INFO2001
Topic 3:Solution Design
Lecture 3:Searching

Thomas Grace

Recap from Lecture 2

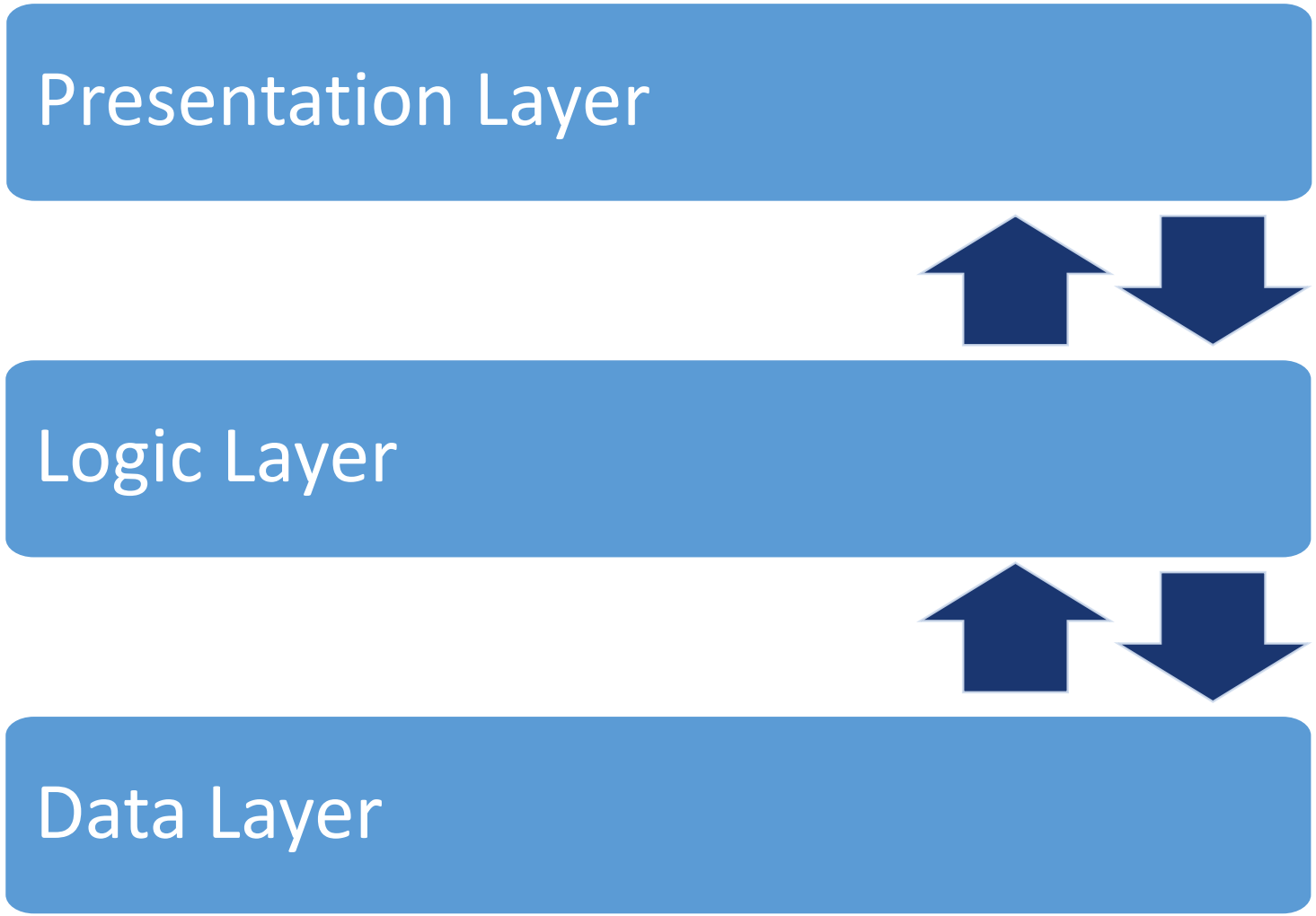
- Explanation of:
  - Table adapters, datasets and binding sources
  - Input validation
- Example
  - Data grids
  - Using the table adapter
    - Creating a query
    - Creating a parameter based query
  - Adding & Editing data through Visual Studio
  - Passing values between forms

Lecture 3 - Agenda

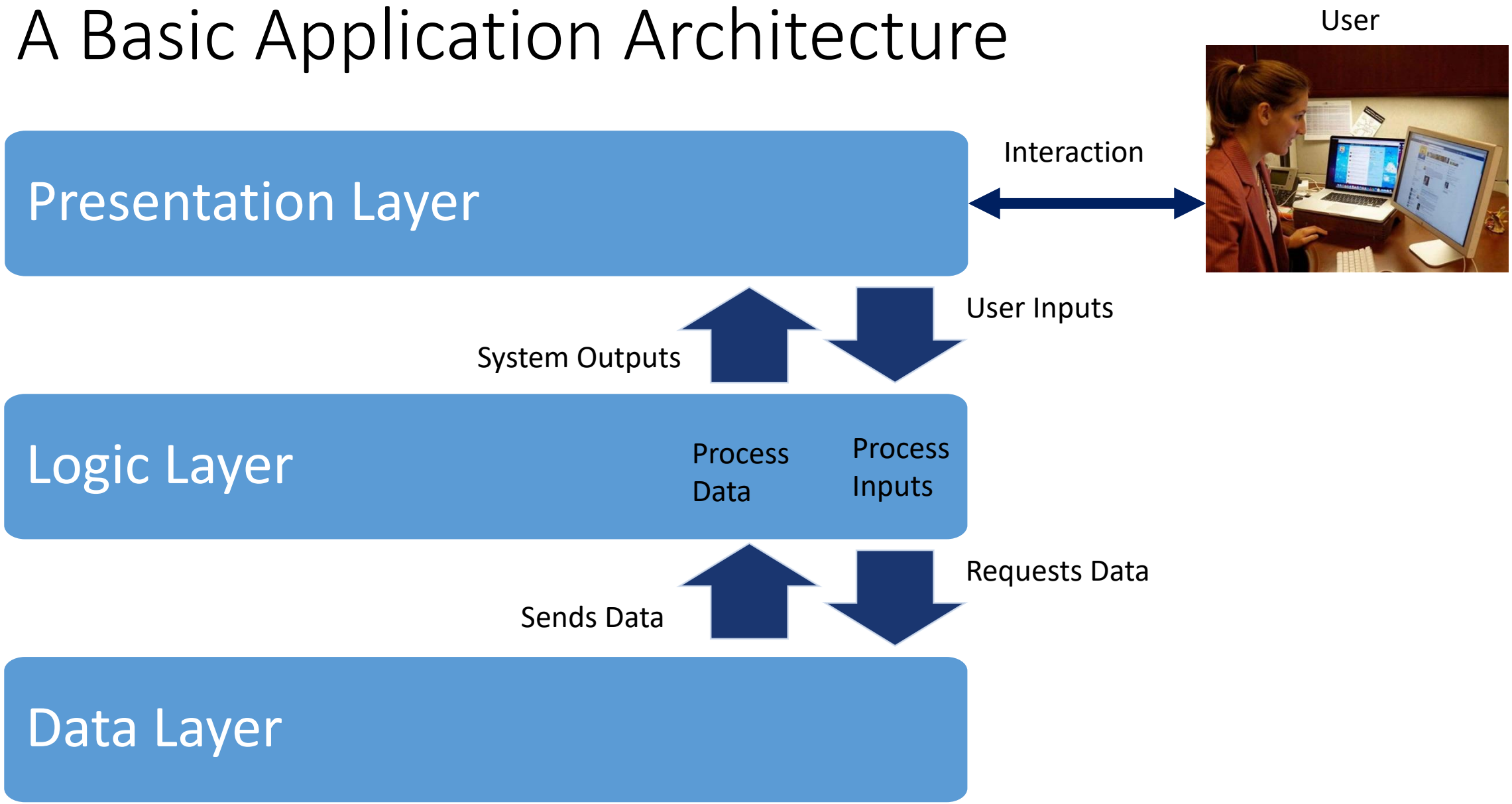
- Explanations:
  - Basic Application Architecture
- Examples
  - Access Control
  - Searching using the table adapter
  - Combo boxes
- Exercise:
  - Do Exercise 1 & 2

A Basic Application Architecture

User

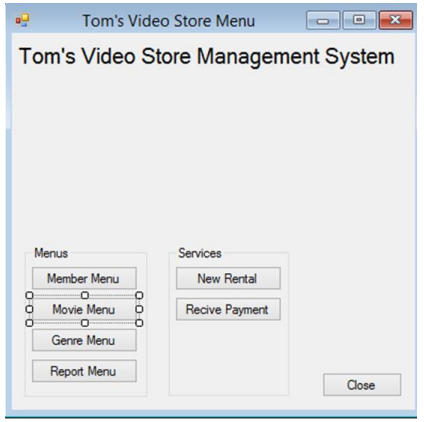


A Basic Application Architecture



Presentation Layer

- Involves User Interaction & Reporting of Information
  - Accepts **inputs** into the solution:
    - User interactions with the system through form controls (e.g. buttons, menus, textboxes, combo boxes, touchscreen)
  - Displays any **output** after processing performed by the logic layer
    - Displays output to the user through reports, message boxes, datagrid, new forms
- Interaction in our system occurs through the interaction with the form
  - Face of the system



Logic Layer

- Involves Processing of user input and data
  - Input from the presentation layer (the user) generally initiatives a set of logical instructions
    - Your code – C#
    - Requires a framework to run – Microsoft uses a .net
  - This might require data from the data layer
    - Requests data through SQL and data connection
  - Performs any processing needed on the data
    - Insert, edit, delete

```
private void button1_Click(object sender, EventArgs e)
{
    Form GenreMenu = new frmGenreMenu();
    GenreMenu.Owner = this;
    this.Hide();
    GenreMenu.Show();



}

private void button2_Click(object sender, EventArgs e)
{
    this.Close();
}

private void button3_Click(object sender, EventArgs e)
{
    Form MovieMenu = new frmMovieMenu();
    MovieMenu.Owner = this;
    MovieMenu.Show();
    this.Hide();
}

private void btnReport_Click(object sender, EventArgs e)
{
    Form Report = new frmReportMenu();
    Report.Owner = this;
    Report.Show();
    this.Hide();
}

private void frmMenu_Load(object sender, EventArgs e)
{
    switch (PrsentUser)
```

