data.ParameterDirection.Input, 0, 0,
"Students", System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@LectID", System.Data.SqlDbType.Int, 4,
System.Data.ParameterDirection.Input, 0, 0, "LectID",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
private void InitConnection() {
    this._connection = new System.Data.SqlClient.SqlConnection();
    this._connection.ConnectionString =
global::DBUpdate1.Properties.Settings.Default.TestSQL2ConnectionString;
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
private void InitCommandCollection() {
    this._commandCollection = new System.Data.SqlClient.SqlCommand[1];
    this._commandCollection[0] = new System.Data.SqlClient.SqlCommand();
    this._commandCollection[0].Connection = this.Connection;
    this._commandCollection[0].CommandText = "SELECT LectID, Name, Course,
Students FROM dbo.Lecturers";
    this._commandCollection[0].CommandType = System.Data.CommandType.Text;
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectM
ethodType.Fill, true)]
public virtual int Fill(TestSQL2DataSet.LecturersDataTable dataTable) {
    this.Adapter.SelectCommand = this.CommandCollection[0];
    if ((this.ClearBeforeFill == true)) {
        dataTable.Clear();
    }
    int returnValue = this.Adapter.Fill(dataTable);
    return returnValue;
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectM
ethodType.Select, true)]
public virtual TestSQL2DataSet.LecturersDataTable GetData() {
    this.Adapter.SelectCommand = this.CommandCollection[0];
    TestSQL2DataSet.LecturersDataTable dataTable = new
TestSQL2DataSet.LecturersDataTable();
    this.Adapter.Fill(dataTable);
    return dataTable;
}

```

```

    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]
public virtual int Update(TestSQL2DataSet.LecturersDataTable dataTable) {
    return this.Adapter.Update(dataTable);
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]
public virtual int Update(TestSQL2DataSet dataSet) {
    return this.Adapter.Update(dataSet, "Lecturers");
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]
public virtual int Update(System.Data.DataRow dataRow) {
    return this.Adapter.Update(new System.Data.DataRow[] {
        dataRow});
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]
public virtual int Update(System.Data.DataRow[] dataRows) {
    return this.Adapter.Update(dataRows);
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodType.Delete, true)]
public virtual int Delete(int Original_LectID, string Original_Name,
string Original_Course, System.Nullable<int> Original_Students) {
    this.Adapter.DeleteCommand.Parameters[0].Value =
((int)(Original_LectID));
    if ((Original_Name == null)) {
        throw new System.ArgumentNullException("Original_Name");
    }
    else {
        this.Adapter.DeleteCommand.Parameters[1].Value =
((string)(Original_Name));
    }
    if ((Original_Course == null)) {
        throw new System.ArgumentNullException("Original_Course");
    }
    else {
        this.Adapter.DeleteCommand.Parameters[2].Value =
((string)(Original_Course));
    }
    if ((Original_Students.HasValue == true)) {
        this.Adapter.DeleteCommand.Parameters[3].Value = ((object)(0));
        this.Adapter.DeleteCommand.Parameters[4].Value =
((int)(Original_Students.Value));
    }
    else {
        this.Adapter.DeleteCommand.Parameters[3].Value = ((object)(1));
        this.Adapter.DeleteCommand.Parameters[4].Value =
System.DBNull.Value;
    }
}

```

