DATABASE CONTROL SYSTEM

By

Capt Muhammad Salman

Capt Muhammad Musa Khan

Submitted to the Faculty of Computer Science
Military College of Signals, National University of Sciences and Technology,
Rawalpindi in Partial Fulfillment for the Requirements of B.E Degree in
Computer Software Engineering
May 2005

ABSTRACT

DATABASE CONTROL SYSTEM

By

Capt Muhammad Salman
Capt Muhammad Musa Khan

There are organizations and institutions with sub units located remotely either in the same city or in different places. These sub units are maintaining their own databases using a variety of database management systems (DBMS) depending upon their own requirements. The authorities at the headquarters require accessing and updating the data of the sub-units frequently. Thus they require data timely and also need to know about the syntax and semantics of the DBMS being used by the sub units. This is a very tedious job. DBCS has been designed to make the job easy for such organizations. It is a web based application and hence one can access the databases from anywhere. The application hides the complexities of DBMS behind one interface so any user can use the software without really having knowledge of the syntax and semantics of the underlying DBMS. The GUI is user friendly. The concept of warehouse has been implemented as well so that all the registered databases can be queried simultaneously. The DBCS is designed for providing maximum help to people required to access data frequently stored in DBMS of different vendors.

DEDICATIONS

This project is dedicated to the instructors of Computer Science Department, Military College of Signals whose patience and love has always been a motivating factor during the course.

ACKNOWLEDGEMENTS

We are thankful to Allah Almighty who gave us courage and wisdom to complete this project. We are thankful to our parent whose prayers have always been with us. We are also very thankful to our family for their love and patience shown during this period of extreme commitment.

We are thankful to our project advisor Lec Aihab Khan for his able guidance and patience. We are grateful to all the faculty members specially Lt Col Naveed Sarfraz Khattak and Asst Prof Tauseef Ahmad Rana for their kind guidance. We are also grateful to Mr Shahid Jamil of FAST institute, Islambabad, for his precious advises and kind guidance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Overview

Just about every part of our lives is controlled by a database. Everything that people do is categorized and persisted and then analyzed and returned back to them as a mail shot or a marketing campaign. If one develop software, then one will find that a database will be somewhere in the solution. Clearly databases are important and a deep knowledge of how to access them effectively and to manipulate the data that they hold is vitally important. Data comes in all shapes and sizes, and there are various different ways of storing data. If someone is lucky enough to work with just one database then he can become an expert with the API to access that database. However, few solutions use a single data source, and even with those that do one cannot guarantee that he will always use that single data source: A client may choose to use a database from a different vendor for a later version of the solution.

The database control system (DBCS) has been designed for the organizations which have sub units located remotely. These sub units are maintaining their own databases. The data is thus distributed. The sub units are using a variety of database management systems (DBMS) depending upon their own requirements. The authorities at the headquarters require accessing the data of the sub-units frequently in order to keep a close watch on their progress and performance. Sometimes they are required to update their database as well. Thus they really require data timely and also need to know about the syntax and semantics of the DBMS being used by the sub units. This is a very tedious job. DBCS has been designed to make the job easy for such organizations. It is a web based application and hence one can access the databases provided he has access to the internet and has the rights to access the data. Moreover the application is designed to hide the complexities of a verity of DBMS behind one interface so any user can use the software without really having knowledge of the syntax and semantics of the underlying DBMS. The GUI is user friendly. The application allows for accessing, querying updating, deleting existing tables and also creating new tables. The compiled data can be stored on any machine as well as sent to the nearest printer. The application is password protected. It can register computers the first time when one visits it so that the next time they are connected to the database automatically. Also it can un-register computers already registered. Moreover it has the capability to block any computer. Moreover as part of the warehouse implementation, it can multicast queries to all the databases, registered with it, simultaneously.

1.2 Aim

The DBCS is designed for providing maximum help to people required to access distributed data frequently stored in DBMS of different vendors.

1.3 Scope

The application covers the database access, manipulation, and querying of Oracle, Microsoft Access (MS Access), SQL Server and converting the data to a common excel format. The data is distributed and the use of a verity of DBMS by the database managers is possible. Also the concept of data warehousing has been implemented.

1.4 Deliverables

At the end of the project, the development team shall be able to deliver the Software, Project documentation and User manual.

Chapter 2

Design of the Application

2.1 Requirement Analysis

There are a number of technologies available for database construction and access. Also there are modules for development of web based database access and manipulation. There was a requirement to develop software which can handle a variety of software within the same interface. Moreover the software is aimed at making the DBMS transparent to the user so that he doesn't have to learn a variety of DBMS. This makes it easy for the administration to change the DBMS. More over the application was required to be web based so that access to data is very easy and is in line with the future developments in this field.

2.2 Working of DBCS

The basic idea behind the software is that a user sitting in front of the interface can communicate with databases stored in a variety of DBMS. User is not required to learn the tools of each individual DBMS. User is required to click the mouse and the Structured Query Language Engine (SQL Engine) will generate the query. The mapping module will map the query to the DBMS and the communication module will communicate the query to the DBMS. The result will be communicated to the user by the communication module. Figure 2.1 shows the working of the software.

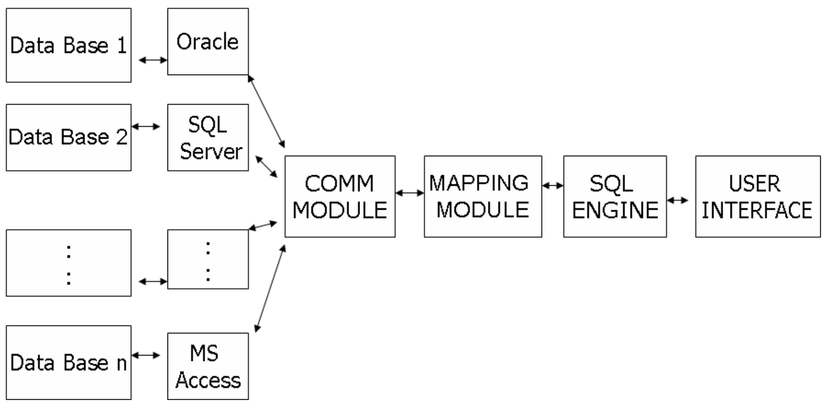


Figure 2.1 Working of the DBCS

2.3 Design of the Software.

The design of the software was a very delicate and tricky issue since this had a direct impact on the performance of the software. Moreover it also affected the choice of technology including the choice of the programming language. Moreover the design of the software was required to be modular since it was required to be extensible to allow flexibility. This would allow the software to be modified to meet any specific requirement. After careful and detailed studies it was decided that the design of the software should be composed of three layers. The First layer composed of the front-end and is the interface which is used by the users. It was required to be very simple informative and user friendly. The back-end is the platform used for storing the data and the choice rest with the database manager for selecting and using a suitable DBMS. The middle layer is the programming language which maps the user actions to the DBMS operations. Figurer 2.1 shows the three Layers of the development.

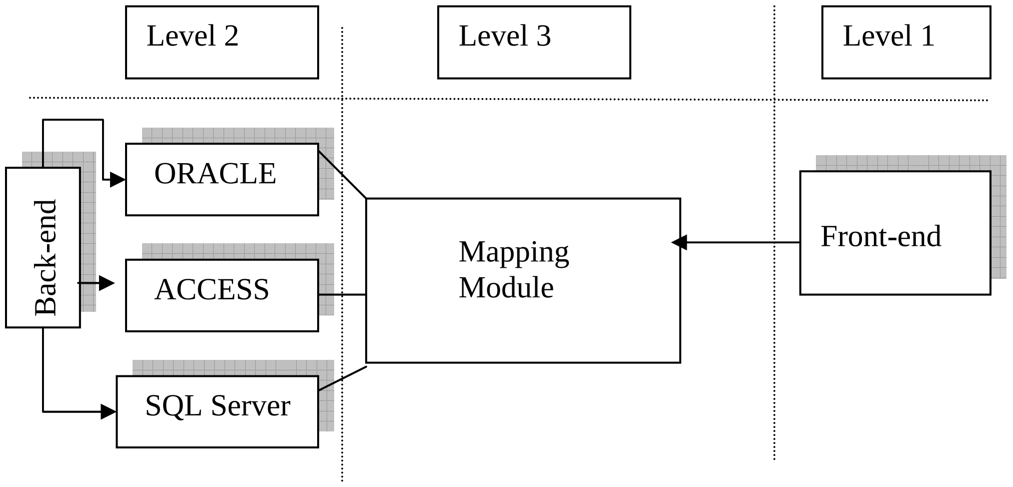


Figure 2.2 Layers of Development

2.3.1 Front-end

The front-end is the user interface and it needed to be very user friendly. The front-end hides the details of the DBMS and the programming language from the user. It is this level to which he is exposed to. Therefore it needed to be simple and required a very careful study for choice of the development technology. There were many choices to include Java, HTML, XML and ASP. All had their own advantages and shortcomings and needed a careful deliberation.

2.3.2 Back-end

At the back-end is the data stored in a DBMS. There is a variety of DBMS available for development of the back-end. Most important of these are the MS Access, SQL Server and Oracle. The choice of selecting a suitable DBMS wholly rests with the database manager. The choice of DBMS however depends upon the nature of the security required and the amount of the data involved. MS Access is not a secure database and any important information cannot be stored there. Oracle and SQL Server are very secure to store classified data. Both do have multiple security checks. So the choice of back end is totally the choice of the database manager and user really has no hand in it.

2.3.3 Programming Language.

This is also a very important aspect of software design. It has a direct bearing upon the efficiency and performance of the software. There are a variety of programming languages available for use in such software. The obvious choices being Java, C++, VB, C# etc. The important aspect to be kept in mind is whether the programming language will add to the efficiency of the front-end development tool or it will have any negative impact.

2.4 Technology Used.

Keeping in view the requirements of the software a detailed study was carried out and it was found that the Microsoft‘s emerging technology, the ‘Visual Studio.NET’ which is mostly spelled as .NET was a perfect platform for this purpose. Once the choice of the platform was over an important choice to make was the programming language. This is no big deal in the context of .NET but still has to do with the programming skills of the developer. After a lot of deliberations it was decided that following technology will be used.

2.4.1 Front-end Development

The best choice for the development of the front is the .NET’s web developing technology, the ‘ASP.NET. ASP stands for ‘active server pages’. ASP.NET is an updated form of the ASP. In fact Microsoft developed this language for web design. ASP.NET will be described in detailed later in the report.

2.4.2 Back-end Development

DBCS is going to support three DBMS for the time being. Since the team is using the modular programming so it is hoped that if need arise one can add more DBMS. For data, not so important, one normally uses the MS Access. For classified data organizations are going to use either SQL Server or Oracle. The latter two are the most secure DBMS.

2.4.3 Programming Language Used.

The choice of Programming language is of less importance in the .NET technology but since it has a bearing on the programmer’s skills one has to be careful in this respect also. .NET supports visual Java, visual C++, visual Basic and visual C#. Our obvious choice is C# for the reason that its syntax is very close to Java and any person who is familiar with Java and C++ can easily shift to this Programming language.

2.5 Summary

The design phase is the most important phase in the development of the software. Not only that the architecture of the application is decided but also the technology to be used is also considered here. The software has been developed a three level architecture. After giving consideration to the available technologies, the use of the most powerful and fast growing Microsoft technology has been used. This technology is known as Microsoft Visual Studios .NET which is mostly referred to as .NET.

Chapter 3

Review of Literature

3.1 Overview

While the DBCS team carried out extensive study of the design and the technology, it was tried to consult various people and experts in the field. At the same time the team remained in touch with the books as which proved to be a very good source of knowledge. At the same time the team visited various web sites to gather invaluable information in this regard.

3.2 ASP.NET Database Programming Weekend Crash Course

Written by Jason Butler and Tony Caudill, this book introduces some very important aspects of .NET technology in a brief and precise manner. It is basically written for those programmers who want to learn this technology in a short time. It is therefore a book which gives the basic details of .NET.

3.3 Programming Microsoft.NET

Written by Prosise Jeff, this book has introduced the ASP.NET in very simple and illustrious way. It is more detailed book than the one discussed in section 3.2. In this book, the author has amply described the advanced techniques and has given examples.

3.4 SAMS Teach Yourself .NET Windows Forms in 21 Days

This book of SAMs series is really a good book. It introduces the basic concepts of the .NET technology in a very clear way and explains the entire step required to carry out any operation with examples. One can make the basic concepts very clear by reading this book and following the steps given in the examples. It is a very good book for the naïve programmers. This book also discusses the basics of the .NET technology. This book is very helpful in learning the .NET technology in a very short time. However advanced programmers are not advised to read this book as it does not enhance their knowledge.

3.5 Essential ADO.NET

By Bob Beauchemin explains the database access part of the .NET framework. ADO stands for active data objects and they are responsible for accessing the data stored in the database.

This book has been written for relatively new programmers. The book has described various connections and is full of examples which make the job of learners very easy.

3.6 ASP.NET by Example

Steven A. Smith is a Managing Consultant and Internet Specialist for Software Architects, Inc., a Chicago-based consulting firm. He has explained ASP.NET in light of his own experience. ASP stands for “active server pages” and it is used for the development of the front-end. The author has explained the use of various functions by giving examples. This book is ideally suited to the programmers who have no prior knowledge of this technology.