Data Layer

- Any Data the systems needs will be stored here
  - Relational Databases
    - SQL Server
  - Media
    - Videos, images, audio
  - Anything that does not fit into a standard Database
  -

Desktop Application

Local Machine

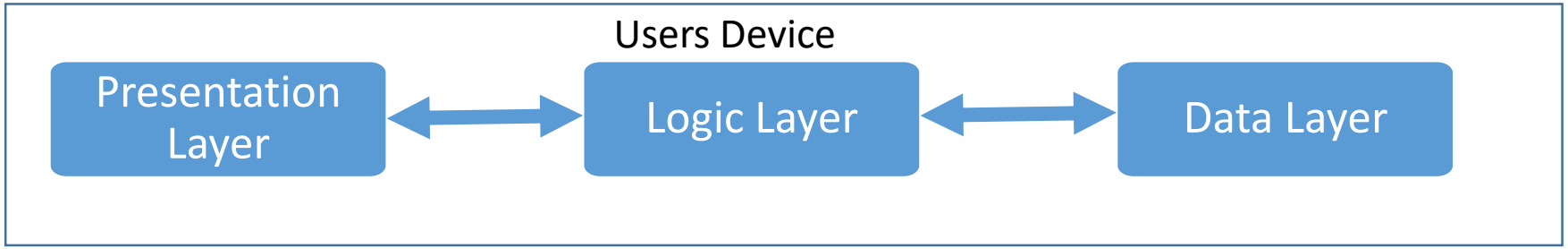
Presentation Layer

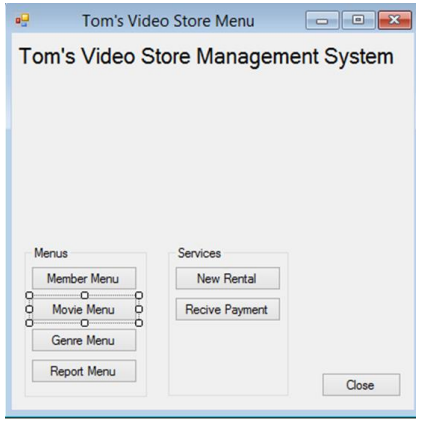
Logic Layer

Data Layer

Desktop Application Architecture

Desktop application – all layers reside on the same physical hard drive, or disk storage (e.g. “Tom’s Video Store”).






```
private void button1_Click(object sender, EventArgs e)
{
    Form GenreMenu = new frmGenreMenu();
    GenreMenu.Owner = this;
    this.Hide();
    GenreMenu.Show();



}

private void button2_Click(object sender, EventArgs e)
{
    this.Close();
}

private void button3_Click(object sender, EventArgs e)
{
    Form MovieMenu = new frmMovieMenu();
    MovieMenu.Owner = this;
    MovieMenu.Show();
    this.Hide();
}

private void btnReport_Click(object sender, EventArgs e)
{
    Form Report = new frmReportMenu();
    Report.Owner = this;
    Report.Show();
    this.Hide();
}

private void frmMenu_Load(object sender, EventArgs e)
{
    switch (PrsentUser)
```

Desktop Application

Characteristics & Limitations

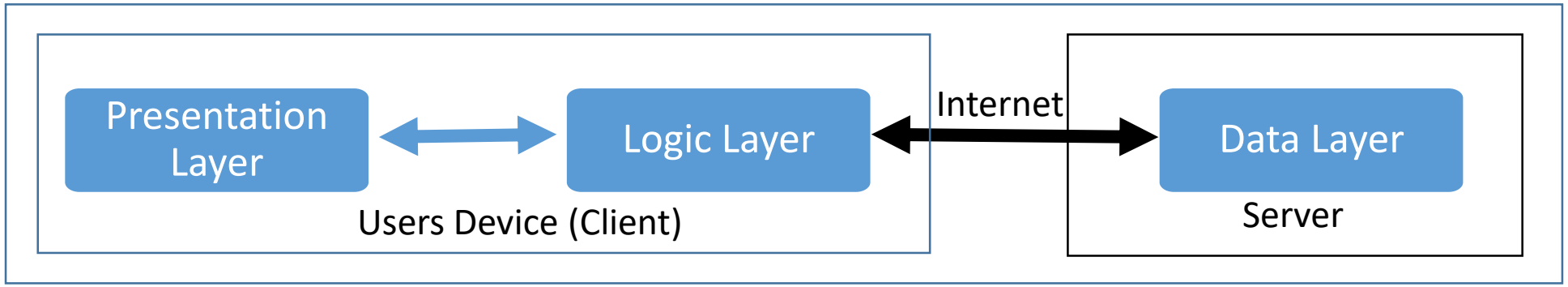
Characteristics:

- Easier to develop than web
  - Interface can be manipulated through drag and drop
  - Fewer languages needed to develop (C# & SQL)
- Interacts directly with hardware and software
  - Games and CAD
  - Scanners or “special devices”
- Faster performance
  - Processing occurs on the machine
- No internet connection is required
- Reliable and secure
- Generally requires no subscription fee
- Applications are comprehensive with features

Limitations:

- Physical location
  - Need to have physical access to the PC to use the system
- Physical hardware and software
- Upgrades can be costly
  - Upgrades of software and hardware need to be done individually on each machine
- Might need a license

Hybrid Application

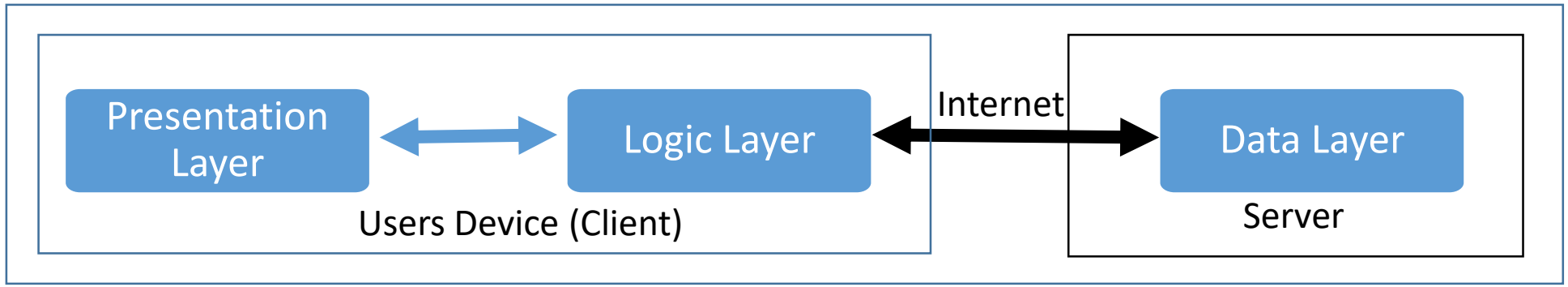


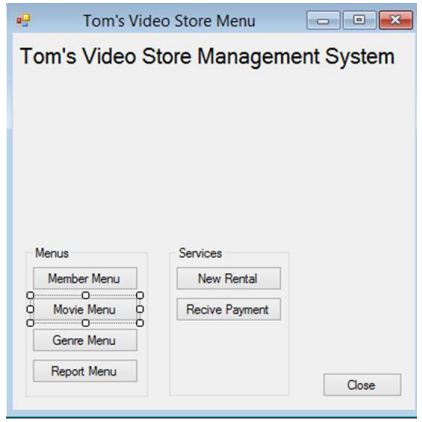
- Users Device has a client application, which consists of the presentation and logic layers (mobile apps are generally like this)
  - The “Presentation” and “Logic” layer are on the local machine (client) BUT the “Data” layer is located on a server
- Examples Include:
  - Online games
  - Desktop software connecting to an online database (Your Project)
  - Most Mobile Apps

Hybrid application

- Posses similar characteristics and limitations to a desktop application
- Therefore: One can take advantage of the benefits of a desktop application while having a database that is stored on the internet
- Additional Characteristics
  - Separate database prevents lose of data due to damage to the machine
  - More than one person can use the same application as they will update to the same database
- Aditional Limitations
  - An active internet connection is required for the application to work
  - Less reliable and secure than a desktop application, due communication occurring over the internet
  - Requires subscription fees to host the database on a server

Your Project



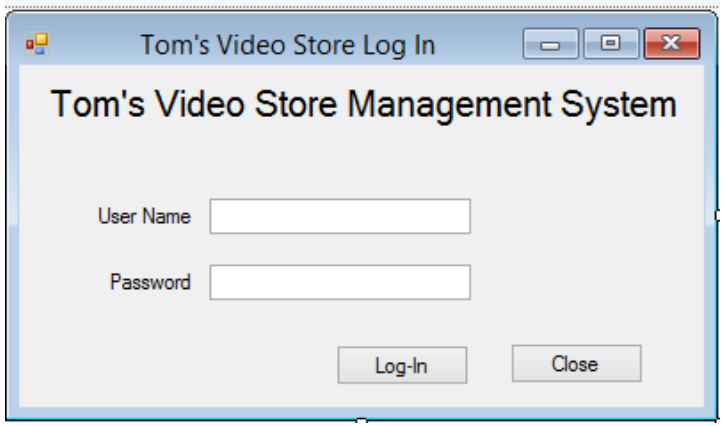



```
private void button1_Click(object sender, EventArgs e)
{
    Form GenreMenu = new frmGenreMenu();
    GenreMenu.Owner = this;
    this.Hide();
    GenreMenu.Show();



}

private void button2_Click(object sender, EventArgs e)
{
    this.Close();
}

private void button3_Click(object sender, EventArgs e)
{
    Form MovieMenu = new frmMovieMenu();
    MovieMenu.Owner = this;
    MovieMenu.Show();
    this.Hide();
}

private void btnReport_Click(object sender, EventArgs e)
{
    Form Report = new frmReportMenu();
    Report.Owner = this;
    Report.Show();
    this.Hide();
}

private void frmMenu_Load(object sender, EventArgs e)
{
    switch (PrsentUser)
```