```

    }
    System.Data.ConnectionState previousConnectionState =
this.Adapter.DeleteCommand.Connection.State;
    if (((this.Adapter.DeleteCommand.Connection.State &
System.Data.ConnectionState.Open)
        != System.Data.ConnectionState.Open)) {
        this.Adapter.DeleteCommand.Connection.Open();
    }
    try {
        int returnValue = this.Adapter.DeleteCommand.ExecuteNonQuery();
        return returnValue;
    }
    finally {
        if ((previousConnectionState ==
System.Data.ConnectionState.Closed)) {
            this.Adapter.DeleteCommand.Connection.Close();
        }
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectM
ethodType.Insert, true)]
public virtual int Insert(string Name, string Course, System.Nullable<int>
Students) {
    if ((Name == null)) {
        throw new System.ArgumentNullException("Name");
    }
    else {
        this.Adapter.InsertCommand.Parameters[0].Value = ((string)(Name));
    }
    if ((Course == null)) {
        throw new System.ArgumentNullException("Course");
    }
    else {
        this.Adapter.InsertCommand.Parameters[1].Value =
((string)(Course));
    }
    if ((Students.HasValue == true)) {
        this.Adapter.InsertCommand.Parameters[2].Value =
((int)(Students.Value));
    }
    else {
        this.Adapter.InsertCommand.Parameters[2].Value =
System.DBNull.Value;
    }
    System.Data.ConnectionState previousConnectionState =
this.Adapter.InsertCommand.Connection.State;
    if (((this.Adapter.InsertCommand.Connection.State &
System.Data.ConnectionState.Open)
        != System.Data.ConnectionState.Open)) {
        this.Adapter.InsertCommand.Connection.Open();
    }
    try {
        int returnValue = this.Adapter.InsertCommand.ExecuteNonQuery();
        return returnValue;
    }
    finally {
        if ((previousConnectionState ==
System.Data.ConnectionState.Closed)) {
            this.Adapter.InsertCommand.Connection.Close();

```



```

        if (((this.Adapter.UpdateCommand.Connection.State &
System.Data.ConnectionState.Open)
            != System.Data.ConnectionState.Open)) {
            this.Adapter.UpdateCommand.Connection.Open();
        }
        try {
            int returnValue = this.Adapter.UpdateCommand.ExecuteNonQuery();
            return returnValue;
        }
        finally {
            if ((previousConnectionState ==
System.Data.ConnectionState.Closed)) {
                this.Adapter.UpdateCommand.Connection.Close();
            }
        }
    }
}

```

```

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGe
nerator", "2.0.0.0")]

```

```

[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.ComponentModel.ToolboxItem(true)]
[System.ComponentModel.DataObjectAttribute(true)]

```

```

[System.ComponentModel.DesignerAttribute("Microsoft.VisualStudio.DataSource.Design.T
ableAdapterDesigner, Microsoft.VisualStudio" +
    ", Version=8.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a")]
[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]

```

```

public partial class OfficesTableAdapter : System.ComponentModel.Component {

```

```

    private System.Data.SqlClient.SqlDataAdapter _adapter;

```

```

    private System.Data.SqlClient.SqlConnection _connection;

```

```

    private System.Data.SqlClient.SqlCommand[] _commandCollection;

```

```

    private bool _clearBeforeFill;

```

```

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

```

```

    public OfficesTableAdapter() {
        this.ClearBeforeFill = true;
    }

```

```

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

```

```

    private System.Data.SqlClient.SqlDataAdapter Adapter {
        get {
            if ((this._adapter == null)) {
                this.InitAdapter();
            }
            return this._adapter;
        }
    }

```

```

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

```

```

    internal System.Data.SqlClient.SqlConnection Connection {
        get {
            if ((this._connection == null)) {
                this.InitConnection();
            }
            return this._connection;
        }
        set {
            this._connection = value;
        }
    }

```

```

        if ((this.Adapter.InsertCommand != null)) {
            this.Adapter.InsertCommand.Connection = value;
        }
        if ((this.Adapter.DeleteCommand != null)) {
            this.Adapter.DeleteCommand.Connection = value;
        }
        if ((this.Adapter.UpdateCommand != null)) {
            this.Adapter.UpdateCommand.Connection = value;
        }
        for (int i = 0; (i < this.CommandCollection.Length); i = (i + 1))
    {
        if ((this.CommandCollection[i] != null)) {

            ((System.Data.SqlClient.SqlCommand)(this.CommandCollection[i])).Connection =
value;