3.7 Professional ASP.NET 1.0 Special Edition

This book of the Wrox series has been written by five very well known persons and gives in depth knowledge of the ASP.NET. This book has described advanced topics in programming with the ASP.NET. Like all other Wrox series books, this book has described the topics in detail and has included ample number of examples. Programmers, who want to master this technology, must read this book.

3.8 Dietel and Dietel - C Sharp How to program.

C#, pronounced as “C Sharp” is one of the programming languages supported by the .NET platform. This book introduces the .NET environment and teaches C# in a very efficient way. Dietel father and son have written many good books on different programming languages. This book is also a very good one. Like the other books, this book has covered all the topics of the language in depth and has included a very large number of examples. This book will help new programmers as well as those who have fair bit of knowledge about this language.

3.9 www.google.com

A very well known search site and has ample of helping material on the issue. One can search any topic related to database and .NET technology. Also one can find demonstration projects which enable the programmers to make suitable changes to their programming style. Apart from this one can find many people who can be asked to give their expert opinion in case the programmers find any difficulty.

3.10 www.Microsoft.com

The official website of the Microsoft has very valuable information about all the products of the Microsoft Company. On this website one can read the documentation, get the answers to the frequently asked questions and can ask the help of expert people. This site has been designed for providing on line help to the people using products developed by Microsoft incorporation.

3.11 MSDN library

This come with the .NET product and is a very good source of information provided by Microsoft. This library is part of the visual studio .NET documentation. It is available in three compact disks and can be installed at the time of installing the visual studio .NET. This library can answer almost all queries pertaining to the visual studio .NET technology. The users of the visual studio .NET technology are advised to make full use of this source of information.

3.12 Summary

Review of literature gives the developers a broad spectrum of design techniques and development tools. In view of these obvious benefits a thorough study of various literature was carried out which helped in the overall design of the application. In the light of the studies carried out the choice of technology was decided and three level architecture was used for the development of the application software.

Salient Features of the software.

4.1 Overview of the Software

The software has been designed to meet the requirements of the authorities who require frequent access to the data bases of their sub-units. These data bases are maintained by different people and the data is stored in different DBMS. The application has been designed to be web based so the user can access the require data from anyware provided he has access to the internet. The software has been designed and developed by using the most modern Microsoft technology. Though much similar software are available in the market, the DBCS has some very good features which make it an obvious choice for organizations/ institutions who have data distributed at remote places store in different DBMS.

4.2 Salient Features of the Software

The software has some very good features which make it a viable option for organizations and institutions with data distributed over variety of DBMS. Some of the salient features are discussed below.

4.2.1 Web Based

The application has been designed to be web based so that it can meet the requirement of the users who need to access and manipulate data frequently these data are store in a verity of DBMS and are distributed. By placing the application on the web the user can access the applications to access and manipulate data from any where he has access to the net. This will allow authorities at the headquarters to get data of there sub-units located in remote areas in time and as per on their requirements. This is very important in cases where sub-units are located in backward areas and timely flow of information is not possible continuously due to disruption of communication channel**.**

4.2.2 Transparency of DBMS.

The DBCS has been designed to work with a verity of DBMS. The user is not requiring learning the intricacies of the DBMS. The interface is user friendly and it helps the user in performance of his job. The user is required only that he should know the location, type and name of the data base. If user has the required permission to access the data, he should register the data bases making use of the user interface. This registration of the data base is required to be carried out only for the first time. Ones a data base has been registered, the applications registers all the required parameters in a log file. Ones the user access the same data base, he is connected to the data base automatically. The user is not aware of the actions being performed at the back end. User is only seeing the operations at the front end so the DBMS operations are hidden from him.

4.2.3 Data Export.

The application has been designed in such a way that it can export the data in excel format. The data may be stored in any DBMS, the application view the data in table format and can save it on any machine in excel format. This characteristic of the software is very important since we are dealing with data store in verity of DBMS. A common format for export of data will help in compilation of required results to be printed in a presentable form.

4.2.4 Database Security.

DBCS has used the .NET technology. This technology is very strong and its lays great importance on the security of data. The administrator is responsible for creating and users and providing login and passwords. At the same time the ASP.NET has its own security mechanism which comes into play and it uses its built-in methods for user authentication and user authorization. These two levels of security ensure that the data is not modified unauthorizely. The ASP.NET is also in the forefront of the DBCS to ensure that the data is exported securely.

4.2.5 User Friendly Interface.

The GUI is very user friendly. It guides the user in his functions of data access and data manipulation. Even a person who does not know anything about the application can be made to work well in a few hours time. The interface displays messages to tell the user that the connection has been successful or failed, the data manipulation operations are performed well or there was an error and so on. The application keeps a log of the registered databases and it does not bother the user to register a database which has already been registered.

4.2.6 Querying the Database.

The user writes his query in a simple way and the interface maps it to the desired database. Most of the common data manipulation queries are mapped and the user has to click the button only. Instead of writing a lengthy SQL query, the user has only to click insert button and select the table. The query is written for the user by the application. Thus naïve user can query verity of databases to get the desire info.

4.2.7 Manipulation of Table

With the help of the interface user can create and delete tables in any DBMS. On clicking create table button, the interface guides the user to set the table column and rows. Then user can insert delete or update a values of column. This requires a lengthy piece of code to be written in the query language it is very hard to remember the query language for all the DBMS. In case of DBCS he query is generated at the back on the click of mouse hence the user is free to divert his attention towards creation of a table with correct values instead of wasting his energies in writing lengthy SQL queries.

4.2.8 Modular Programming.

The programming is modular. This ensues that the application can be modified to meet special requirement of a user. New DBMS can be added or existing DBMS can be deleted. The application can be modified to allow any special query or data manipulation.

4.2.9 Data Warehouse.

The concept of data warehouse has been also implemented in this software. By data warehouse we mean data stored in various data bases at different times using different DBMS. For retrieving data from data ware house we put forward query of the data bases registered with the user. The query is multicast to all the registered DBMS and the required result is obtained

4.3 Analysis of the Software

Though many similar software are available for accessing and manipulating data base. Many of the software are web base as well. The DBCS has the advantage over other software. Firstly, the DBCS deals with a verity of DBMS through the same interface. Secondly, it makes use of the power of .NET. Thirdly, it implements the concept of data ware house. Fourthly, it allows the user to query the data base on the click of the mouse. Fifth, it brings together many characteristics which are found in different software. Last, DBCS is very user friendly and cost effective.

4.4 Summary

The software has been designed to meet the requirements of the users who require frequent access to the data stored in distributed bases. These data bases are maintained by different people and the data is stored in different DBMS. The application has been designed to be web based so the user can access the require data from anywhere provided he has access to the internet. The software has been designed and developed by using the most modern Microsoft technology. Though much similar software are available in the market, the DBCS has some very good features which make it an obvious choice for organizations/ institutions who have data distributed at remote places store in different DBMS.

Chapter 5

Visual Studio.NET

5.1 Overview

The term “Microsoft .NET” refers to a massive effort on Microsoft’s part to get away from traditional software development and to build—with help from partners all over the industry—the Internet into a service-oriented software platform. Web services and Web service clients use XML to exchange data. As XML Web services proliferate, the Internet will become a software platform with an API far richer than any operating system. Today’s applications rely primarily on operating system services. Tomorrow’s applications will use Web services to validate credit card purchases, check the status of airline flights, and perform other everyday tasks.

5.2 .NET Framework

The .NET Framework is a platform for building and running applications. Its chief components are the common language runtime (CLR, or simply “the runtime”) and the .NET Framework class library (FCL). The CLR abstracts operating system services and serves as an execution engine for managed applications—applications whose every action is subject to approval by the CLR. The FCL provides the object-oriented API that managed applications write to. When user write .NET Framework applications, he leave behind the Windows API and other tools and technologies he is familiar with, and use the FCL instead. Sure, he can call a Windows API function or a COM object if he want to, but he don’t want to because doing so requires transitioning from managed code (code run by the CLR) to unmanaged code (native machine code that runs without the runtime’s help). Such transitions impede performance and can even be vetoed by a system administrator.

Microsoft .NET is chiefly about XML Web services, but the .NET Framework supports other programming models as well. In addition to writing Web services, user can write console applications, GUI applications (“Windows Forms”), Web applications (“Web Forms”), and even Windows services, better known as NT services. The framework also helps user consume Web services—that is, write Web service clients. Applications built on the .NET Framework are not, however, required to use Web services.

Next to XML Web services, the portion of the framework with the greatest potential to change the world is ASP.NET. The name comes from Active Server Pages (ASP), which revolutionized Web programming in the 1990s by providing an easy-to-use model for dynamically producing HTML content on Web servers using server-side script. ASP.NET is the next version of ASP, and it provides a compelling new way to write Web applications that's unlike anything that has preceded it. Because ASP.NET is such an important part of the framework, a complete chapter of this document is devoted to it. The key to understanding the .NET Framework and the various programming models it supports is to understand the common language runtime and the FCL.

5.3 The Common Language Runtime

If the .NET Framework were a living, breathing human being, the common language runtime would be its heart and soul. Every byte of code that user write for the framework either runs in the CLR or is given permission by the CLR to run outside the CLR. Nothing happens without the CLR's involvement. The CLR sits atop the operating system and provides a virtual environment for hosting managed applications. When user run a managed executable, the CLR loads the module containing the executable and executes the code inside it. Code that targets the CLR is called managed code, and it consists of instructions written in a pseudo-machine language called common intermediate language, or CIL. CIL instructions are just-in-time (JIT) compiled into native machine code (typically x86 code) at run time. In most cases, a given method is JIT compiled only one time—the first time it's called—and thereafter cached in memory so that it can be executed again without delay. Code that isn't called is never JIT compiled. While JIT compilation undeniably impacts performance, its negative effects are mitigated by the fact that a method is compiled only once during the application's lifetime and also by the fact that the CLR team at Microsoft has gone to extraordinary lengths to make the JIT compiler as fast and efficient as possible. In theory, JIT compiled code can outperform ordinary code because, the JIT compiler can optimize the native code that it generates for the particular version of the host processor that it finds itself running on. JIT compiled code is not the same as interpreted code.

The benefits of running code in the managed environment of the CLR are legion. For starters, as the JIT compiler converts CIL instructions into native code, it enacts a code verification process that ensures the code is type safe. It's practically impossible to execute an instruction that accesses memory that the instruction isn't authorized to access. User will never have problems with stray pointers in a managed application because the CLR throws an exception before the stray pointer is used. User can't cast a type to something it's not because that operation isn't type safe. And user can't call a method with a malformed stack frame because the CLR simply won't allow it to happen. In addition to eliminating some of the most common bugs that afflict application programs, code verification security makes it eminently more difficult to write malicious code that intentionally inflicts harm on the host system. In the unlikely event that user don't want code verification security, he can have it turned off by a system administrator.

Code verification security is also the chief enabling technology behind the CLR's ability to host multiple applications in a single process—a feat of magic that it works by dividing the process into virtualized compartments called application domains. Windows isolates applications from one another by hosting them in separate processes. An unfortunate side effect of the one-process-per-application model is higher memory consumption. Memory efficiency isn't vital on a stand-alone system that serves one user, but it's paramount on servers set up to handle thousands of users at once. In certain cases (ASP.NET applications and Web services being the prime examples), the CLR doesn't launch a new process for every application; instead, it launches one process or a handful of processes and hosts individual applications in application domains. Application domains are secure like processes because they form boundaries that managed applications can't violate. But application domains are more efficient than processes because one process can host multiple application domains and because libraries can be loaded into application domains and shared by all occupants.

Another benefit of running in a managed environment comes from the fact that resources allocated by managed code are garbage collected. In other words, users allocate memory, but don't free it; the system frees it. The CLR includes a sophisticated garbage collector that tracks references to the objects the code creates and destroys those objects when the memory they occupy is needed elsewhere. Thanks to the garbage collector, applications that consist solely of managed code don't leak memory. Garbage collection even improves performance because the memory allocation algorithm employed by the CLR is fast—much faster than the equivalent memory allocation routines in the C runtime. The downside is that when a collection does occur, everything else in that process stops momentarily. Fortunately, garbage collections occur relatively infrequently, dramatically lessening their impact on performance.

5.4 Programming Language

Yet another benefit of running applications in a CLR-hosted environment is that all code reduces to CIL, so the programming language that users choose is little more than a lifestyle choice. The "common" in common language runtime alludes to the fact that the CLR is language-agnostic. In other environments, the language users use to implement an application inevitably affects the application's design and operation. Code written in Visual Basic 6, for example, can't easily spawn threads. To make matters worse, modern programming languages such as Visual Basic and Visual C++ use vastly different APIs, meaning that the knowledge users gain programming Windows with Visual Basic is only marginally helpful if users write a DLL in C++. With the .NET Framework, all that changes. A language is merely a syntactic device for producing CIL, and with very few exceptions, anything users can do in one language, can be done in all the others, too. Moreover, regardless of what language they're written in, all managed applications use the same API: that of the .NET Framework class library. Porting a Visual Basic 6 application to Visual C++ is only slightly easier than rewriting the application from scratch. But porting a Visual Basic .NET application to C# (or vice versa) is much more reasonable. In the past, language conversion tools have been so imperfect as to be practically useless. In the era of the .NET Framework, someone might write a language converter that really works. Because high-level code ultimately compiles to CIL, the framework even lets users write a class in one language and use it (or derive from it) in another.

