**SPORTS CLUB MANAGEMENT SYSTEM**

**By Pritish Arora**

**INDEX**

1.SYSTEMANALYSIS……………………………………………………………….…...15

2.DBMS ARCHITECTURE…………………………………………………………..…...17

3.DATA MODELLING………………………………………………………………..…..20

4.DATABASE DESIGN……………………………………………………………….......22

5.MODEL MAPPING………………………………………………………………...…....26

6.USER INTERFACE DESIGN………………………………………………………..….31

7.DATABASE CONNECTIVITY………………………………………………….……...35

8.SYSTEM IMPLEMENTATION…………………………………………………….…38

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Topic | Page No. |
| 1 | DBMS 2-Tier Architecture | 18 |
| 2 | Two-Tier Architecture | 19 |
| 3 | E-R Diagram | 21 |
| 4 |  | 28 |
| 5 | Login Details | 29 |
| 6 |  | 30 |
| 7 | Attendance Records | 31 |
| 8 | Data Adapter | 37 |
| 9 | System Implementation | 40 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Topic | Page No. |
| 1 | Unnormalized Form | 23 |
| 2 | First Normalized Form | 23 |
| 3 | Employee1(2NF) | 24 |
| 4 | Employee2(2NF) | 24 |
| 5 | Employee3(2NF) | 24 |
| 6 | Attendance(3NF) | 25 |
| 7 | Salary(3NF) | 25 |

**INTRODUCTION**

**Introduction of System:**

“Sports Club Management System” is a distributed application, developed to evaluate the performance of players working in any club. It maintains the information about a club, personal details of their employees and players, also the project details assigned to particular developer. The application is actually a suite of applications developed using C#.

It is simple to understand and can be used by anyone who is not even familiar with simple management system. It is user friendly and just asks the user to follow step by step operations by giving him few options. It is fast and can perform many operations of a company.

This software package has been developed using the powerful coding tools of C# at Front End and Oracle SQL at Back End. Because of the Visual features, the software is very user friendly. The package contains different modules like Contacts, Search for property and other useful Links. This version of the software has multi-user approach. For further enhancement or development of the package, user’s feedback will be considered.

**Problem Definition**

The problem assigned to us is as follows:

“We are to design and implement a SPORTS CLUB MANAGEMENT SYSTEM that should accept employee details, days worked, compute net pay and gross pay and record all the Sports Club data for subsequent processing. The system should prepare pay cheques, and maintain data on a sequential Sports Club file.

The Sports Club data are employee id, employee name, password, no. of working days and pay rate.”

**Objective**

**The main objective of SPORTS CLUB MANAGEMENT SYSTEM is to help** organization to manage employee’s salary monthly cutting, tax information salary slips. The other objectives are:

* To improve efficiency of company’s administration
* To store up-to-date information of the employees.
* To make the employees understand the rules of the company.
* To reduce extra cost for security of database.
* Employees can easily put feedback through online support

CHAPTER 1

SYSTEM ANALYSIS

**System Analysis**

* Platform

1. Windows Operating system

* Software Requirements

1. Microsoft Visual Studio Professional (2010)
2. Microsoft Windows
3. SQL Server (with SQL Server Management Studio)
4. Selenium IDE

* Hardware Specifications

1. Processor: Intel Pentium or more
2. Motherboard: Intel® Chipset Motherboard.
3. RAM: 2 GB or more
4. Cache: 512 KB
5. Hard disk: 16 GB hard disk recommended
6. Monitor: 1024 x 720 Display

CHAPTER 2

DATABASE ARCHITECTURE

**DBMS Architecture**

**2-Tier Architecture**

* The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
* The user interfaces and application programs are run on the client-side.
* The server side is responsible to provide the functionalities like: query processing and transaction management.
* To communicate with the DBMS, client-side application establishes a connection with the server side.

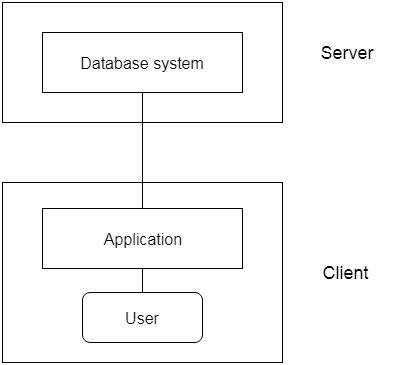


Fig.1 DBMS 2-Tier Architecture

Two tier architecture is similar to a basic client-server model. The application at the client end directly communicates with the database at the server side. API’s like ODBC, JDBC are used for this interaction. The server side is responsible for providing query processing and transaction management functionalities. On the client side, the user interfaces and application programs are run. The application on the client side establishes a connection with the server side in order to communicate with the DBMS.

An advantage of this type is that maintenance and understanding is easier, compatible with existing systems. However, this model gives poor performance when there are a large number of users.

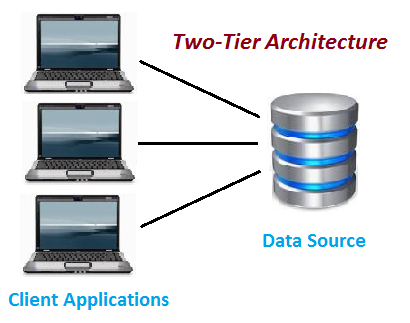


Fig. 2 Two-Tier Architecture

The project is a based on two tier architecture because as it does not contain application server such as apache tomcat or any other web server. also, there is no interface between client and server.

