Comparative Study of Web Application Development with SQL Server and Db4o

MASTER'S THESIS IN COMPUTER SCIENCE
30 CREDITS, ADVANCED LEVEL

Sreedhar Danturthi

Supervisor: Frank Lüders
Examiner: Sasikumar Punnekkat

Mälardalen University
School of Innovation, Design and Engineering
Västerås, Sweden
Abstract

This thesis work is a comparative study of technical features between three distinctive methods for database access. A web store which offers software components to download, for registered users has been built with three different database access methods to draw comparisons.

1. SQL Server 2005 with stored procedures
2. LINQ to SQL
3. Db4o

Working in Visual Studio with three different methods for database access was not difficult. SQL Server with Stored procedures is efficient in performance and offers maintainability of the database access logic. LINQ to SQL makes queries to database as first-class language construct providing type-safety and Intellisense. LINQ to SQL has relatively low performance compared to Stored procedures because of the translation of LINQ queries to SQL Queries to communicate with the database. Db4o solves the problem of impedance mismatch and making the development of Database model much simpler and similar to the application domain model. Although Db4o has its own drawbacks with database size and concurrency. Db4o can be considered as an alternative to Relational Database Management Systems. Db4o works particularly well with applications which have a small data footprint.
Acknowledgements

I would like to thank my supervisor Frank Lüders for his constant support and guidance during this thesis work. I also would like to thank my examiner Sasikumar Punnekkat for his invaluable suggestions and feedback. Last but not least I would like to thank my friends and family who have given me love and strength during this thesis.
# Table of Contents

1. Introduction ......................................................................................................................5  
2. Background..........................................................................................................................6  
   2.1 Web Applications .........................................................................................................6  
      2.1.1 Features of Web Applications ..............................................................................6  
      2.1.2 Architecture of Web Application ........................................................................7  
      2.1.3 ASP.NET Architecture .......................................................................................10  
   2.2 Database Management Systems (DBMS) ..................................................................15  
      2.2.1 Relational Database Management Systems (RDBMS) .........................................15  
      2.2.2 Object Oriented Database Management Systems (OODBMS) ................................19  
3. Web Store ...............................................................................................................................25  
   3.1 Application Architecture ...............................................................................................26  
   3.2 User Interface .................................................................................................................27  
   3.3 Database Design .............................................................................................................35  
      3.3.1 SQL Server .............................................................................................................36  
      3.3.2 Db4o .......................................................................................................................37  
   3.4 Implementation ...............................................................................................................39  
      3.4.1 Getting Started with ASP.NET Web-applications ..................................................39  
      3.4.2 WebStore using SQL Server 2005 .......................................................................41  
      3.4.3 WebStore using LINQ to SQL ............................................................................43  
      3.4.4 WebStore using Db4o .........................................................................................45  
4. Discussion .............................................................................................................................48  
5. Conclusions and Future work ............................................................................................52  
References .................................................................................................................................53
1. Introduction

In present day web application development, databases have become omnipresent. Most of the time both developer community and customers have fixated opinion on usage of database-model regardless of the requirement.

**Problem Statement:** There are different solutions for storing persistent data in web-based systems. It is most commonly done using some database management system (DBMS), and the most common type of Database systems is relational database. It is quite common to use relational databases with the web application development. An alternative solution is object-oriented databases. This thesis investigates alternative database model (Object Oriented database model) to the relational database-model for making a comparative analysis.

**Method:** An e-commerce web application using ASP.NET with C# has been developed with the following variations:

1. SQL Server 2005 (Relational Database) is used as database and stored procedures were used to interact with the SQL Server
2. SQL Server 2005 (Relational Database) is used as database and LINQ to SQL is used to interact with the SQL Server
3. Db4o (Object Oriented Database) is used as database and LINQ has been used as standard query language to interact with the Db4o object database.

The application model nor the presentation model has been changed, it is the underlying data layer and the methodology to query the database which has been altered in these three variants.

Then the comparative study has been made discussing the technical details and performance issues associated with these three distinctive methods for database access.

**Organization of Thesis:** In section 2, web-application general features were discussed along with architecture of ASP.NET and Web-applications. A general introduction about Relational Database Management Systems and Object-oriented Database Management Systems was made. In section 3 web store description, architecture, application User Interface, Database design for the database-models used in application along with implementation details are discussed. Finally in section 4 a comparative analysis of the web store is done. Finally in section 5 conclusions and Future work are discussed.
2. Background

2.1. Web Applications

2.1.1. Features of Web Applications

Web applications can be classified as

1. Static web applications
2. Dynamic web applications

Static web applications: These are the Web applications where the elements comprising the web application are static HTML pages. They don't give end user any chance to interact or modify the application behaviour.

Dynamic web applications: On the other hand these applications give the end user the capability to interact with the application behaviour. The interaction and behaviour of application is based on the end user input and requests in the client browser. The input entered by the user could be parsed both in the client end or at the server end. The input that is entered by the end user can be executed at the client end using web browser. To parse and interact according to the client input we need the help of the client scripting languages. Examples of client side scripting languages include VBScript and Java Script. These languages could be embedded in the HTML script which as a whole is executed by the browser or they could be placed in a separate file which when requested by the browser could be sent by the web server and which is then executed on the web browser. An example of this could be after the user enters all the text box fields on registration page (in order to be a registered user), to validate the data entered by him, to check if he has left any blanks in the text box fields, if the data types entered are correct, and to reset the text box fields after pressing the submit button we could normally take the help of scripting languages like Java Script or VBScript to perform these actions. Usage of the client side scripting languages reduces the network traffic and the throughput of the web applications is normally enhanced.[1]

If the action or the request initiated by the user is executed at the web server then that is called server side scripting. Normally when a html file is requested by the client the server just returns the html file with out any further processing, in the case of html file with server scripts, the server further processes the file, deciphers the server scripts and returns the result as a plain html file which is presented to the browser. In the case of ASP.NET when .aspx files are requested by the browser, the request is sent to the IIS (Internet Information server) web server, which is then sent to the .NET engine, where each line in aspx file is
parsed converted to HTML and returned to IIS, the IIS then returns the HTML to the browser.

**Advantages:** There are several advantages in using Server side scripting, as response to the client can be highly customized, the data accessed by user is based on the privileges given according to the user role, the html page can be presented so that data specific to a particular client can be shown, the data that is submitted by the user can be stored in database and the data can be queried and presented to the user depending on user request. For security reasons the server side scripts cannot be viewed in the client side html page. The client request and server responses are handled by the HTTP (Hyper text transfer protocol) and FTP(File Transfer Protocol).

2.1.2. Architecture of Web Application

The web-application structure can be divided into different tiers dependent on the functionality of each tier.