Coding Examples

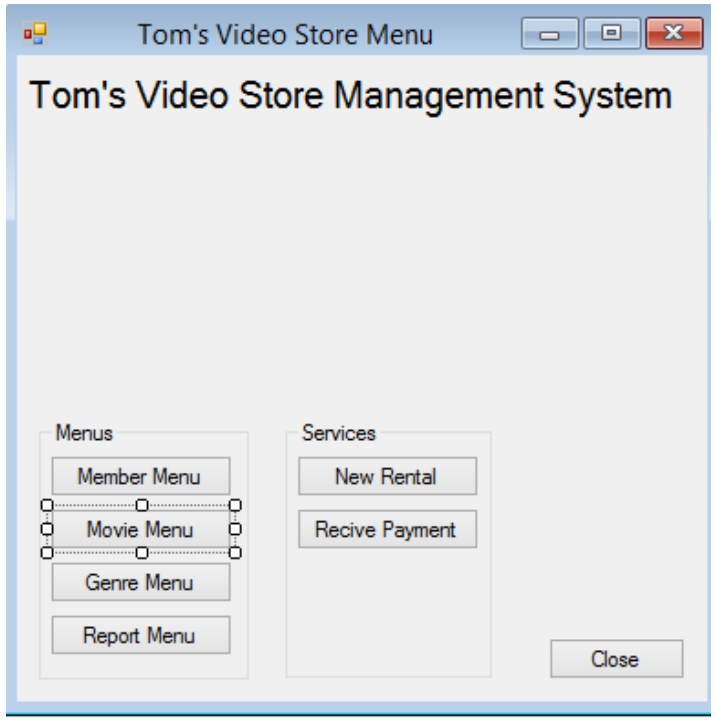
- Examples
  - Access Control
  - Searching using the table adapter
  - Combo boxes

Access Control

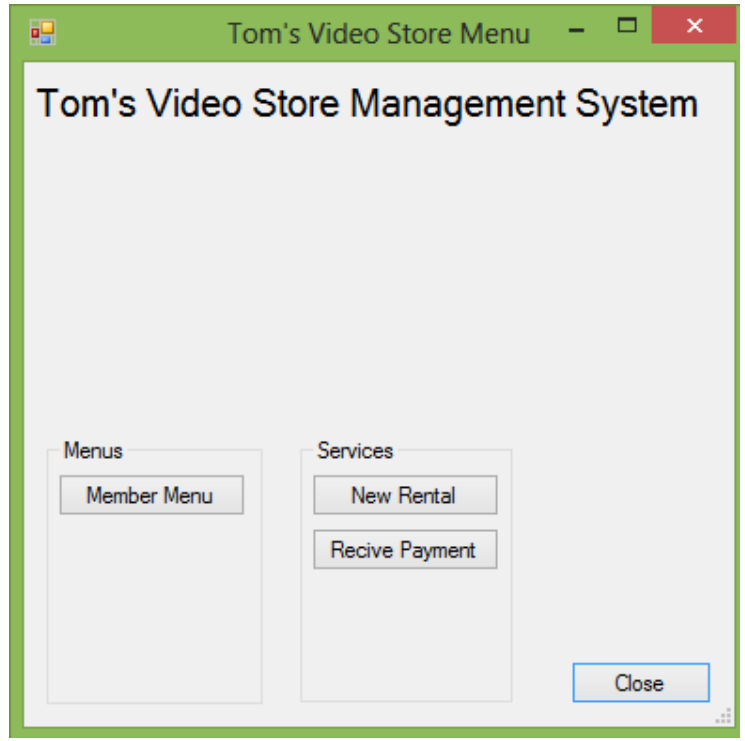


- The Administrator has more functionality than the average employee
- This is achieved through hard coding
- Normally we would use an access control group with a database table
  - We would have a table that stores the different users

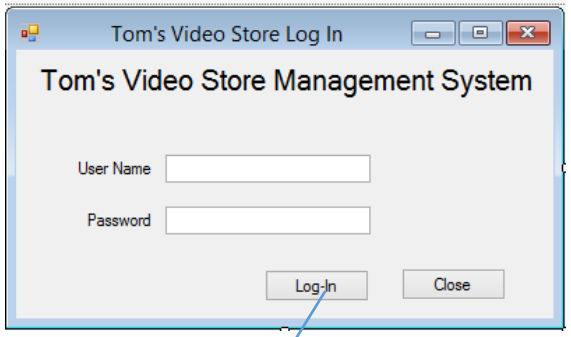
Admin View



Employee View



How to make a log in




```
private void btnLogIn_Click(object sender, EventArgs e)
{  string PresentUser;
    if ((txtUserName.Text == "Admin") && (txtPassword.Text == "adm123"))
    {
        PresentUser = "Admin";
        Form Menu = new frmMenu(PresentUser);
        this.Hide();
        Menu.ShowDialog();
        txtUserName.Text = "";
        txtPassword.Text = "";
        txtUserName.Focus();
        this.Show();
    }
    else if ((txtUserName.Text == "Employee") && (txtPassword.Text == "emp123"))
    {
        PresentUser = "Emp";
        Form Menu = new frmMenu(PresentUser);
        this.Hide();
        Menu.ShowDialog();
        txtUserName.Text = "";
        txtPassword.Text = "";
        txtUserName.Focus();
        this.Show();
    }
```

```
else
{
    MessageBox.Show("Please enter a valid username and password", "Error", MessageBoxButtons.OK);
```

- Both of those if statements are identical so we will look at the first one
- The variable has been created within this “btnLogIn” class
- We use an ‘if’ statement to check if the username and password are matched
- We assign a value to the “PresentUser” (Admin or Emp) Variable and then pass it to the main menu form with Show Dialog
- When control is returned then the form is cleared
- Validation will display an error message if the log-in details are anything other than those in the code

The Main Menu Code Needs Some Changes

```
11  namespace Video_Store
12  {
13      public partial class frmMenu : Form
14      {
15          string PrsentUser;
16          public frmMenu(string User)
17          {
18              PrsentUser = User;
19              InitializeComponent();
20          }
21
```

- The “PresentUser” variable is declared in the global space at the top
- The variable is received from the “LogIn” form and then assigned to the “PresentUser”

```
62          private void frmMenu_Load(object sender, EventArgs e)
63          {
64              switch (PrsentUser)
65              {
66                  case "Admin":
67
68                      break;
69                  case "Emp":
70                      btnReportMenu.Visible = false;
71                      btnGenreMenu.Visible = false;
72                      btnMovieMenu.Visible = false;
73                      break;
74
75
76              }
77          }
```

- In the form Load class
- We select the “PresentUser” variable with a switch (Could use an if statement as well)
- There are two cases at the moment, ‘Admin’ or ‘Emp’
- Depending on if the “PresentUser” is “Admin” or “Emp” we will hide buttons or various controls on a form

Main Menu Show needs to become Show Dialog (Logic Reasons)

```
private void button1_Click(object sender, EventArgs e)
{
    Form GenreMenu = new frmGenreMenu();
    GenreMenu.Owner = this;
    this.Hide();
    GenreMenu.Show();



}

private void button2_Click(object sender, EventArgs e)
{
    this.Close();
}

private void button3_Click(object sender, EventArgs e)
{
    Form MovieMenu = new frmMovieMenu();
    MovieMenu.Owner = this;
    MovieMenu.Show();
    this.Hide();
}

private void btnReport_Click(object sender, EventArgs e)
{
    Form Report = new frmReportMenu();
    Report.Owner = this;
    Report.Show();
    this.Hide();
}

private void frmMenu_Load(object sender, EventArgs e)
{
    switch (PrsentUser)
```

- We need to move to each of the other menu forms with showdialog, as the log-In form is now the parent form
- All our forms would close and revert to the log in with the previous logic
- So change from the left to the right

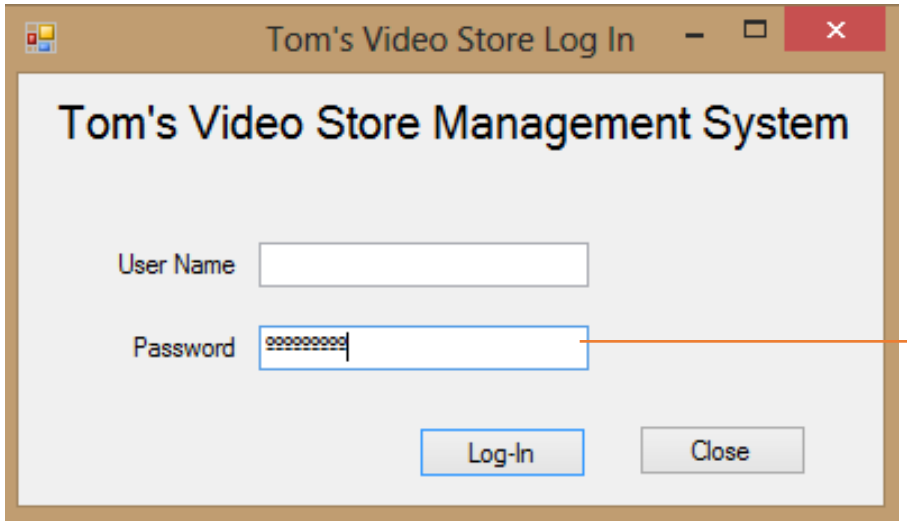
```
}

private void button1_Click(object sender, EventArgs e)
{
    Form GenreMenu = new frmGenreMenu();

    this.Hide();
    GenreMenu.ShowDialog();
    this.Show();


}

private void button2_Click(object sender, EventArgs e)
{
    this.Close();
}

private void button3_Click(object sender, EventArgs e)
{
    Form MovieMenu = new frmMovieMenu();
    this.Hide();
    MovieMenu.ShowDialog();
    this.Show();
}

private void btnReport_Click(object sender, EventArgs e)
{
    Form Report = new frmReportMenu();
    this.Hide();
    Report.ShowDialog();
    this.Show();
}

private void frmMenu_Load(object sender, EventArgs e)
{
```