CHAPTER 3

DATA MODELLING

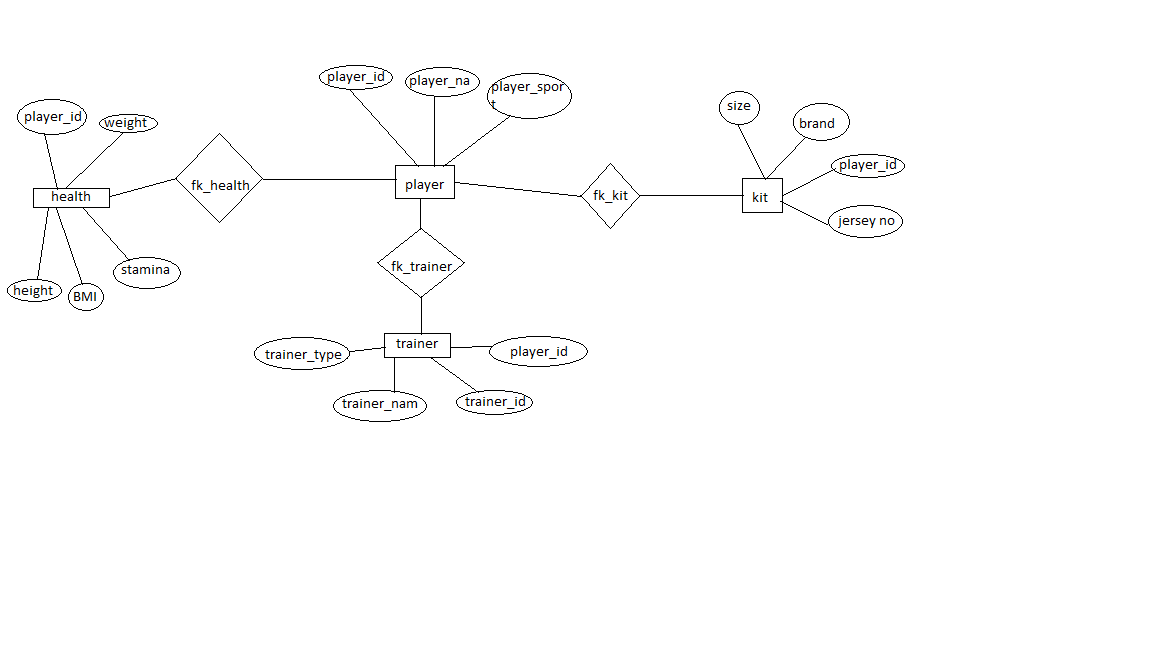


Fig.3 E-R Diagram

CHAPTER 4

DATABASE DESIGN

SPORTS CLUB MANAGEMENT SYSTEM

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Player\_id | Player\_nm | Sport | Weight | Height | Stamina | BMI | Brand | Size | Jersey\_no |
| 1 | Pratik | Football | 60 | 165 | 40 | 1.34 | Nike | 32 | 7 |
| Trainer\_id | Trainer\_nm | Trainer\_type |  |  |  |  |  |  |  |
| 101 | Pritish | Manager |  |  |  |  |  |  |  |

Master Table(Unnormalized Form)

1NF (Removing all multi-value attributes)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Player\_id | Player\_nm | Sport | Weight | Height | Stamina | BMI | Brand | Size | Jersey\_no |
| 1 | Pratik | Football | 60 | 165 | 40 | 1.34 | Nike | 32 | 7 |
| 1 | Pratik | Football | 60 | 165 | 40 | 1.34 | Nike | 32 | 7 |
| 1 | Pratik | Football | 60 | 165 | 40 | 1.34 | Nike | 32 | 7 |
| Trainer\_id | Trainer\_nm | Trainer\_type |  |  |  |  |  |  |  |
| 101 | Pritish | Manager |  |  |  |  |  |  |  |
| 101 | Pritish | Manager |  |  |  |  |  |  |  |
| 101 | Pritish | Manager |  |  |  |  |  |  |  |

2NF(Removing Dependency of non prime attributes)

|  |  |  |
| --- | --- | --- |
| Player\_id | Player\_nm | Sports |
| 1 | Pratik | Football |
| 2 | Nishant | Cricket |

PLAYERS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Player\_id | Height | Weight | Stamina | BMI |
| 1 | 165 | 60 | 40 | 1.34 |
| 2 | 175 | 60 | 70 | 2.4 |

HEALTH

|  |  |  |  |
| --- | --- | --- | --- |
| Player\_id | Brand | Size | Jersey no |
| 1 | Nike | 34 | 7 |
| 2 | Puma | 30 | 10 |

KIT

|  |  |  |  |
| --- | --- | --- | --- |
| Player\_id | Trainer\_id | Trainer\_nm | Trainer\_type |
| 1 | 101 | Pritish | Manager |
| 2 | 102 | Puskar | Coach |

TRAINER

3NF(Remove The Transitive Dependecy)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Player\_id | Player\_nm | Sports | Jersey\_no | Trainer\_id |
| 1 | Pratik | Football | 7 | 101 |
| 2 | Nishant | Cricket | 10 | 102 |

CHAPTER 5

MODEL MAPPING

**MODEL MAPPING**

ER Model, when conceptualized into diagrams, gives a good overview of entity-relationship, which is easier to understand. ER diagrams can be mapped to relational schema, that is, it is possible to create relational schema using ER diagram. We cannot import all the ER constraints into relational model, but an approximate schema can be generated. There are several processes and algorithms available to convert ER Diagrams into Relational Schema. Some of them are automated and some of them are manual. We may focus here on the mapping diagram contents to relational basics. ER diagrams mainly comprise of −

* Entity and its attributes
* Relationship, which is association among entities.

**Mapping Entity**

An entity is a real-world object with some attributes.