        }
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
protected System.Data.SqlClient.SqlCommand[] CommandCollection {
    get {
        if ((this._commandCollection == null)) {
            this.InitCommandCollection();
        }
        return this._commandCollection;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
public bool ClearBeforeFill {
    get {
        return this._clearBeforeFill;
    }
    set {
        this._clearBeforeFill = value;
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
private void InitAdapter() {
    this._adapter = new System.Data.SqlClient.SqlDataAdapter();
    System.Data.Common.DataTableMapping tableMapping = new
System.Data.Common.DataTableMapping();
    tableMapping.SourceTable = "Table";
    tableMapping.DataSetTable = "Offices";
    tableMapping.ColumnMappings.Add("OfficeID", "OfficeID");
    tableMapping.ColumnMappings.Add("LectID", "LectID");
    tableMapping.ColumnMappings.Add("Building", "Building");
    tableMapping.ColumnMappings.Add("Room", "Room");
    this._adapter.TableMappings.Add(tableMapping);
    this._adapter.DeleteCommand = new System.Data.SqlClient.SqlCommand();
    this._adapter.DeleteCommand.Connection = this.Connection;
    this._adapter.DeleteCommand.CommandText = "DELETE FROM [dbo].[Offices]
WHERE (([OfficeID] = @Original_OfficeID) AND ([LectID] +
    ") = @Original_LectID) AND ([Building] = @Original_Building) AND
([Room] = @Original_Room))";
    this._adapter.DeleteCommand.CommandType =
System.Data.CommandType.Text;
    this._adapter.DeleteCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_OfficeID",

```

```

System.Data.SqlDbType.Int, 0, System.Data.ParameterDirection.Input, 0, 0,
"OfficeID", System.Data.DataRowVersion.Original, false, null, "", "", "");
    this._adapter.DeleteCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_LectID", System.Data.SqlDbType.Int,
0, System.Data.ParameterDirection.Input, 0, 0, "LectID",
System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._adapter.DeleteCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_Building",
System.Data.SqlDbType.Char, 0, System.Data.ParameterDirection.Input, 0, 0,
"Building", System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._adapter.DeleteCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_Room", System.Data.SqlDbType.Char,
0, System.Data.ParameterDirection.Input, 0, 0, "Room",
System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._adapter.InsertCommand = new System.Data.SqlClient.SqlCommand();
    this._adapter.InsertCommand.Connection = this.Connection;
    this._adapter.InsertCommand.CommandText = "INSERT INTO [dbo].[Offices]
([LectID], [Building], [Room]) VALUES (@LectID, @Buil" +
        "ding, @Room);\r\nSELECT OfficeID, LectID, Building, Room FROM
Offices WHERE (Offic" +
        "eID = SCOPE_IDENTITY())";
    this._adapter.InsertCommand.CommandType =
System.Data.CommandType.Text;
    this._adapter.InsertCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@LectID", System.Data.SqlDbType.Int, 0,
System.Data.ParameterDirection.Input, 0, 0, "LectID",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._adapter.InsertCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Building", System.Data.SqlDbType.Char, 0,
System.Data.ParameterDirection.Input, 0, 0, "Building",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._adapter.InsertCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Room", System.Data.SqlDbType.Char, 0,
System.Data.ParameterDirection.Input, 0, 0, "Room",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._adapter.UpdateCommand = new System.Data.SqlClient.SqlCommand();
    this._adapter.UpdateCommand.Connection = this.Connection;
    this._adapter.UpdateCommand.CommandText = @"UPDATE [dbo].[Offices] SET
[LectID] = @LectID, [Building] = @Building, [Room] = @Room WHERE (([OfficeID] =
@Original_OfficeID) AND ([LectID] = @Original_LectID) AND ([Building] =
@Original_Building) AND ([Room] = @Original_Room));
SELECT OfficeID, LectID, Building, Room FROM Offices WHERE (OfficeID =
@OfficeID)";
    this._adapter.UpdateCommand.CommandType =
System.Data.CommandType.Text;
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@LectID", System.Data.SqlDbType.Int, 0,
System.Data.ParameterDirection.Input, 0, 0, "LectID",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Building", System.Data.SqlDbType.Char, 0,
System.Data.ParameterDirection.Input, 0, 0, "Building",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Room", System.Data.SqlDbType.Char, 0,
System.Data.ParameterDirection.Input, 0, 0, "Room",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_OfficeID",
System.Data.SqlDbType.Int, 0, System.Data.ParameterDirection.Input, 0, 0,
"OfficeID", System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_LectID", System.Data.SqlDbType.Int,

```