The evolution of the x-tier'ed applications started as

1. Single tier applications.
2. Two-tired applications.
3. Three-tired applications

**Single tier applications:** As shown in Figure 1, in these applications presentation logic, business logic and the data logic are placed in mainframe server. The processing time is distributed to various dumb client machines based on a time-sharing algorithm. Applications based on such architecture are obsolete these days. [3]

![Figure 1: Single-Tier Application Architecture](image-url)
**Two tier application:** As demonstrated in Figure 2, in these applications the presentation logic and business logic is placed in a single layer and the data layer is placed in another layer. This layer too has its own disadvantages as the business logic that is used to interact with the data layer is placed in the same layer as presentation layer so the database connections are handled in a haphazard manner. This will put more load on the database in connection handling, as much of the database resources would be expended for connection management when actually database resources should be devotced for query handling and data manipulation. [3]

![Two-Tier Application Architecture](image)

**Figure 2:** Two-Tier Application Architecture

**Three-tier application:** As shown in the Figure 3, this kind of application has a clear demarcation between presentation layer, application layer and data layer. The presentation layer is only limited to presenting the data and Graphical User Input to the end user, all the evaluation logic that interacts with the database is placed in a Application layer. The data layer only handles the request from the application layer. This model has many advantages where the application layer which is independent of presentation layer can be replicated at multiple machines depending upon the user load, the clients don't manipulate the database connections, so any modifications to the application logic can be handled independently of the GUI. Although the complexity to clearly define each layer could be difficult to design and implement. [3]
Figure 3: Three-tier Application Architecture
2.1.3 ASP.NET Architecture

ASP.NET is a server side scripting technology. ASP.NET is a application framework that was developed by Microsoft for building desktop applications, web-applications and dynamic websites. ASP.NET is based on Common Language Runtime (CLR) which gives developers the freedom to develop applications in multiple languages like Visual C#, VB.NET, Visual J#, Visual C++ and various other languages that are supported by the .NET framework.

.NET Run Time Environment:

Common Language Infrastructure (CLI): CLI is a specification of set of standards and rules, when applications are written conforming to the set standards of CLI, applications can be developed in multiple programming languages and can be executed in various platforms with out any additional customization of code specific to the platform [4]. CLI provides 2 step compilation procedure where in the first step the .NET Language is compiled to a neutral Microsoft Intermediate Language (MSIL) or Common Intermediate Language(CIL) and in the second step the CIL is converted to native machine code specific to the architecture of machine which in turn is executed to produce desired output [4] [5].

Common Language Runtime (CLR): CLR is the run time environment that is provided by the .NET frame work. In the CLR application source code is converted to Microsoft Intermediate language (MSIL) or the Common Intermediate language (CIL). This intermediate assembly language is neutral of the application source code. When the application is run then the Just in time compiler converts the Assembly language to the architecture specific machine language which is processed by central processing unit. The main advantage of CLR is the provision of interoperability. Applications could be developed in multiple languages as all the languages comply to the CLI so the intermediate language is language-neutral and more over the libraries or classes from one language could use the classes of other languages with out any additional overhead. [6]

ASP.NET Application Framework:

As shown in Figure 4 the top layer comprises of elements in ASP.NET application which include

1. configuration files which are XML files with extension .config as in web.config where we can specify all the configuration settings pertinent to the application.
2. Files with extension .asmx which are the end point for consumption of web services by another application.
3. Web forms which are the basic building block for application development in ASP.NET. These have the extension .aspx and classes associated with the web form has an extension of .aspx.cs. [8]
The second block is the ASP.NET run-time services which include security features, caching and providing state management using sessions and view state. The web forms and state management together encompasses the ASP.NET page framework. [8]

The third block is the ASP.NET base classes also called as base class library which is used by the ASP.NET run time services during compile time. The CLR is invoked during the application run time when the application source code from the top two layers is converted to the language neutral Intermediate language [8].

Figure 4: Elements in a ASP.NET Framework [7]
ASP.NET Web Page Processing

Step-1: The web page processing starts by the user typing http request in the form of a url in web browser (client) which sends request to the web server IIS (Internet Information Server) that is installed along with the Visual Studio IDE.

![Figure 5: ASP.NET Web Page Processing](image)

Step-2: As shown in the Figure 5 when the request from the browser arrives at the Web server (IIS) the IIS has to decipher the extension of the file that has sent the request and then IIS routes the file accordingly. If it has .aspx extension (in this case), IIS has a component aspnet_isapi.dll which routes the request to ASP.NET engine. This component acts as a bridge between the unmanaged code and managed code of ASP.NET engine. Before the control is passed to the ASP.NET engine the application object is instantiated and the .config files (both the machine.config and web.config) in the application are parsed to get a clear picture on how to handle this particular request.

Step-3: When request reaches the Web server (IIS in this case), then the request is routed to the .NET engine where the request passes through the HTTP Pipeline, as the request goes through a set of various HTTP modules as shown in Figure 6. HTTP modules are basically a set of classes which have access to the incoming request and can manipulate the request which can affect its sequential flow through the HTTP pipeline. And after going through the HTTP Modules then the request lands into HTTP Handler which is responsible for producing the HTML output and delivering it back to the Web browser (Client). HTTP modules can be modified by manipulating the machine.config file associated with the web application otherwise the default HTTP modules that are executed are enlisted in the
LocalDrive\Windows\Microsoft.NET\Framework\VersionNumber\Config\machine.config under the <httpmodules> section. [10]

As depicted in the Figure 6 HTTP Handlers on the other hand act as the final transit point for the HTTP request. When the request reaches the HTTP handler its responsibility is to deliver the output. If the request is a web page then all the web controls in that page should be changed to HTML and delivered as output. If the arrived request is a web service the relevant method has to be executed and the output from that method is delivered in the SOAP format. HTTP handlers are classes which get executed when the request reaches them or they can be factory handlers which merely return an instance of HTTP handler. HTTP handlers that are executed by default can be found in this directory LocalDrive\Windows\Microsoft.NET\Framework\VersionNumber\Config\machine.config under the <httphandlers> section. [10]

![HTTP Pipeline inside .NET Engine](image)

**Figure 6:** HTTP Pipeline inside .NET Engine [10]

HTTP modules and HTTP handlers can be customized to affect the application flow and cater to the specific needs of the application. If the httphandlers enlisted in the machine.config are modified it affects all the applications running under the web server, if the httphandlers enlisted in the web.config are modified then it affects the relevant web application. [10]

**Step-4:** When the request arrives from the Client to web server, which then is redirected to the HTTP Pipeline as shown in Figure 6. After passing through the http modules then the request is sent to the http handler. Then the http handlers takes the incoming request depending on the input generates the output and delivers it to the HTTP modules which is passed through various HTTP modules and delivered to web server (IIS). The IIS then delivers the output to the client (which is normally web browser). [10]
Features of ASP.NET

Compiled Code: The code in ASP.NET is compiled not interpreted as in ASP scripts which makes the ASP.NET applications run faster. [11]

Enriched Tool Support: The Visual Studio Integrated Development Environment (IDE) has Tool box which gives the developer an opportunity to drag and drop various web controls, HTML Elements. As a result the application development time gets faster. [11]

Flexibility: As ASP.NET applications run on CLR, the developers can choose to work on any language supported by the .NET Framework, which gives the developer an enhanced flexibility during application development phase. [11]

Simplicity: ASP.NET has a clear demarcation between the application layer and presentation layer. More over ASP.NET has automated features like garbage collection, authentication of client, web site configuration and deployment, Form Submission. [11]

Security: ASP.NET provides security for the applications developed in Visual studio by implementing user privileges and roles. [11]

Extensibility: ASP.NET provides a plethora of features to extend the features of application. ASP.NET gives the developer scope to create custom controls, create custom HTTP modules and HTTP handlers to alter the flow of application path in HTTP pipeline. [11]
2.2 Database Management Systems (DBMS)

A database is storage of inter-related persistent data. The database systems give us a provision to store the data and management of these database systems include giving us a possibility to manipulate and access the data through various database functions like Create, Retrieve, Update and Delete and also providing structures so that we can store the persistent data. The data should be persisted in a way that it with stands system crashes, provides security against the potentially harmful users and anomalous data should not be stored when the database is being accessed by multiple users. [11]

Database model defines the underlying infrastructure on which the database is constructed. It defines the conceptual rules concerning the database behaviour, such as data storage, data manipulation, data constraints. The underlying data model defines the type of database management systems.

There are a variety of database management systems categorized on the database model. Two of the popular database management systems are

1. Relational Database management Systems.
2. Object oriented Database management Systems.

2.2.1 Relational Database Management Systems (RDBMS)

Relational Database Management Systems incorporates the software and structure to handle and maintain a Relational database. Relational Database is a logical grouping of tables with unique name for each table. Each table holds certain number of unique columns. Each table is a set of relationships established between the columns and each row represents a singular relationship. There is a “correlation between the table in a database and the mathematical concept of relation”, that is the reason for calling the database based on relational data model as relational databases. And to access the database system we need Structured Query Language (SQL) a language which is used to access and manipulate the data in the database. RDBMS forms the underlying foundation for the SQL. [14][17]

The most popular Relational Database Management System is Microsoft SQL Server. Most of the web applications developed using Visual Studio framework use Microsoft SQL Server as the Database Management System. Microsoft .NET framework has provided a set of class libraries to communicate with the Microsoft SQL Server, and databases that are exposed through OLEDB and XML. [15]

Figure 7: ADO.NET Architecture [16]
As shown in Figure 7, to access data in the database primarily connection to the database has to be established through the command object which represents a SQL statement or stored procedure. As you can see the data can be accessed and manipulated either through the Data Reader object or through DataSet object.

**Data Reader:** Data Reader is provider dependent data object. For interacting through SQLClient Provider, the objects created to interact with SQL server would be SQLCommand, SQLDataReader, SQLConnection, SQLDataAdapter. For interacting through Oledb Data Provider, the objects created to interact with the data source would be OleDbCommand, OleDbDataReader, OleDbConnection, OleDbDataAdapter. Data Reader in general is a connected data object, which means that connection to the database should be open for it to fetch the data from data source. It is a read only, forward only data object. Data cannot be manipulated using Data Reader object. Data Reader fetches only one record at a time and when the read() method is called it clears the current record and bring the next record. Data Reader is much faster in accessing the data as there is less overhead in retrieving the information. [18]

**Data Set:** As shown in Figure 7 Dataset uses a intermediary Data Adapter to populate the results from the database into DataSet. And the changes that are made to the data in the DataSet like deleting some records or inserting a new record or updating the existing records with new values are actually reflected onto the database from dataset with the help of Data Adapter. Datasets are disconnected data objects as the data that is requested is populated into dataset with the help of connection object and then connection is closed. Datasets act like in-memory database because what ever constraints and data integrity rules that exist in the data source can also exist in the dataset. Data can be manipulated in the dataset and then be updated in the database, data can also be randomly accessed from the dataset. Datasets are independent of the data providers. It is the responsibility of the data adapter to convert the data provider dependent data into data independent data and place it in the dataset. But the only drawback is the performance when using datasets over a large data record set. [18]

![DataReader vs. DataSet](image)

**Figure 8:** DataReader vs DataSet [18]
The empirical experiments indicate that Data providers is thirty times faster than dataset and in the above Figure 8, is the graph to illustrate the performance where the pink line indicates the time taken by the dataset and blue line indicates the time taken by the time taken by the data reader on a graph with Time in sec on Y-axis and Number of records on X-axis. [18]

**Stored Procedures:** They are similar to the functions in any programming language except that they execute the group of SQL statements against the relational database. Stored procedures can return values in the form of output parameters and take input parameters as well, just as functions do. [28]

There are several advantages in using stored procedures when compared to using embedded SQL statements.

1. Single point of access: When a stored procedure is created it creates a single point of reference and the same stored procedure can be called multiple times with out replication of the SQL code.
2. Increased Query Response Time: The stored procedure when executed for the first time is parsed and and compiled code is cached so when the stored procedure is executed again the cached compiled code can be used with out having to recompile which improves the query response time.
3. Reduced Network Traffic: The network traffic is reduced as the control is transferred only once to stored procedure for execution and the results are sent at the end of the execution rather than data being ferried for each and every SQL statement. Moreover only the name of the stored procedure and input parameters (if any) are sent from Business layer to data layer, so this reduces network traffic as we are not sending any SQL queries and we get the results back as output parameters. This also improves the performance as a result of this reduced network traffic.[28]

**Language Integrated Query (LINQ):** Usually Queries to the database require learning a query language to interact with data source. And the queries that are used during the development lack any "compile-time checking or intellisense". From .NET 3.5 framework Microsoft has introduced a Language integrated Query commonly called as LINQ so that queries to the datasource can be written as part of the application development language. LINQ also has intellisense feature. Previously there was an additional overhead to learn the query language dependent on the data source. For example to access data from a SQL server SQL has to be learnt, for interacting with a XML file XQuery has to be learnt but with the advent of LINQ the same query language could be used to interact with any data source be it XML files, SQL Server, objects or any other database that has LINQ provider, as the LINQ queries interact with objects in the queries. [19] [20] [21] [22]

**LINQ to SQL:** To interact with the SQL Server, Microsoft .NET framework has provided a set of class libraries grouped as a LINQ to SQL component. The LINQ to SQL provides functionality to map the Database tables into LINQ to SQL entities and associations through the Object relational designer (O/R Designer). O/R designer is a visual designer where we can drag and drop the database tables onto the entities pane of the visual surface. We can also drag and drop the stored procedures and views from the database onto the methods
pane of the visual surface. As soon as we drag and drop the database tables (database objects from LINQ point of view) onto the methods pane a DataContext is created which acts as a monitoring authority for all the entity objects that are created under a particular database connection. It keeps track of all the changes made by the entities and submits those changes to the LINQ to SQL component which converts the data from an object type to the data compatible to the SQL Server. The data context also holds a “Identity cache” which is a grouping of individual identities created for each entity object that is created while dragging and dropping the tables onto the entities pane, no matter how many times an entity object is used during development the same identity would be reused without any additional overhead. The LINQ to SQL component acts as an interface between the data context object and the SQL Server database, its main functionality would be to convert the relational data to object data when submitting the data to DataContext, and it would be converting the object data back to the relational data when submitting the data to SQL Server database. [19] [20] [21] [22]

![Object Relational Designer](image-url)

**Figure 9:** Object Relational Designer
As shown in the Figure 9 when the tables from the server explorer side window are dragged and dropped onto the methods surface pane a DBML file is created under which the “.designer.cs” file is created, which is a translation of the relational Database schema to object model which in turn is C#/VB.NET understandable code. The code is generated in the following steps as shown in Figure 10 where the DBML schema is extracted into XML file which is sent to XSD Validator which validates the XML Schema Document (XSD) and if it conforms then it is sent to DBML Validator where the XSD is validated for any errors or warnings, if it passes then the XSD is sent to code generator.[23]  

![Figure 10: DBML Extractor](image) 

As shown in Figure 11 the code generator simply converts the dbml file to relevant file of C#/VB.NET type. During the run time the file is sent to the VB.NET/VC# compiler and if it has no errors it is sent to the LINQ to SQL component which converts the object data to the relational data and sends to the SQL Server database. [23]  

![Figure 11: Code generator](image) 

### 2.2.2 Object Oriented Database Management Systems (OODBMS)

Object Oriented Programming Languages have become the basis for most of the web application development. The application developers usually model their objects to resemble the real life model. There are concepts like encapsulation, inheritance, polymorphism which the developer would be using to produce the desired structure. And for the data persistence, mostly application layer related application development has to focus with the relational database management systems. The object model in the application layer has to be mapped onto the relational model of the Database, this is often referred to as impedance mismatch. More over the objects in the domain model has dependencies and relationships with other objects, these relationships between tables in relational database
have to be modelled in accordance with the object model in the application domain. This mapping is necessary, because when the data is retrieved from the database it has to reconstruct the exact object with its inter dependencies. Some relationships like inheritance cannot be mapped onto the database structure as relational databases don't support it. [24]

**Transparent Persistence:** The usual practice to interact with the databases is to use Structure Query Language (SQL). The problem with this is impedance mismatch as discussed earlier. The run time errors and the syntax related errors are unknown until the application is run. There is no transparency into the working of the SQL commands that we run from the application and how they interact with the Database.

Transparent persistence is introduced to resolve these problems. In this concept we interact with the database using our application development language, there is no additional overhead of using SQL. We use objects to retrieve and manipulate the data in data source. To accomplish this we need the help of Object Relational mapping (ORM) software which acts as a kind of adapter between the Application objects and Database, in case of .NET framework LINQ to SQL is an example and in Java platform nHibernate is such example. But using Object Relational mapping software we are not solving the real problem we are just shifting the problem to another component which is doing the object to relational mapping in an XML file. The application performance will get degraded as the ORM software has to generate to do the mapping behind the scenes every time there is an interaction between the Application and Database. [25]

**Database for Objects (Db4o):** This is an example of a Object Oriented Database. In Db4o the objects from the application can interact with the database in their native state. There is no conceptual difference between the database model and the application domain model. It has a low footprint, has maintenance requirement simple to use, short learning curve. The only action required during development is installing a add-on to visual studio IDE, and use it as a database from the application. There is also a object manager which gives an opportunity to edit and delete the objects that are stored in the object files (tables from an Relational database terminology). [29]

There are different ways to query the Db4o database:

1. **Query By Example:** In this a template, an object is created for the object on which the search is based. If the object that is being searched has some attributes then we can isolate the search based on those attributes otherwise the search will return all the objects of the search object type. The result of the query should be stored in IObjectResult as shown in following code which is adapted from [29]. The main drawback with this is lack of advanced querying capabilities. [29]

   ```java
   Person p1 = new Person("Sree", 29);
   IObjectResult result = db.QueryByExample(p1);
   ListResult(result);
   ```

2. **Native Queries:** This is the preferred way to interact with the database as the queries are type-safe, pre-compiled and the queries can be written as native language of application development. The Native queries have to be converted to SODA queries to execute and obtain the results, the performance due to this translation is often times negligible, but may get deteriorated when executing more complex queries[29].
An example of the native queries is shown below which is adapted from [29].

```csharp
IList <Customer> result =
    db.Query<Customer> (delegate ( Customer cust)) {
        return cust.Age > 25 && cust.Age < 40
        || cust.Name == "Rachel";
    });
```

3. **LINQ Queries:** Db4o can also be queried using LINQ queries the syntax is similar to the normal LINQ queries with the only difference being adding two namespaces in the .cs file: [29] An example of the LINQ queries is shown below which is adapted from [29].

```csharp
using Db4objects.Db4o;
using Db4objects.Db4o.Linq;

IObjectContainer loginCustomerDb =
    CustomerConnection.CustomerClient;

var result = (from Customer cust in loginCustomerDb where
    cust.UserName == userName && cust.Password == password  select
    cust.FirstName).Count();
```

4. **SODA Queries:** The low level API of the querying interface is SODA. The queries written in the SODA are converted into graphs with each node represented by constraints and elements of SODA queries as shown in Figure 12. The evaluation of constraints decides if the lower level of nodes have to be traversed or not and there by deciding which of the elements have to be in the result set. The drawbacks associated with this type of queries is the fields are searched with the strings, so they are not pre-compiled, and they are not type-safe, as a result the queries are error prone. [29]. An example of the SODA queries is shown below which is adapted from [29].

```csharp
Iquery query = db.Query();
query.Constrain(typeof(Customer));
query.Descend("_Name").Constrain("Rachel");
IobjectSet result = query.Execute();
ListResult(result);
```
Client/Server mode in Db4o:

In this mode there will be multiple client processes which are usually requests emanating from the web browsers (in case of web applications) which will be reaching the Db4o Server process which will be constantly listening for the incoming connection requests as shown in Figure 13. The server then provides the access to Db4o object Database so that Clients can perform CRUD operations on the Database. [30]
There are three variants in the Client/Server mode.

1. **Embedded Server**: In this mode the client and server are located on the same machine, although client and server run as different processes. This kind of methodology is mostly used for the desktop applications. This kind of methodology can be used in scenario's when the application can be developed as a desktop application but later with a changed requirement scenario or during deployment phase if the application need to be changed as server on one machine and client processes running on different machines then this could done with minimal alteration of code with only change being adding of port numbers and IP address of host machine. [30]. A syntax of this implementation is shown below which is adapted from [30]. As shown below in the code a server is opened on a particular Db4o file which has an extension .yap, zero specifies that the file is situated in the local machine. Then on this particular server instance a client connection is opened to interact with the yap file.

```java
// accessLocalServer
IObjectServer server =
Db4oClientServer.OpenServer(Db4oClientServer.NewServerConfiguration(),
YapFileName, 0);
try
{
    IObjectContainer client = server.OpenClient();
    // Do something with this client, or open more clients
    client.Close();
}
finally
{
    server.Close();
}
```

2. **Networking Mode**: This mode is similar to the Embedded server mode except that the client and server are located in different machines. We also specify the Port number, host IP address. The client interacts with server through TCP/IP protocol. [30]. The syntax for implementation of this mode is shown below which is adapted from [30]. In the following code we create a server with similar configuration as we have created in the embedded mode with only difference being specifying a port number instead of zero (in the earlier case). Then we specify the user name and password on the server port. The client is created on this server instance where we specify the IP address of the machine on which server is running in the network, then we specify the port number. The user name and password are also specified along with this information which should match the credentials specified by the server machine.
// accessRemoteServer

IObjectServer server =
Db4oClientServer.OpenServer(Db4oClientServer.NewServerConfiguration(),
    YapFileName, ServerPort);
server.GrantAccess(ServerUser, ServerPassword);
try
{
    IObjectContainer client =
Db4oClientServer.OpenClient(Db4oClientServer.NewClientConfiguration(),
    IPAddress, ServerPort, ServerUser, ServerPassword);
    // Do something with this client, or open more clients
    client.Close();
}
finally
{
    server.Close();
}

3. **Out Of Band Signalling:** This kind of communication is done in asynchronous
mode. This mode can be useful when client wants to send special messages to server
like if the client wants the server to shut down or do some special operations like de-
fragmentation etc. [30]
3. Web Store

This web application allows the registered user to select software applications from the grid view and then add them to the shopping cart. After adding the items to the shopping cart, the user can edit the items. After adding and editing the items to the shopping cart, the user proceeds to check out the items. Although the payment process is dummy, the user can view the total cost of the software components and enters a dummy credit card number to take him to the orders list page from where the user can download the selected software components. The application also archives the software components that were bought by the user, the logged in user can view list of all the components that were previously bought by him ordered by data of transaction.

The other user role is administrator who is a root user. As the user logged in as administrator can add new software components, set the attributes of the software components. The administrator can disable or re-enable the software components and there by setting the visibility of the software components in the grid view. The administrator can also view the list of transactions made by all the users ordered by date of transaction.
3.1 Application Architecture

The web store has three tier architecture in which the presentation layer is placed in the .aspx forms where as the business logic and the data logic is placed in the .aspx.cs side. The database tables and object files are placed in the database. Upon closer look the web store has the following structure as shown in Figure 14.

Figure 14: Application Structure

The Login.Master is the root master page, The Admin.Master and Webstore.Master inherit Login.Master. Login.Master basically contains the functionality to login and register. Admin.Master inherits Login.Master and the defines the sidebar to browse to various web pages that are under the purview of the administrator. Webstore.Master defines sidebar to browse between the web pages that are under the purview of Registered User. More description on the web forms under the Admin.Master and Webstore.Master are given in the next section 3.3 User Interface with clear pictorial description.
3.2 User Interface

The Web store is implemented in three methods for database access namely

1. SQL Server 2005 with stored procedures
2. LINQ to SQL
3. Db4o with LINQ queries

The Graphical User Interface is common in these three methods for database access only the underlying architecture to query the database be it Relational Database (SQL Server 2005) or Object oriented database (Db4o) that is varied.

The user who visits the web application would be viewing the start.aspx by default which allows the user to either log-in or register (if he is not a registered customer) and view all the available software components along with its attributes as shown in Figure 15. The user should be registered in order to download the components. Total downloads specifies the total number of times a particular component has been downloaded by the users.

![Start.aspx](image)

**Figure 15:** Start.aspx
Register.aspx: If the user is not a registered customer the user can register by clicking on the register link button on the start.aspx which takes him to the register.aspx where he can provide all the details to get registered as shown in Figure 16. The username has to be between 6 and 14 characters. It has to start with an alphabet and only "_" and "." are allowed as special characters. The password can start either with an alphabet or numeric and the password has to be between 8 and 14 characters. And again the password is only allowed to have "_" and "." as special characters.

![Register.aspx](image.png)

**Figure 16: Register.aspx**

Login.aspx: After the user gets registered the user can login with his username and password by clicking on the login link button on Register.aspx as shown in Figure 16.

The user can login with two different user roles

1. Administrator
2. Customer

If the user is logged in as Administrator these are the pages that are under his purview

1. Admin.aspx
2. CustomerTransactions.aspx
3. AddNewComponent.aspx
**Admin.aspx:** In this page the Administrator can disable or re-enable the software components that are displayed in the gridview as shown in Figure 17. If the software component is disabled by clicking on the checkbox, it is not displayed in the grid view on the Start.aspx. The administrator can re-enable a component by unchecking the checkbox. The component that is disabled is unavailable only for the new users but the existing users would still be able to download the component from their transaction history page MySoftware.aspx. The administrator can edit the attributes of the components displayed in gridview and update the attributes of the Software components displayed in gridview. The administrator would be able to navigate to other pages “CustomersTransactions.aspx” and ”AddNewComponent.aspx” from the sidebar available in Admin.aspx.

![Image of Admin.aspx](image)

**Figure 17:** Admin.aspx

**Customers Transaction.aspx:** In this page the administrator can view all the records bought by all the customers in the form of gridview as shown in Figure 18 ordered by date of transaction in descending order. The customer transaction history is grouped by the date of transaction and invoice number with total amount of all the components bought with that particular invoice number.
Figure 18: CustomerTransactions.aspx

Figure 19: AddNewComponent.aspx
**AddNewComponent.aspx:** In this page the new Software component and the description can be uploaded to the special folder in the root directory of the application. The component name, file size are extracted on uploading the software component. The user can set the other attributes like price, edit the component name and description if needed and then insert the details into Components table of the database as shown in Figure 19. By default the number of downloads will be zero. On adding the new component and its attributes, immediately the new component is visible in the start.aspx grid view.

If the registered customer is logged in then he will be accessing the following pages:

1. Start.aspx with sidebar
2. ShoppingCart.aspx
3. CheckOut.aspx
4. MySoftware.aspx
5. OrderDetails.aspx

**Start.aspx:** When the registered user logs in with his credentials he would be re-directed to the start.aspx with the side bar to navigate to other pages as shown in Figure 20.

![Start.aspx](image)

**Figure 20:** Start.aspx

**ShoppingCart.aspx:** After the user selects required software components they are added to the shopping cart and after adding them to the shopping cart, the user can edit the shopping cart by navigating to ShoppingCart.aspx by clicking on the side bar link button “Edit Shopping Cart”. As shown in Figure 21 the shopping cart can be edited by clicking on the remove button, which removes the particular item from the shopping cart.
Figure 21: ShoppingCart.aspx

Figure 22: CheckOut.aspx
Checkout.aspx: After editing the shopping cart the user can navigate to the CheckOut.aspx as shown in Figure 22 where he can view the total amount incurred, checked out items and submit the dummy credit card number to be able to download the items. On submitting, a unique Invoice number is created for every transaction and stored in database. This unique Invoice number would be used to distinguish between the various transactions.

OrdersList.aspx: The user navigates to this page after submitting the credit card number (dummy) in the Checkout.aspx, in this page he would be able to download the selected software components by clicking on the download link button as shown in Figure 23.

![OrderDetails.aspx](image)

**Figure 23:** OrderDetails.aspx

MySoftware.aspx: The customers records are archived and stored as history, when the user logs in later, he would be able to view and download all the previously "checked out" software components ordered by the data of transaction in descending order as shown in Figure 24.
**Figure 24:** MySoftware.aspx
3.3 Database Design

For the web store, database design for the three different database access methods is same. Specifically the database design for SQL Server, LINQ to SQL and Db4o are similar with only the underlying database querying that differs. The terminology is different, for example the tables in SQL Server database are referred as Object files in Db4o.

The Database has three tables (object files in case of Db4o) namely Customer, CustomersRecord, Components.

**Customer:** It has the fields SNo, First Name, Last Name, UserName, Password, Sex, Age which stores all the details of the registered customer. The same UserName, Password would be used as an identity information for authentication of an individual customer during log-in and to associate all the Customer transaction History pertinent to the particular user.

**Component:** It has the fields SNo, Component Name, Size (In MB), Total Downloads, FilePath, Price (in SEK), Description, Disable Flag. The Component table stores the relevant information for each individual software component that allows the user to download. The Size (in MB) has the file size of each software component. FilePath stores Component Path in this format “~/ComponentsFolder/CCCleaner.exe” where tilde (~) is used to specify that the Software Component is stored in the root of the web application. Total Downloads indicates the count of number of times an individual software component is downloaded by the registered customers. It maintains this count by counting all the occurrences of the software component in the CustomersRecord table (object file in Db4o). DisableFlag is a boolean variable when it is false (when the check box is unchecked) the component is visible in the grid view of the start.aspx and when it is true (when the check box is checked) the component is invisible in the grid view of the start.aspx. The visibility of the component is set by administrator. Even though the component is invisible in the grid view of start.aspx, the software component is still available for download by the logged in user from the registered user's transaction archive that is available from his MySoftware.aspx web page.

**Customers:** It has the fields SNo, CustomerName, Invoice Number, Date Of Transaction, Software Title, Total Amount. It archives the individual registered Customer's checked out items so that he might be able to view the history of his transactions and be able to download the software components. This Customers Record table also prevents the user from checking out the same software component twice. When the user checks out the items from the shopping cart, the Customer Name, an unique invoice number (that is created for each user transaction), date of transaction, software title, price for each software component is inserted in the Customers Record database table (Object file in Db4o). This information would be used to show the record history for logged in registered user by grouping the data of transaction and invoice number and by invoking the sum of all the software components that have same invoice number and date of transaction. This information is viewed on the MySoftware.aspx page.
3.3.1 SQL Server

**Figure 25:** Relational Database Design

**Using SQL DataReader with Stored Procedures:** It has the database design as depicted in the Figure 25. The customer and CustomersRecord have one to many relationship as one customer can have multiple customer records. The CustomersRecord and components have many to many relationship as customers records maintain a multiple copies of the components and vice versa. The select, insert and update functions for data access and data
manipulation have been performed using the Stored procedures with SQL Data Reader class.

**LINQ to SQL:** In this same relationships hold between the database tables as shown in Figure 26. The Object to relational mapper comes into play here. It converts the database tables shown in the above Figure 25 to the object mapping as shown in Figure 26 while holding the relationships between the objects. Here LINQ queries are used to perform the operations on translated object model for data access and data manipulation which in turn are converted to the SQL queries by LINQ to SQL component.

### 3.3.2 Db4o

![Figure 26: Object Oriented Database Design](image)

---

37
In Database for objects (Db4o) the database holds object files which in turns holds objects. The Customer, CustomersRecord and Components are referred as database object files. The same relationships exist between the object files as shown in Figure 27. The object files have the same functionality and they have the same behaviour of persisting the objects.
Db4o provides various querying interfaces like Query-By-Example (QBE), Native Queries (NQ), LINQ and the SODA API of which LINQ is the preferred querying interface for its advanced querying capabilities, pre-compilation, querying optimization mechanisms. In this there is no need for the data modelling or Object to relational mapper. The object model for the LINQ and the Db4o are similar except that in LINQ to SQL the object model has to be converted to relational model for every interaction with the SQL Server database and the output from SQL Server has to be converted to the application compatible data. This translation is not required in Db4o as the application object model and data model are similar. This solves the problem of impedance mismatch.
3.4 Implementation

As explained earlier the user interface is similar in all the three different variants i.e., Web store using SQL server with stored procedures, LINQ to SQL and Db4o.

**Project Environment:** The project has been developed in Visual studio 2008 IDE with ASP.NET using C# on Vista Operating System. The databases that are used are SQL Server 2005 and Db4o 7.2.

3.4.1 Getting Started with ASP.NET Web-applications

**Creating Web Application:** Choose Create Project, in the pop up window select ”web” from the project types and from the Templates sub window choose the ASP.NET Web Application as shown in Figure 27. And give the name of your choice.

![Creating Web Application](image)

**Figure 27:** Creating Web Application

After creation of the web application, then default.aspx webform is created with the code behind cs file with default.aspx.cs. To create another web form right click on the namespace
which opens a context menu with option "Add New item" as shown in Figure 28. On clicking you will be able to add new web form or new master page. The new classes can be created by the same context menu by clicking on the Class option as shown in Figure 28.

Figure 28: Adding a new web form or Master Page

Adding Web Server Controls: The new web controls can be added by dragging and dropping the elements from the tool box as shown in Figure 29 or the new web server controls can be created manually by the code, for example if the new grid view has to be created in the web form either the grid view can be dragged and dropped from the tool box which results in the following code

```xml
<asp:GridView ID="GridView1" runat="server">
</asp:GridView>
```

or the new gridview can be created by the c# code as

```csharp
GridView gridview1 = new GridView();
```
3.4.2 WebStore using SQL Server 2005

The SQL Server 2005 provides a management Studio which can be downloaded from Microsoft.com/downloads. The management studio provides the necessary tools to create new database, new database tables. The developer can connect to the database server either through SQL Server authentication or windows authentication. The new database can be created by right clicking on the databases node on the object explorer sidebar. The new table can be created by right clicking on the tables node under the database created as shown in Figure 30. Later on the fields for the database table can be set by the prompting window where the Column name and data type can be set.

To connect to the database that is created through the SQL Server Management Studio from the ASP.NET application, the following xml node is changed in the Web.config file.

```xml
<connectionStrings>
  <add name="componentConnString" providerName="System.Data.SqlClient"
   connectionString="Data Source = SDANTURTHI-PC; Initial Catalog = ComponentDB;"
</add>
</connectionStrings>
```
Integrated security= SSPI; MultipleActiveResultSets=true"/>
</connectionStrings>

Figure 30: Creating new Database Table

Later on componentConnString is used in the C# code to extract information from the Web.Config <connectionStrings> in the following manner:

```csharp
string connectionString = ConfigurationManager.ConnectionStrings["componentConnString"].ConnectionString;
```

In the web application stored procedures were used as they are more efficient than embedded SQL code for the reasons mentioned earlier in the section 2.2.2 under Stored Procedures.

For example a stored procedure can be created by right clicking on the “stored procedures” under the programmability node which enables you to create new stored procedure where you can write the stored procedure. A sample stored procedure would look like as follows. The name of this stored procedure is “NewUserInsertion” and it has several IN parameters like fName, lName, uName, pwd, sex, age which takes these values from the C# code and inserts the values into the SQL Server Database table Customer.

```sql
Create procedure [dbo].[NewUserInsertion] (@fName nvarchar(max), @lName nvarchar(max), @uName nvarchar(max), @pwd nvarchar(max), @sex varchar(max), @age int) as insert into [dbo].[Customer] values (@fName, @lName, @uName, @pwd, @sex, @age)
```

This stored procedure is used in the C# code as follows which basically creates a new connection while supplying the “connectionstring” which is a information that is in the Web.Config <connectionStrings> node. Later on the stored procedure name along with the
Connection is supplied to the SqlCommand class. And then we supply all the parameters in the stored procedure with the values and execute the stored procedure against the database as shown below in the code.

```csharp
Connection con = new Connection();
string conString = con.connectionString();
SqlConnection newConnection = new SqlConnection(conString);
SqlCommand cmd2 = new SqlCommand("NewUserInsertion", newConnection);

cmd2.CommandType = CommandType.StoredProcedure;

cmd2.Parameters.Add("@fName", SqlDbType.VarChar).Value = fName.Text.Trim();
cmd2.Parameters.Add("@lName", SqlDbType.VarChar).Value = lName.Text.Trim();
cmd2.Parameters.Add("@uName", SqlDbType.VarChar).Value = uName.Text.Trim();


cmd2.ExecuteReader();
```

3.4.3 WebStore using LINQ to SQL

It's another technology to query with the SQL Server database. The advantage in using LINQ to SQL is using the same native development language with out any need of learning a query language to interact with the SQL Server database.

![Image of LINQ to SQL demonstration in Visual Studio](image_url)

**Figure 31:** Adding DBML class to ASP.NET Web Application
To use LINQ to SQL queries in the ASP.NET web application, it requires creating an Object relational mapper, which is defined through the .dbml classes. For creating .dbml classes, right-click on the namespace in the solution explorer, which will open the context menu to “add new item”. On clicking “Add new item”, it opens a new window as shown in Figure 31. To create dbml class, select the LINQ to SQL classes option, which will result in a new form with Server Explorer. When we click on the Server Explorer, it opens the existing Data Connections from where we can browse to the appropriate Database and relevant tables. Then we can drag and drop the tables onto the dbml pane as shown in Figure 32.

**Figure 32:** Adding Tables to the DBML Pane using Server Explorer

Again referring to the Figure 32 along with .dbml class .designer.cs classes are also created.
from where we get the reference to the datacontext class, here in this case the “ComponentDBDataClassesDataContext” provides the main entry point to the LINQ to SQL framework.