**Mapping Process (Algorithm)**

* Create table for each entity.
* Entity's attributes should become fields of tables with their respective data types.
* Declare primary key.

**Mapping Relationship**

A relationship is an association among entities.

**Mapping Process**

* Create table for a relationship.
* Add the primary keys of all participating Entities as fields of table with their respective data types.
* If relationship has any attribute, add each attribute as field of table.
* Declare a primary key composing all the primary keys of participating entities.
* Declare all foreign key constraints.

**Mapping Weak Entity Sets**

A weak entity set is one which does not have any primary key associated with it.

**Mapping Process**

* Create table for weak entity set.
* Add all its attributes to table as field.
* Add the primary key of identifying entity set.
* Declare all foreign key constraints.

**Mapping Hierarchical Entities**

ER specialization or generalization comes in the form of hierarchical entity sets.

**Mapping Process**

* Create tables for all higher-level entities.
* Create tables for lower-level entities.
* Add primary keys of higher-level entities in the table of lower-level entities.
* In lower-level tables, add all other attributes of lower-level entities.
* Declare primary key of higher-level table and the primary key for lower-level table.
* Declare foreign key constraints.

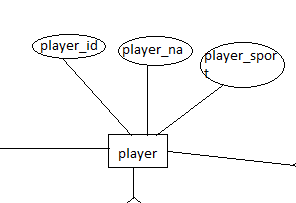


Fig.4 Player Entity

|  |  |  |
| --- | --- | --- |
| Player\_id | Player\_nam | Player\_sports |
| 1 | Pratik Mali | Football |
| 2 | Nishant Singh | Cricket |

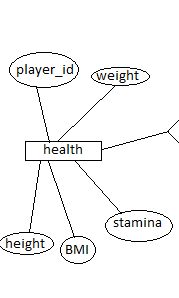


Fig.6 Health Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Player\_id | Weight | Height | BMI | Stamina |
| 1 | 60 | 165 | 1.34 | 40 |
| 2 | 60 | 175 | 2.1 | 80 |

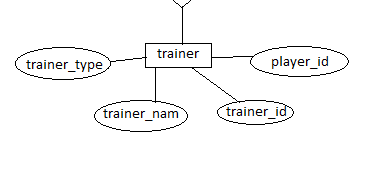


Fig.7 Trainer Records

|  |  |  |  |
| --- | --- | --- | --- |
| Player\_id | Trainer\_id | Trainer\_nam | Trainer\_type |
| 1 | 12 | Abc | Manager |
| 2 | 13 | Xyz | Coach |

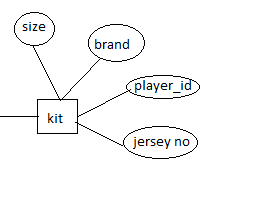


Fig.8 Kit Records

|  |  |  |  |
| --- | --- | --- | --- |
| Player\_id | Size | Brand | Jersey no |
| 1 | 32 | Nike | 7 |

CHAPTER 6

USER INTERFACE DESIGN

**USER INTERFACE DESIGN**

**Windows Form Controls**

At the centre of most Visual Basic Windows applications stands the form designer. You create a user interface by dragging and dropping controls from a toolbox to your form, placing them where you want them to be when you run the program, and then double-clicking the control to add handlers for the control. The controls provided out of the box by Microsoft along with custom controls that can be bought at reasonable prices, have supplied programmers with an unprecedented pool of reusable, thoroughly tested code that is no further away than a click with the mouse. What was central to Visual Basic is now, through Visual Studio.NET, available to C# programmers.

**The Windows Form Designer**

Start Visual Studio.NET and create a new C# Windows Application project by selecting File | New | Project. In the dialog that appears, click Visual C# Projects in the tree to the left and then select Windows Application in the list to the right. For now, simply use the default name suggested by Visual Studio and click OK. This should bring up a window much like the one shown below:

In the centre of the screen is the form that you are designing. You can drag and drop controls from the toolbox onto the form. The toolbox is collapsed in the picture above, but if you move the mouse pointer to the far left of the screen over the Toolbox tab, it will unfold. You can then click the pin at the top right of the panel to pin it down. This will rearrange the work area so that the toolbox is now always on top, and isn't obscuring the form. We'll take a closer look at the toolbox and what it contains shortly.

Also collapsed on the left-hand bar is the Server Explorer – represented by the computers icon on top of the toolbox tab. You can think of this as a small version of the Windows Control Panel. From here, you can browse computers on a network, add and remove database connections, and much more.

To the right of the window are two panels. The top-right one is the Solution Explorer and the class view. In the Solution Explorer, you can see all open projects and their associated files. By clicking the tab at the bottom of the Solution Explorer, you activate the Class Viewer. In this, you can browse all of the classes in your projects and all of the classes that they are derived from.

At the bottom right of the screen, is the Properties panel. This panel will contain all of the properties of the selected item for easy reference and editing. We'll be using this panel quite a bit in this chapter.

**The Toolbox**

Let's have a closer look at the toolbox. If you haven't already, move your mouse pointer over the toolbox on the left of the screen, and pin it to the foreground by clicking the pin at the top right of the panel that unfolds:

If you accidentally remove the toolbox by clicking the X instead, you can make it reappear by selecting Toolbox from the View menu, or by pressing Ctrl-Alt-X.

**Controls**

Most controls in .NET derive from the System.Windows.Forms.Control class. This class defines the basic functionality of the controls, which is why many properties and events in the controls we'll see are identical. Many of these classes are themselves base classes for other controls, as is the case with the Label and TextBoxBase

**Properties**

All controls have a number of properties that are used to manipulate the behaviour of the control. The base class of most controls, Control, has a number of properties that other controls either inherit directly or override to provide some kind of custom behaviour.