```

0, System.Data.ParameterDirection.Input, 0, 0, "LectID",
System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_Building",
System.Data.SqlDbType.Char, 0, System.Data.ParameterDirection.Input, 0, 0,
"Building", System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_Room", System.Data.SqlDbType.Char,
0, System.Data.ParameterDirection.Input, 0, 0, "Room",
System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._adapter.UpdateCommand.Parameters.Add(new
System.Data.SqlClient.SqlParameter("@OfficeID", System.Data.SqlDbType.Int, 4,
System.Data.ParameterDirection.Input, 0, 0, "OfficeID",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
private void InitConnection() {
    this._connection = new System.Data.SqlClient.SqlConnection();
    this._connection.ConnectionString =
global::DBUpdate1.Properties.Settings.Default.TestSQL2ConnectionString;
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]
private void InitCommandCollection() {
    this._commandCollection = new System.Data.SqlClient.SqlCommand[2];
    this._commandCollection[0] = new System.Data.SqlClient.SqlCommand();
    this._commandCollection[0].Connection = this.Connection;
    this._commandCollection[0].CommandText = "SELECT OfficeID, LectID,
Building, Room FROM dbo.Offices";
    this._commandCollection[0].CommandType = System.Data.CommandType.Text;
    this._commandCollection[1] = new System.Data.SqlClient.SqlCommand();
    this._commandCollection[1].Connection = this.Connection;
    this._commandCollection[1].CommandText = @"UPDATE [dbo].[Offices] SET
[LectID] = @LectID, [Building] = @Building, [Room] = @Room WHERE (([OfficeID] =
@Original_OfficeID) AND ([LectID] = @Original_LectID) AND ([Building] =
@Original_Building) AND ([Room] = @Original_Room));
SELECT OfficeID, LectID, Building, Room FROM Offices WHERE (OfficeID =
@OfficeID)";
    this._commandCollection[1].CommandType = System.Data.CommandType.Text;
    this._commandCollection[1].Parameters.Add(new
System.Data.SqlClient.SqlParameter("@LectID", System.Data.SqlDbType.Int, 4,
System.Data.ParameterDirection.Input, 0, 0, "LectID",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._commandCollection[1].Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Building", System.Data.SqlDbType.Char, 10,
System.Data.ParameterDirection.Input, 0, 0, "Building",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._commandCollection[1].Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Room", System.Data.SqlDbType.Char, 10,
System.Data.ParameterDirection.Input, 0, 0, "Room",
System.Data.DataRowVersion.Current, false, null, "", "", ""));
    this._commandCollection[1].Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_OfficeID",
System.Data.SqlDbType.Int, 4, System.Data.ParameterDirection.Input, 0, 0,
"OfficeID", System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._commandCollection[1].Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_LectID", System.Data.SqlDbType.Int,
4, System.Data.ParameterDirection.Input, 0, 0, "LectID",
System.Data.DataRowVersion.Original, false, null, "", "", ""));
    this._commandCollection[1].Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_Building",
System.Data.SqlDbType.Char, 10, System.Data.ParameterDirection.Input, 0, 0,
"Building", System.Data.DataRowVersion.Original, false, null, "", "", ""));

```

```

        this._commandCollection[1].Parameters.Add(new
System.Data.SqlClient.SqlParameter("@Original_Room", System.Data.SqlDbType.Char,
10, System.Data.ParameterDirection.Input, 0, 0, "Room",
System.Data.DataRowVersion.Original, false, null, "", "", ""));
        this._commandCollection[1].Parameters.Add(new
System.Data.SqlClient.SqlParameter("@OfficeID", System.Data.SqlDbType.Int, 4,
System.Data.ParameterDirection.Input, 0, 0, "OfficeID",
System.Data.DataRowVersion.Original, false, null, "", "", ""));
    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectM
ethodType.Fill, true)]
    public virtual int Fill(TestSQL2DataSet.OfficesDataTable dataTable) {
        this.Adapter.SelectCommand = this.CommandCollection[0];
        if ((this.ClearBeforeFill == true)) {
            dataTable.Clear();
        }
        int returnValue = this.Adapter.Fill(dataTable);
        return returnValue;
    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectM
ethodType.Select, true)]
    public virtual TestSQL2DataSet.OfficesDataTable GetData() {
        this.Adapter.SelectCommand = this.CommandCollection[0];
        TestSQL2DataSet.OfficesDataTable dataTable = new
TestSQL2DataSet.OfficesDataTable();
        this.Adapter.Fill(dataTable);
        return dataTable;
    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]
    public virtual int Update(TestSQL2DataSet.OfficesDataTable dataTable) {
        return this.Adapter.Update(dataTable);
    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]
    public virtual int Update(TestSQL2DataSet dataSet) {
        return this.Adapter.Update(dataSet, "Offices");
    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]
    public virtual int Update(System.Data.DataRow dataRow) {
        return this.Adapter.Update(new System.Data.DataRow[] {
            dataRow});
    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]

```

```

public virtual int Update(System.Data.DataRow[] dataRows) {
    return this.Adapter.Update(dataRows);
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectM
ethodType.Delete, true)]
public virtual int Delete(int Original_OfficeID, int Original_LectID,
string Original_Building, string Original_Room) {
    this.Adapter.DeleteCommand.Parameters[0].Value =
((int)(Original_OfficeID));
    this.Adapter.DeleteCommand.Parameters[1].Value =
((int)(Original_LectID));
    if ((Original_Building == null)) {
        throw new System.ArgumentNullException("Original_Building");
    }
    else {
        this.Adapter.DeleteCommand.Parameters[2].Value =
((string)(Original_Building));
    }
    if ((Original_Room == null)) {
        throw new System.ArgumentNullException("Original_Room");
    }
    else {
        this.Adapter.DeleteCommand.Parameters[3].Value =
((string)(Original_Room));
    }
    System.Data.ConnectionState previousConnectionState =
this.Adapter.DeleteCommand.Connection.State;
    if (((this.Adapter.DeleteCommand.Connection.State &
System.Data.ConnectionState.Open)
!= System.Data.ConnectionState.Open)) {
        this.Adapter.DeleteCommand.Connection.Open();
    }
    try {
        int returnValue = this.Adapter.DeleteCommand.ExecuteNonQuery();
        return returnValue;
    }
    finally {
        if ((previousConnectionState ==
System.Data.ConnectionState.Closed)) {
            this.Adapter.DeleteCommand.Connection.Close();
        }
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectM
ethodType.Insert, true)]
public virtual int Insert(int LectID, string Building, string Room) {
    this.Adapter.InsertCommand.Parameters[0].Value = ((int)(LectID));
    if ((Building == null)) {
        throw new System.ArgumentNullException("Building");
    }
    else {
        this.Adapter.InsertCommand.Parameters[1].Value =
((string)(Building));
    }
}

```

```

        if ((Room == null)) {
            throw new System.ArgumentNullException("Room");
        }
        else {
            this.Adapter.InsertCommand.Parameters[2].Value = ((string)(Room));
        }
        System.Data.ConnectionState previousConnectionState =
this.Adapter.InsertCommand.Connection.State;
        if (((this.Adapter.InsertCommand.Connection.State &
System.Data.ConnectionState.Open)
            != System.Data.ConnectionState.Open)) {
            this.Adapter.InsertCommand.Connection.Open();
        }
        try {
            int returnValue = this.Adapter.InsertCommand.ExecuteNonQuery();
            return returnValue;
        }
        finally {
            if ((previousConnectionState ==
System.Data.ConnectionState.Closed)) {
                this.Adapter.InsertCommand.Connection.Close();
            }
        }
    }

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectM
ethodType.Update, true)]
    public virtual int Update(int LectID, string Building, string Room, int
Original_OfficeID, int Original_LectID, string Original_Building, string
Original_Room, int OfficeID) {
        this.Adapter.UpdateCommand.Parameters[0].Value = ((int)(LectID));
        if ((Building == null)) {
            throw new System.ArgumentNullException("Building");
        }
        else {
            this.Adapter.UpdateCommand.Parameters[1].Value =
((string)(Building));
        }
        if ((Room == null)) {
            throw new System.ArgumentNullException("Room");
        }
        else {
            this.Adapter.UpdateCommand.Parameters[2].Value = ((string)(Room));
        }
        this.Adapter.UpdateCommand.Parameters[3].Value =
((int)(Original_OfficeID));
        this.Adapter.UpdateCommand.Parameters[4].Value =
((int)(Original_LectID));
        if ((Original_Building == null)) {
            throw new System.ArgumentNullException("Original_Building");
        }
        else {
            this.Adapter.UpdateCommand.Parameters[5].Value =
((string)(Original_Building));
        }
        if ((Original_Room == null)) {
            throw new System.ArgumentNullException("Original_Room");
        }
        else {

```

```

        this.Adapter.UpdateCommand.Parameters[6].Value =
((string)(Original_Room));
    }
    this.Adapter.UpdateCommand.Parameters[7].Value = ((int)(OfficeID));
    System.Data.ConnectionState previousConnectionState =
this.Adapter.UpdateCommand.Connection.State;
    if (((this.Adapter.UpdateCommand.Connection.State &
System.Data.ConnectionState.Open)
        != System.Data.ConnectionState.Open)) {
        this.Adapter.UpdateCommand.Connection.Open();
    }
    try {
        int returnValue = this.Adapter.UpdateCommand.ExecuteNonQuery();
        return returnValue;
    }
    finally {
        if ((previousConnectionState ==
System.Data.ConnectionState.Closed)) {
            this.Adapter.UpdateCommand.Connection.Close();
        }
    }
}

[System.Diagnostics.DebuggerNonUserCodeAttribute()]

[System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")]

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectM
ethodType.Update, false)]
public virtual int NewUpdateQuery(int LectID, string Building, string
Room, int Original_OfficeID, int Original_LectID, string Original_Building, string
Original_Room, int OfficeID) {
    System.Data.SqlClient.SqlCommand command = this.CommandCollection[1];
    command.Parameters[0].Value = ((int)(LectID));
    if ((Building == null)) {
        throw new System.ArgumentNullException("Building");
    }
    else {
        command.Parameters[1].Value = ((string)(Building));
    }
    if ((Room == null)) {
        throw new System.ArgumentNullException("Room");
    }
    else {
        command.Parameters[2].Value = ((string)(Room));
    }
    command.Parameters[3].Value = ((int)(Original_OfficeID));
    command.Parameters[4].Value = ((int)(Original_LectID));
    if ((Original_Building == null)) {
        throw new System.ArgumentNullException("Original_Building");
    }
    else {
        command.Parameters[5].Value = ((string)(Original_Building));
    }
    if ((Original_Room == null)) {
        throw new System.ArgumentNullException("Original_Room");
    }
    else {
        command.Parameters[6].Value = ((string)(Original_Room));
    }
    command.Parameters[7].Value = ((int)(OfficeID));
    System.Data.ConnectionState previousConnectionState =
command.Connection.State;
    if (((command.Connection.State & System.Data.ConnectionState.Open)

```

```
        != System.Data.ConnectionState.Open)) {
            command.Connection.Open();
        }
        int returnValue;
        try {
            returnValue = command.ExecuteNonQuery();
        }
        finally {
            if ((previousConnectionState ==
System.Data.ConnectionState.Closed)) {
                command.Connection.Close();
            }
        }
        return returnValue;
    }
}
```

```
#pragma warning restore 1591
```