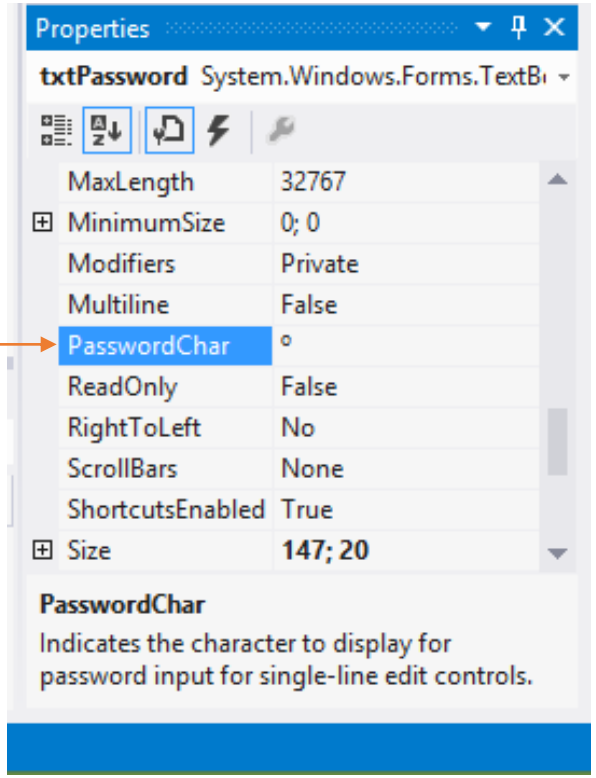
Don’t forget to delete the line from the other menu’s close buttons

```
public partial class frmGenreMenu : Form
{

    public frmGenreMenu()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        this.Owner.Show();
        this.Close();
    }

    private void frmGenreMenu_Load(object sender, EventArgs e)
    {

        // TODO: This line of code loads data into the 'tomsDB_Vide
        this.gENRETableAdapter.FillByGenreName(this.tomsDB_VideoDat
```

- Delete in genre menu and movie menu
- Delete in any new screens you have built

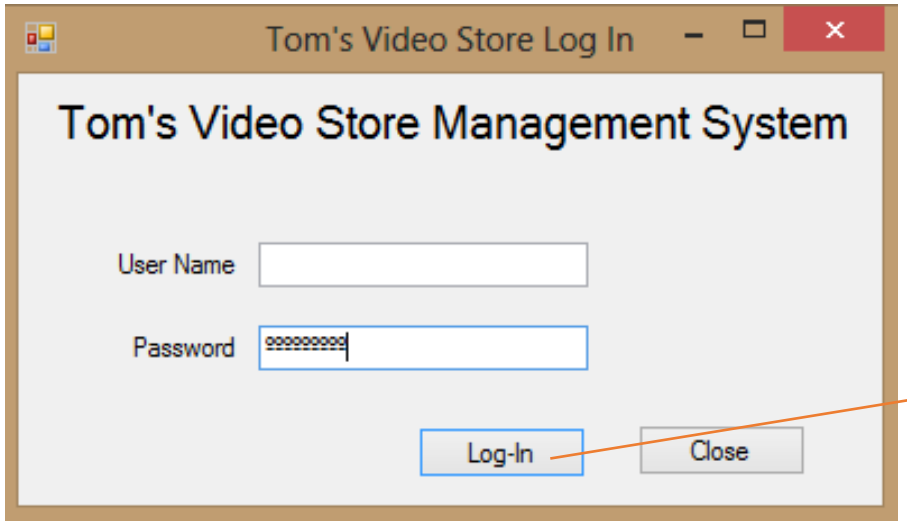
The Password Character Property

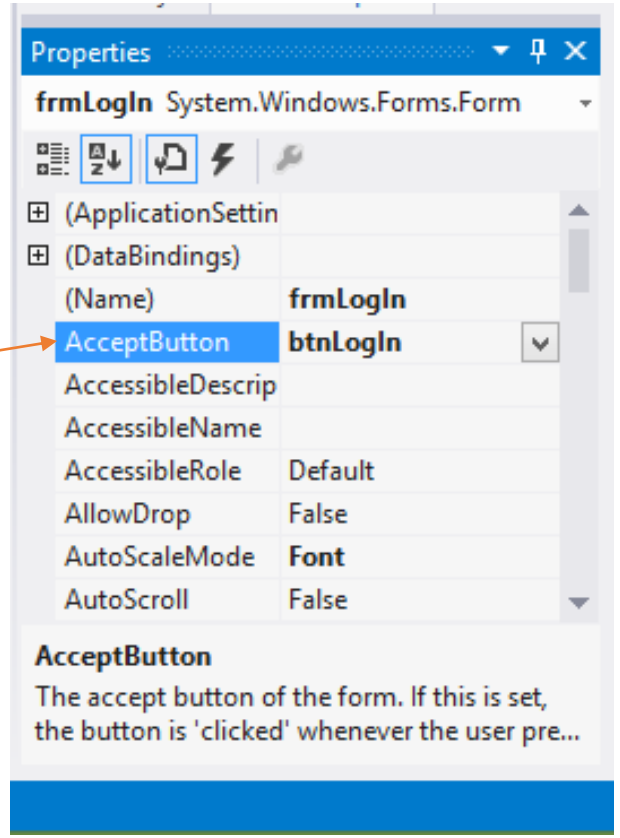




This property will assign a character to your password textbox.

Accept Button Property

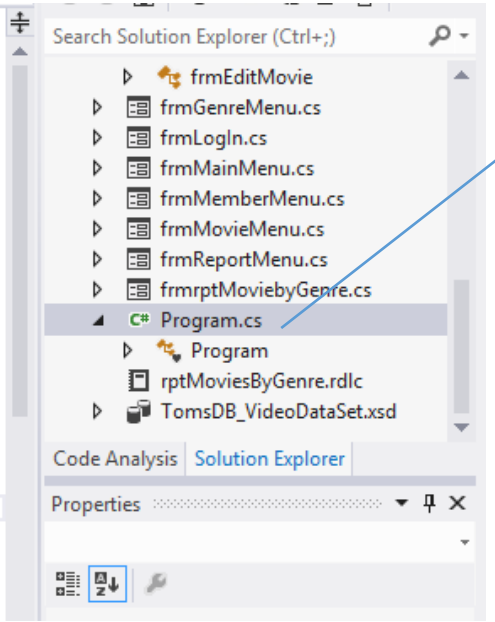




This property will activate the “LogIn” button when enter is pushed. You can assign any button/event to the enter button

Changing the form that loads on run

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using System.Windows.Forms;
6
7  namespace Video_Store
8  {
9      static class Program
10     {
11         /// <summary>
12         /// The main entry point for the application.
13         /// </summary>
14         [STAThread]
15         static void Main()
16         {
17             Application.EnableVisualStyles();
18             Application.SetCompatibleTextRenderingDefault(false);
19             Application.Run(new frmLogIn());
20         }
21     }
22 }
23
```



Go to your “program.cs” in your main project solution

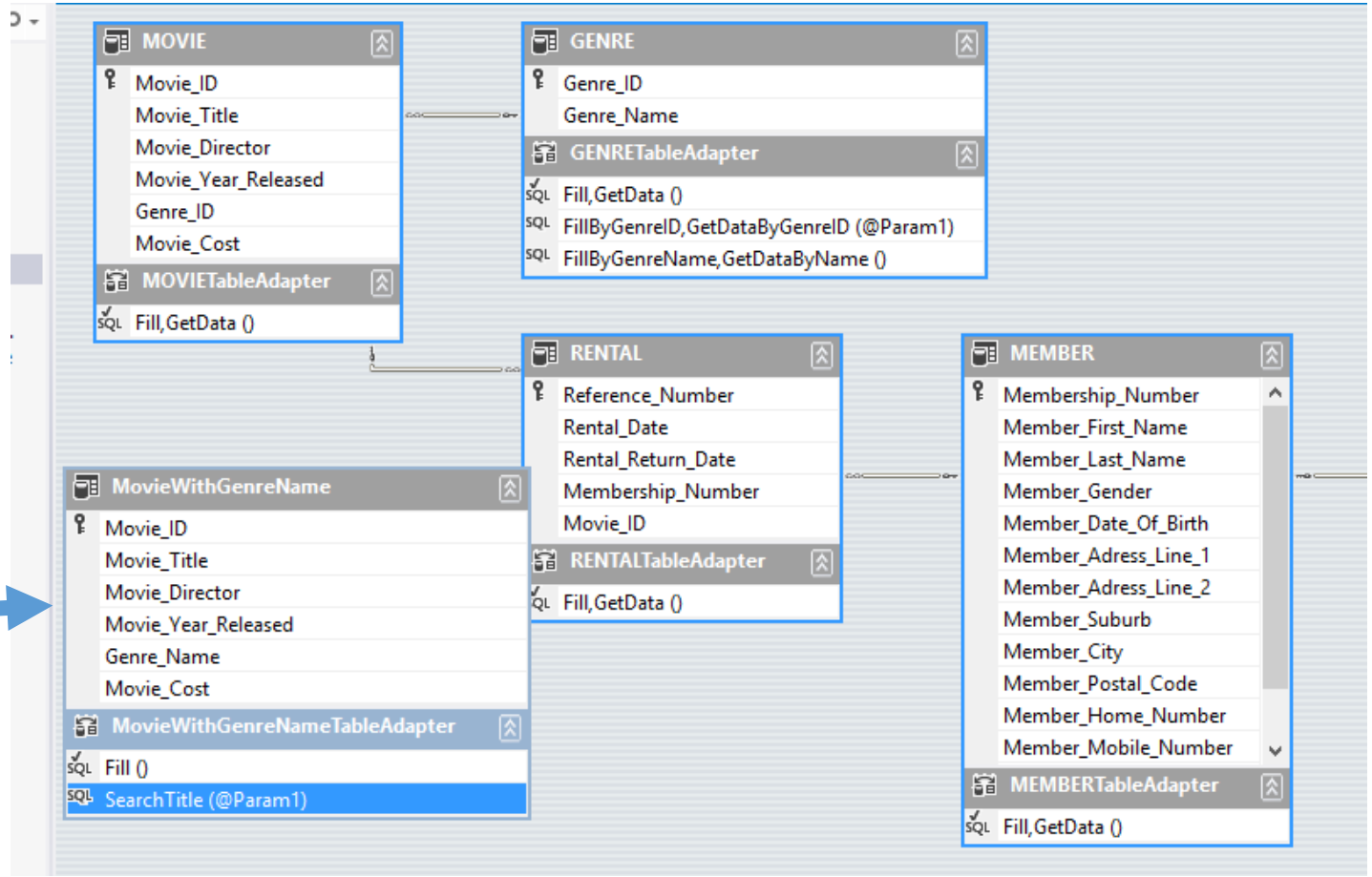
Change the form in the brackets to the form you want to load when you run the application. (“Application.Run(new [form you want to run first]())

First Step: Need to think about dataset

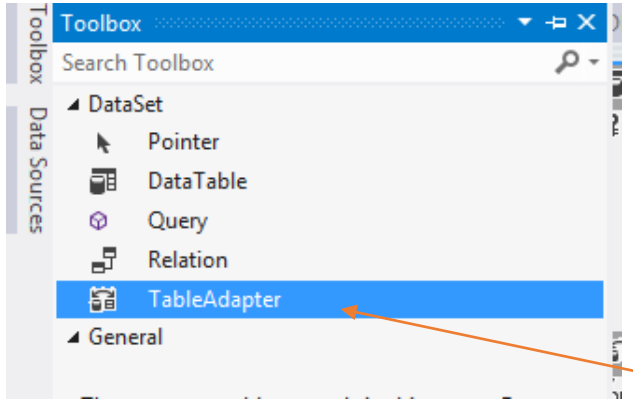
- If we are displaying data from only one table, it is not a problem as the table already exists
- If we display data from two tables then we need to make a new table adapter with that information inside
- An Example is the “Genre_Name” should be displayed instead of the “Genre_ID” when dealing with the “Movie” table
- We must create a new table adapter “MovieWithGenreName”
  - Name them so you can easily remember what their purpose was
- Important concept for reporting next week

You can create you own custom tables in the dataset

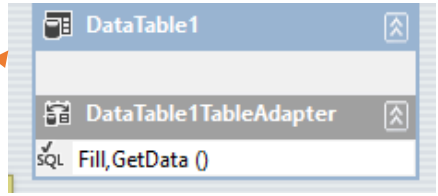
Notice this table adapter does not reflect in the database

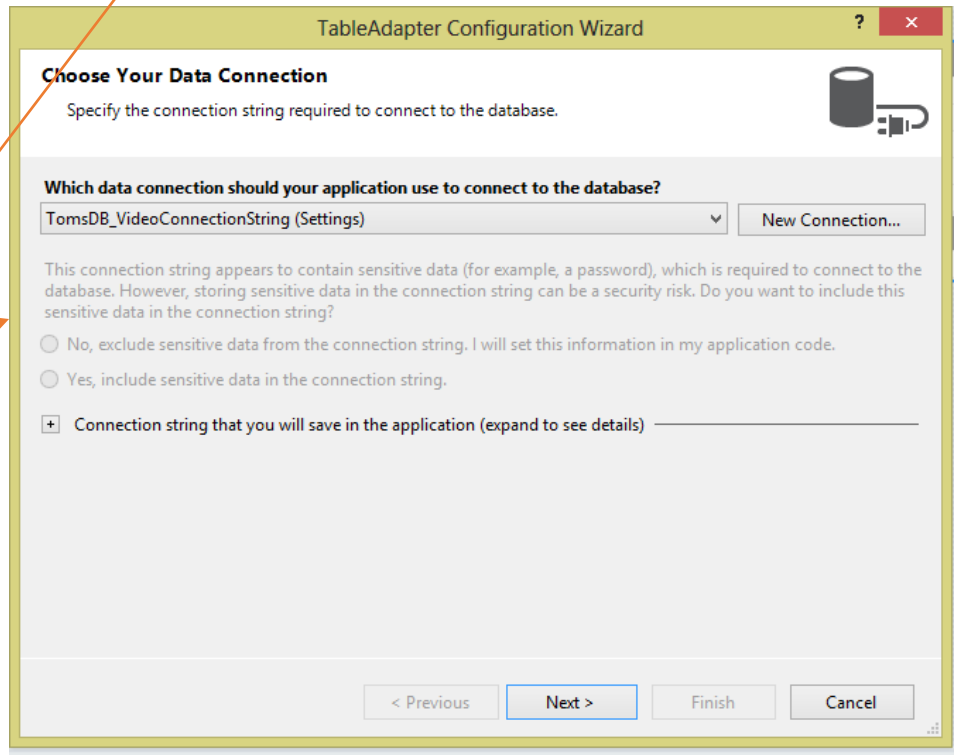


The “MovieWithGenreName” Table Adapter

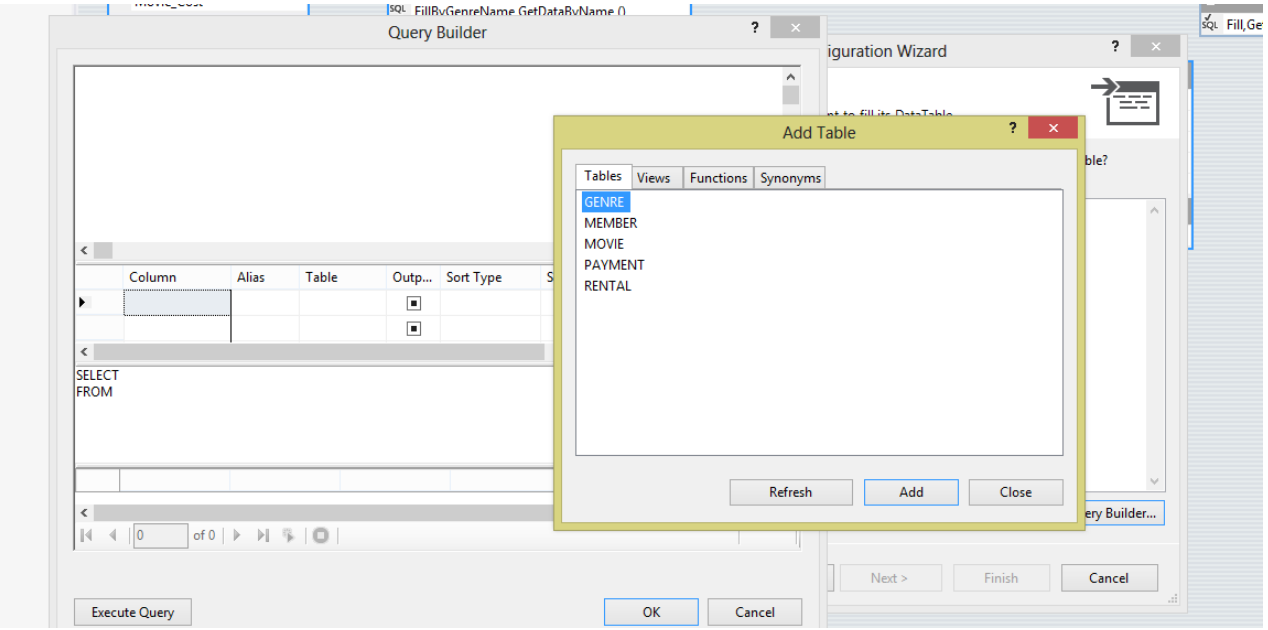


- Go to the data designer (Generally you can right-click any data source or table adapter and it will give you an option: “edit in data designer”)
- Drag a new table adapter onto the data area (Where all the datasets and relationships are)