The table below shows some of the most common properties of the Control class. These properties will be present in most of the controls we'll visit in this chapter, and they will therefore, not be explained in detail again, unless the behaviour of the properties is changed for the control in question. Note that this table is not meant to be exhaustive; if you want to see all the properties in the class, please refer to the MSDN library.

**ContextMenuStrip Control**

The ContextMenuStrip control provides functionality of context menus in Visual Studio 2010 and .NET 4.0. A context menu is also known as a popup menu. A context menu appears when you right click on a Form or on a control.

**Creating a Context Menu**

To create a ContextMenuStrip control at design-time, you simply drag and drop a ContextMenuStrip control from Toolbox onto a Form in Visual Studio. After you drag and drop a ContextMenuStrip on a Form, the ContextMenuStrip1 is added to the Form and looks like Figure 1. Once a ContextMenuStrip is on the Form, you can add menu items and set its properties and events.

**What is a dialog box?**

* A dialog box is a type of window, which is used to enable common communication or dialog between a computer and its user.
* A dialog box is most often used to provide the user with the means for specifying how to implement a command or to respond to a question.
* Windows.Form is a base class

MessageBox class is used to display messages to the user. The show() method is used to display a message box with the specified text, caption, buttons and icon. There are other overloads also available.

Dialog boxes are of two types, which are given below.

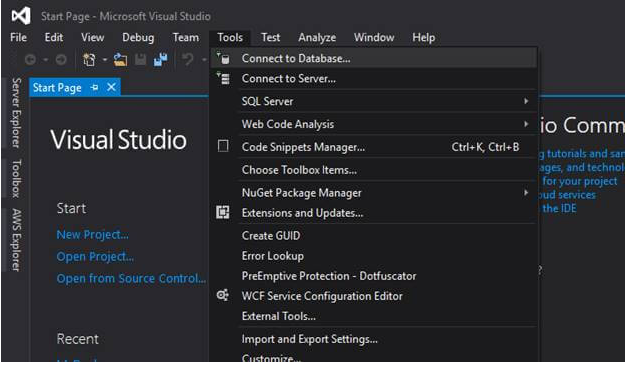
1. Modal dialog box
2. Modeless dialog bo

CHAPTER 7

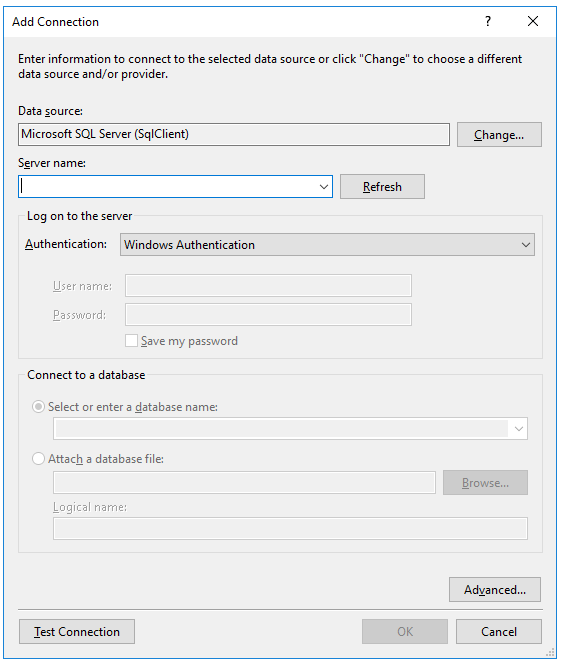
DATABASE CONNECTIVITY

**Database Connectivity**

Go to "Tools" menu and then click on "Connect to Database".

****

The "Add Connection" window opens.

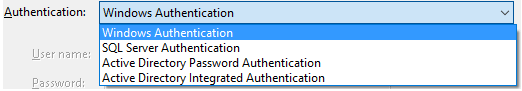
****

First, you need to choose the type of database to which you want to connect. MS SQL Server is pre-selected.

In the Server Name, type the IP and Instance of your Server. In this example, we will be connecting to a local instance of SQL Server, so I’ve entered dot (.).

Select Authentication. (Again, since it's local, I’ve selected Windows Authentication.). Selecting SQL Server Authentication, you can enter username and password.

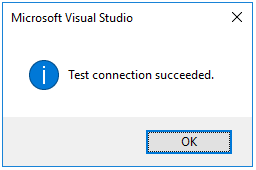
Below are the authentication types available.



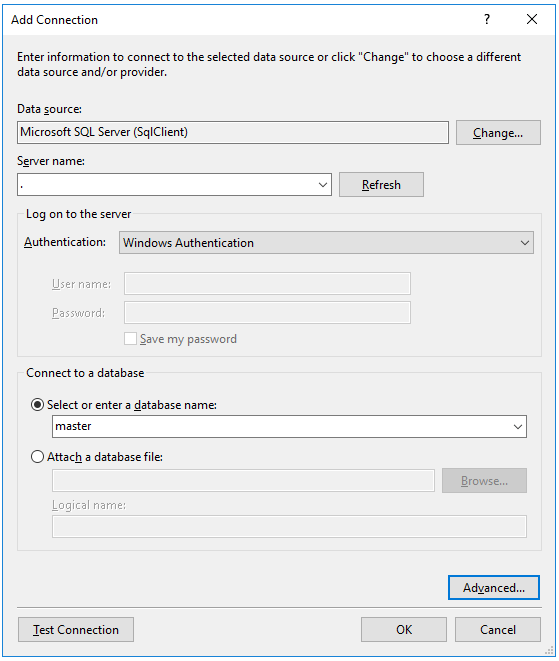
Then, select the required database from the drop down or you can type it.