Microsoft provides CIL compilers for five languages: C#, J#, C++, Visual Basic, and JScript. The .NET Framework Software Development Kit (SDK) even includes a CIL assembler called ILASM, so users can write code in raw CIL if want to. Third parties provide compilers for other languages, including Perl, Python, Eiffel, and, even COBOL. No matter what language users are most comfortable with, chances are there’s a CIL compiler that supports it. And even if users prefer COBOL, they can do almost anything those snobby C# programmers can do.

5.5 Managed Modules

When users build a program with the C# compiler, the Visual Basic .NET compiler, or any other compiler capable of generating CIL, the compiler produces a managed module. A managed module is simply an executable designed to be run by the CLR. It typically, but not always, has the file name extension EXE, DLL, or NETMODULE. Inside a managed module are four important elements: One, a Windows Portable Executable (PE) file header. Two, a CLR header containing important information about the module, such as the location of its CIL and metadata. Three, metadata describing everything inside the module and its external dependencies. Four, the CIL instructions generated from the source code

Every managed module contains metadata describing the module’s contents. Metadata is not optional; every CLR-compliant compiler must produce it. That’s important, because it means every managed module is self-describing. Think about it this way. If someone hands users an ordinary EXE or DLL today, can users easily crack it open and Figure out what classes are inside and what members those classes contain? No way! If it’s a managed module, however, no problem. Metadata is like a COM type library, but with two important differences: First, type libraries are optional; metadata is not. And second metadata fully describes a module; type libraries sometimes do not.

Metadata is important because the CLR must be able to determine what types are present in each managed module that it loads. But it’s also important to compilers and other tools that deal with managed executables. Thanks to metadata, Visual Studio .NET can display a context-sensitive list of the methods and properties available when users type the name of a class instance into the program editor, a feature known as IntelliSense. And thanks to metadata, the C# compiler can look inside a DLL containing a class written in Visual Basic .NET and use it as the base class for a derived class written in C#.

5.6 Common Intermediate Language

CIL is often described as a pseudo-assembly language because it defines a native instruction set for a processor. In this case, however, the processor is the CLR, not a piece of silicon. You don't have to know CIL to program the .NET Framework any more than you have to know x86 assembly language to program Windows. But a rudimentary knowledge of CIL can really pay off when a method in the FCL doesn't behave the way you expect it to and you want to know why. You don't have the source code for the FCL, but you do have the CIL.

In all, CIL includes about 100 different instructions. Some are the typical low-level instructions found in silicon instruction sets, like instructions to add two values together (ADD) and to branch if two values are equal (BEQ). Other instructions work at a higher level and are a typical of those found in hardware instruction sets. For example, NEWOBJ instantiates an object, and THROW throws an exception. Because the CIL instruction set is so rich, code written in high-level languages such as C# and Visual Basic .NET frequently compiles to a surprisingly small number of instructions.

CIL uses a stack-based execution model. Whereas x86 processors load values into registers to manipulate them, the CLR loads values onto an evaluation stack. To add two numbers together, one has to copy them to the stack, call ADD, and retrieve the result from the stack. Copying a value from memory to the stack is called loading, and copying from the stack to memory is called storing. CIL has several instructions for loading and storing. LDLOC, for example, loads a value onto the stack from a memory location, and STLOC copies it from the stack to memory and removes it from the stack.

5.7 The .NET Framework Class Library

Windows programmers, who code in C, tend to rely on the Windows API and functions in third-party DLLs to get their job done. C++ programmers often use class libraries of their own creation or standard class libraries such as MFC. Visual Basic programmers use the Visual Basic API, which is an abstraction of the underlying operating system API. Using the .NET Framework means one can forget about all these anachronistic APIs. Users have a brand new API to learn, that of the .NET Framework class library, which is a library of more than 7,000 types—classes, structs, interfaces, enumerations, and delegates (type-safe wrappers around callback functions)—that are an integral part of the .NET Framework. Some

FCL classes contain upward of 100 methods, properties, and other members, so learning the FCL isn't a chore to be taken lightly. The bad news is that it's like learning a brand new operating system. The good news is that every language uses the same API, so if you decide to switch from Visual Basic to C++ or vice versa, the hard work users make in learning the FCL isn't lost.

To make learning and using the FCL more manageable, Microsoft divided the FCL into hierarchical namespaces. The FCL has about 100 namespaces in all. Each namespace holds classes and other types that share a common purpose. For example, much of the window manager portion of the Windows API is encapsulated in the System.Windows.Forms namespace. In that namespace you'll find classes that represent windows, dialog boxes, menus, and other elements commonly used in GUI applications. A separate namespace called System.Collections holds classes representing hash tables, resizable arrays, and other data containers. Yet another namespace, System.IO, contains classes for doing file I/O. Users will find a list of all the namespaces present in the FCL in the .NET Framework SDK's online documentation. Their job as a budding .NET programmer is to learn about those namespaces. Fortunately, the FCL is so vast and so comprehensive that most developers will never have to tackle it all.

Table 5.1 lists a few of the FCL's namespaces and briefly describes their contents. The term "et al" refers to a namespace's descendants. For example, System.Data et al refers to System.Data, System.Data.Common, System.Data.OleDb, System.Data.SqlClient, and System.Data.SqlTypes.

Table 5.1 The .NET Class Libraries

Namespace	**Contents**
System	Core data types and auxiliary classes
System.Collections	Hash tables, resizable arrays, and other containers
System.Data et al	ADO.NET data access classes
System.Drawing	Classes for generating graphical output (GDI+)
System.IO	Classes for performing file and stream I/O
System.Net	Classes that wrap network protocols such as HTTP
System.Reflection et al	Classes for reading and writing metadata
System.Runtime.Remoting et al	Classes for writing distributed applications
System.ServiceProcess	Classes for writing Windows services

System.Threading	Classes for creating and managing threads
System.Web	HTTP support classes
System.Web.Services	Classes for writing Web services
System.Web.Services.Protocols	Classes for writing Web service clients
System.Web.UI	Core classes used by ASP.NET
System.Web.UI.WebControls	ASP.NET server controls
System.Windows.Forms	Classes for GUI applications

The first and most important namespace in the FCL—the one that every application uses—is System. Among other things, the System namespace defines the core data types employed by managed applications. The System namespace is also home for many of the exception types defined in the FCL (for example, InvalidCastException) and for useful classes such as Math, which contains methods for performing complex mathematical operations; Random, which implements a pseudo-random number generator; and GC, which provides a programmatic interface to the garbage collector.

5.8 Summary

To summarize, the .NET Framework is a platform for Web services and other kinds of applications. Applications that target the .NET Framework are managed applications. They're made of CIL and metadata, and they're JIT compiled at run time and executed by the CLR. Languages such as C# and Visual Basic .NET are syntactic tools for generating CIL. For the first time in programming history, language is unimportant because, at the end of the day, all languages exercise the same set of features in the CLR and FCL.

Ironically, the .NET Framework grew out of an effort by a lot of smart people at Microsoft to make COM programming easier. Little did they know that the solution they'd come up with wasn't to fix what was broken, but to tear it down and start over again. That's what the .NET Framework is: a new beginning. The sooner you can let go of the old ways of writing and executing code, the more quickly you'll adapt to the .NET way.

Chapter 6

Accessing the Database

6.1 Overview of the ADO.NET

Access to the database is carried out using the active data objects .NET (ADO.NET). ADO.NET refers to a set of classes that ship with Visual Studio .NET that allows developers to access data typically stored in relational databases. ADO.NET allows interacting with a database directly using objects of the *managed provider* classes. These objects allow, to connect to the database and execute, SQL statements while directly connected to the database. It also allows working in a disconnected manner. When doing this, store information is stored from a database locally in the memory of the computer on which user program is running. Users store that information using objects of the *data set* classes. Once users have that information in the memory, they can then read and manipulate that information. For example, they can display the columns for the rows, add new rows, modify rows, and delete rows. Periodically, they'll reconnect to the database to synchronize changes made locally with the database. This disconnected model allows users to write applications that run on the Internet, as well as for devices that aren't always connected to the database-PDAs such as the Palm and the Pocket PC, for example. This capability to store a local copy of rows retrieved from the database is one of the main strengths of ADO.NET.

6.2 The Managed Provider and Generic Data Set Classes

To provide both connected and disconnected database access, ADO.NET defines two sets of classes: managed provider and generic data. Users use objects of the managed provider classes to directly connect to a database and to synchronize the locally stored data with the database. Users can use the managed provider classes to read rows from the database in a forward-only direction. Users use a different set of managed provider classes depending on the database in use. They use objects of the generic data classes to store a local copy of the information retrieved from the database. This copy is stored in the memory of the computer where the C# program is running. The main generic data class is the System.Data.DataSet class. The generic data classes, as their name suggests, are not specific to any database, and they always use the same classes regardless of the database you use. The generic data classes represent information retrieved from the database as XML.

6.3 The Managed Provider Classes

The managed provider objects allow users to directly access a database. They use the managed provider objects to connect to the database and read and write information to and from the database. Figure 6.1 illustrates some of the managed provider objects and how they relate to each other.

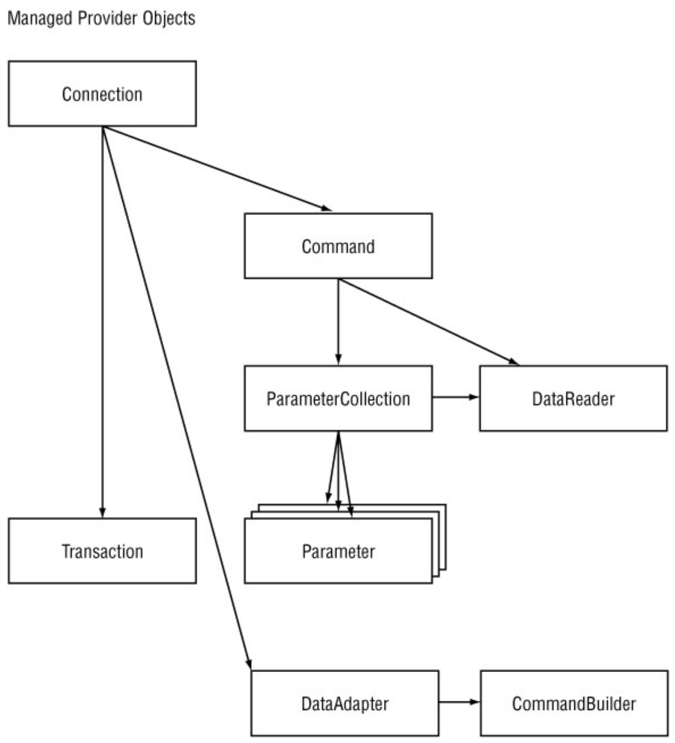


Figure 6.1: Some of the Managed Provider Objects

There are currently three sets of managed provider classes, and each set is designed to work with different database standards. These three sets of classes all implement the same basic functionality: First is *SQL Server Managed Provider* Classes which users use the SQL Server managed provider classes to connect to a SQL Server database. Second, *OLE DB Managed Provider* Classes, which they use the OLE DB (Object Linking and Embedding for

Databases) managed provider classes to connect to any database that supports OLE DB, such as Access or Oracle. Third, *ODBC Managed Provider* Classes, which is used with the ODBC (Open Database Connectivity), managed provider classes to connect to any database that supports ODBC. All the major databases support ODBC, but ODBC is typically slower than the previous two sets of classes when working with .NET. Users should use the ODBC managed provider classes only when there aren't any alternative OLE DB managed provider classes. Some of the Managed Provider Classes are described in the next lines.

The *Connection* Classes are used to make connections. There are three Connection classes: SqlConnection, OleDbConnection, and OdbcConnection. Object of the SqlConnection class is used to connect to a SQL Server database. Object of the OleDbConnection class is used to connect to any database that supports OLE DB, such as Access or Oracle. Object of the OdbcConnection class is used to connect to any database that supports ODBC. Ultimately, all communication with a database is done through a Connection object. Next are the Command Classes**.** There are three Command classes: SqlCommand, OleDbCommand, and OdbcCommand. A Command object is used to run a SQL statement, such as a SELECT, INSERT, UPDATE, or DELETE statement. Users can also use a Command object to call a stored procedure or retrieve rows from a specific table. They run the command stored in a Command object using a Connection object. Then the *Parameter* Classes. There are three Parameter classes: SqlParameter, OleDbParameter, and OdbcParameter. A Parameter object is used to pass a parameter to a Command object. A Parameter is used to pass a value to a SQL statement or a stored procedure call. User can store multiple Parameter objects in a Command object through a ParameterCollection object. *The ParameterCollection Classes.* There are three ParameterCollection classes: SqlParameterCollection, OleDbParameterCollection, and OdbcParameterCollection. Users use a ParameterCollection object to store multiple Parameter objects for a Command object. The *DataReader* Classes. There are three DataReader classes: SqlDataReader, OleDbDataReader, and OdbcDataReader. A DataReader object is used to read rows retrieved from the database using a Command object. DataReader objects can only be used to read rows in a forward direction. DataReader objects act as an alternative to a DataSet object. One cannot use a DataReader to modify rows in the database. Next comes the The *DataAdapter* Classes. There are three DataAdapter classes: SqlDataAdapter, OleDbDataAdapter, and OdbcDataAdapter. A

DataAdapter object is used to move rows between a DataSet object and a database. A DataAdapter is used object to synchronize your locally stored rows with the database. This synchronization is performed through a Connection object. For example, users can read rows from the database into a DataSet through a DataAdapter, modify those rows in DataSet, and then push those changes to the database through a Connection object. Next in the list is the *CommandBuilder* Classes. There are three CommandBuilder classes: SqlCommandBuilder, OleDbCommandBuilder, and OdbcCommandBuilder. A CommandBuilder object is used to automatically generate single-table INSERT, UPDATE, and DELETE commands that synchronize any changes made to a DataSet object with the database. This synchronization is performed through a DataAdapter object. The transaction classes are used to carryout transactions. There are three Transaction classes: SqlTransaction, OleDbTransaction, and OdbcTransaction. Users use a Transaction object to represent a *database transaction*. A database transaction is a group of statements that modify the rows in the database. These statements are considered a logical unit of work. For example, in the case of a banking transaction, one might want to withdraw money from one account and deposit it into another. He would then commit both of these changes as one unit, or if there’s a problem, roll back both changes. Namespaces for the Managed Provider Classes have three categories. The managed provider classes for SQL Server (SqlConnection and so on) are declared in the System .Data.SqlClient namespace. The classes for OLE DB-compliant databases (SqlDbConnection and so on) are declared in the System.Data.OleDb namespace. The classes for ODBC-compliant databases (OdbcConnection and so on) are declared in the System.Data.Odbc namespace.