- A new table adapter and data set will add, and you then need to edit it (These open simultaneously)

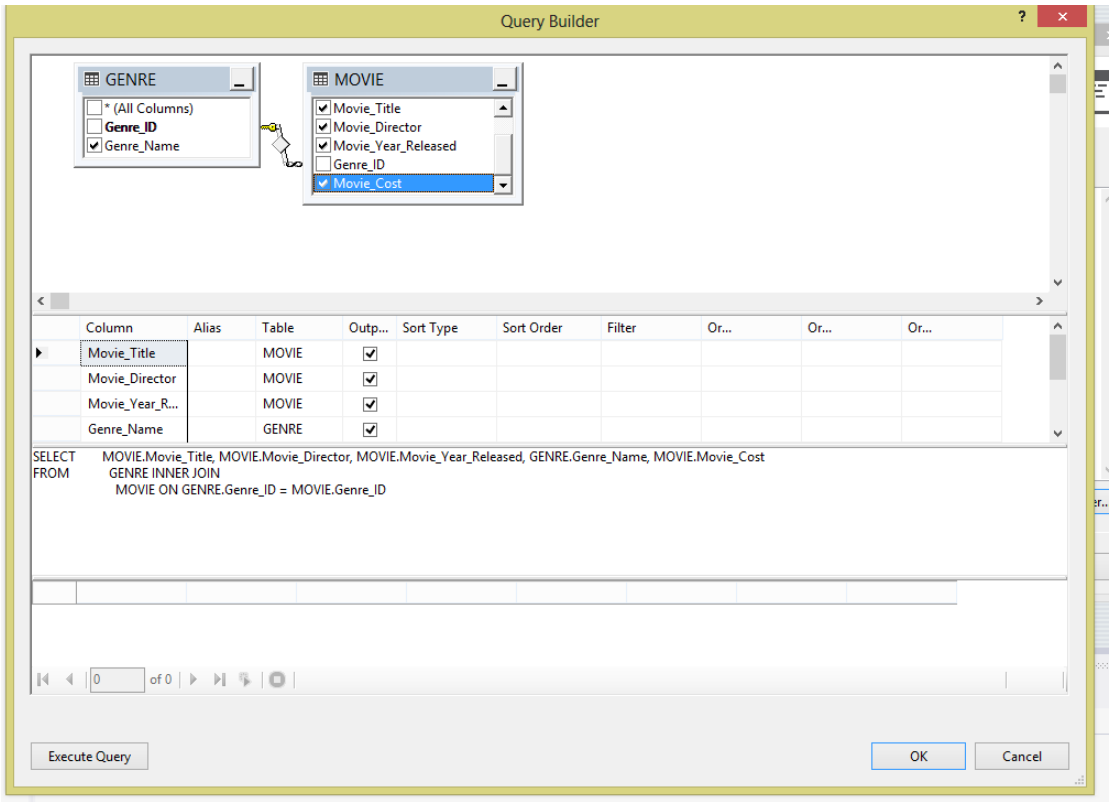




Create the “Fill” statement that controls the fields in your data table

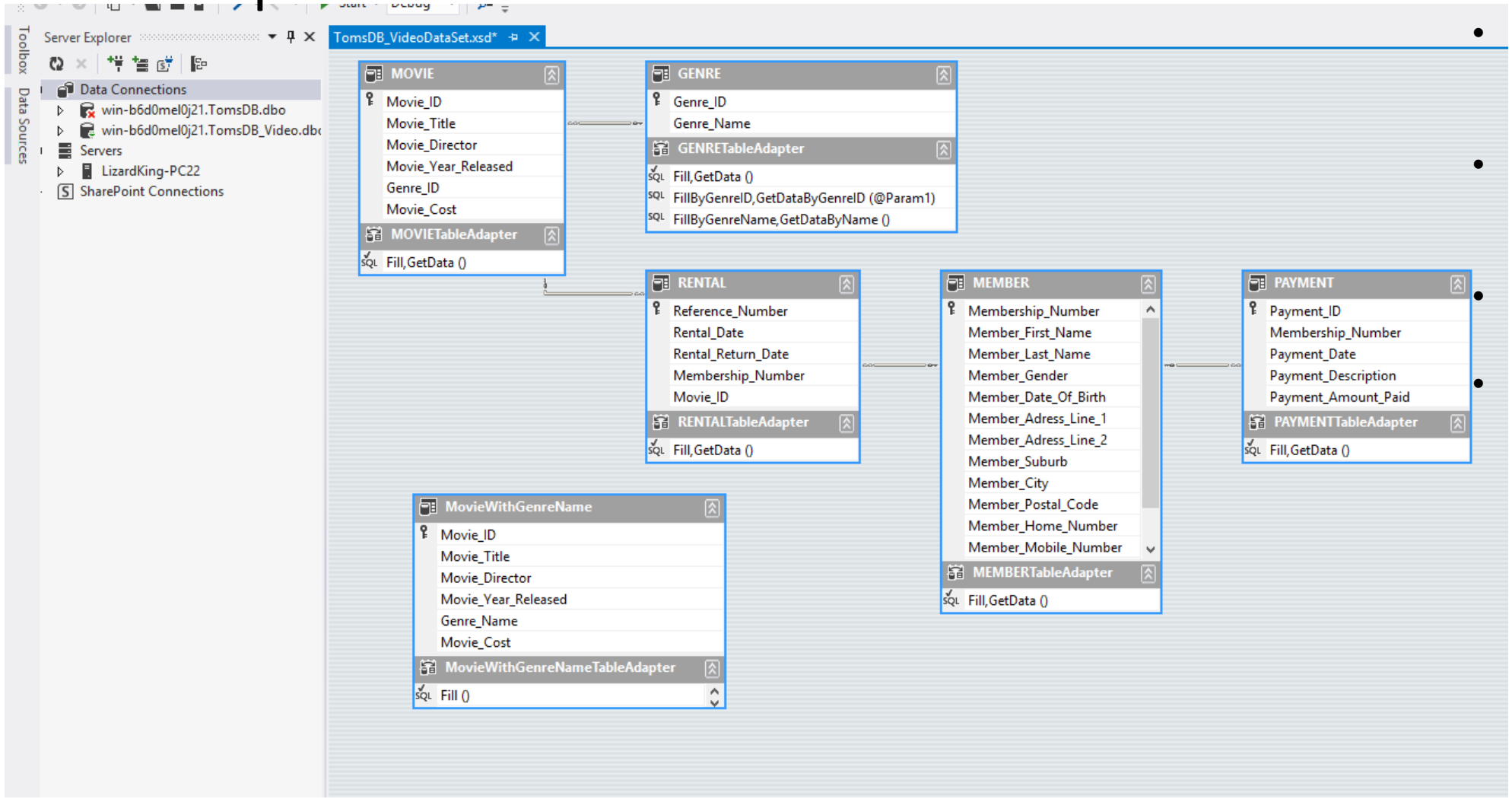


Instead of adding one table, we add two and select the collums we want. The query builder does all the SQL joins for us, as noted in the left



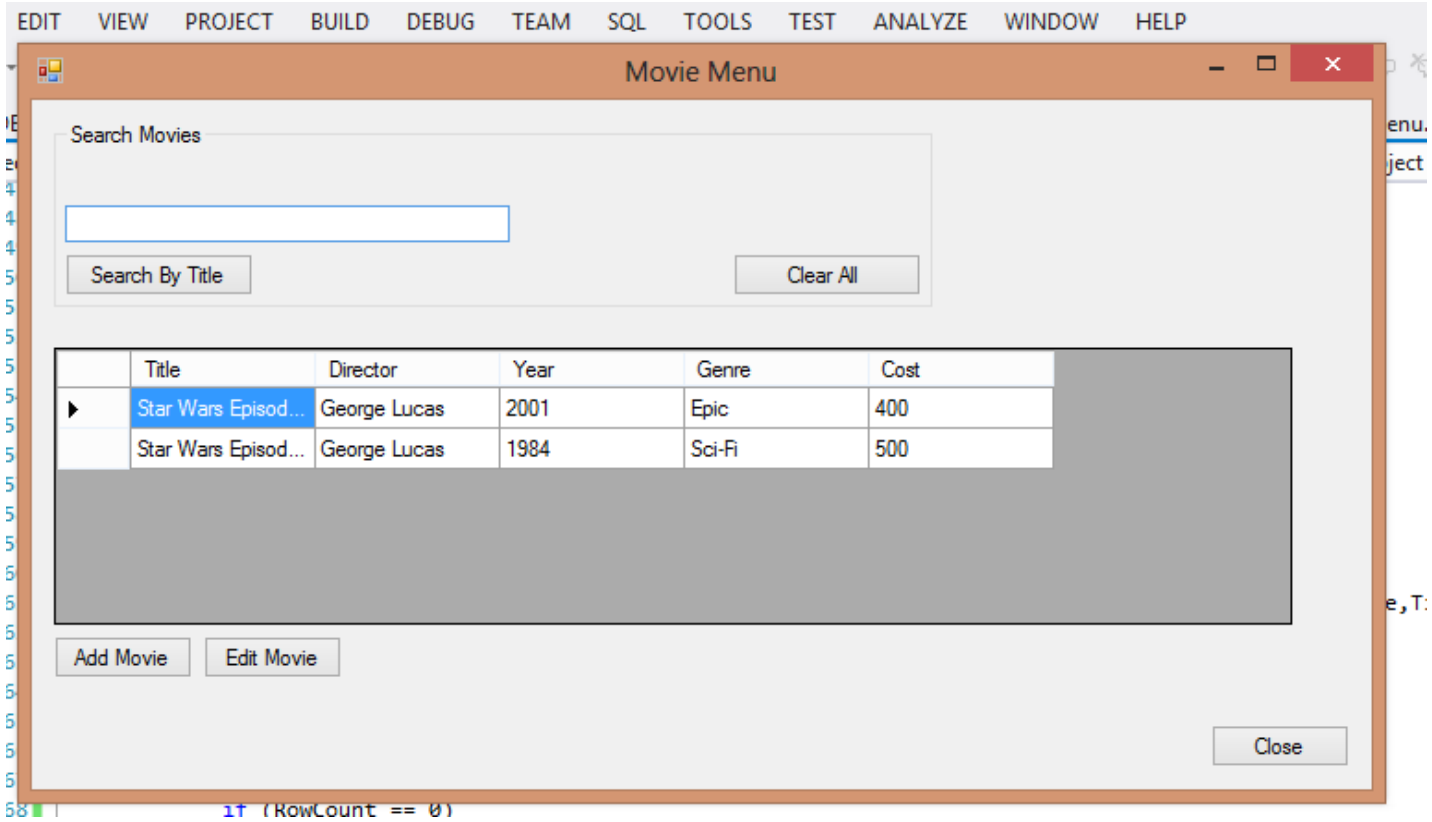
Tick the columns you want, so replace the “Genre_ID” with the “Genre_Name”

The dataset designer and our new table adapter



- The new table adapter has been filled by our fill statement
- Remember to give it a name, here it is “MovieWithGenreName”
- This is the dataset designer
- You will have to make a new table adapter for any report that needs additional information

Searching



We want to be able to type part of a title into the textbox and then populate the data grid with only those movies that include what has been typed.

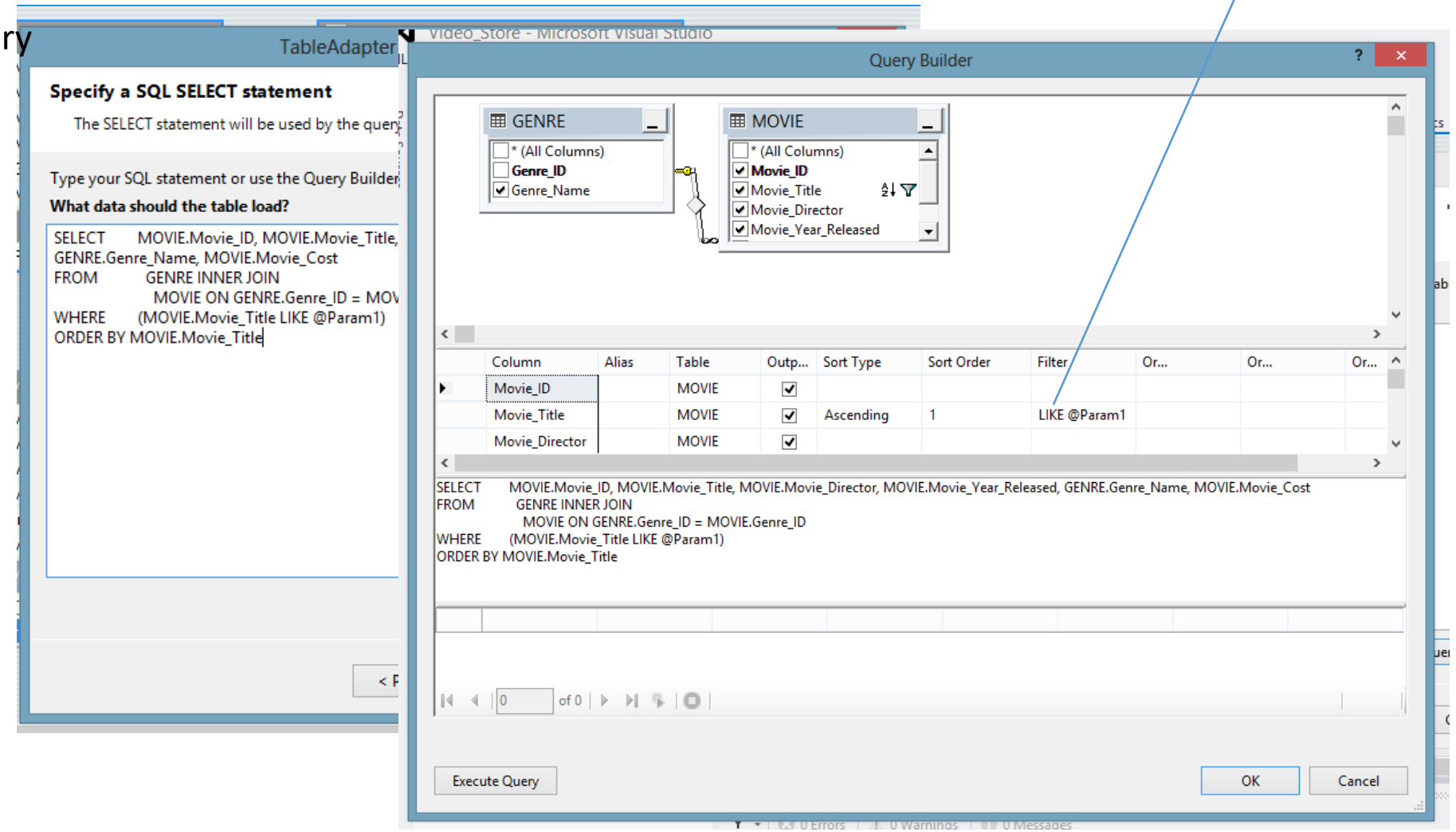
Implementing the search function

- Create a parameter based query in a table adapter (covered last week)
- Replace the = with LIKE from SQL
- When inserting the parameter variable ensure that you have % symbols between, after or before the variables
  - Eg parameter = “%”+parameter+”%” will search for anything including the parameter
- % is a wildcard
- http://www.w3schools.com/sql/sql_wildcards.asp has more info on wildcard use

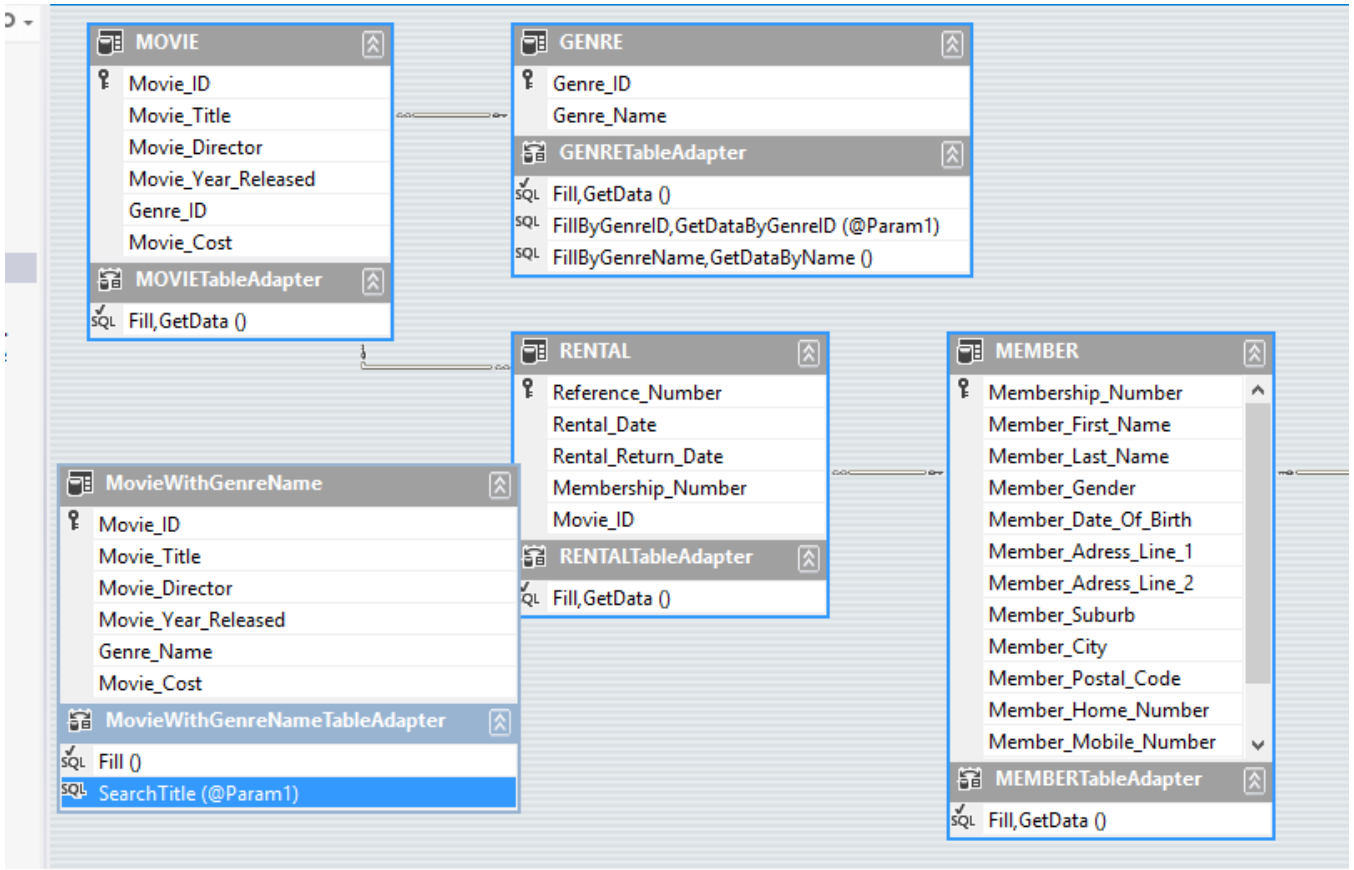
Add a parameter based query to the table adapter

Create a parameter based query

NB Replace = with LIKE

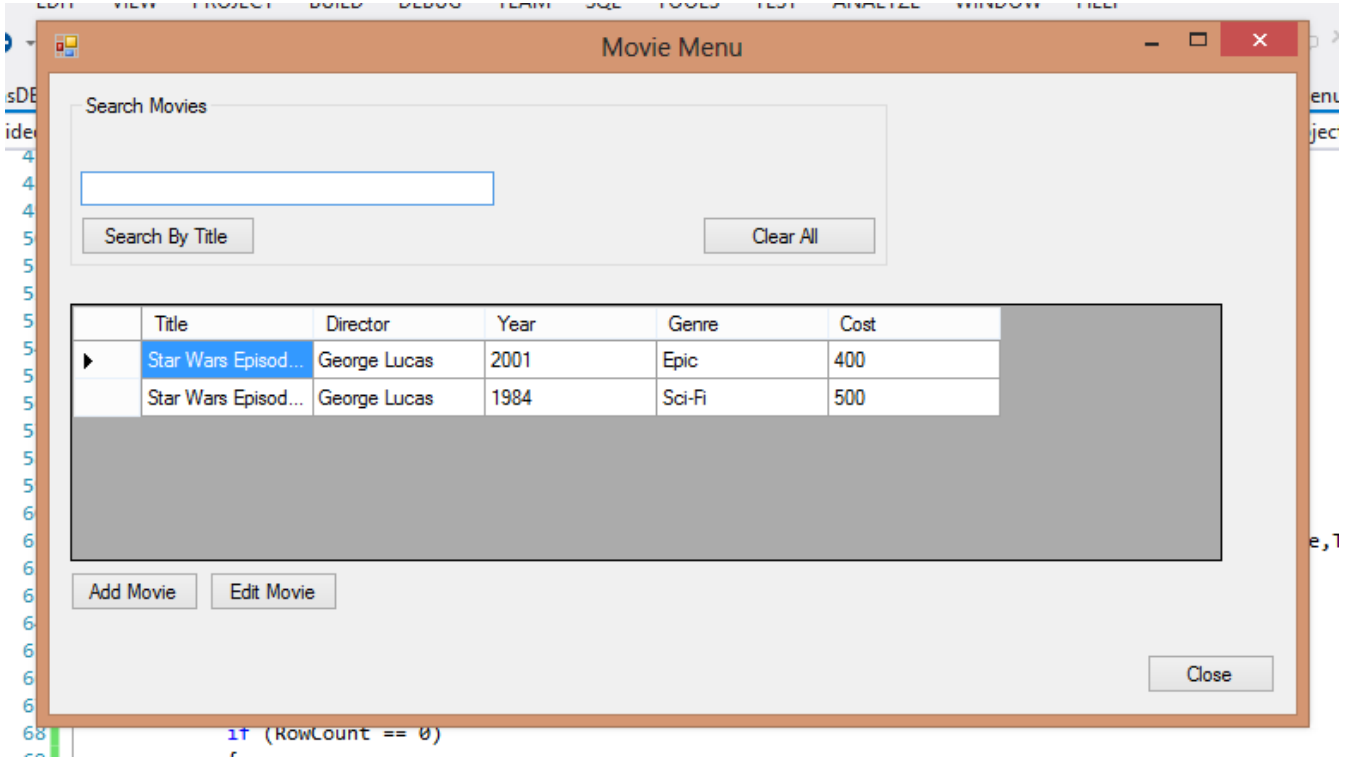


New query has been added



Generally name them by the type of search you are going to do, this will help you if you have made many search queries

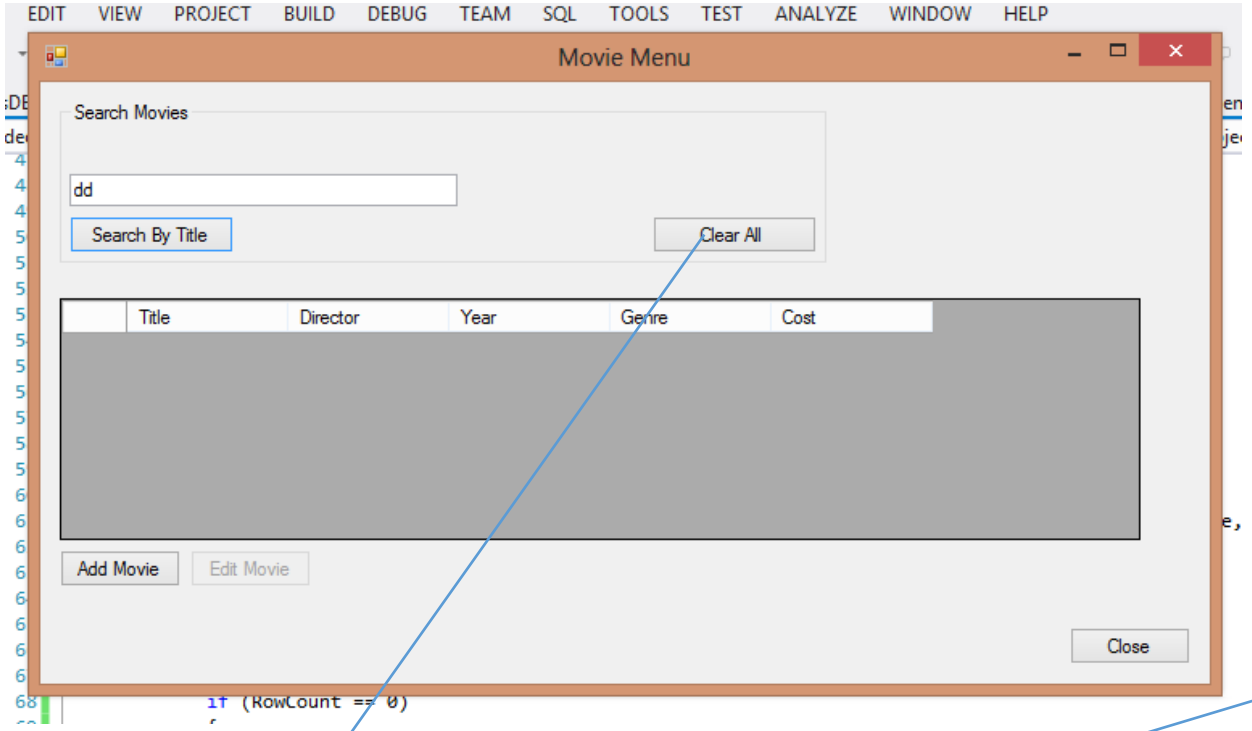
Lets see how it works




```
private void btnSearchTitle_Click(object sender, EventArgs e)
{
    string Title = txtSearch.Text;
    Title = '%' + Title + '%';
    this.movieWithGenreNameTableAdapter.SearchTitle(this.tomsDB_VideoDataSet.MovieWithGenreName,Title
    EditButtonValidation();
}
```

- First line declares a string, we need to do this as we are going to change the text with sql operators
- The next line does just that, notice we are placing a '%' symbol between the title. This means that it will search for a word containging anything in the string title.
- If we were to use only one % sign, we would be looking for word starting with title
- The third line shows the query we just made
- The fourth line is explained on the next slide

Edit Button Validation

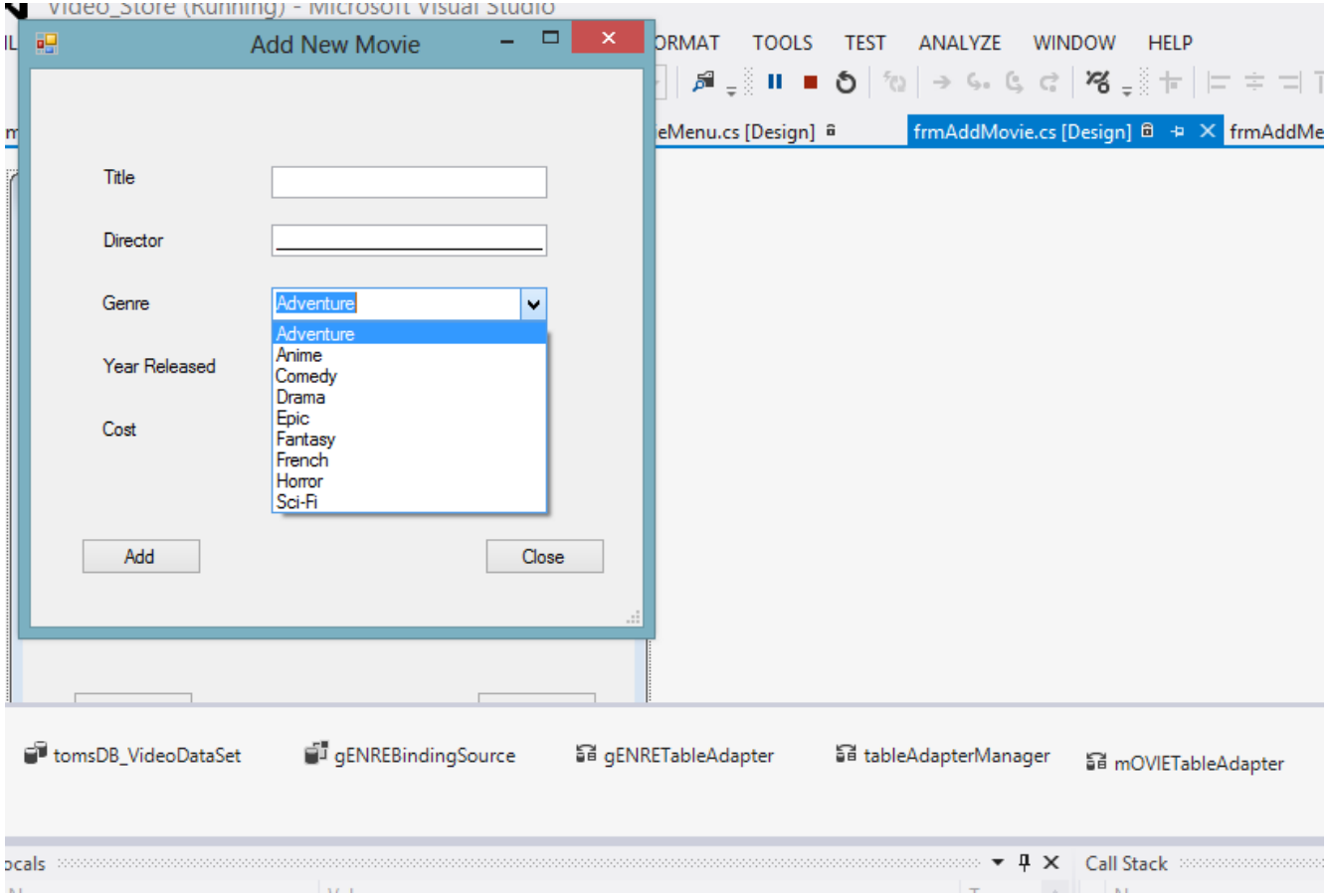


- EditValidation checks whether there are rows in our datagrid to select, and if there are none it disables the edit button.
- The first line counts the rows
- The “Clear All” button just runs the old fill statement that will clear all search results and populate the grid with all movies
- It has to run after the clear button to enable the button again.

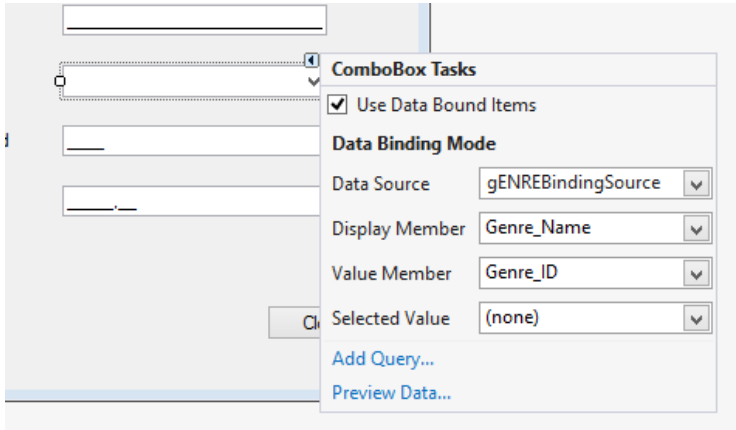
```
private void button1_Click(object sender, EventArgs e)
{
    this.movieWithGenreNameTableAdapter.Fill(this.tomsDB_VideoDataSet.MovieWithGenreName);
    EditButtonValidation();
}
```