For inserting the data into Customer database table entered by the user for registration in Register.aspx would look like:

```csharp
ComponentDBDataClassesDataContext db = new ComponentDBDataClassesDataContext();
string firstName = fName.Text.Trim();
string lastName = lName.Text.Trim();
string userName = uName.Text.Trim();
string password = pwd.Text.Trim();
string selectedSexValue = sexList.SelectedItem.Text.Trim();
int ageValue = Convert.ToInt32(age.Text.Trim());
Customer cust1 = new Customer { FirstName = firstName, LastName = lastName, UserName = userName, Password = password, Sex = selectedSexValue, Age = ageValue }; 
db.Customers.InsertOnSubmit(cust1);
db.SubmitChanges();
```

In this case we create a datacontext object and then we create a object for the Customer class that was mapped by DBML LINQ to SQL class, assign the values that were entered by the user to the properties of the customer class and then use the datacontext instance to insert the object properties in the database with the help of Object Relational Mapper. Later on we commit the changes done to the database.

### 3.4.4 WebStore using Db4o

Db4o is used in this web application in the client/server Networking Mode. For more information please refer to the 2.2.3 Object Oriented Database Management Systems (OODBMS) under the section Client-Server mode.

The Db4o can be freely downloaded from the [http://www.Db4o.com/DownloadNow.aspx](http://www.Db4o.com/DownloadNow.aspx) and later on be installed as a plug in for the Visual studio 2008.

First of all object files (Database tables in SQL Server terminology) are nothing but class objects with properties. For example to create Customer object file (Database table in SQL Server terminology), we have to create a class in C# with all the properties which would later on be converted as fields in database object file. The following code is a sample (not complete) code from the customer class:

```csharp
public class Customer
{
    private string firstName;
    private string lastName;

    public Customer(string firstName, string lastName)
    {
        this.firstName = firstName;
        this.lastName = lastName;
    }
}
```
After the creation of the Customer object the “CustomerConnection” is created where the server and client are created. When ever the data access or data manipulation has to be made on the customer table, the “CustomerConnection” method is called to open server and client using the following code:

```csharp
string yapFilePath = context.Server.MapPath(ConfigurationManager.AppSettings["KEY_Db4o_CUSTOMER_FILE_NAME"]);
objectServer = Db4oFactory.OpenServer(yapFilePath, 8111);
objectServer.GrantAccess("anonymous", "everyone");

objectClient = Db4oFactory.OpenClient("localhost", 8111, "anonymous", "everyone");
context.Items["KEY_DB4O_CUSTOMER_CLIENT"] = objectClient;
```

where the `KEY_DB4O_CUSTOMER_FILE_NAME` stores the relative path to Customer object file. Then we open the server using the filename and the portnumber. The server is granted access to everyone as there are no restrictions for users to be able to register.

Then we open the client with localhost(which can be replaced with IP address if application is hosted on a production server), port number on which the server is opened, username and password (which are same as the one with which server instance is opened) which don't serve any purpose in this case and later on reference to the object client is maintained though context.items.

To insert values entered by the user from the register.apsx web form into Db4o database object file, the following steps are needed:

```csharp
string firstName = fName.Text.Trim();
string lastName = lName.Text.Trim();
```
```csharp
string password = pwd.Text.Trim();
string sex = sexList.SelectedValue.Trim();
int userAge = Convert.ToInt32(age.Text.Trim());
Customer cust1 = new Customer(firstName, lastName, userName, password, sex, userAge);
IObjectContainer registerConnectionDb = CustomerConnection.CustomerClient;
registerConnectionDb.Store(cust1);
registerConnectionDb.Commit();
```

Which involves creating an object of the Customer Class created earlier which takes all the values from the text boxes in the register.aspx web form. Later on “CustomerConnection” class is opened and its reference is stored in the IObjectContainer which is a interface for “storing, querying and deleting objects and to commit and roll back transactions”. [32]

To use LINQ queries which are the preferred way to interface Db4o database in the .NET platform we use two namespaces:

```csharp
using Db4objects.Db4o;
using Db4objects.Db4o.Linq;
```

A sample LINQ query in Db4o would look like any other LINQ query which would be:

```csharp
IObjectContainer loginCustomerDb = CustomerConnection.CustomerClient;
var result = (from Customer cust in loginCustomerDb where cust.UserName == userName
&& cust.Password == password select cust.FirstName).Count();
```

In the above query the CustomerConnection opens the static method CustomerClient which in turn opens a server and client for interfacing with the Customer Object File. The above query just checks if there a matching user name and password in the Customer Object file with the username and password provided by the user in the Login.aspx webform to be able to log in.
4. Discussion

The web store has been developed in three different methods for database access. The following technical characteristics have been compared among these technologies.

**Transparent Persistence:**

In relational database management systems like SQL Server the relational data has to be converted back to object model when the application needs to retrieve or manipulate the persisted objects from the database. The objects for the application have to be reconstructed with all the dependencies and relationships from the relational data which is a overhead. Moreover we interact from the application using Structured Query Language (SQL) which is not a native development language. It is hard to spot errors in SQL during compilation process.[33]

In ORM tools like LINQ to SQL the developer interacts with the database objects constructed by the mapping tool and interacts with them as objects in main memory using the application development language but behind the scenes the queries written to interact with the database are converted to SQL Queries and again the output from the SQL queries is converted back to the application compatible output by the LINQ to SQL component. This translation degrades the performance and the mapping has to be still maintained and kept in synchronization with the database and application changes which is an additional overhead. [33]

In Object Oriented Database Systems like Db4o the objects from the application model are stored in their native state in the database and the data from the database is retrieved in their native state without any need for conversion. The application can interact with the database objects as the main memory objects and the same application development language can be used to interact with the database as a first class language construct. [33]

**Connection Management:**

SQL Server and Visual Studio have been developed by the same vendor, this automates some of the development features and capabilities. SQL Server implements connection pooling which is a set of active connections that have been opened during the application runtime. When a SQL connection is closed, the connection information is cached and returned to a set of active connections in a pool, so when the same connection is re opened later, this cached information regarding the connection is used from the active connections in connection pool. This concept is referred to as connection pooling. Connection pooling saves a lot of overhead when dealing with SQL Server and thereby improves the performance and scalability. [34]

LINQ to SQL is rather unique way of querying the database as a first class language construct. The datacontext object is our entry point to the LINQ to SQL. It is primarily our object from which we enumerate, update, select results from the database. It can be treated as a resource which can be treated as a connection.open in SQL Server connection process.
But we don't need to close it every time we open it. The queries that are written in the LINQ get translated into SQL queries, after the query translation the SQL Connection is opened and the translated query is executed. After the query execution the SQL Connection is closed and result is returned. Although DataContext implements IDisposable interface, there is no need to call the Dispose() method to close the connection associated with the Datacontext object as it is already closed. [35]

In Db4o the connection information for the database object file is not cached any where, so when a connection is established with an object file, it is not disposed off but instead retained. Db4o allows simultaneous multiple connections and the connections are only closed during events such as application being closed, session out. These events are handled by implementing IHttpModule interface and the client and server connection is closed when these events occur.

Impedance Mismatch:

Impedance mismatch is the difference between the application model and database model. The application object model follows object oriented concepts like Inheritance, Polymorphism and encapsulation but the same concepts cannot be extended onto the Database model which requires radical shift in the way Database modelling is done to cater to the needs of the application model. [33]

Developing web applications with SQL Server as a back end database has this problem of mismatch which reduces the performance. The object data from the application has to be converted to the relational data to be stored in the database. The output from the relational database has to be converted again to the application model compatible data. A lot of emphasis is placed in the metadata modelling in the relational databases for implementing the database model in line with the application object model. This places lot of overhead and allocation of resources during the software engineering project life cycle. [33]

LINQ to SQL tries to overcome this problem with the help of Object relational mapper but it shifts the problem from the ambit of application developer to database administrator, but the problem still remains. The LINQ queries are converted to SQL queries and executed against the SQL Server and the output from SQL Server has to be still converted to the application model compatible data. This conversion of data degrades the performance. [33]

Db4o solves this problem the application object model and the database model are similar. There is no impedance mismatch here. [33]

Programmability:

Microsoft SQL Server has vendor support so lot of features like user management, security are supported by SQL Server. Db4o on the other hand does not have these features and have to be implemented by the developer. SQL Server is a Database management system, so there is in house functionality and support provided for the web applications development using SQL Server in Visual Studio. This support is lacking when developing web applications with Db4o in conjunction with Visual Studio. Moreover Db4o is a Object database and it is not a database management system.
Isolation:

Isolation is one of the ACID (Atomicity, Consistency, Isolation and Durability) properties. When concurrent transactions try to access the records in a database, Isolation level decides when the changes committed by one transaction are visible to other transactions. [38]

Both SQL Server and Db4o have read committed isolation level. This concurrency level is lock-based meaning that when a transaction tries to access a data record it obtains both write-lock and read-lock on the data record or set of data records. The write-lock is not released until the end of the transaction but the read-lock is released as soon as there is select operation to be performed by another transaction simultaneously. This relaxation of read locks lead to non-repeatable reads.[38] [39] [40]

Indexing:

In SQL Server there are two ways indexes can be implemented using clustered indexing and non-clustered indexing. Both kinds of indexing use b-trees to arrange the columns that are used for indexing. In non-clustered indexing the column that is set as index is sorted and contains reference to the corresponding row entries. While the clustered indexing contains the entire row of the indexed column sorted. Clustered indexing is much more efficient than non-clustered indexing but the problem is maintaining an additional indexed table with the entries. [41]

Db4o has the capability of indexing for speeding up the data retrieving and searching capabilities. Db4o inherently uses b-tree for storing the columns used for indexing as a separate object-file which will be referenced during data retrieval or searching. There is a trade-off that should be kept in mind upon performance enhancement due to indexing but also a overhead for maintaining multiple indexing tables for speedy data retrieval and search. [42]

Concurrency:

In SQL Server the default concurrency level is optimistic concurrency level. It is implemented using the concept of row level Versioning. In this concept multiple copies of the values in the table are maintained. When ever a value is being updated the old values are retained and the new value still maintains a link to the old value in the form of linked list.

So when ever there is read request on a data (new version) with write lock, the old version of the data is returned. In this still simultaneous write requests are not possible and need to be in the wait state when there is write lock on a particular data record. The trade-off between overhead of updating a record has to be kept in mind as the old versions of the data have to be linked with the new versions and also maintaining multiple versions of the data in the form tempdb table has to be understood. [43] SQL Server is multi threaded, so there is adequate mechanism to address and handle deadlocks, race conditions when multiple threads try access the SQL Server.

In Db4o the concurrency mechanism is quite limited which is the biggest drawback while using Db4o with web applications. As Db4o is single threaded the concurrency is quite limited and the performance is degrading. Db4o uses overly optimistic control as default
meaning the collisions are neither detected nor avoided as said earlier Db4o core is inherently single threaded so only one client is handled at a time. In this mechanism when a object in a object file is to be updated, the write lock is obtained, then all the concurrent transactions are kept in a wait state until the transaction is completely finished and committed. After committing the transaction then a commit-time roll back is done, in which the transactions that are kept in the wait state (transactions that have subscribed for the event) are notified about the completion of commit operation. [44]

**Performance:**

When considering SQL Server with stored procedures the performance is superior when compared to LINQ to SQL because in Stored procedures the network traffic is minimal as we call the stored procedure with its name and send the input parameters over the connection. The stored procedure is executed on the database server and the output is sent over the connection which minimizes the traffic and improves performance. In LINQ to SQL the LINQ queries are converted to SQL queries and the queries are sent over the network, the performance degrades for converting the LINQ queries to SQL queries and the SQL queries have to be sent over the network. Again the SQL query output have to be converted to object model compatible data. This degrades the performance. [45] [46]

When considering the Db4o there is no conversion of LINQ queries (which is a standard way to query the Db4o database when developing applications with Visual Studio .NET environment) needed and the objects from Db4o database can be worked as in-memory objects. But the performance gradually degrades when the database size is over the 2-16 GB. The maximum file size is limited to 254 GB. The concurrency is also quite limited when working with multiple users. Db4o works particularly well with embedded applications, mobile applications, PDA applications. Db4o can be used as a caching mechanism for industrial applications when storing the data onto the relational databases. [47]
Building web application with SQL Server has lot of in-house support but at the same time lot of time is devoted to creating a correlation between the Application object model and database model. The same issue is dealt with in the ambit of LINQ to SQL as the Object to relational mapper, although it simplifies the job of developers to interact with the database, it shifts the problem rather than solving the core problem of impedance mismatch. With the Object Relational Mapper (ORM) lot of things happen behind the scenes, there is not a full understanding over the Object relational mapping from the developer point of view. And the performance degradation comes with the mapping issues. With Db4o much of the resources spent and maintaining on mapping issues and impedance mismatch can instead be devoted to building richer and powerful applications and it also cuts short the application development time. But there are some drawbacks with Db4o like limited database size to 2-16 GB, limited simultaneous access of multiple users to the web application. Db4o is not an substitute to powerful SQL Server though it has its advantages, but it can be considered as an alternative where the web applications are less user-intensive and have small data footprint.

More over the when the web applications evolve with time both the application data model and database model have to be synchronized at all times which could be a overhead with SQL Server. With LINQ to SQL the mapping has to maintained it addition to the database model. As the application data model and the database model are similar in object oriented databases like Db4o, it lead to easier maintainability of the changing requirements of the application. This transpires to lower maintenance and development cost with object oriented databases.

There are quite many features that need to be added to make this web application a full fledged credible application. The payment process has to be added, searching functionality of the components has to be incorporated. The application has to be deployed and empirical analysis has to be done on parameters like scalability, throughput, query read, inserting the data, searching the data. The performance measurement has to be done on these three distinct database methods.
References

1. Dave Ragget, W3C, Client-side Scripting and HTML, Available at http://www.w3.org/TR/WD-script-970314 accessed on 16th may 2011


Chapter-1 Introduction to Db4o, Page-5, 6, 7


29. Db4o Tutorial, Chapter-4, Querying.

30. Db4o Tutorial, Chapter-12, Client/Server.


35. Stephe Walther, ASP.NET MVC Dispose of your Datacontext (or Don't) http://stephenwalther.com/blog/archive/2008/08/20/asp-net-mvc-tip-34-dispose-of-your-datacontext-or-don-t.aspx accessed on 20th June, 2011


42. Stefan Edlich, Eric Falsen, RootSecurity, Getting started with Db4o, http://r00tsecurity.org/db/cheat-sheets/Programming/db4o/rc053-010d-db4o_1.pdf accessed on 21st June, 2011


