Programming with Event Driven in C#

Working with Files and Database Programming

Chalew Tesfaye
Department of Computer Science
Debre Berhan University
2017

Objectives

- After the end this lesson the student will be able to
  - ✓ Understand file system
  - ✓ Understand database system
  - ✓ Write a program that uses file system as a backed end
  - ✓ Write a program uses database system as a backed end

Lesson Outline

- File system and IO stream
- Read/write file
- Relational Database system
- ADO.Net programming
- Connect a project to database
- Using database objects in application program

DATABASE PROGRAMMING

Working with relational database system

DATABASE PROGRAMMING

- Introduction to Relational Database System
- Relational Database System Management System Software
- Structured Query Language
- Connecting C# application to Database System
- Creating and executing statements
- Working on populated data/Result set
- Using database objects in application program

Database System

- A database is
  - a collection of related,
  - logically coherent data used by the application programs in an organization.
- Advantages of databases
  - Less redundancy
  - Inconsistency avoidance
  - Efficiency
  - Data integrity
  - Confidentiality

DBMS

- A database management system (DBMS)
  - ✓ defines,
  - ✓ creates and
  - ✓ maintains a database.
  - ✓ allows controlled access to data in the database.
- A DBMS is a combination of five components:
  - ✓ hardware,
  - ✓ software,
  - ✓ data,
  - ✓ users and
  - ✓ procedures

Relational Database Model

- Data are organized in two-dimensional tables called relations.
- The tables are related to each other.
- A relation in an RDBMS has the following features:
  - ✓ Name of entity
  - ✓ Attributes
  - ✓ Tuples

Relational Integrity Constraints

- Constraints are **conditions** that must hold on **all** valid relation states.
- There are three *main types* of constraints in the relational model:
  - ✓ **Key** constraints
    - Primary Key
    - Unique Key
    - Foreign Key
  - ✓ **Entity integrity** constraints
  - ✓ **Referential integrity** constraints
- Another implicit constraint is the **domain** constraint
  - ✓ Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)
- Check constraints
- Default value constraint

Introduction to SQL

- SQL is the language standardized by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) for use on relational databases.
- It is a *declarative* rather than *procedural* language,
  - ✓ which means that users declare what they want without having to write a step-by-step procedure.

Introduction to SQL …

- In a relational database, we can define several operations to create new relations out of the existing ones.
- Basic operations:
  - ✓ Insert
  - ✓ Delete
  - ✓ Update
  - ✓ Select
  - ✓ Project
  - ✓ Join
  - ✓ Union
  - ✓ Intersection
  - ✓ Difference

Possible violations for each operation

- INSERT may violate any of the constraints:
  - ✓ Domain constraint:
    - if one of the attribute values provided for the new tuple is not of the specified attribute domain
  - ✓ Key constraint:
    - if the value of a key attribute in the new tuple already exists in another tuple in the relation
    - Primary Key
    - Unique Key
  - ✓ Referential integrity:
    - if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation
  - ✓ Entity integrity:
    - if the primary key value is null in the new tuple

Violations …

- DELETE may violate only referential integrity:
  - ✓ If the primary key value of the tuple being deleted is referenced from other tuples in the database
    - > Can be remedied by several actions: RESTRICT, CASCADE, SET NULL, SET DEFAULT
      - RESTRICT/NO ACTION option: reject the deletion
      - CASCADE option: propagate the new primary key value into the foreign keys of the referencing tuples
      - SET NULL option: set the foreign keys of the referencing tuples to NULL
      - SET DEFAULT option : set the foreign keys of the referencing tuples to the default value if any
  - ✓ One of the above options must be specified during database design for each foreign key constraint

Violations …

- UPDATE may violate domain constraint and NOT NULL constraint on an attribute being modified
- Any of the other constraints may also be violated, depending on the attribute being updated:
  - ✓ Updating the primary key (PK):
    - Similar to a DELETE followed by an INSERT
    - Need to specify similar options to DELETE
  - ✓ Updating a foreign key (FK):
    - May violate referential integrity
  - ✓ Updating an ordinary attribute (neither PK nor FK):
    - Can only violate domain constraints

Relational Database Management System

- Networked RDBMS
  - Oracle
    - For large mission critical systems runs on Linux
  - MS SQL Server
    - Used in small to medium sized systems run in Windows server
  - DB2
    - For large mission critical systems runs on IBM OS
  - MySQL
    - Open source, runs on major operating system, commonly used in web application
  - Apache Derby/Java DB
    - New java based relation database, can be embedded in application
  - …
- Embedded RDBMS
  - Apache Derby/Java DB
  - MS Access
  - Compact/SQLite version of MS SQL, Oracle