6.4 The Generic Data Classes

As user can use the managed data provider objects to connect to the database through a Connection object, issue a SQL statement through a Command object, and read retrieved rows using a DataReader object; however, they can read rows only in a forward only direction and you must be connected to the database. The generic data objects allow users to store a local copy of the information stored in the database. This allows users to work the information while disconnected from the database

Figure 6.2 illustrates some of the generic data set objects and how they relate to each other. The bridge between the managed provider and generic data set objects is the DataAdapter, which users use to synchronize changes between your DataSet and the database.

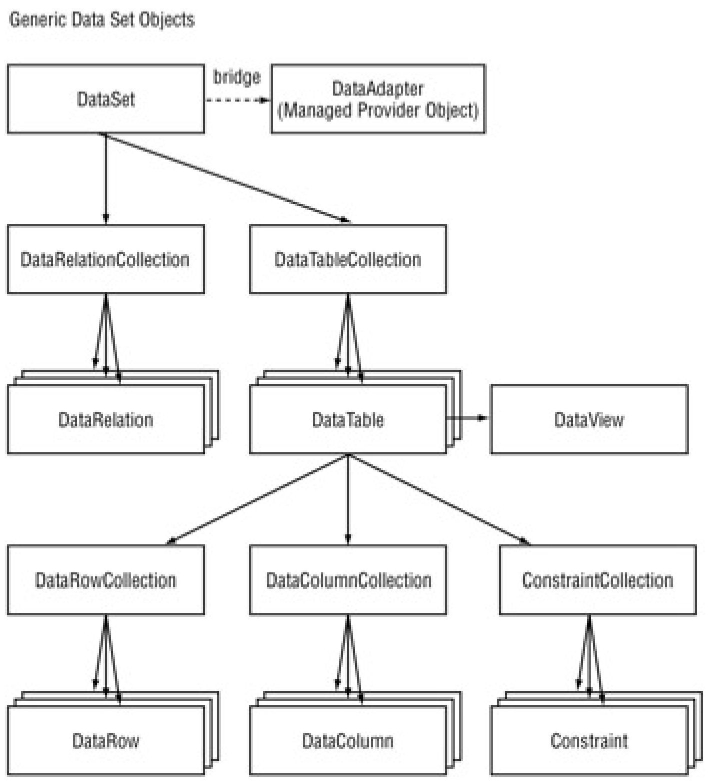


Figure 6.2: Some of the Generic Data Set Objects

The following lines outline some of the generic data classes. First comes the *DataSet* Class.You use an object of the DataSet class to represent a local copy of the information stored in the database. One can make changes to that local copy in your DataSet and then later synchronize those changes with the database through a managed provider DataAdapter object. A DataSet object can represent database structures such as tables, rows, and columns. One can even add constraints to your locally stored tables to enforce unique and foreign key constraints. They can also use a DataSet object to represent XML data. In fact, all information stored in a DataSet is represented using XML, including information retrieved from the database. Next is the *DataTable* Class. Users use an object of the DataTable class to represent a table. They can store multiple DataTable objects in a DataSet through a DataTableCollection object. A DataSet object has a property named Tables, which they use to access the DataTableCollection containing the DataTable objects stored in that DataSet.

Next is the *DataRow* Class. Users an object of the DataRow class to represent a row. They can store multiple DataRow objects in a DataTable through a DataRowCollection object. A DataTable object has a property named Rows, which they use to access the DataRowCollection containing the DataRow objects stored in that DataTable. Next is the *DataColumn* Class. Users use an object of the DataColumn class to represent a column. They can store multiple DataColumn objects in a DataTable through a DataColumnCollection object. A DataTable object has a property named Columns, which they use to access the DataColumnCollection containing the DataColumn objects stored in that DataTable. Next class is the *Constraint* Class. One uses an object of the Constraint class to represent a database constraint that is to be enforced on one or more DataColumn objects of a DataTable. he can store multiple Constraint objects in a DataTable through a ConstraintCollection object. A DataTable object has a property named Constraints, which you use to access the ConstraintCollection containing the Constraint objects stored in that DataTable. Next in the list is the *DataView* Class. A DataView class is used to view only specific rows in a DataTable object using a filter, which specifies the criteria to restrict the rows. The *DataRelation* Class use an object of the DataRelation class to represent a relationship between two DataTable objects. One can use a DataRelation object to model parent-child relationships between two database tables. One can store multiple DataRelation objects in a DataSet through a DataRelationCollection object. A DataSet object has a property named Relations, which one use to access the DataRelationCollection containing the DataRelation objects stored in that DataSet. He use an object of the UniqueConstraint class to represent a database constraint that enforces that the value stored in a DataColumn object is unique. The UniqueConstraint class is derived from the Constraint class. You can store multiple UniqueConstraint objects in a DataTable through a ConstraintCollection object. Users use an object of the ForeignKeyConstraint class to specify the action performed when the column values in the parent table are updated or deleted. The ForeignKeyConstraint class is derived from the Constraint class. One can either have the child rows deleted (cascading action), set the child columns to null, or set the child columns to a default value. He can store multiple ForeignKeyConstraint objects in a DataTable through a ConstraintCollection object. Now something about the namespaces for the Generic Data Classes. The DataSet, DataTable, DataRow, DataColumn, DataRelation, Constraint, and DataView classes are all declared in the System.Data namespace. This namespace contains other classes that users can use in their programs.

6.5 Performing a SQL SELECT Statement

To see, how to connect to the database and perform a SQL SELECT statement to retrieve the data, one should proceed as explained. The following steps allow to retrieve the rows into a DataSet object: First formulate a string containing the details of the database connection. Then create a SqlConnection object to connect to the database, passing the connection string to the constructor. Then formulate a string containing a SELECT statement to retrieve the columns for the rows from the Customers table. Now create a SqlCommand object to hold the SELECT statement. Now set the CommandText property of the SqlCommand object to the SELECT string. Next create a SqlDataAdapter object. Then set the SelectCommand property of the SqlAdapter object to the SqlCommand object. Now create a DataSet object to store the results of the SELECT statement. Now open the database connection using the Open() method of the SqlConnection object. Then call the Fill() method of the SqlDataAdapter object to retrieve the rows from the table, storing the rows locally in a DataTable of the DataSet object. Next close the database connection, using the Close() method of the SqlConnection object. Now get the DataTable object from the DataSet object. Now display the columns for each row in the DataTable, using a DataRow object to access each row in the DataTable.

6.6 Summary

Users store that information using objects of the *data set* classes. Once they have that information in the memory, one can then read and manipulate that information. For example, one can display the columns for the rows, add new rows, modify rows, and delete rows. Periodically, one will reconnect to the database to synchronize the changes made locally with the database. All such access to the database is possible by use of the ADO.NET.

Chapter 7

Connecting to a Database

7.1 Overview

To manipulate data stored in databases, users need to make connections to the database. This is done by using objects of a Connection class. There are three Connection classes: SqlConnection, OleDbConnection, and OdbcConnection. Users use an object of the SqlConnection class to connect to a SQL Server database, an object of the OleDbConnection class to connect to any database that supports OLE DB, such as Oracle or Access, and an object of the OdbcConnection class to connect to any database that supports ODBC. Ultimately, all communication with a database is done through a Connection object.

7.2 Understanding the *SqlConnection* Class

Users use an object of the SqlConnection class to connect to a SQL Server database, and this object handles the communication between the database and your C# program. Although the *SqlConnection* class is specific to SQL Server, many of the properties, methods, and events in this class are the same as those for the *OleDbConnection* and *OdbcConnection* classes.

7.3 Connecting to a SQL Server Database

Users create a SqlConnection object using the SqlConnection() constructor. This constructor is *overloaded*, meaning that there are multiple versions of the constructor that you can call. The SqlConnection() constructors are as follows:

SqlConnection()
SqlConnection(string *connectionString*)

where *connectionString* contains the details for the database connection. Assuming users have imported the System.Data.SqlClient namespace, they can create a new SqlConnection object using the following statement:

```
SqlConnection mySqlConnection = new SqlConnection();
```

They can then set the details for the database connection using the ConnectionString property of mySqlConnection. For example:

```
mySqlConnection.ConnectionString =
  "server=localhost;database=Northwind;uid=sa;pwd=sa";
```

where **server** specifies the name of the computer on which SQL Server is running, **database** specifies the name of the database, **uid** specifies the name of the database user and **pwd** specifies the password for the user. One thing users need to bear in mind is that they can set the ConnectionString property only when your Connection object is closed.

7.4 Opening and Closing a Database Connection

Once users have created your Connection object and set its ConnectionString property to the appropriate details for the database connection, they can open the connection to the database. Users do this by calling the Open () method of their Connection object. Once they have finished with their database connection, they call the Close () method of their Connection

7.5 Connection Pooling

Opening and closing a database connection is a relatively time-consuming process. For this reason, ADO.NET automatically stores database connections in a pool. *Connection pooling* offers a great performance improvement because users don't have to wait for a brand new connection to the database to be established when there's a suitable connection already available. When users close a connection, that connection isn't actually closed; instead, their connection is marked as unused and stored in the pool, ready to be used again. If users then supply the same details, in the connection string (same database, username, password, and so on), then the connection from the pool is retrieved and returned to them. They then use that same connection to access the database. When using a SqlConnection object, users can indicate the maximum number of connections allowed in the pool by specifying max pool size in their connection string

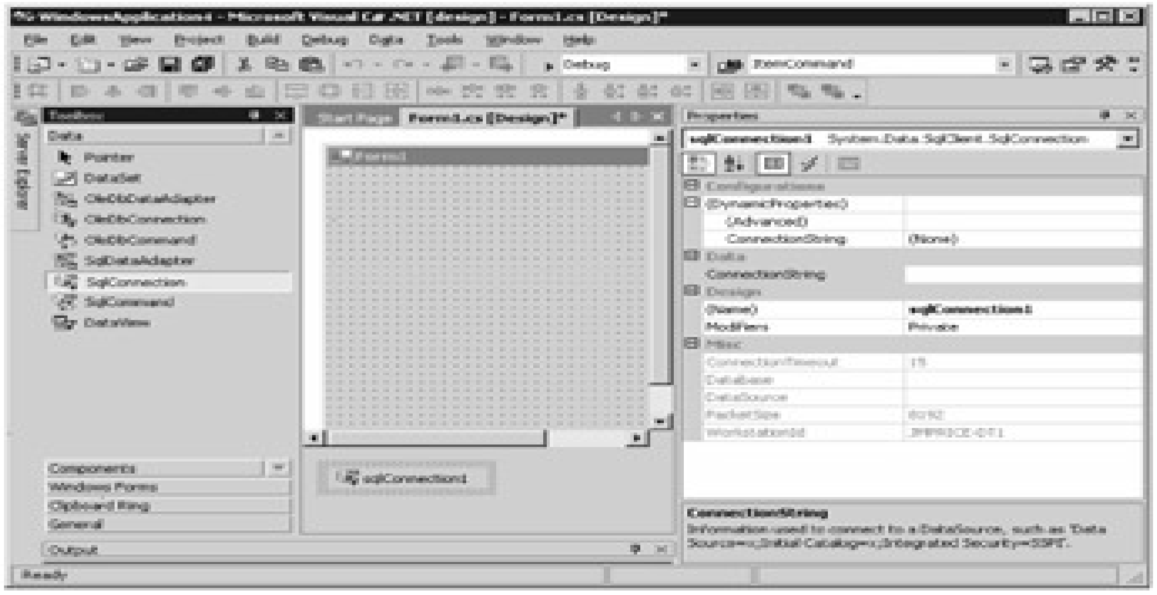
7.6 Getting the State of a *Connection* Object

The state of a connection enables users to know the progress of their connection request to the database; two examples of states are open and closed. Users use the Connection object's

State property to get the current state of the connection to the database. The State property returns a constant from the ConnectionState enumeration.

7.7 Creating a *Connection* Object Using Visual Studio .NET

To create a SqlConnection object using Visual Studio .NET, users drag a SqlConnection object from the Data tab of the Toolbox to their form. Users will recall that a SqlConnection object allows them to connect to a SQL Server database. They can also drag an OleDbConnection object from the Toolbox to their form to connect to a database that supports OLE DB. Figure 7.1 shows a form with a SqlConnection object. This object is assigned the default name of sqlConnection1.



sqlConnection1 System.Data.SqlClient.SqlConnection

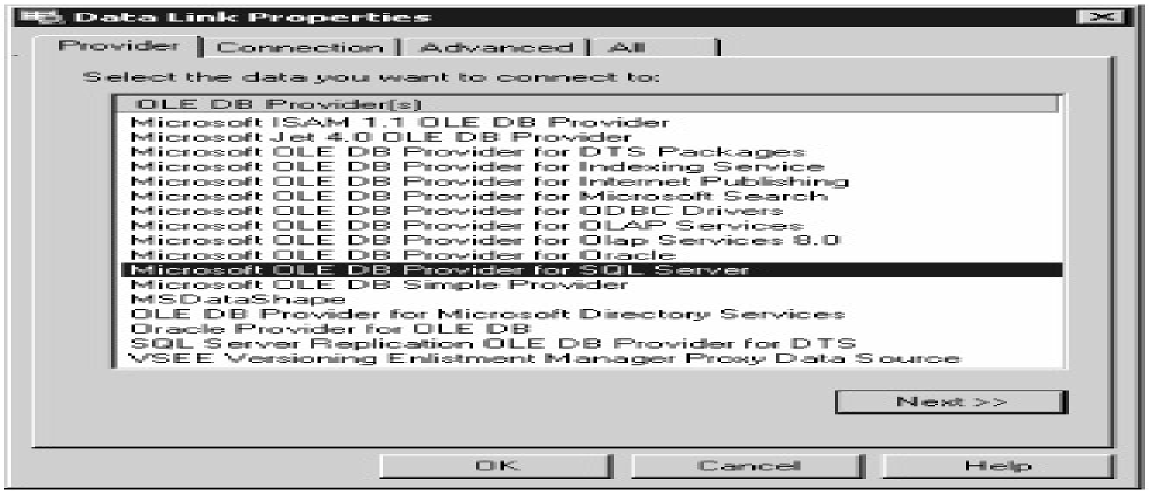
Configurations	
(DynamicProperties)	
(Advanced)	
ConnectionString	(None)
Data	
ConnectionString	
Design	
(Name)	sqlConnection1
Modifiers	Private
Misc	
ConnectionTimeout	15
Database	
DataSource	
PacketSize	8192
WorkstationId	[illegible]

ConnectionString
Information used to connect to a DataSource, such as 'Data Source=x;Initial Catalog=x;Integrated Security=SSPI'.

Figure 7.1: Creating a SqlConnection object with Visual Studio .NET

Once they've created a SqlConnection object, that object appears in the "tray" below the form. The tray is used to store nonvisual components like SqlConnection objects. Other objects that appear in the tray are SqlCommand objects. These objects are considered nonvisual because they don't see them when you run their form. They can of course still work with them visually when designing their form.

To the right of the form, They'll notice the Properties window, which they use to set the properties for their SqlConnection object as shown in Figure 7.2.

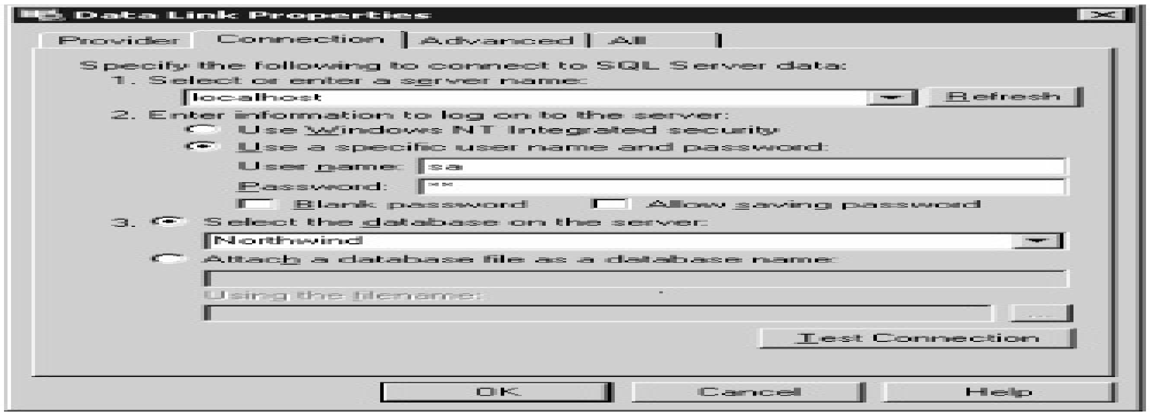


Select the data you want to connect to:

OLE DB Provider(s)
Microsoft ISAM 1.1 OLE DB Provider
Microsoft Jet 4.0 OLE DB Provider
Microsoft OLE DB Provider for DTS Packages
Microsoft OLE DB Provider for Indexing Service
Microsoft OLE DB Provider for Internet Publishing
Microsoft OLE DB Provider for Microsoft Search
Microsoft OLE DB Provider for ODBC Drivers
Microsoft OLE DB Provider for OLAP Services
Microsoft OLE DB Provider for Olap Services 8.0
Microsoft OLE DB Provider for Oracle
Microsoft OLE DB Provider for SQL Server
Microsoft OLE DB Simple Provider
MSDataShape
OLE DB Provider for Microsoft Directory Services
Oracle Provider for OLE DB
SQL Server Replication OLE DB Provider for DTS
VSEE Versioning Enlistment Manager Proxy Data Source

Figure 7.2: Selecting the Provider

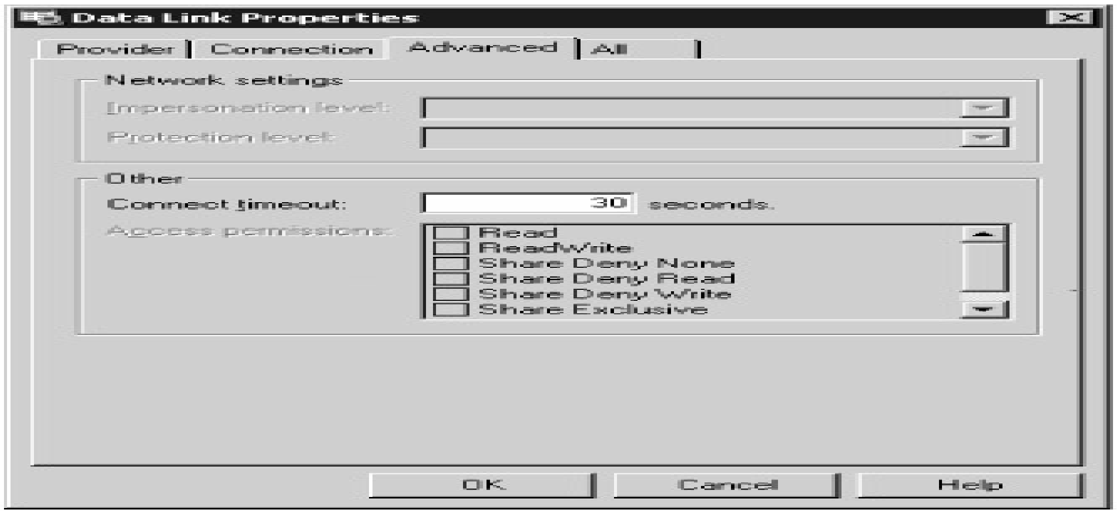
Click the Next button to continue to the Connection tab, where users enter the details for their database connection, as shown in Figure 7.3.



Specify the following to connect to SQL Server data:
1. Select or enter a server name:
localhost
2. Enter information to log on to the server:
Use Windows NT Integrated security
[✓ Use a specific user name and password:]
User name: sa
Password: **
Blank password
Allow saving password
3. [✓ Select the database on the server:]
Northwind
Attach a database file as a database name:
Using the filename:

Figure 7.3: Entering the Connection Details

Once users have entered their connection details, they can press the Test Connection button to ensure their details are correct. At this point, they have entered all the mandatory details, and they can choose to save their details by clicking OK, or can click Advanced to enter additional details such as the connection timeout, as shown in Figure 7.4.


Data Link Properties

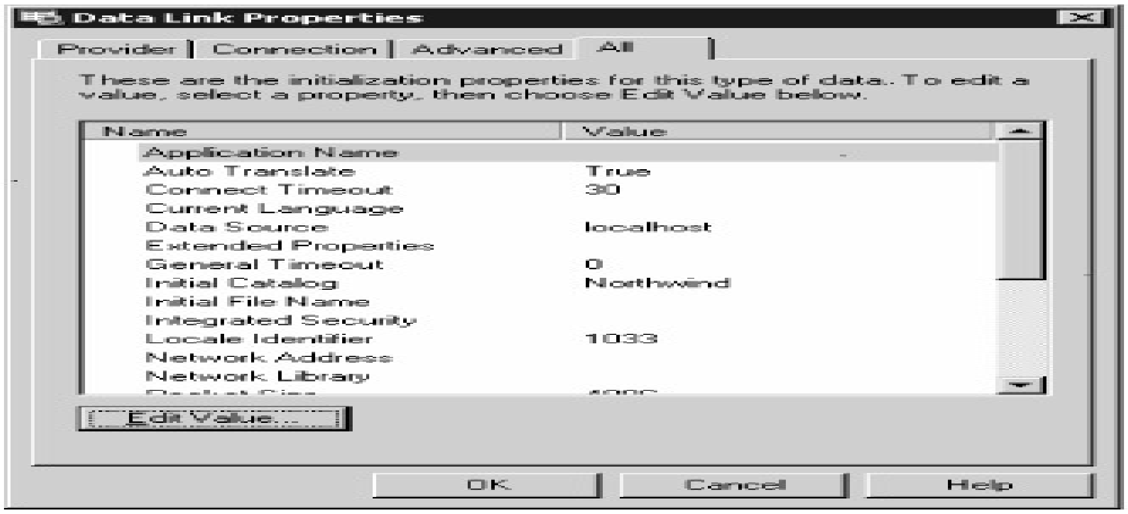
Provider | Connection | Advanced | All

Network settings
Impersonation level:
Protection level:

Other
Connect timeout: 30 seconds.
Access permissions:
- [ ] Read
- [ ] ReadWrite
- [ ] Share Deny None
- [ ] Share Deny Read
- [ ] Share Deny Write
- [ ] Share Exclusive

Figure 7.4: Entering the Advanced Connection Details

Users can also click the All tab to view and edit all the values for the connection, as shown in Figure 7.5. To edit a value, click Edit Value.


Data Link Properties

Provider | Connection | Advanced | All

These are the initialization properties for this type of data. To edit a value, select a property, then choose Edit Value below.

Name	Value
Application Name	
Auto Translate	True
Connect Timeout	30
Current Language	
Data Source	localhost
Extended Properties	
General Timeout	0
Initial Catalog	Northwind
Initial File Name	
Integrated Security	
Locale Identifier	1033
Network Address	
Network Library	

Figure 7.5: Entering the Advanced Connection Details

Click the OK button to save connection details. On my computer, the ConnectionString property for the SqlConnection object that connects to the SQL Server Northwind database is set to

data source=localhost;initial catalog=Northwind;persist security info=False;

user id=sa;pwd=sa;workstation id=JMPRICE-DT1;packet size=packetsize

7.8 Summary

Once the database accessed the user can carry out all the data manipulation and data definition operations on the stored data. But for this purpose the user needs to connect to the database. This is done by the use of the connection classes of the base library provided by the .NET framework. One can really choose the way the database is to be accessed. Separate connection classes are used for connecting to DBMS.

Chapter 8

Development of the Web Applications

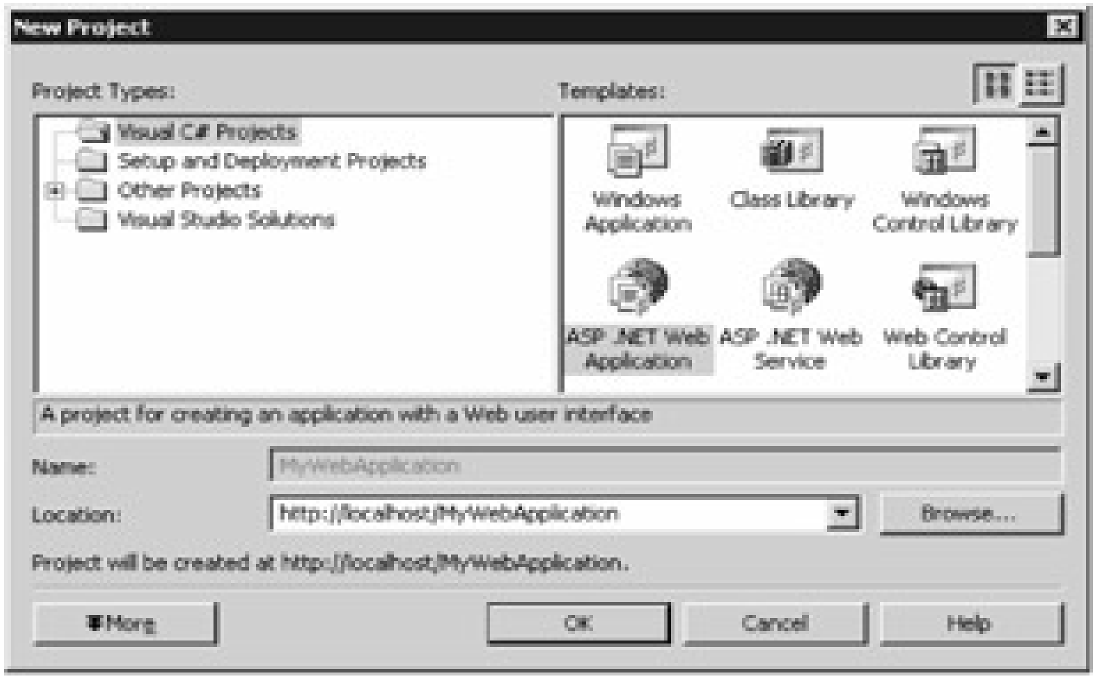
8.1 Overview

Active Server Pages for .NET (ASP.NET) enables users to create dynamic Web pages with content that can change at runtime and to develop applications that are accessed using a Web browser. For example, users could develop an e-commerce application that allows users to order products over the Web, or a stock-trading application that allows users to place trades for shares in companies. ASP.NET is conceptually similar to its rival JavaServer Pages (JSP) in that users request a page from a server using a Web browser, and the server responds by running the ASP.NET page. The server then sends back HTML that is displayed in their browser.

There are two main parts to an ASP.NET form: the .aspx file, which contains HTML and ASP.NET code, and the .aspx.cs file, which contains C# code that supports the Web form. Just think of this C# code as *running behind* the form, and for this reason the .aspx.cs file is known as the *code-behind file*. One can view the HTML containing the ASP.NET code for your form by clicking the HTML link at the bottom of the form designer. One also can view the code-behind file by selecting View Code, or can press F7 on keyboard.

8.2 Creating a Simple ASP.NET Web Application Using VS .NET

To create a simple ASP.NET Web application that contains a text box and a button using VS .NET. When users press the button, a string of text will appear in the text box. Perform the following steps: First start Visual Studio .NET (VS .NET) and select File , New Project. Select Visual C# Projects from the Project Types area on the left of the New Project dialog box, and select ASP .NET Web Application from the Templates area on the right. Enter **http://localhost/MyWeb-Application** in the Location field, as shown in Figure 8.1.



A project for creating an application with a Web user interface

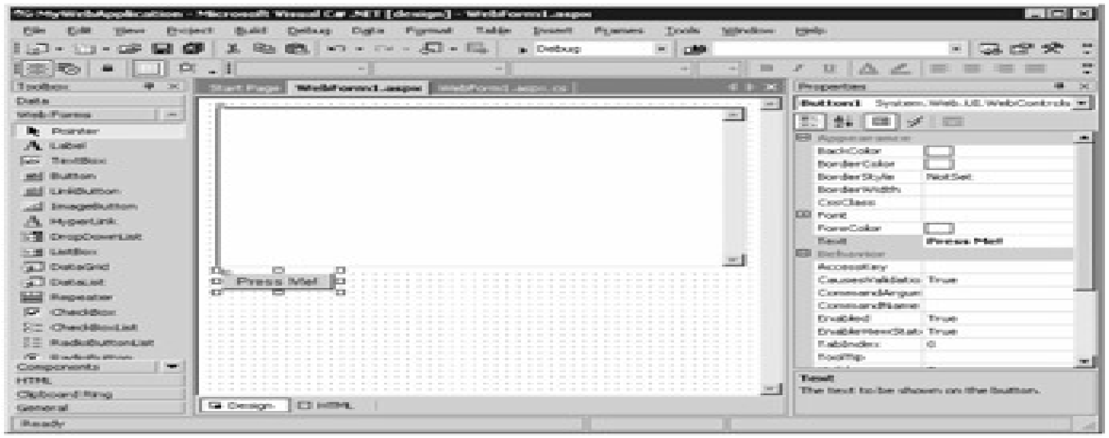
Name: MyWebApplication

Location: http://localhost/MyWebApplication

Project will be created at http://localhost/MyWebApplication.

Figure 8.1: Creating an ASP.NET Web Application in Visual Studio .NET

Next click the OK button to continue. VS .NET will create a new directory named MyWebApplication in the wwwroot directory; this is the directory where IIS stores published Web pages and applications. After click the OK button, users will see the new application being sent to IIS. Once the application has been deployed to IIS, VS .NET will display a blank Web form. One can think of the Web form as the canvas on which you can place controls, such as text boxes and buttons. When users later run the form, they'll see that the page displayed by the Web browser is laid out in a similar manner to the form. Now add a TextBox control to the form. The default value for the ID property of your TextBox control is TextBox1. Now set the TextMode property for TextBox1 to MultiLine; this allows the text to be displayed on more than one line. Next, add a Button control to the form. The default ID for your Button control is Button1. Set the Text property for Button1 to Press Me! Figure 8.2 shows the form with the TextBox and Button controls.

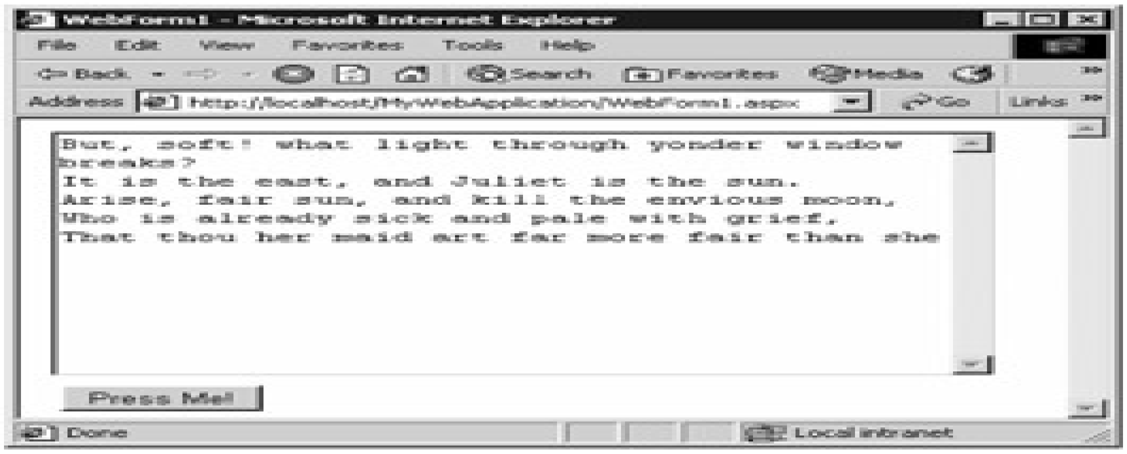


Button1	System.Web.UI.WebControls
Appearance	
BackColor	
BorderColor	
BorderStyle	NotSet
BorderWidth	
CssClass	
Font	
ForeColor	
Text	Press Me!
Behavior	
AccessKey	
CausesValidatio	True
CommandArgum	
CommandName	
Enabled	True
EnableViewStat	True
TabIndex	0
ToolTip	

Text
The text to be shown on the button.

Figure 8.2: Adding TextBox and Button Controls to the Form

Next, add a line of code to the Button1_Click() method. This method is executed when Button1 is pressed in your running form. The statement added to Button1_Click() will set the Text property of TextBox1 to a string. Now form is ready to be run. Select Debug, Start Without Debugging, or press Ctrl+F5 on the keyboard to run the form (see Figure 8.3).


WebForm1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
http://localhost/MyWebApplication/WebForm1.aspx
Go
Links
Press Me!
Done
Local intranet

```
But, soft! what light through yonder window
breaks?
It is the east, and Juliet is the sun.
Arise, fair sun, and kill the envious moon,
Who is already sick and pale with grief,
That thou her maid art far more fair than she
```

Figure 8.3: The Running Form

Now that the form is created and run the form, let's examine the code generated by VS .NET. There are two main parts to the code. The WebForm1.aspx file, which contains HTML and ASP.NET code. The WebForm1.aspx.cs file, which contains C# code that supports the Web form. One can think of this C# code as *running behind* the form, and for this reason the WebForm1.aspx.cs file is known as the *code-behind file*.

8.3 The Web Form Controls

A summary of the various Web form controls that users can pick from the Toolbox's Web Forms Section. Table 8.1 summarizes the controls.

Table 8.1: Web Form Controls

CONTROL	DESCRIPTION
Label	Displays text. Users set the text that they want to display using the Text.
TextBox	A box containing text that the user may edit at runtime.
Button	A clickable button. The Text property determines the text shown on it.
LinkButton	A LinkButton appears as a hypertext link.
ImageButton	An ImageButton shows an image.
HyperLink	A hyperlink. Users set the hyperlink using the NavigateUrl property.
DropDownList	A list of options that drops down when clicked.
ListBox	A list of options which case the user can select only one option.
DataGrid	A grid containing data retrieved from a data source, such as a database.
DataList	A list containing data retrieved from a data source.
Repeater	A list containing data retrieved from a data source that users set using the DataSource property.

8.4 Summary

The web is the level which is in direct contact to the user. Through this interface the user interacts with the DBMS and carryout the data manipulation and data definition operations. Web application is developed using the ASP.NET. The .NET framework provides ASP.NET which is very easy to use. One needs to “drag and drop” the functionalities and the code in HTML is automatically generated. Then this web is connected to the back-end through a programming language supported by the .NET framework such as C# in this software.

The DBCS Application

In this chapter the functionalities of the DBCS application will be explained so that one can have a fair idea of how this is going to work. Figure 9.1 shows the first screen which the user comes across. User is required to login by supplying the password and the user name. The user can be administrator or a normal user.

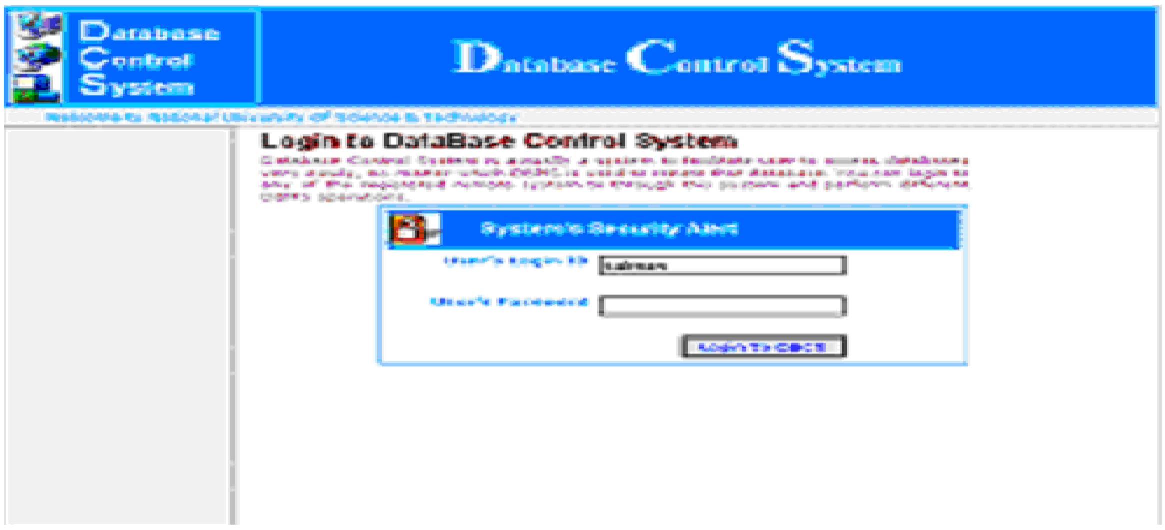


Figure 9.1 Login

Figure 9.2 shows the functionalities. This is the screen which users come across once his user name and password are accepted and he is logged in. This screen is for the normal users and he can carry out following operations. **Register new computers** is used to registered new computers. A log of registered computer is kept and used for future connections. **Un-registered computers** is used to remove an already registered computer. **Block computers** is used to block a computer. The block computer will not be shown in the list of computers. Blocking is for a specific period. **Un-block computers**. A blocked computer is un-blocked so that it becomes a member of the group again. **Insert**. This functionality is used to write a value to a column or row. **Update.** Update is used to change the value of a row or column which is already existing. **Delete**. Delete is used to remove a value. **View schema**. It is used to view the schema. **View Data**. Data is shown as table on this command. **Change Password** is used to change the password of the users. **sign out** is command that ends a session. Once sign out, a user has to login again. **New table** is used to create a new table. The user is required to provide the column names and adjust parameters. **Delete table**. Deletes an existing table. **Add column**. Modifies a table by adding a column. **Delete column**. Deletes a column. **Select Data.** It used to select a specific data from the data ware house. **Insert Data** is used to insert a data in a data warehouse. **Delete Data** deletes data from data ware house. **Update Data** modifies a data from ware house.