For advanced setting, click "Advanced…"

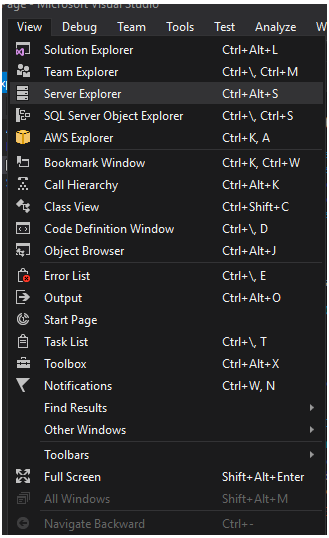
You can also click on "Text Connection" to see if the details entered by you are correct and are able to connect to the SQL Server Instance.



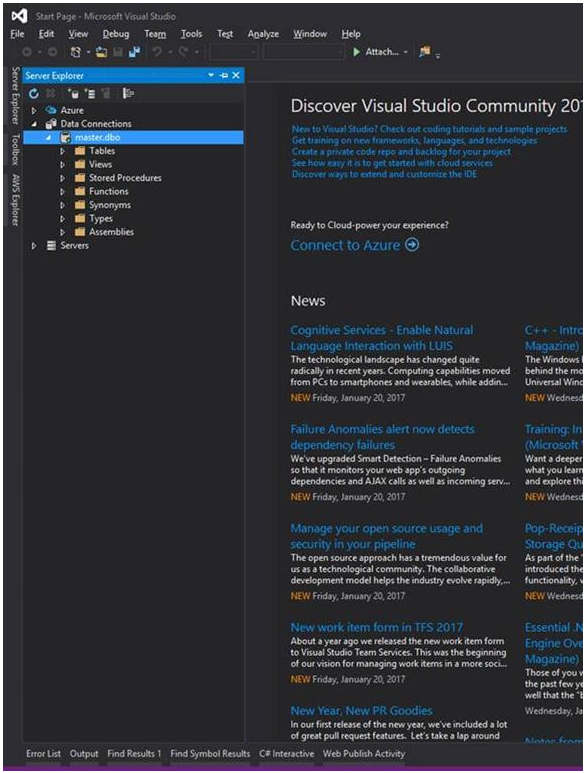
Once done, click OK.



The database connection will now show in the "Server Explorer" tab. If you can’t see it, go to "View" menu and click "Server Explorer".

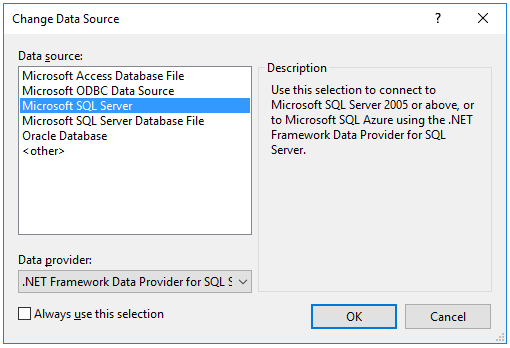


It will open the tab.



**Note**

By default, you can connect to the below types of databases.



**Data Adapter**

DataAdapter is a part of the ADO.NET Data Provider. DataAdapter provides the communication between the Dataset and the Data source. We can use the DataAdapter in combination with the DataSet Object. That is these two objects combine to enable both data access and data manipulation capabilities.

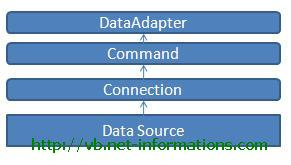


Fig.8 Data Adapter

The DataAdapter can perform Select, Insert, Update and Delete SQL operations in the Data Source. The Insert, Update and Delete SQL operations , we are using the continuation of the Select command perform by the DataAdapter. That is the DataAdapter uses the Select statements to fill a DataSet and use the other three SQL commands (Insert, Update, delete) to transmit changes back to the Database. From the following links describe how to use SqlDataAdapter and OleDbDataAdapter in detail.

The Select Command property of the DataAdapter is a Command Object that retrieves data from the data source. The Insert Command, Update Command, and Delete Command properties of the DataAdapter are Command objects that manage updates to the data in the data source according to modifications made to the data in the DataSet. From the following links describes how to use SqlDataAdapter and OleDbDataAdapter in detail.

string constring = ConfigurationManager.ConnectionStrings["constr"]. ConnectionString;

using (SqlConnection con = new SqlConnection(constring))

{

using (SqlCommand cmd = new SqlCommand("SELECT Name, City FROM Persons", con))

{

cmd.CommandType = CommandType.Text;

using (SqlDataAdapter sda = new SqlDataAdapter(cmd))

{

DataSet ds = new DataSet();

sda.Fill(ds);}

**Data Grid View**

The DataGridView control available as a part of Windows Forms controls in Visual Studio 2008 is much more powerful than its previous versions. This tutorial is a basic introduction of how to write a simple data driven application using Visual Studio 2010 wizards without writing a single line of code. In my next articles, I will talk about more detailed features of the DataGridView control.

The DataGridView control provides a powerful and flexible way to display data in a tabular format. You can use the DataGridView control to show read-only views of a small amount of data, or you can scale it to show editable views of very large sets of data. This control also lets you display data in a master-details view.

**SQL Data Reader**

A SqlDataReader is a type that is good for reading data in the most efficient manner possible. You can \*not\* use it for writing data. SqlDataReaders are often described as fast-forward firehose-like streams of data.

You can read from SqlDataReader objects in a forward-only sequential manner. Once you’ve read some data, you must save it because you will not be able to go back and read it again.