Essential SQL Skill

- Creating and/or modifying database objects
- Retrieving data from a table
- Insert, Delete, update data
- Using functions
  - ✓ String function
  - ✓ Numeric functions
  - ✓ Date/Time functions
  - ✓ Other functions
    - > CASE, IF, IFNULL
- Creating View
- Transact-SQL programming
  - ✓ Scripts
  - ✓ Creating triggers
  - ✓ Stored procedures
  - ✓ Functions
- Working with Transaction

ADO.NET

Connecting a project to Database system and access database objects

ADO.NET

- ActiveX Data Object .NET is the primary data access API for the .NET Framework
- provides the classes that you as you develop database application with C# and other .Net languages
- These classes can be divided into two
- **The .NET data provider**
  - ✓ which provide the classes that you use to access the data in database
- **Dataset**
  - ✓ which provide the classes that you can use to store and work with data in your application

The .NET data providers

- A .NET data provider is a set of classes that enables you to access data that's managed by a particular database server.
- All .NET data providers must include core classes for creating the four types of objects.
  - **Connection**
    - establishes a connection to a database server
  - **Command**
    - represents an individual SQL statement that can be executed against the database
  - **Data reader**
    - provides read-only forward access to the data in a database
  - **Data adapter**
    - provides the link between the command and connection objects and a dataset object

The .NET data providers

- Data provider included with the .NET Framework
- **SQL Server**
  - ✓ System.Data.SqlClient
    - > enables access MS SQL Server database
- **OLEDB**
  - ✓ System.Data.OleDb
    - > enables access to any database that supports OLEDB
- **ODBC**
  - ✓ System.Data.Odbc
    - > enables access to any database that supports ODBC
- **Oracle**
  - ✓ System.Data.OracleClient
  - ✓ Oracle.DataAccess.Client;
    - > enables access to Oracle database
- Class name for data provider

Object	**SQL Server**	**OLEDB**	**ODBC**	**Oracle**
Connection	SqlConnection	OleDbConnection	OdbcConnection	OracleConnection
Command	SqlCommand	OleDbCommand	OdbcCommand	OracleCommand
Data Reader	SqlDataReader	OleDbDataReade	OdbcDataReader	OracleDataReader
Data adapter	SqlDataAdapter	OleDbDataAdapter	OdbcDataAdapter	OracleDataAdapter

The .NET data providers

ADO.NET….

- The steps for executing database command in ADO.NET are:
  1. Create connection object
  2. Open the connection
  3. Create a command object encapsulating both Sql command and the connection
  4. Call a method on the command object to execute the command
  5. Close the connection by calling the close on the connection object
- You can create ADO.NET objects
  - ✓ By Writing the code yourself
  - ✓ From data source listed in the Data Sources window
    - > Add Data Source using Data Source Configuration wizard

Creating ADO.NET MS SQL Server Database objects

- **Create a connection object**
- The SqlConnection Object is handling the part of physical communication between the C# application and the SQL Server Database.
- An instance of the SqlConnection class in C# is supported the Data Provider for SQL Server Database.
- The SqlConnection instance takes Connection String as an argument and passes the value to the Constructor statement.

```
connetionString = "Data Source=ServerName; Initial Catalog=DatabaseName; User ID=UserName;
                   Password=Password“;
conn = new SqlConnection(connetionString);
cnn.Open();
```

- When the connection is established,
  - ✓ SQL Commands will execute with the help of the Connection Object and retrieve or manipulate the data in the database.
- Once the Database activities over,
  - ✓ Connection should be closed and release the Data Source resources.
- conn.Close();

Create a command object

- The Command Object in ADO.NET executes
  - ✓ SQL statements and
  - ✓ Stored Procedures against the data source specified in the C# Connection Object.
- The Command Object requires an instance of a C# Connection Object for executing the SQL statements.
- Some important built in methods uses in the Command Object to execute the SQL statements

ExecuteNonQuery

- used for executing statements that do not return result sets (i.e. statements like insert data, update data etc.)
- returns the no. of rows affected by the command

```
String connetionString = "Data Source=ServerName; Initial Catalog= SISDB; User ID=UserName;
Password=Password“;
 conn = new SqlConnection(connetionString);
SqlConnection conn = new SqlConnection (connetionString );
try {
   conn.Open ();
   SqlCommand cmd = new SqlCommand ();
   cmd.CommandText = "delete from Course where Code = ‘CoSc1002'";
   cmd.Connection = conn;
   cmd.ExecuteNonQuery (); // Execute the command
}
catch (SqlException ex) {
   // TODO: Handle the exception
}
finally {
   conn.Close ();
}
```

ExecuteReader

- to perform database queries and obtain the results
- Returns a DataReader object
- SqlCommand. ExecuteReader();
- SqlCommand. ExecuteReader(behavior);
  - Behavior – CloseConnection, Default, SingleRow
- DataReader has methods and properties that you can call to iterate over the result set, such as GetValue, GetName, Read, GetDecimal, …

```
SqlConnection conn = new SqlConnection(connetionString );
try {
   conn.Open ();
   SqlCommand cmd = new SqlCommand ("select * from Course", conn);
   SqlDataReader reader = cmd.ExecuteReader();
   while (reader.Read ())
      listBox1.Items.Add(reader["Title"].ToString());
}
catch (SqlException ex) {
   MessageBox.Show(ex.Message);
}
finally {
   conn.Close ();
}
```

Closing a DataReader

- DataReader.Close does not close the connection encapsulated in the command object that created the DataReader.
- reader.Close (); // Does NOT close the connection!
- DataReader.Close
  - ✓ closes the DataReader, which frees the connection associated with the DataReader so that it can be used again.

ExecuteScalar

- to retrieve a single value from Database after the execution of the SQL Statement
- returns the first row of the first column in the result set
- used to execute SQL functions such as COUNT, AVG, MIN, MAX, and SUM which returns a single row single, column result set

```
SqlConnection conn = new SqlConnection(connetionString );
try {
   conn.Open ();
   SqlCommand cmd = new SqlCommand
      ("select max (credit) from Course", conn);
   int  amount = (int) cmd.ExecuteScalar();//cast object to int
   txtCredit.Text = amount.ToString();
}
catch (SqlException ex) {
   MessageBox.Show(ex.Message);
}
finally {
   conn.Close ();
}
```

ExecuteScalar …

- Another common use for ExecuteScalar is to retrieve BLOBs (binary large objects) from databases.
- The following example retrieves an image from the “photo” field of the SISDB database’s “Student” table and encapsulates it in a bitmap:

```
MemoryStream stream = new MemoryStream ();
SqlConnection conn = new SqlConnection(connectionString);
try {
   conn.Open ();
   SqlCommand cmd = new SqlCommand("select photo from student where studid='DBU1001'", conn);
   byte[] blob = (byte[]) cmd.ExecuteScalar ();
   stream.Write (blob, 0, blob.Length);
   Bitmap bitmap = new Bitmap (stream);
   // TODO: Use the bitmap
                pictureBox1.Image = bitmap;
   bitmap.Dispose ();
}
catch (SqlException ex) {
   // TODO: Handle the exception
}
finally {
   stream.Close ();
   conn.Close ();
}
```

ExecuteScalar …

- You might be interested in knowing how to write BLOBs to databases.
- The secret is to call **ExecuteNonQuery** on a command object that wraps an INSERT command containing an input parameter whose type is byte[].
- To demonstrate, the following example inserts a record into the SISDB database's "Student" table and includes a BLOB in the record's "photo" field:

```
FileStream stream = new FileStream ("photo.jpg", FileMode.Open);
byte[] blob = new byte[stream.Length];
stream.Read (blob, 0, (int) stream.Length);
stream.Close ();
SqlConnection conn = new SqlConnection(connectionString);
try {
   conn.Open ();
   SqlCommand cmd = new SqlCommand("insert into Student (StudId, photo) values ('DBU1001', @photo)",
      conn);
   cmd.Parameters.Add ("@photo", blob);
   cmd.ExecuteNonQuery ();
}
catch (SqlException ex) {
   // TODO: Handle the exception
}
finally {
   conn.Close ();
}
```

Transacted Commands

- Transacted database operations are an important element of many data-driven applications.
- A transaction is simply two or more otherwise independent units of work grouped together into one logical unit.
- A classic example is an application that transfers funds from one bank account to another by debiting money from one account (that is, one database record) and crediting it to another.
- The updates should be performed within the scope of a transaction.
- Why? So that if one of the operations fails, the other will fail (or be rolled back) too.

Transacted Commands

```
SqlTransaction trans = null;
SqlConnection conn = new SqlConnection(connectio
nString);
try {
   conn.Open ();
   // Start a local transaction
   trans = conn.BeginTransaction (IsolationLevel.Seri
alizable);

   // Create and initialize a SqlCommand object
   SqlCommand cmd = new SqlCommand ();
   cmd.Connection = conn;
   cmd.Transaction = trans;

   // Debit $1,000 from account 1111
   cmd.CommandText = "update accounts set balance = " +
      "balance - 1000 where account_id = '1111'";
   cmd.ExecuteNonQuery ();

   // Credit $1,000 to account 2222
   cmd.CommandText = "update accounts set balance = " +
      "balance + 1000 where account_id = '2222'";
   cmd.ExecuteNonQuery ();

   // Commit the transaction (commit changes)
   trans.Commit ();
}
catch (SqlException) {
   // Abort the transaction (roll back changes)
   if (trans != null)
      trans.Rollback ();
}
finally {
   conn.Close ();
}
```

Parameterized Commands

- Parameter is a variable in SQL Statement
- It's not unusual for an application to execute the same command on a database repeatedly,
  - ✓ varying only the value or values used in the command.
- SQL programmers often use parameterized commands
  - ✓ frequently referred to as "parameterized queries"
  - ✓ to code redundant commands, especially commands whose input values come from user input.
- More secure, they prevent SQL injection attacks
- Use named variable to identify a parameter
  - ✓ The name of the variable begins with at sign @ symbol

SqlParameter class

- new SqlParameter ()
- new SqlParameter (name, value)
- new SqlParameter (name, type)
- new SqlParameter (name, type, size)

```
SqlParameter  paramCode = new SqlParameter ()
paramCode.ParameterName = "@CourseCode"
parameCode.value = courseCode
```

- Another way

```
SqlParameter  paramCode = new SqlParameter ("@CourseCode", courseCode)
```

Common members of parameters collection

- Indexer
  - [parametername]
  - [index]
- Methods
  - Add(parameter)
  - Add(name, type)
  - Add(name, type, size)
  - AddWithVale(name, value)
    - cmd.Parameters.Add("@CourseCode");
    - cmd.Parameters.AddWithValue("@CourseCode", courseCode);
    - cmd.Parameters.["@CourseCode"].value = courseCode;

Parameterized Queries

```
try {
   conn.Open ();

   // Create and initialize a SqlCommand object
   SqlCommand cmd = new SqlCommand
      ("update accounts set balance = balance + @amount "
+
      "where account_id = @id", conn);
   cmd.Parameters.Add ("@amount", SqlDbType.Money);
   cmd.Parameters.Add ("@id", SqlDbType.Char);
  // Debit $1,000 from account 1111
   cmd.Parameters["@amount"].Value = -1000;
   cmd.Parameters["@id"].Value = "1111";
   cmd.ExecuteNonQuery ();

   // Debit $1,000 from account 1111
   cmd.Parameters["@amount"].Value = -1000;
   cmd.Parameters["@id"].Value = "1111";
   cmd.ExecuteNonQuery ();

   // Credit $1,000 to account 2222
   cmd.Parameters["@amount"].Value = 1000;
   cmd.Parameters["@id"].Value = "2222";
   cmd.ExecuteNonQuery ();
}
catch (SqlException ex) {
   // TODO: Handle the exception
}
finally {
   conn.Close ();
}
```

Lab Exercise

- **Using appropriate connection, command, and data reader objects,**
- Create a database “SISDB” contains
- To create a database
  - ✓ use the Management Studio if you have installed in your computer or
  - ✓ Use Server Explorer tab or use **View-> Server Explorer**
    - > Right Click on **Data Connection** -> click “**Create New SQL Server database”**
    - > Enter Server name and Database name ->Click **OK**

Working with Data Source and Datasets

- Data source
  - ✓ Specifies the source of data for an application
  - ✓ shows all the tables and columns in the dataset that are available to your application
  - ✓ Easiest way to generate Windows forms that work with the database
  - ✓ Especially for small application or prototyping large application

Data Source and Datasets …

- **Create a data source**
- Open the Data Source windows
  - ✓ **Click** on the **Data Source** tab or **Data ->Data Source** command
  - ✓ Click on the Add New Data Source link, if your adding data source for the first time to your project, or
  - ✓ use **Data->Add New Data Source** command
- Choose the data source type, Click **Next**
- Choose the database model, (Dataset or Entity Data Model), select **dataset**, click Next
- Choose the data connection, Click New Connection -> select Data source and click Continue
- Add Connection: Select/Enter database server, Select/Attach Database, click Ok
- Choose your data connection, have a look at your connection string, click Next
- Save connection string to application configuration file, Click Next
- Choose Database object you need to add to your project
- Click Finish, Enjoy with drag and drop of your database object

Handling errors

- Data errors
  - SqlException
- ADO.Net error
  - DBConcurrencyException
  - DataException
  - ConstraintException
  - NoNullAllowedException
- DataGridView data error
  - DataError Event handler

Lab Exercise

- **Working on Data Source**
- Add a Student information display form to your project.
- Add the customer database object from the data source to the form in order to display customer information in a GridView.
- Run and test your project.
- Have a look how the data source works.

DataSet and DataAdapters

- ADO.NET's SqlDataReader class provides stream-based access to the results of database queries
- Streaming access is fast and efficient, but it's also read-only and forward-only.
- ADO.NET supports set-based data access as well as stream-based data access.
- Set-based accesses capture an entire query in memory and support backward and forward traversal through the result set.
  - ✓ edit the data obtained through database queries, propagate the changes back to the data source, and much, much more.
- Set-based data accesses revolve around two classes:
  - ✓ DataSet, which is the equivalent of an in-memory database and is defined in the System.Data namespace, and
  - ✓ DataAdapter, which serves as a bridge between DataSets and physical data sources.

DataSet

- DataSet provides a disconnected representation of result sets from the Data Source, and it is completely independent from the Data Source.
- DataSet provides much greater flexibility when dealing with related Result Sets.
- Dataset is structured much like a relational database.
- It can contain one or more tables, and each table can contain one or more columns and rows.

DataSet …

DATASET
DataTableCollections
DataTable
DataRowCollection
DataDColoumnCollection
ConstraintCollection
DataRelationCollection

DataSet …

- You can also work with the data in a database without using a data adapter.
- Use command and connection objects to access database.
- Instead of using a data adapter to execute the commands, you can execute the commands directly.
- When you do that, you also have to provide code to handle the result of the command.
- If you issue a command that contains an
  - ✓ Insert, Update, or Delete statement
    - the result is an integer that indicates the number of rows that were affected by the operation.
  - ✓ You can use that operation to determine if the operation was successful.
- If you execute a command that contains a Select statement,
  - ✓ the result is a result set that contains the rows you requested.
  - ✓ To read through the rows in the result set, use a data reader object

DataSet Class

- Think of a DataSet as an in-memory database.
- The actual data is stored in DataTable objects, which are analogous to tables in a database.
- The DataSet.Tables property exposes a list of the DataTables in a DataSet.
- Records in a DataTable are represented by DataRow objects, and fields are represented by instances of DataColumn.
- DataTable properties named Rows and Columns expose the collections of DataRows and DataColumns that make up the table.

```
DataSet ds = new DataSet ();
// TODO: Initialize the DataSet
MyDataGrid.DataSource = ds;
MyDataGrid.DataBind ();
```

DataSets vs. DataReaders

- DataReader
  - simply to query a database and read through the records one at a time until you find the one you're looking for,
  - retrieve only the data that you actually use, and they don't consume memory by storing every record that you read.
- DataSet
  - to use all the query results (perhaps because you're displaying them in a table), you need the ability to iterate backward and forward through a result set
  - to cache the result set in memory
  - DataSet ds = new DataSet ();
  - // TODO: Initialize the DataSet
  - MyDataGrid.DataSource = ds;
  - MyDataGrid.DataBind ();
- the same code can be implemented more efficiently with a DataReader:
  - SqlDataReader reader = cmd.ExecuteReader ();
  - MyDataGrid.DataSource = reader;
  - MyDataGrid.DataBind ();

DataSet …

- purpose is to perform database queries and create DataTables containing the query results.
- It's also capable of writing changes made to the DataTables back to the database.
- **Initializing a DataSet from a Database: DataAdapter.Fill**

DataAdapter Classes

- ✓ SqlDataAdapter adapter = new SqlDataAdapter ("select * from course", conn);
- ✓ DataSet ds = new DataSet ();
- ✓ adapter.Fill (ds, “Course");

- Fill opens a connection to the database using the connection string passed to SqlDataAdapter’s constructor.
- It performs a query on the Pubs database using the query string passed to SqlDataAdapter’s constructor.
- It creates a DataTable named “Course” in the DataSet.
- It initializes the DataTable with a schema that matches that of the “Course” table in the database.
- It retrieves all the records produced by the query and writes them to the DataTable.
- It closes the connection to the database.

DataTable

- The following code iterates through the Tables collection of a DataSet named ds and outputs the name of each table that it encounters:
  - ✓ foreach (DataTable table in ds.Tables)
  - ✓ ltxtBox1.Items.Add (table.TableName.ToStriing());
- Individual DataTables in a DataSet can be referenced by name or 0-based index
- The next example retrieves the first DataTable from a DataSet and writes the value of the first column in every row to a listBox:
  - ✓ DataTable table = ds.Tables[0];
  - ✓ foreach (DataRow row in table.Rows)
  - ✓ ltxtBox1.Items.Add(row[0].ToString());
- Columns, too, can be referenced by name as well as numeric index.
- Thus, if the name assigned to the first column in the DataTable is “CourseCode” the preceding example could be rewritten this way:
  - ✓ DataTable table = ds.Tables[0];
  - ✓ foreach (DataRow row in table.Rows)
  - ✓ ltxtBox1.Items.Add(row["CourseCode"].ToString());
- Enumerating a DataTable’s columns is equally simple:
  - ✓ DataTable table = ds.Tables[0];
  - ✓ foreach (DataColumn col in table.Columns)
  - ✓ MessageBox.Show("Name=“ + col.ColumnName + ”Type=“ + col.DataType);

Inserting Records into a DataTable

- One way to insert records into a database is to call ExecuteNonQuery on a Command object wrapping an INSERT command
- You can also insert records into databases using DataSets.
- The general approach is to
  - ✓ perform a query with DataAdapter.Fill,
  - ✓ add records to the resulting DataTable, and
  - ✓ write the changes to the database

Inserting Record …

- The following example adds a record to a DataTable created from the SISDB database's "Course" table:
  - SqlDataAdapter adapter = new SqlDataAdapter ("select * from Course", connString)
  - DataSet ds = new DataSet ();
  - adapter.Fill (ds, "Course");
  -
  - // Create a new DataRow
  - DataTable table = ds.Tables["Course"];
  - DataRow row = table.NewRow ();
  -
  - // Initialize the DataRow
  - row["Code"] = "InSy1001";
  - row["Title"] = "C# Programming";
  - row["Credit"] = 5;
  - // Add the DataRow to the DataTable
  - table.Rows.Add (row);

Selecting Records in a DataTable

- A smarter way to find the records is to use the DataTable.Select method.
- The following statement returns an array containing a single DataRow
  - ✓ the one added to the table with Add:
    - DataRow[] rows = table.Select (“Code = ‘InSy1001'");
- This statement selects (returns) all DataRows whose “Credit” field contains a value less than 5:
    - DataRow[] rows = table.Select (“Credit < 5");
- If you want to know how many rows Select returned, read the array’s Length property.

Updating Records in a DataTable

- The following example selects all the records in the SISDB database's "Course" table
  SqlDataAdapter adapter = new SqlDataAdapter ("select * from Course", ConnString);
  - ✓ DataSet ds = new DataSet ();
  - ✓ adapter.Fill (ds, "COurse");
  - ✓ DataRow[] rows = table.Select ("credit < 4");
  - ✓ foreach (DataRow row in rows)
  - ✓ row["Credit"] = (int) row["Credit"] + 2;

Deleting Records from a DataTable

- Deleting records from a DataTable is a simple matter of calling Delete on each DataRow that you want to remove.
- The next example deletes all rows whose credit equal to null:
  - SqlDataAdapter adapter = new SqlDataAdapter ("select * from course ", connString);
  - DataSet ds = new DataSet ();
  - adapter.Fill (ds, “Course");
  - DataRow[] rows = table.Select ("isnull (Credit, 0) = 0");
  - foreach (DataRow row in rows)
  - row.Delete ();

Propagating Changes Back to a Database: DataAdapter.Update

- Inserts, updates, and deletes performed on a DataTable do not automatically propagate back to the database.
- If you want changes written back to the database, you have to take matters into your own hands.
- Fortunately, the DataAdapter.Fill method makes your job incredibly simple.
- Here's a code sample demonstrating how to make changes to a database using a DataSet and a DataAdapter:

```
> SqlDataAdapter adapter =  new SqlDataAdapter ("select * from Course", connString);
>
> SqlCommandBuilder builder = new SqlCommandBuilder (adapter);
> DataSet ds = new DataSet ();
> adapter.Fill (ds, "Course");
> // Insert a record
> DataTable table = ds.Tables["Course"];
> DataRow row = table.NewRow ();
> row["Code"] = "InSy1001";
> row["Title"] = "C# Programming";
> row["Credit"] = 7;
> table.Rows.Add (row);
> // Update the database
> adapter.Update (table);
```

CommandBuilder Classes

- How does Update physically update the database?
- The short answer is that it executes SQL INSERT commands for rows added to a DataTable,
  - ✓ UPDATE commands for rows that were modified, and
  - ✓ DELETE commands for rows that were deleted.
- But where do the INSERT, UPDATE, and DELETE commands come from? Are they manufactured out of thin air?
- Close. They're manufactured by a SqlCommandBuilder object.
- Note the following statement from the previous code sample:

Command Builder …

- SqlCommandBuilder builder = new SqlCommandBuilder(adapter);
- If you omit this statement, Update throws an exception.
- A DataAdapter has four very important properties that control how it communicates with a database:
  - ✓ SelectCommand
    - > which encapsulates the command the DataAdapter uses to perform queries
  - ✓ InsertCommand
    - > which encapsulates the command the DataAdapter uses to insert rows
  - ✓ UpdateCommand
    - > which encapsulates the command the DataAdapter uses to update rows
  - ✓ DeleteCommand
    - > which encapsulates the command the DataAdapter uses to delete rows

Lab Exercise

- **Using DataSet and DataAdapter**
- Add a Course information display form to your project.
- Add GridView control on the Form.
- Use DataSet and DataAdapter code to bind data source for the GridView and display book information.
- Write a code for deleting and updating book information using Dataset and DataAdapter.
- Run and test your project.

Making Reports in event driven

Generate decision supporting information

Report

- Report is a
  - ✓ Meaningful,
  - ✓ well summarized
  - ✓ nicely formatted way of presenting data from the database.
- Usually the routine activities are automated and data summarized into a **decision-supporting format** "Reports".
- Reports act as wonders when it converts the usual messy data into attractive charts, and graphical representations.

Reporting tools

- Commercial tools
  - Crystal Reports
  - fyiReporting
  - VS Reporting feature
  - …
- Open source /Free
  - ReportingMax
  - Seal Report
  - JasperReports
  - iReport and
  - JFreeChart

Using Reporting Features of Visual Studio

- Based on Microsoft's SQL Server Reporting Service
  - ✓ Provide tools to
    - > Create
    - > Manage
    - > Display
    - > Deliver reports from SQL Server
- Visual Studio offers
  - ✓ Report Wizard
  - ✓ Report Designer to design report
  - ✓ **ReportViewer** control to display and print local or remote report

Types of Reports

- Table
  - ✓ Display data in tubular form
- Matrix
  - ✓ Crossbar or pivot table report
  - ✓ Both the rows and columns can be expanded
- Chart
  - ✓ display data in a variety of graphical representation
- Gauge
  - ✓ Represents data graphically with an indicator to highlight a value

Components of a report page

- Many of the available reporting tools have the following layout for a report page.
- **Title**
  - ✓ appears only once at the very beginning of the report.
  - ✓ Title of the report is written in this part.
- **Page Header**
  - ✓ appears at the top of each page.
  - ✓ This part may contain date and time information and/or organization name.
- **Column Header**
  - ✓ lists names of those specific fields which you want to display.
- **Details**
  - ✓ the part where those fields of the entries are shown.
- **Column Footer**
  - ✓ may display summation of any of the fields.
- **Page Footer**
  - ✓ appears at the bottom of each page.
  - ✓ This part may contain page count information
- **Summary**
  - ✓ is the part where information inferred from the data in the “detail” part is displayed.

Start a local Report

- Project -> Add New Item, select Reporting -> Select Report
- Project -> Add New Item, select Reporting -> Select Report Wizard
- Drag ReportViewer control -> Select Design New Report from the smart tag
- New Project -> Select Reporting -> Reports Application

Report Wizard

- Project -> Add New Item, select Reporting -> Select Report Wizard
- Set/Select Dataset properties
- Select Rows and columns
- Arrange fields
- Choose layout and style

ReportViewer

- Drag and Drop ReportViewer control onto a Form
- Assign report to the control
  - ✓ Click the smart tag
  - ✓ Choose a report from report list
  - ✓ If report not available
    - > Select Design New Report option to design new report

Report Designer

- Components
  - ✓ Report Data pane
  - ✓ Grouping Pane
  - ✓ Report Designer
- Add/Delete a column or row
- Add/Delete a field value
- Insert grouping and grand total
- Create expressions

Improve the look of a Report

- Apply data formatting
  - ✓ Date
  - ✓ Currency
  - ✓ Number
- Add headers and Footer
- Add image to a report

Build other type of Reports

- Create Report from scratch
- Create a matrix report
- Create a list report
- Create a chart report

Crystal Report

- Crystal Report is a Reporting Application that can generate reports from various Data Sources like Databases, XML files etc..
- The Visual Studio.NET Integrated Development Environment comes with Crystal Reports tools.
- The Crystal Reports
  - ✓ makes it easy to create simple reports, and
  - ✓ also has comprehensive tools that you need to produce complex or specialized reports in csharp and other programming languages.

Creating Crystal Report

- Start your first Crystal Reports
- Create a new simple Crystal Reports for Book table from the database SISDB you created earlier.
- The report shows the whole table data in the Crystal Reports.
- From **Project -> Add New Item** or Right Click on SISDB project **Add->Add new Item** Command to bring Add New Item Dialog window
- Select **Reporting** tab and Select **Crystal Reports,** available if you installed Crystal report on your computer

Creating Crystal Report ….

- Select Report type from Crystal Reports gallery, accept default and click OK
- Select the appropriate connection to your database, Click Next
- Select all fields from Course table to the right side list, Click Next
- Click Finish Button.
- Then you can see the Crystal Reports designer window.
- You can arrange the design according your requirements.
- Add a Form to your project and drag a CrystalReportViewer control to your form
- Select Form's source code view and put the code on top

Creating Crystal Report ….

- Using CrystalDecisions.CrystalReports.Engine;
- Put the following source code in the button click event
  - ✓ ReportDocument cryRpt = new ReportDocument();
  - ✓ cryRpt.Load(@"D:\Projects\CSharp\StudentInformationSystem\StudentInformationSystem\crpCourseList.rpt");
  - ✓ crystalReportViewer1.ReportSource = cryRpt;
  - ✓ crystalReportViewer1.Refresh();
- Run your project, look at your report.

Passing Parameters

- Declare/Define a parameter in report designer
- Add parameters to the report
- Pass parameter from C# app

Passing parameter sample code…

```
private void btPrint_Click(object sender, EventArgs e)
        {
            if (cmbCurriculum.SelectedIndex > -1)
            {
                Faculty fclty = new Faculty();
                Department dept = new Department();
                FieldOfStudy field = new FieldOfStudy();
                StudyProgram study = new StudyProgram();
                AdmissionProgram admission = new AdmissionProgram();
                Curriculum curr = new Curriculum();
                try
                {
```

Sample code ..

```
fclty = (Faculty)cmbFaculty.SelectedItem;
                dept = (Department)cmbDepartment.SelectedItem;
                field = (FieldOfStudy)cmbFieldOfStudy.SelectedItem;
                study = (StudyProgram)cmbStudyProgram.SelectedItem;
                admission = (AdmissionProgram)cmbAdmProgram.SelectedItem;
                curr = (Curriculum)cmbCurriculum.SelectedItem;
                string[] currName = curr.CurrNomenclature.Split('(');
this.viewCourseListAsIsTableAdapter.FillByCurrCode(this.sISBDDataSet.ViewCourseListAsIs, curr.CurrCode);
                ReportParameter[] paramCourse = new ReportParameter[7];
                paramCourse[0] = new ReportParameter("ParamFacultyName", fclty.FacultyName);
                paramCourse[1] = new ReportParameter("ParamDeptName", dept.DeptName);
                paramCourse[2] = new ReportParameter("ParamFieldOfStudy", field.FieldName);
```

Sample code ..

```
paramCourse[3] = new ReportParameter("ParamStudyProgram", study.StudydName);
                paramCourse[4] = new ReportParameter("ParamAdmissionProgram", admission.AdmissionName);
                paramCourse[5] = new ReportParameter("ParamCurriculum", currName[0]);
                paramCourse[6] = new ReportParameter("ParamDuration", currName[1].TrimEnd(')'));
                reportViewerCourseList.LocalReport.SetParameters(paramCourse);
                this.reportViewerCourseList.RefreshReport();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
```

Lab Exercise

- **Work with Reporting feature of visual studio**
  - ✓ Create a report that displays data
- **Working on Crystal report**
  - ✓ Download and install SAP crystal crystal report for visual studio
  - ✓ Create a crystal report that displays data

For more information

- Anne Boehm, Ged Mead. ADO.NET 4 Database programming with C# 2010
- Joel Murach, Anne Boehm. Murach C# 2012, Mike Murach & Associates Inc USA, 2013
- Deitel, C#-How to Program. USA, 2010
- Svetlin Nakov *et al*. Fundamentals of Computer Programming With C#. Sofia, 2013
- www.csharp.net-informations.com

QUESTION!