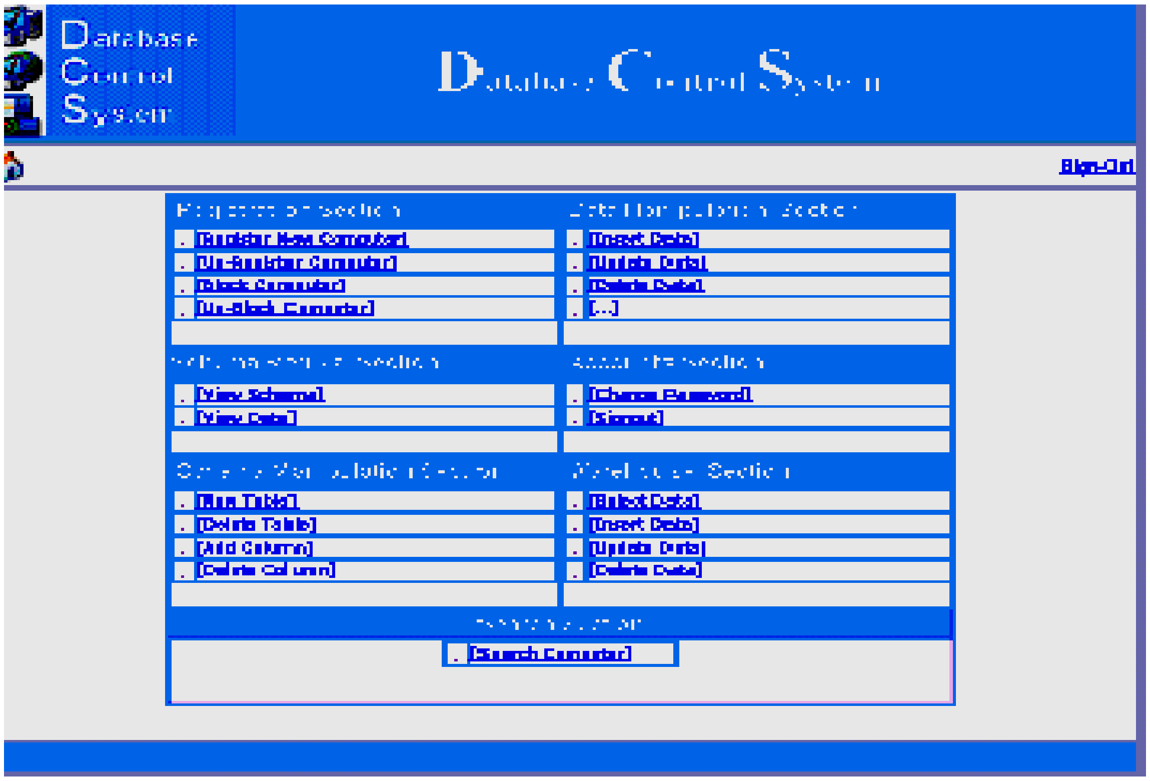
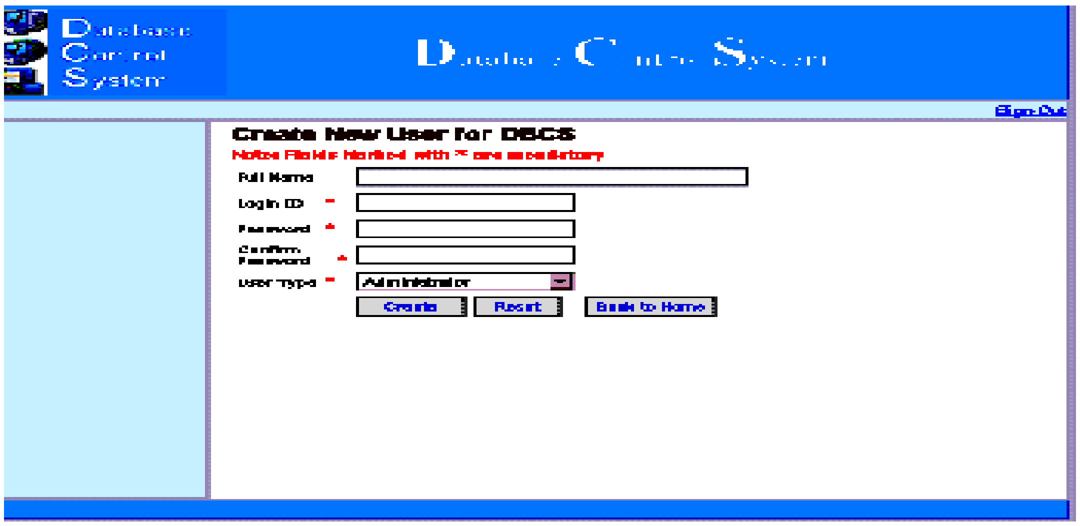


Figure 9.2 Functionalities

Apart from these above functionalities, an administrator has additional functionality of create user which creates a new user by assigning user name and a password. The fields which are required to be filled for the creation of a new user is shown in Figure 9.3 which includes user name, user ID, password and user type.



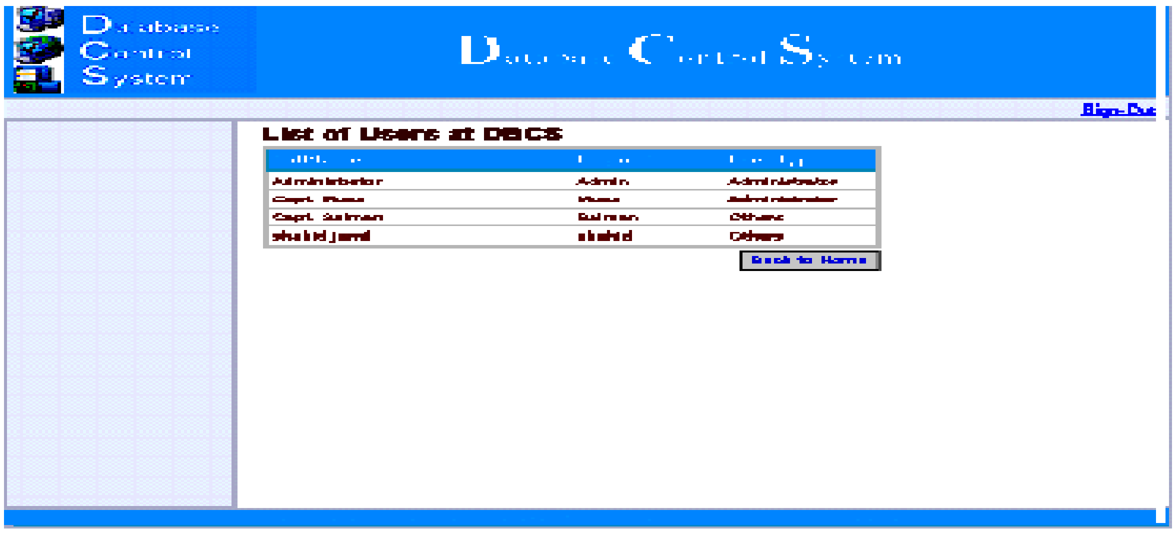
Create New User for DBCS

Note: Fields Marked with * are mandatory

Full Name	
Login ID *	
Password *	
Confirm Password *	
User Type *	Administrator

Figure 9.3 Create User

Figure 9.4 shows the list of users at the DBCS. It shows all the users who have been created by supplying a password and a user name by the administrator. The list also includes the administrator.

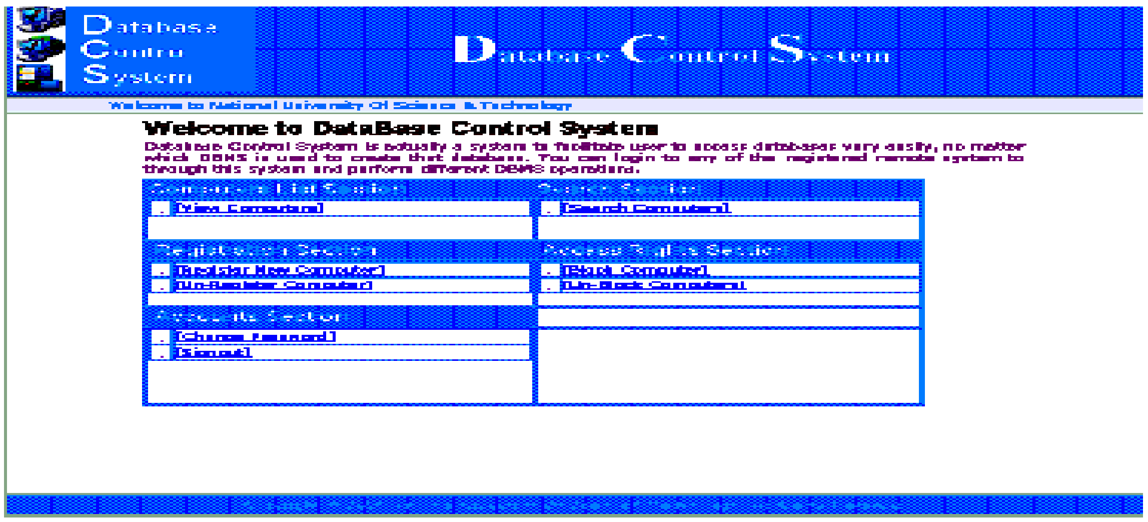


List of Users at DBCS

[illegible]	[illegible]	[illegible]
Administrator	Admin	Administrator
Capt. [illegible]	[illegible]	Administrator
Capt. Salman	Salman	Others
shahid jamil	shahid	Others

Figure 9.4 List of Users

Figure 9.5 is the administrators screen once he login. The administrator has the functionalities of register, un-register, Block, Un-block, Change password and sign out functionalities which have been discuss earlier. The additional functionalities shown here are; View computer. Used to view the computers already registered with the DBCS. Search Computers. It is used to search a computer by its name or IP.


Database Control System

Database Control System

Welcome to National University Of Science & Technology

Welcome to DataBase Control System

Database Control System is actually a system to facilitate user to access databases very easily, no matter which DBMS is used to create that database. You can login to any of the registered remote system to through this system and perform different DBMS operations.

Computers List Section
- [View Computers]

Search Section
- [Search Computers]

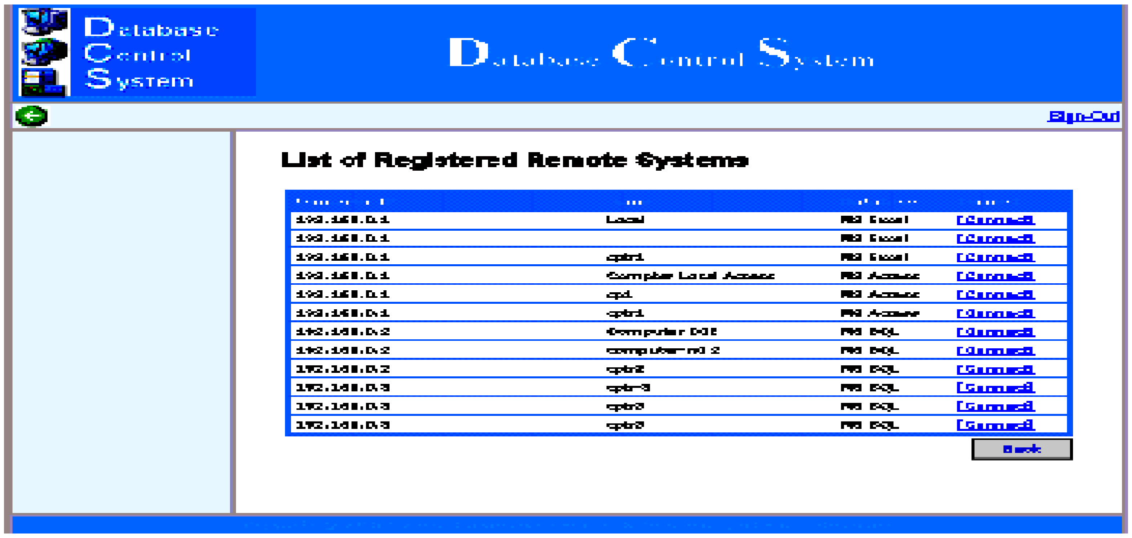
Registration Section
- [Register New Computer]
- [Un-Register Computer]

Access Rights Section
- [Block Computer]
- [Un-Block Computers]

Accounts Section
- [Change Password]
- [Signout]

Figure 9.5 Welcome

Figure 9.6 displays the list of remote systems which has been registered by the DBCS.


Database Control System

Database Control System

Sign-Out

List of Registered Remote Systems

[illegible]	[illegible]	[illegible]	[illegible]
192.168.0.1	Local	MS Excel	[Connect]
192.168.0.1		MS Excel	[Connect]
192.168.0.1	cptr1	MS Excel	[Connect]
192.168.0.1	Computer Local Access	MS Access	[Connect]
192.168.0.1	cp1	MS Access	[Connect]
192.168.0.1	cptr1	MS Access	[Connect]
192.168.0.2	Computer DB2	MS SQL	[Connect]
192.168.0.2	computer no 2	MS SQL	[Connect]
192.168.0.2	cptr2	MS SQL	[Connect]
192.168.0.3	cptr-3	MS SQL	[Connect]
192.168.0.3	cptr3	MS SQL	[Connect]
192.168.0.3	cptr3	MS SQL	[Connect]

Back

Figure 9.6 List of Remote Computers

Figure 9.7 shows the fields required to be supplied to the user by the remote system wishing to register itself.

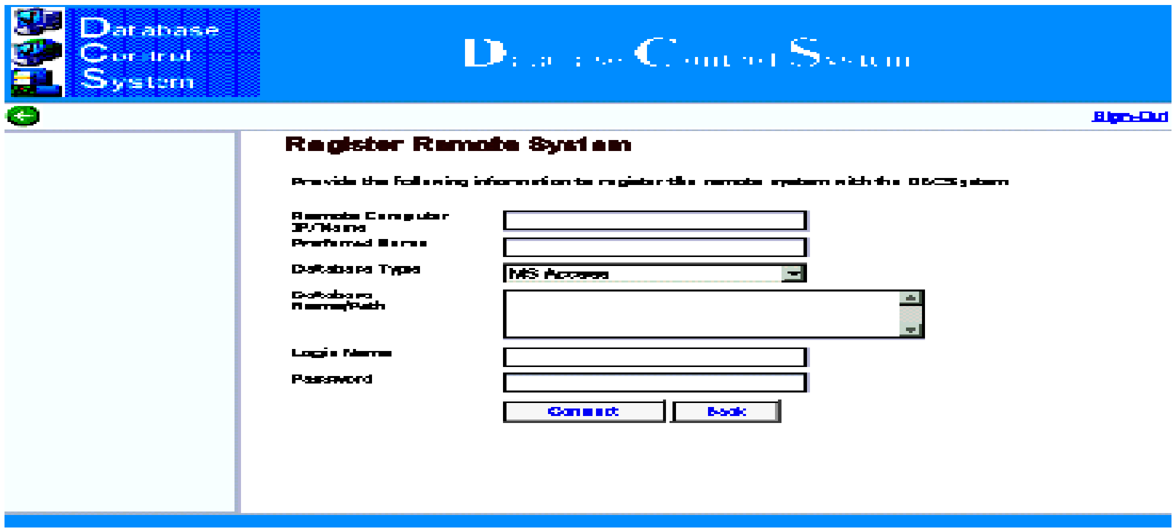


Figure 9.7 Register Remote System

The users can change the passwords as well. Figure 9.8 shows the screen which appears once a user wants to change his password.

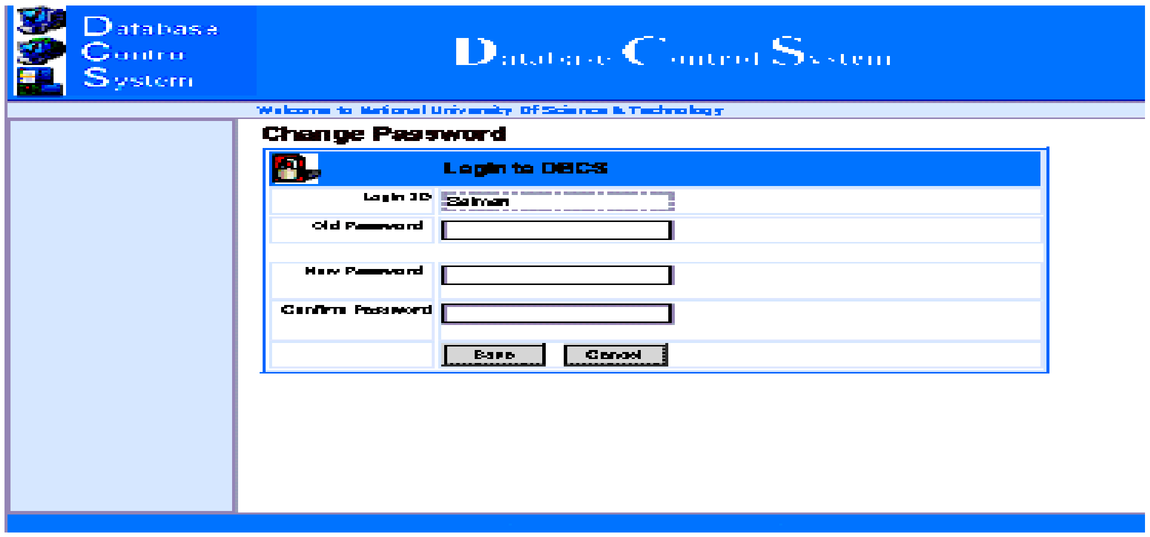


Figure 9.8 Change Password

The data ware house concept has been implemented in this project. A user can send a query to all the data bases to enquire a specific data by using the window shown in Figure 9.9.

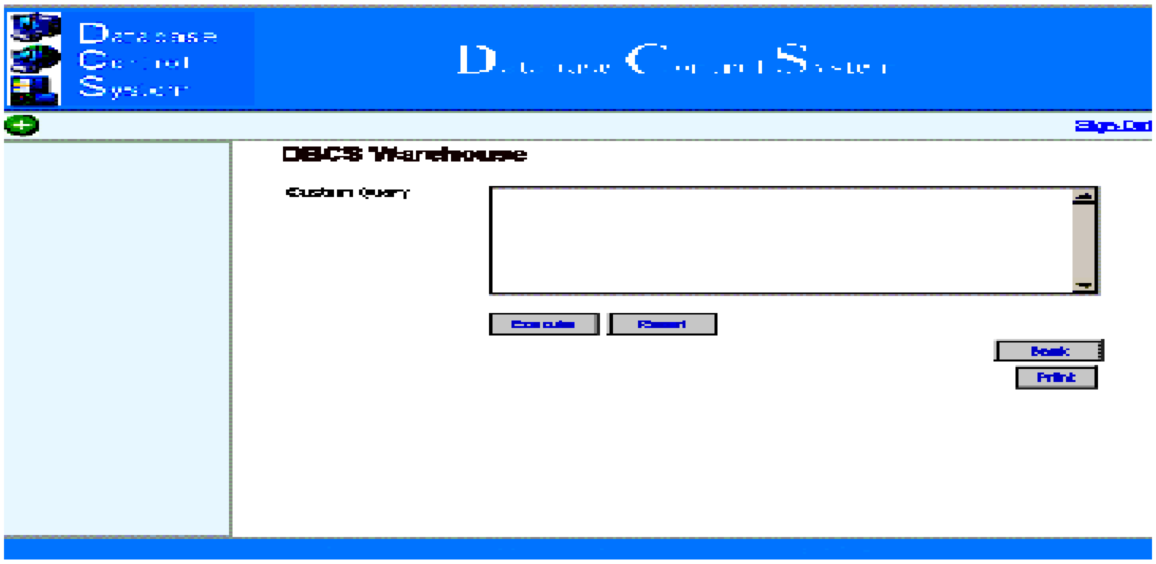


Figure 9.9 Warehouse

Chapter 10

Results and Analysis

10.1 Introduction

Testing of software is a very important part of the software development activity. During this stage many modifications are also carried out. As a result of this phase, a programmer may be required to completely rewrite the code. If the results are not as per the expectations of the user, the software may not be accepted by the user. Testing is carried out throughout the development phase. Once developed, software is integrated and a new series of testing is required to be carried out. As such the application software was also passed through a series of tests starting with the testing of the first module. This continued till the software was completed. After the completion of the software, it was tested on line as well.

10.2 Testing Model Used

Testing was carried out using the ‘waterfall’ model. Here as soon as the very first model was written, it was tested. Subsequently all other modules were tested and then integrated in the overall code and tested. Once the software was ready, testing on LAN and workgroup was also planned and executed. The use of this model did consume lot of time however, most importantly, at each stage of the development, the software was checked and in the end no extra effort was required to correct any error. Thus it reduced the risk factor. Also it was possible to accommodate the changes in the in the design and the functionalities. This also allowed changes to technology and style of coding.

10.3 Stages of Testing.

During the development and integration of the software three levels testing was carried out. In the first level, each module was tested for errors and the output was observed. Any changes required were made. This consumed a lot of time but the time spent did pay in the end. This also allowed refining the overall design of the DBCS. This highlighted the powers and shortcomings of the programming languages and helped in the choice of an appropriate programming language. The second level of testing was carried out to see if the individual modules fit in the overall coding. This in fact had a direct bearing on the efficiency of the software and needed more care. Any changes required to be made to the modules as a result of this level of testing was carried out immediately. This testing caused a delay in the development of software but the risk of error was reduced to a minimum. Till this level the testing was carried out on a single PC. In the next level, the software was tested on workgroup and also on the LAN in the computer laboratory. By this time the software was completed and integrated. This level of testing not only tested the output and performance of the software but also checked the hardware and software requirements of the machines on which the software was to be run.

10.4 Results and Analysis

Once the software was tested on a single PC, it met all the required parameters. Its performance was satisfactory and the output was as per the expectations. Once the test was carried out in the computer laboratory, the results were not really encouraging. The reason was that the strict administrative rights were enforced and the software was not really allowed to carry out the database access and manipulation operations. This denial of access means the software was not able to connect to the database stored in a DBMS running on a remote machine. After obtaining the administrative rights, the software was able to connect to remote machines. Yet problems were faced which was mainly pertaining to the networking and network security. The software was tested on a 'workgroup' and the results were very much satisfactory. It was able to access and connect register DBMS running on remote machines and carrying out the data manipulation operations. The software can give good results if it is placed on the server machine and the administrative and network security restrictions are not a hurdle in the way of the user forbidding the software to access DBMS running on a remote machine and stopping the user from connecting to remote databases. The best way to use the application is to place it on web so that it can be accessed and run from anywhere. This kind of testing was not carried out due to the fact that web hosting was a costly affair and also needed time to carryout the necessary coordination. However the results shown by the software once tested in a workgroup was very encouraging and the same result is expected once the software is run from a website hosting the software.

10.5 Limitations and Future Work

Despite showing good results, the software has few inherent weaknesses. These weaknesses have not been catered for in the design of the software. Moreover few additional features could not be incorporated due to time limitations. First on the list are the key board shortcut keys. All software, whether operating system or any other utility software or special software, always contains key board shortcut keys, which make the job of the user very easy and in certain cases if the mouse stop functioning, the user is not paralyzed and can continue working by making use of the key board shortcut keys. For example while typing a letter in MS Word, a person can save his work by pressing 'CTRL +S' and thus saves time and effort which is required for saving the document by using the mouse. In this way the person ensures that his tempo of work is not broken. This aspect has not been catered for in the DBCS and is a shortcoming of the software. This can be addressed to in future works. If this is done it will add to the performance of the software and will help the user to carryout the job very smoothly.

Another limitation of the software is the inability to convert from one DBMS format to the other. The software can convert data stored in any format to a standard format but it cannot convert a table from MS Access format to a table in Oracle or vice versa. Routines for the inter-conversion are available. One need to first convert the data from one format to an intermediate standard format and then this intermediate format is converted to the required format. Since all the data are exported to a standard format by this software the job of inter-conversion is half done. In a future work, the inter-conversion of various data storage formats may be implemented. This will allow easy implementations of the database operations.

Yet another limitation has been the lack of the database security. The software is password protected and the DBMS is also secure. The built-in security measures of the .NET ensure security of the database. Need is to enhance the security level to the table and the field level. Thus in any future work this aspect of the database security may be implemented making the database secure up to the table and the field level. This aspect of database security is very helpful in ensuring secure data access and manipulation in case the software is hosted on web.

The basic database operations have been implemented and are well functioning. The result of a search on multi DBMS is stored in a simple format and is printed as such. This results in data redundancy. Duplicity of data is not required. This can be avoided by incorporating the basic functionalities of ‘JOIN’, ‘DIVIDE’ and ‘INTERSECT’. Also the functionality of ‘UNION’ is required to be implemented in the software. Any future work may map these four main functions along with the derivative functions. This will deny any chance of duplicity of data.

In this software the communication module is responsible to map the queries generated by the SQL Engine to three DBMS i.e. MS Access, SQL Server and Oracle. The allows for extension of the software and in any future work more DBMS can be added to the software. This also depends upon the requirement of a particular user.

10.6 Conclusion

DBCS is Software developed by the DBCS team. It is a platform independent product, which is capable of manipulating different databases of various vendors. Since there is no such software (platform) having these facilities. So there was a great need to design such software. Nowadays database development can be done using Microsoft Access, SQL Server and SQL/PLSQL and Oracle etc. Each platform has its own features. Access is very user friendly and it takes very less time to build the back-end of database. SQL/PLSQL, Oracle and SQL Server are more secure and handle millions of records. The decision of using any backend platform depends upon the requirement. A small database where security is not of great importance, using ORACLE is a waste of resources, therefore, all these three have their own respective applications.

Our application makes the job of learning and implementing the database operations much easier. The user of this application will no longer have to learn each of these technologies separately. The user will perform all his operations on this generic platform. All his operations will be mapped to the respective back-end database development platform. How the operations of the platform are mapped, is not the concern of the end user. He just selects the database application onto which he wants to map his operations and all of his operations will be mapped to the selected platform. Moreover, implementing these operations is much straight forward as same interface is used for all database applications and the platform is very user friendly. By mapping all the desired operations onto any of the back-end platform using a common interface user does not have to learn all the backend development tools. Only this software can do their job much quicker as it is a very user-friendly interface. Moreover the software has been developed using a very powerful and emerging technology, the MS Visual Studio.NET. The software can be web hosted and can be accessed and run from anywhere.

BIBLIOGRAPHY

[1] ASP.NET Database Programming Weekend Crash Course by Jason Butler and Tony Caudill.

[2] Programming Microsoft.NET by Prosise Jeff.

[3] SAMS, Teach Yourself .NET Windows Forms in 21 Days, by SAMS series

[4] Essential ADO.NET by Bob Beauchemin explains the database access part of the .NET framework.

[5] ASP.NET by Example by Steven A. Smith

[6] Professional ASP.NET 1.0 Special Edition a wrox series book.

[7] Dietel-C Sharp How to programme by Dietel and Dietel

[8] *http://www.c-sharpcorner.com/*

[9]*http://www.codeguru.com/*

[10] *http://www.dotnet-fr.org/*

[11] *http://microsoft.com*

[12] *http://msdn.microsoft.com/net/*