The forward only design of the SqlDataReader is what enables it to be fast. It doesn’t have the overhead associated with traversing the data or writing it back to the data source. Therefore, if your only requirement for a group of data is for reading one time and you want the fastest method possible, the SqlDataReader is the best choice. Also, if the amount of data you need to read is larger than what you would prefer to hold in memory beyond a single call, then the streaming behaviour of the SqlDataReader would be a good choice.

**Creating a SqlDataReader Object**

Getting an instance of a SqlDataReader is a little different than the way you instantiate other ADO.NET objects. You must call ExecuteReader on a command object, like this:

SqlDataReader rdr = cmd.ExecuteReader();

The ExecuteReader method of the SqlCommand object, cmd, returns a SqlDataReader instance. Creating a SqlDataReader with the new operator doesn’t do anything for you. As you learned in previous lessons, the SqlCommand object references the connection and the SQL statement necessary for the SqlDataReader to obtain data.

**Reading Data**

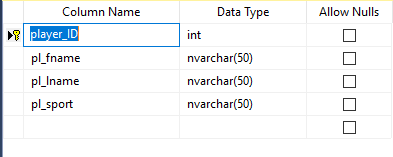
The SqlDataReader returns data iia a sequential stream. To read this data, you must pull data from a table row-by-row Once a row has been read, the previous row is no longer available. To read that row again, you would have to create a new instance of the SqlDataReader and read through the data stream again.

CHAPTER 8

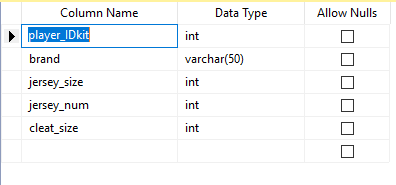
SYSTEM IMPLEMENTATION

**STRUCTURE OF TABLE**

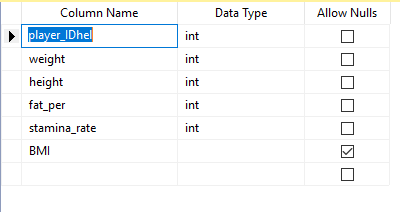
1.Player:



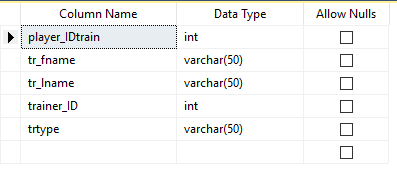
2.Kit:



3.health:

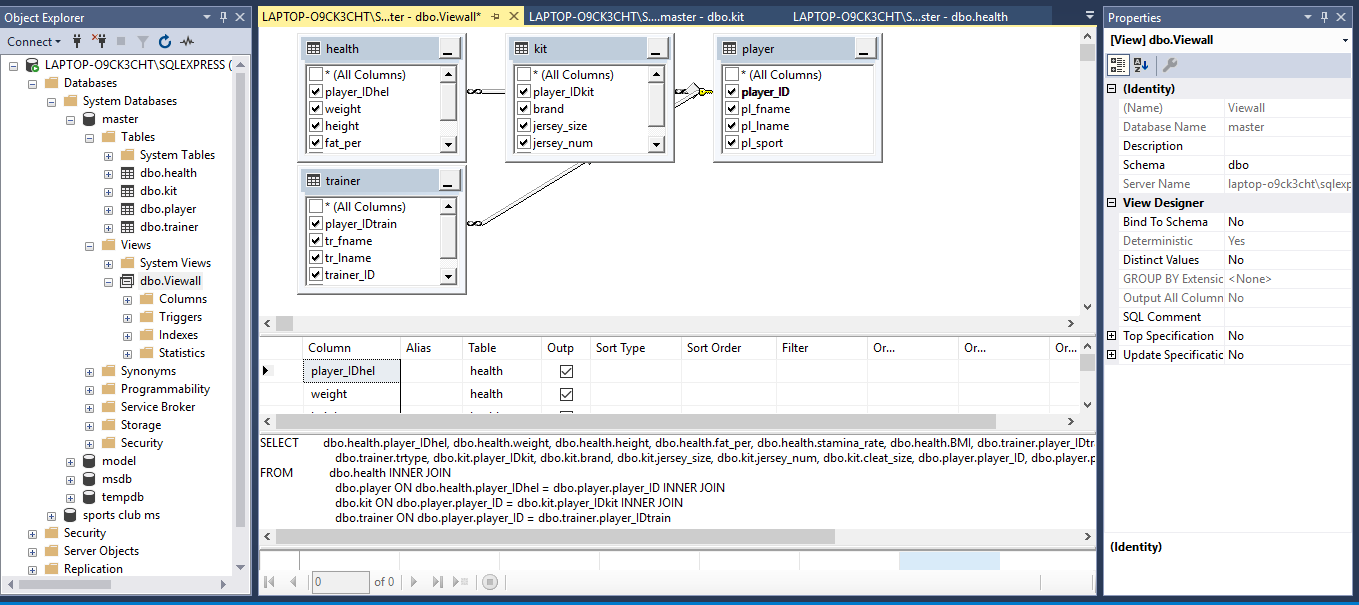


4.Trainer:

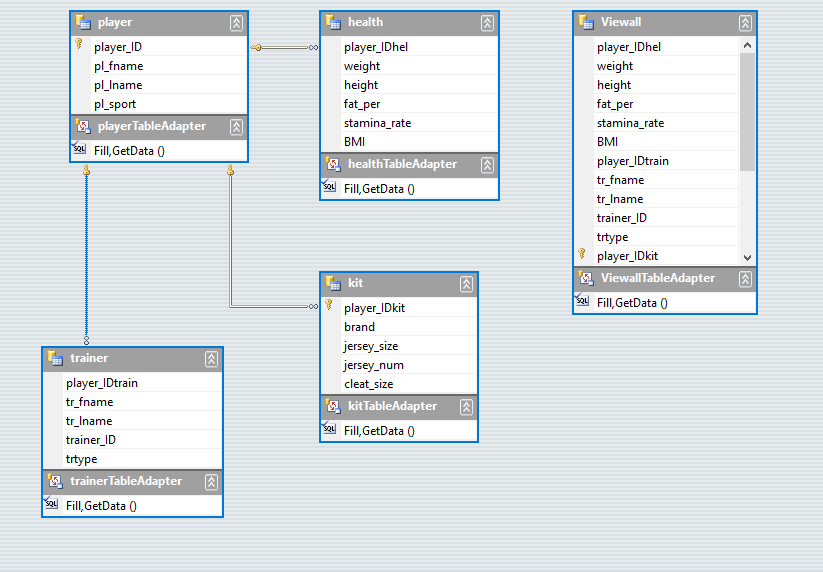


**VIEW**

1. **Viewall**



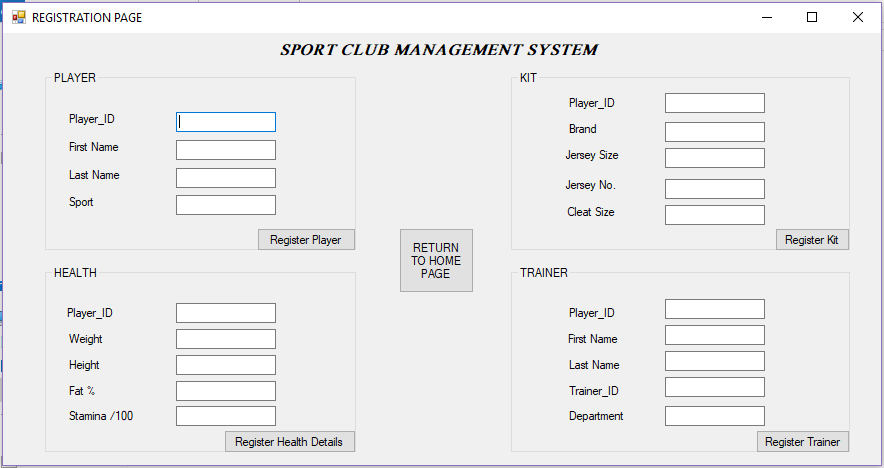
**ER Diagram**



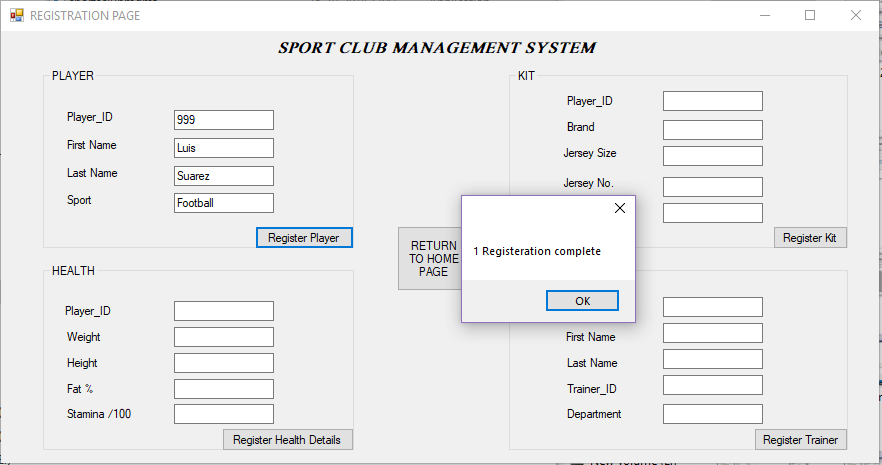
HOME PAGE



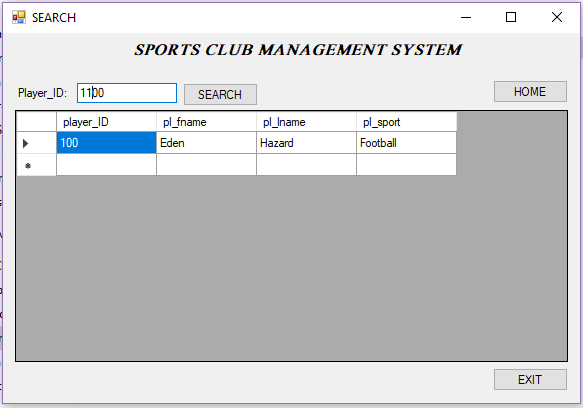
Entering the user details in the Registration Form:



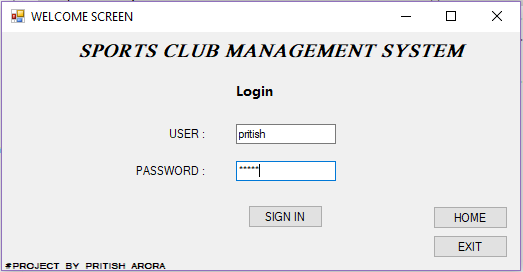
Registration is successful :