```
private void EditButtonValidation()
{
    int RowCount = dtgMovie.BindingContext[dtgMovie.DataSource].Count;

    if (RowCount == 0)
    {
        btnEditMovie.Enabled = false;
    }
    else
    {
        btnEditMovie.Enabled = true;
    }
}
```

Combo Boxes



- Notice the drop down list is populated with each of the genre's we have created
- The display member is the Genre_Name while the value member is the Genre_ID
- This allows us to put foreign keys into other tables while still displaying the name to the user, and not some numbers
- Below are the display and value member settings.



Code to add the foreign key

```
string MovieTitle = txtTitle.Text;
string MovieDirector = msktxtDirct.Text;
string MovieYear = msktxtYear.Text;
int GenreID = Convert.ToInt32(cmbbxGenre.SelectedValue);
double MovieCost = Convert.ToDouble(msktxtCost.Text);

TomsDB_VideoDataSet.MOVIERow NewMovieRow = tomsDB_VideoDa

NewMovieRow.Movie_Title = MovieTitle;
NewMovieRow.Movie_Director = MovieDirector;
NewMovieRow.Movie_Year_Released = MovieYear;
NewMovieRow.Genre_ID = GenreID;
NewMovieRow.Movie_Cost = MovieCost;
```

- The selected value is converted to an integer
- This is then added to the new row.

Exercise

- Now complete exercise 1 & 2
- Add Access Control