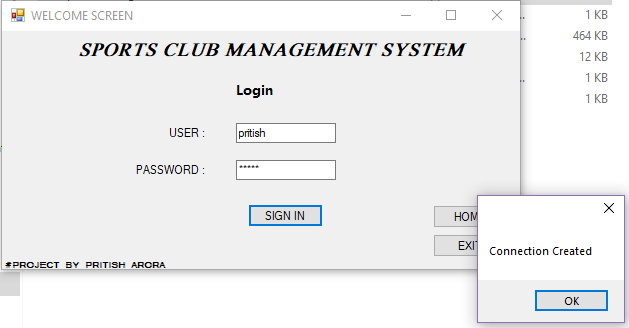
Searching for a player details using ID:



Entering the login credentials on the Authorize Login Form:

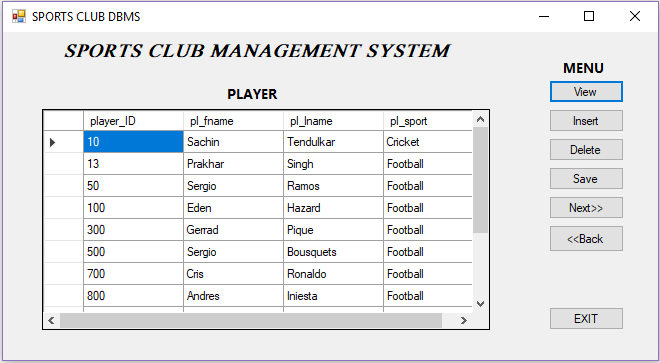


Connection to server successfully created:

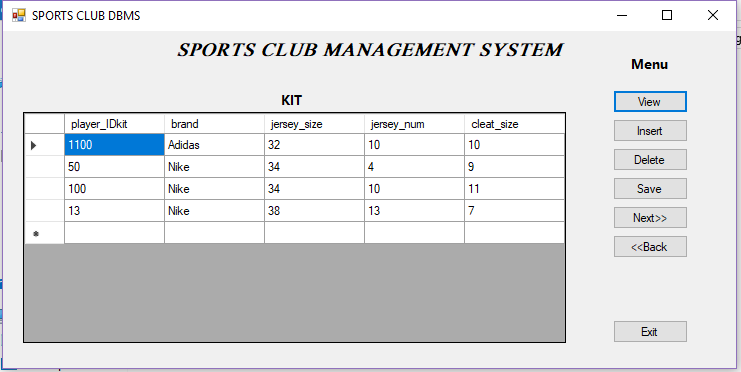


Viewing and manipulating different tables for all entities:

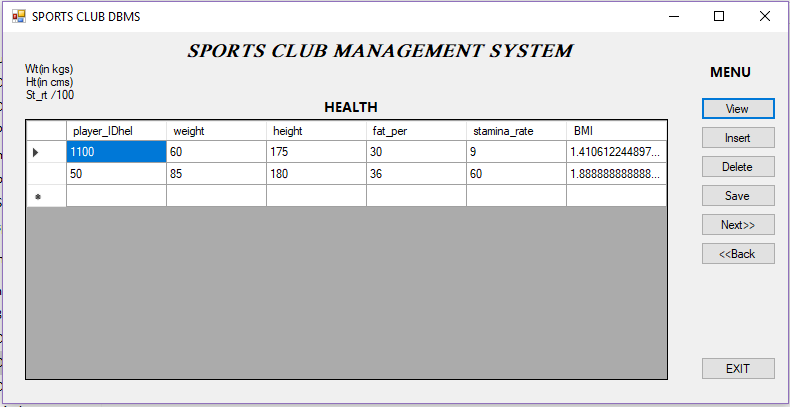
Player:



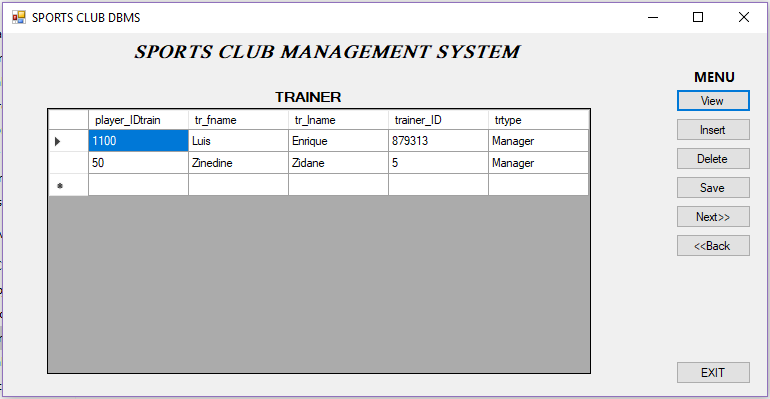
Kit:



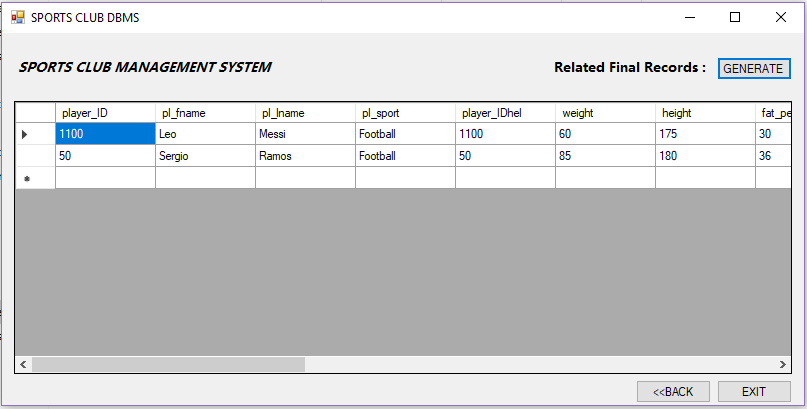
Health:



Trainer:



Master table(Joined relationships):



Search Results:

Reflection in the player table in master database:

