A TOOL FOR HETEROGENEOUS DATABASE COMMUNICATION

Krishna Deepak Gujjar
B.E., Visveswaraiah Technological University, Karnataka, India, 2006

PROJECT

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

FALL
2009
A TOOL FOR HETEROGENEOUS DATABASE COMMUNICATION

A Project

by

Krishna Deepak Gujjar

Approved by:

_______________________________, Committee Chair
Dr. William Mitchell

_______________________________, Second Reader
Dr. Meiliu Lu

_______________________________
Date
Student: Krishna Deepak Gujjar

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the Project.

__________________________, Graduate Coordinator       ________________
Dr. Cui Zhang       Date

Department of Computer Science
Abstract

of

A TOOL FOR HETEROGENEOUS DATABASE COMMUNICATION

by

Krishna Deepak Gujjar

Today there are several different databases systems in the market and many organizations work with multiple databases, heterogeneous in nature. Each of these systems store data and have applications running with them. In such a scenario there are instances where these organizations want their databases to interoperate and cross-populate the back end data repositories. Few of such instances when organizations would need heterogeneous database systems to communicate with each other are during mergers and acquisitions, during legacy system modernization or during an enterprise application consolidation. This project aims at implementing a tool which will help in transfer of data from one database to another database which is heterogeneous from the earlier one. The two heterogeneous databases that have been used in this project are MS SQL Server 2008 and MySQL 5.0.

The tool will extract the table definitions including all the primary key, foreign key, index definitions of a particular table or tables in MS SQL Server and convert that definition into a script that can be run on MySQL database to produce a replica of the table/tables present in MS SQL Server. The tool then transfers the data present in the
selected tables from MS SQL Server to MySQL. The tool will have an interactive GUI and will automate the entire process of data transfer with little manual intervention.

__________________________, Committee Chair
Dr. William Mitchell

__________________________
Date
ACKNOWLEDGMENTS

I am thankful to all the people who have helped and guided me through this journey of completing my Masters Project.

My sincere thanks to Dr. William Mitchell, for giving me the opportunity to work on my masters project under him and for guiding me throughout the project. My heartfelt thanks to Dr. Meiliu Lu for agreeing to be my second reader and providing me with her invaluable inputs on revising my report.

My special thanks to my friends Pramukh Jadhav and Samuel Mathison for helping me with their ideas and by reviewing my project report. I would also like to thank my roommates and all my friends who have been there for me throughout this graduate program at California State University Sacramento.

I would like to thank my family here in Sacramento, my uncle Poorna Kale, my aunt Nayana Kale and my cousin Ria for their love, support and encouragement.

Last but not the least I would like to thank my parents Jagdish Gujjar and Uma Gujjar, my brother Sandeep Gujjar and my grandmother Sundari Bai for their unconditional love. They have always motivated me and are the sole reasons for me to have come this far in life.
TABLE OF CONTENTS

Page
Acknowledgements .............................................................................................. vi
List of Tables ........................................................................................................ ix
List of Figures ....................................................................................................... x

Chapter
1. INTRODUCTION .............................................................................................. 1
2. BACKGROUND .................................................................................................... 4
   2.1 Purpose and Scope ....................................................................................... 5
   2.2 Overview ...................................................................................................... 6
3. TECHNOLOGY SURVEY .................................................................................... 8
   3.1 Microsoft SQL Server 2008 ......................................................................... 9
   3.2 SQL Server Management Studio 2008 (SSMS 2008) .................................. 9
      3.2.1 Key Features ....................................................................................... 10
   3.3 MySQL 5.0 .................................................................................................. 11
      3.3.1 Key Features ....................................................................................... 11
   3.4 Query Browser ............................................................................................ 12
   3.5 Microsoft Visual Studio 2008 ..................................................................... 13
   3.6 VB.NET ....................................................................................................... 14
      3.6.1 Key Features ....................................................................................... 14
   3.7 MySQL Connector/NET 6.0.4 ................................................................... 15
4. SYSTEM DESIGN ............................................................................................... 16
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table Number</th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Table 1 MS SQL Server – MySQL Mapping</td>
<td>23</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Basic Design</td>
<td>17</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Functional Flow Diagram</td>
<td>26</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Code Snippet to Connect to MS SQL Server</td>
<td>28</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Example of a Connection String</td>
<td>28</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Code Snippet to Connect to a MySQL Database</td>
<td>30</td>
</tr>
<tr>
<td>Figure 6</td>
<td>MySQL Connection String</td>
<td>30</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Config Page</td>
<td>31</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Table Selection Form</td>
<td>33</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Data Present in the Table INFORMATION_SCHEMA.TABLES</td>
<td>34</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Flowchart of getListOfTables</td>
<td>36</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Flowchart of getSqlServerSchema</td>
<td>39</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Flowchart of CreateTableSchema (Part 1)</td>
<td>40</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Flowchart of CreateTableSchema (Part 2)</td>
<td>41</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Flowchart of ConvertSchema</td>
<td>43</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Flowchart of BuildCreateTableQuery</td>
<td>44</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Configuration Page Screen</td>
<td>45</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Table Selection Screen</td>
<td>46</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Table Definition of the Table Books</td>
<td>47</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Table Definition of the Table Author</td>
<td>47</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Table Definition of the Table AuthorBook</td>
<td>48</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Index Definitions of the Three Tables</td>
<td>48</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Data Present in the Three Tables</td>
<td>49</td>
</tr>
</tbody>
</table>
Figure 23 Table Definitions After Transfer in the MySQL Database .......................... 50
Figure 24 Data Present in the Table Books .......................................................... 51
Figure 25 Data Present in the Table Author ........................................................ 51
Figure 26 Data Present in the Table AuthorBook.................................................. 52
Chapter 1

INTRODUCTION

Heterogeneous databases systems are systems that differ from each other schematically, semantically and syntactically. In other words heterogeneous database systems are different database systems like MySQL, SQL Server, Oracle, Sybase, Derby, PostgreSql etc. These various databases which are all derived from the relational database management system model (RDBMS) are independently created and administered but since almost every organization today uses several of these databases to run their applications there would be a need for these databases to communicate and interoperate with each other. Users might need to access data from a number of databases and applications may require data from several independent databases [1]. To complicate matters more, legacy database systems that are not based on the relational model contain enormous amounts of data, and such systems routinely interoperate with many of today’s relational environments Therefore, an application or a tool is required to manipulate and manage heterogeneous database systems.

Interoperability is defined as “the ability of two or more systems or components to exchange information and to use the information that has been exchanged” [2]. Similarly, database interoperability can be defined as the ability of two or more independently created databases with different architectures and structures to communicate with each other while maintaining their principles, objectives and individuality [1]. This database interoperability or the communication between
heterogeneous databases is a problem that affects a lot of companies today. Almost every organization makes use of multiple database systems depending on the requirements of their projects. If a project requires a light weight database system having a lot of emphasis on performance then a database system like MySQL may be chosen and if another project requires use of lot of reporting tools and has a lot of users logging on to the system then a slightly robust system like MS SQL Server may be used. So this way depending on the needs of the project and their clients’, organizations choose different databases to work with. Now each of these database systems store a lot of data and usually have many applications running against them. Now if these various applications have to communicate with each other they would also need their various database systems to interoperate with each other.

There may be many real time scenarios in organizations that require heterogeneous databases to interoperate or communicate with each other so that they can transfer data from one database system to another. For example, there may be some very old applications which were developed using some historic (legacy) database systems available at that time and now in the present day these historic databases might not give the same desired performance required by the application and hence would need to be migrated to a newer database. One other example could be changes in the requirements of a project after its development and deployment which may need a change in the database system used, which means transferring all the data from the database previously used to the new database system. Another example where we would need data present in one database system to be transferred to another database system is during mergers and
acquisitions when organizations or companies merge with other organizations or are acquired by other companies. These scenarios would require a tool which can transfer and convert data and schema structure from one database system to another. This project deals with building such a tool.
Chapter 2
BACKGROUND

Databases can be defined as a collection of large related data [3]. Collection and storage of information is not something new, the origins go back to ancient ages where information and data were collected and stored in libraries and universities. Then came traditional processing of data where files and tapes were used by operators to store information manually. This trend was slowly replaced by database management systems (DBMS) - software which allows large amount of data to be stored in databases and disks. DBMS not only allowed storage of large amounts of data it also made searching and retrieval of information very easy. This model of collecting data was widely accepted and hence led to its proliferation.

This proliferation has led to many intriguing advances in database systems in recent years such as, multimedia databases which allows media formats like audio and video files to be stored in databases, data warehousing where data stored is used to analyze information for decision making in large companies, fuzzy databases where data is stored and retrieved using active rules so on and so forth. Heterogeneous databases are one more such advancement in this field where there has been lot of research and proliferation. A heterogeneous database management system combines multiple dissimilar models of data within a single integrated system.

Data as we know is a critical resource in every organization today hence there is a lot of emphasis on how efficiently data can be accessed from various sources within the
organization and how easily it can be shared for data processing. As a result there is much effort on interconnecting the increasing number of databases scattered across several sites. In order to reconcile the contrasting requirements of the different database management systems (DBMSs), a tool is required which will transfer data from one system to another system. The fundamental issue in developing such a tool which will help in the integration of heterogeneous databases is how a database object such as a table, is translated from one DBMS architecture to another. We shall discuss the issues in detail as we move further.

2.1 Purpose and Scope

Transfer of data between heterogeneous database systems becomes imperative, as we discussed above. A tool is required which facilitates this transfer from a source database to a target database. The purpose of this project is to build a tool which can accomplish the above.

The scope of this project is to transfer data from Microsoft SQL Server (MS SQL Server) database to MySQL database. The transfer of data between MS SQL and MySQL should be an automated process with little manual intervention. The tool would

1. Allow the user to select one table or multiple tables or the entire database for transfer of data from the source to the target database.

2. Transfer table definitions, index definitions and the data in one single process.
3. Have an interactive GUI which will help the users to select a table or a list of tables from the databases of the source DBMS for the transfer to the target DBMS. The GUI would return messages for successful transfer or errors in the transfer.

2.2 Overview

The following is an overview of the various steps the tool follows to transfer data from one DBMS to the other:

1. Source DBMS connection – The connection parameters to connect to the source DB are specified.

2. Target DBMS connection – The connection parameters to connect to the target DB are specified.

3. Object Selection – The user selects the table or a list of tables to transfer data from.

4. Converting database objects – The tool extracts the table definitions, index definitions and other constraint definitions from the table selected of the source database and runs these definitions on the target database to replicate the database objects of the source on to the target. The tool should take care of the following steps while transferring the database object definitions:
   i. Column names will have to be transferred as they are from the source to the target.
ii. Datatypes of the source DB have to be matched with the equivalent datatypes present in the target DB.

iii. The length of the columns will have to remain the same.

iv. Tool will also look and transfer information like if the column can take null values or not.

v. Tool will then have to pick the index/key definitions of the source such as the primary key, unique key, foreign key definitions and replicate them on the target database

5. Transfer Data - Once the table and index/key definitions are transferred from source to target database the tool then extracts the data from table/tables selected by the user and transfer the data present in those tables from the source database to the target database.
Chapter 3

TECHNOLOGY SURVEY

This chapter will discuss the various technologies that are used in building the application and also gives a brief description on each one of them to familiarize and understanding them better.

The basic architecture of the application can be broken down into three layers and each of these 3 layers uses technologies that will be discussed below.

- **Presentation Layer** – This layer which is also known as the User Interface layer is the topmost layer which will display all the GUI forms to the user and helps the user to interact with the system. This application uses Windows Forms to build the GUI on .Net Framework 3.5 using the Integrated Development Environment (IDE) Visual Studio 2008.

- **Business Layer** – This is the middle tier and holds the main logic of the application. It will take the information provided by the user through the presentation layer and use it to accomplish its function. The business layer is developed using VB.Net on Visual Studio 2008 IDE and the .Net 3.5 framework.

- **Database Layer** – The bottom tier in this application consists of the two heterogeneous databases MS SQL Server 2008 and MySQL 5.0. Throughout the project to query the two databases, graphical tools have been used - SQL Server
Management Studio 2008 and MySQL Query Browser to query MS SQL Server 2008 and MySQL 5.0 databases respectively.

3.1 Microsoft SQL Server 2008

Microsoft SQL Server is a relational model database server produced by Microsoft. Its primary query languages are T-SQL and ANSI SQL. The current version of SQL Server is the SQL Server 2008 which aims to make data management self-tuning, self organizing, and self maintaining. SQL Server 2008 supports structured and semi-structured data like; digital media formats for pictures, audio, video and other multimedia data. It helps to do more with your data such as search, query, synchronize, analyze and report.

SQL Server 2008 reduces test and cost of management in development of applications and provides the highest levels of security, reliability, and scalability for business-critical applications [4].

3.2 SQL Server Management Studio 2008 (SSMS 2008)

MS SQL Server 2008 has a tool SQL Server Management Studio which includes both script editors and graphical tools. It is used for managing, configuring and administering all components within Microsoft SQL Server[5].
This tool combines the features included in previous releases of SQL Server like Enterprise Manager, Query Analyzer, and Analysis Manager, into a single environment and works with all components of SQL Server such as Reporting Services, Integration Services, and SQL Server Compact 3.5 SP1. Developers get a familiar experience, and database administrators get a single comprehensive utility that combines easy-to-use graphical tools with rich scripting capabilities [5].

3.2.1 Key Features

- T-SQL Debugger - a debugging tool has been integrated into the query editor in the SQL Server Management Studio, making it easy to debug the T-SQL code.

- Intellisense in the Query Editor – This feature like in other application development environments will underline incorrect syntax of your SQL statements, complete a word in a variable, command or function as you type, list the available parameters required by a function or SP, opens a list that provides available database objects and user defined variables that have been defined previously.

- Activity Monitor - The Activity Monitor includes performance dashboards with graphs and performance indicators with filtering capabilities, making it easier for DBAs to find important performance metrics.

- Object Explorer Details – Provides lots of details of a particular object selected in the object explorer. The details window will display different information, depending on the type of object chosen (a database, a table, etc).
• Object Search – This new feature allows you to type in a name of an object and perform a search that is based on your current context. For example, to search for a table in a single database, you need to select the database in the explorer window and then perform your search.

• Multi-Server Queries – This is another exciting new feature in SQL 2008 which is to query multiple servers simultaneously and return the results to a single window in SSMS [6].

3.3 MySQL 5.0

MySQL is a database management system that is used for mission-critical, heavy load production systems and delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. The MySQL software is Dual Licensed. Users can choose to use the MySQL software as an Open Source product under the terms of the GNU General Public License or can purchase a standard commercial license from Sun Microsystems, Inc [7].

3.3.1 Key Features

• Multiple storage engines, allowing one to choose the one that is most effective for each table in the application.

• High Performance for variety of workloads.

• Great documentation.
• Connectors for C, ODBC, Java, PHP, Perl,.NET etc.

• Wide range of supported platforms.

• Great community and Commercial Support [8].

3.4 Query Browser

The MySQL Query Browser is an open source graphical tool which is designed to query and analyze data stored in MySQL database and is provided by MySQL AB for creating, executing, and optimizing queries in a graphical environment. The MySQL Query Browser allows querying and editing of data in a more intuitive, graphical manner.

While all queries executed in the MySQL Query Browser can also be run from the command-line using the MySQL utility, the MySQL Query Browser allows for the querying and editing of data in a more intuitive, graphical manner.

MySQL Query Browser is designed to work with MySQL versions 4.0 and higher. Some of the features of MySQL Query Browser are:

• Interactively enter, edit, and execute queries.

• Browse the databases available on the server, the tables and stored routines in databases, and the columns in tables.

• Browse your query history to see what queries you’ve issued, or recall and re-execute previous queries.
• Bookmark queries for easy recall.

• Edit connection profiles that can be used to connect to servers more easily [9].

3.5 Microsoft Visual Studio 2008

Microsoft Visual Studio can be used to develop console and graphical GUI, Windows Forms and web applications, also web sites and web services in both native codes together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight. It is an Integrated Development Environment (IDE) from Microsoft.

Visual Studio code editor supports IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. GUI applications, web designer, class designer, and database schema designer can all be supported and built using a tool like forms designer. Plug-ins are allowed to be added that enhance the functionality at almost every level - including adding support for source control systems (like Subversion and Visual SourceSafe) to adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle.

Visual Studio allows any programming language to be supported and supports languages by means of language services, provided a language-specific service has been authored also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Built-in
languages include C/C++ (via Visual C++), VB.NET (via Visual Basic .NET), and C# (via Visual C#). Support for other languages such as F#, M, Python, and Ruby among others has been made available via language services which are to be installed separately.

3.6 VB.NET

Visual Basic .NET comes with increased application performance, powerful integrated development environment (IDE) and enhanced visual designers. It is the most productive tool used to rapidly build Windows and Web applications. It also supports creation of applications for wireless, Internet-enabled hand-held devices.

Visual Basic .NET comes with features for building more robust applications easily and quickly also powerful new forms designer, an in-place menu editor, and automatic control anchoring and docking. Visual Basic .NET offers fast, automatic formatting of code as you type, XML designer, improved Intellisence and an enhanced object browser, with the help of improved integrated development environment (IDE) and a significantly reduced startup time [10].

3.6.1 Key Features

- Full support for object oriented programming.

- Structured error handling capabilities.

- Access to .NET Framework.

- Inherent support for XML & Web Services.
• Better windows applications with Windows Forms.

• New Web capabilities with Web Forms.

• Immense power of tools & controls (including Server Controls).

• Interoperatibility with other .NET complied languages.


3.7 MySQL Connector/.NET 6.0.4

MySQL Connector/.NET 6.0.4 is a connector which is needed to connect to MySQL database from the .Net Framework 3.5. Connector/.NET enables developers to easily create .NET applications that require secure, high-performance data connectivity with MySQL. It implements the required ADO.NET interfaces and integrates into ADO.NET aware tools. Developers can build applications using their choice of .NET languages. Connector/.NET is a fully managed ADO.NET driver written in 100% pure C# [12].
Chapter 4
SYSTEM DESIGN

This chapter provides details about the design of the system implemented in this project. It discusses the various components of the system, their interaction with each other and how they integrate to form a system. This chapter further outlines the various difficulties faced during designing the system and the eventual solutions to getting over them.

4.1 Basic Design

The basic architecture of the system is as shown in the figure 1. Microsoft SQL Server is the source database from where table definitions and the data present in the table has to be transferred to the target database. The system or the tool is the medium which converts the definitions of the table/tables selected from the source database to their equivalent definitions in the target database. After the table/tables are set up in the target database, the system has to transfer the data present in the selected source tables to their respective tables in the target database. The third component in the basic design of the system as shown in figure 1 is the target database which is MySQL.

To transfer the table definitions and the data from the source database to target database the application or the tool has to communicate with both the MS SQL Server and MySQL databases. The system has to take the parameters required to connect to these two databases from the user. After accepting the parameters, the system will then
try to connect to the databases. The connection is successful if all the parameters are correct, otherwise an error message is displayed that notifies the user that the parameters entered were wrong. Once the parameters entered are authenticated, the system connects to both SQL Server database and MySQL databases. After the connections are established, the user is asked to select the tables from a database in MS SQL Server whose definitions and data have to be transferred to MySQL. After receiving the inputs from the user on the tables selected, the system extracts the definitions of the tables, converts them into equivalent definitions that MySQL understands and executes the converted definitions on MySQL. After the table definition is executed on MySQL, the system then queries the SQL Server database for the data present in the tables and transfers the same onto MySQL database. Figure 1 shows the basic design of the system.

Figure 1 Basic Design
4.2 Detailed Design

This section of the chapter discusses the design of the system in much more detail. It outlines the various steps involved to convert the schema and transfer the data from the source database to the target database.

4.2.1 Source/Target Database Connection

The very first thing that the tool has to do for its successful operation is to connect to the source database, which in our case is the MS SQL Server. So the application has to connect and login into the MS Sql Server database so that it can retrieve the various databases and tables present in each database and display it to the user. We shall discuss how the tool connects to the MS SQL Server in the implementation chapter of this report.

One important point we need to note while trying to connect to MS SQL Server is, that it uses two authentication modes to authenticate the users logging into the database system. The two modes are:

- Windows Authentication
- SQL Server Authentication

In Windows Authentication, the database server verifies to see if you have a valid windows account, in other words if you are connecting into the SQL Server database from a system which has Windows operating system in it, the database will take the credentials that the user exercises to login into the windows system. SQL Server validates the windows account credentials using the windows principal token in the operating
system, which means that the identity of the user is confirmed by the windows operating system itself. Hence, the server does not ask for the password of the windows account. This authentication mode is the default authentication mode of MS SQL Server and is said to be much more secure than the SQL Server Authentication mode [13].

In SQL Server Authentication, the logins are created in the MS SQL Server database and are not related in any way to the Windows user accounts. The user name and the password are set up and created by using the MS SQL server. These user names and passwords are stored in the MS SQL server. Each time users connect using this authentication mode, they would have to input their username and password [13].

The main difference between the above two authentication schemes is that the former relies mainly on the security architecture of your domain whereas the latter adds an additional process for authentication by asking for a hard-coded login and an associated password for connecting to the SQL Servers instances or databases [14].

Since MS SQL Server can be connected using any of these 2 authentication schemes, the system being developed has to cater to this by being able to provide an option to the user to use either of the authentication schemes to connect to the database. Once the authentication scheme to connect to the database is chosen and used to successfully connect into the source database, the user will then have to input the parameters needed to log into MySQL database before going into the next step of selecting the tables that need to be transferred from the source database to the target database. In other words, the system has to establish connections between the source and the target databases before moving on to the next step in the working of the tool.
4.2.2 Source Database Object Selection and Object Definition Extraction

The next step that the tool has to do after successfully connecting into the source and target databases is to select the objects. The objects in our case are the tables whose schema and the data present in them have to be transferred to the target database. These objects or tables are displayed once the user successfully connects to the SQL Server and chooses a particular database. All the tables that are created within that particular database chosen are then displayed to the user in a grid. The user is then allowed to select either one table or multiple tables or the entire list of tables present in that database to go to the next step which is to extract the definitions of the tables selected.

The definition of the table comprises of the following parts - the Create table statement with all the attributes in the table and their corresponding datatypes, the primary and foreign keys and the other definitions on the table like indexes, clauses like default value, null/not null values. Once the definition of a particular table or tables selected from the source database is generated, the keywords and the syntax that are used in MS SQL Server are mapped and modified into keywords and syntax that MySQL understands. The script that results from the mapping and modifications can then be executed on the MySQL database to create a replica of the table present on the MS SQL Server.

4.2.3 Converting Database Objects

In this section we shall discuss about converting the schema or table definitions that are present in MS SQL Server syntax to the syntax that MySQL would accept. The
most important concern in this topic would definitely be the data types. Although, MS
SQL Server and MySQL have many common data types, care has to be taken while
mapping these data types because even if they have the same name, their capacities
would be different. For e.g. the datatype CHAR in MS SQL Server can hold upto 8000
characters whereas the CHAR datatype used in MySQL can just hold 255 characters. We
shall discuss about this in much more detail later in the section. But before that, the other
things that we need to take care of during the conversion process are the database name
and the table names. Once a database is selected in MS SQL Server, the tool will save
this database name and use the same database name to create one on the MySQL side. If
MySQL already has a database created with that name, the tool will use the database that
is already present. Similarly, table names of the tables in MS SQL Server have to be kept
the same while transferring to the MySQL side. Also the column names have to remain
the same while being transferred and definitions on these columns like indexes, primary,
foreign and unique keys have to remain intact while being transferred.

Coming back to the discussion of mapping data types of MS SQL Server to their
counterparts in MySQL, we realize that it is a very profound process and one which is
very essential in the correct and efficient working of the tool. As we were discussing
earlier although MS SQL Server and MySQL have a fair amount of overlap as far as data
types go, there are still a lot of differences that has to be accounted for. MS SQL Server
has a lot many data types that have not been defined in MySQL like for e.g. the data type
MONEY. Now we have to come up with a slight alteration from the list of available data
types in MySQL, which will serve the same purpose. After having a careful look at all the
available options in MySQL we can say a definition like DECIMAL(15,4) will suffice. Here, the numbers 15 and 4 are called precision and scale. Precision is the number of the digits that can be stored before a decimal point and scale is the number of digits after the decimal point. One more example is the data type IMAGE, this is a variable-length binary data with a maximum length of $2^{31}-1$ bytes. The corresponding data type in MySQL that will serve the same purpose is LONGBLOB where the maximum length it can accept is $2^{32}$ bytes.

There are other instances where the names are the same in both the databases but cannot be mapped directly, like the data type TEXT. Although it is present in both MS SQL Server and MySQL it cannot be mapped directly because in MS SQL Server the maximum capacity of this data type is $2^{31}-1$ characters whereas in MySQL the maximum capacity it can hold is 65535 characters. Hence, we will need to map the MS SQL Server data type TEXT to LONGTEXT of MySQL which can hold $2^{32}$ characters. Table 1 shows the other mappings that are used in the conversion process in this tool [15][16][17].
<table>
<thead>
<tr>
<th>MS SQL Server datatype</th>
<th>Length/Storage required</th>
<th>MySQL Server equivalent Mapping</th>
<th>Length/Storage required</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT</td>
<td>Allows 0, 1 or NULL</td>
<td>BIT</td>
<td>8 bytes</td>
</tr>
<tr>
<td>INT</td>
<td>Whole number data from (2^{31}) through (2^{31}) - 1. Storage size is 4 bytes</td>
<td>INT</td>
<td>Whole number data from (2^{31}) through (2^{31}) - 1. Storage size is 4 bytes</td>
</tr>
<tr>
<td>TINYINT</td>
<td>Integer data from 0 through 25. Storage size is 1 byte</td>
<td>TINYINT</td>
<td>Integer data from 0 through 25. Storage size is 1 byte</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>Integer data from (2^{15}) through (2^{15}) - 1</td>
<td>SMALLINT</td>
<td>Integer data from (2^{15}) through (2^{15}) - 1</td>
</tr>
<tr>
<td>BIGINT</td>
<td>Whole number data from (2^{63}) through (2^{63}) - 1. Storage size is 2 bytes</td>
<td>BIGINT</td>
<td>Whole number data from (2^{63}) through (2^{63}) - 1. Storage size is 2 bytes</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>Max number of digits 38</td>
<td>DECIMAL</td>
<td>Max number of digits 65</td>
</tr>
<tr>
<td>MONEY</td>
<td>Monetary data values from (2^{63}) through (2^{63}) - 1.</td>
<td>DECIMAL(15,4)</td>
<td></td>
</tr>
<tr>
<td>SMALLMONEY</td>
<td>-21,478.3648 through +21,478.3648</td>
<td>DECIMAL(5,4)</td>
<td></td>
</tr>
<tr>
<td>FLOAT[(n)]</td>
<td>n must be a value from 1 through 53. 8 bytes</td>
<td>DOUBLE(M,D)</td>
<td>M may be a value from 1 through 53. 8 bytes.</td>
</tr>
<tr>
<td>REAL[(n)]</td>
<td>n must be a value from 1 through 24. Size : 4 bytes</td>
<td>FLOAT(M,D)</td>
<td>M must be a value from 1 through 24. 4 bytes.</td>
</tr>
<tr>
<td>DATETIME</td>
<td>Date &amp; time from Jan 1, 1753 through Dec 31, 9999</td>
<td>DATETIME</td>
<td>Date &amp; time from Jan 1, 1000 through Dec 31, 9999</td>
</tr>
<tr>
<td>SMALL DATETIME</td>
<td>Jan 1, 1900 through Jan 6, 2079</td>
<td>DATETIME</td>
<td>Date &amp; time from Jan 1,1000 through Dec 31,9999</td>
</tr>
</tbody>
</table>

Table 1 MS SQL Server – MySQL Mapping
<table>
<thead>
<tr>
<th>MS SQL Server datatype</th>
<th>Length/Storage required</th>
<th>MySQL Server equivalent Mapping</th>
<th>Length/Storage required</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR</td>
<td>Maximum length of 5,000 characters</td>
<td>TEXT</td>
<td>Maximum length of 65,535 characters</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>Maximum length of 5,000 characters</td>
<td>VARCHAR</td>
<td>Maximum length of 5,000 characters</td>
</tr>
<tr>
<td>TEXT</td>
<td>Maximum length of $2^{31} - 1$ characters</td>
<td>LONGTEXT</td>
<td>Maximum of 4,294,967,295 characters</td>
</tr>
<tr>
<td>NCHAR</td>
<td>Unicode data with a maximum of 4000 characters</td>
<td>TEXT</td>
<td>Maximum length of 65,535 characters</td>
</tr>
<tr>
<td>NVARCHAR</td>
<td>Unicode data with a maximum of 4,000 characters</td>
<td>VARCHAR</td>
<td>Maximum length of 5,000 characters</td>
</tr>
<tr>
<td>BINARY</td>
<td>Fixed length binary data. 8,000 bytes</td>
<td>BLOB</td>
<td></td>
</tr>
<tr>
<td>VARBINARY</td>
<td>Variable length binary data. 8,000 bytes</td>
<td>BLOB</td>
<td></td>
</tr>
<tr>
<td>IMAGE</td>
<td>Variable length binary data with a maximum of $2^{31} - 1$ bytes</td>
<td>LONGBLOB</td>
<td>4,294,967,295 bytes</td>
</tr>
</tbody>
</table>

Table 1 MS SQL Server – MySQL Mapping (contd)

4.2.4 Definition Generation and Execution

Once all the objects in the table definition of a particular table in MS SQL Server is mapped to equivalent objects that MySQL understands, a resulting script is generated that follows the MySQL syntax and keywords. This resulting script is then transferred to the MySQL database and executed there. On executing the script, we have a single table
or multiple tables created on MySQL database whose table definitions are similar to the table/tables that the user selected from MS SQL Server.

4.2.5 Data Extraction and Transfer

After successfully replicating the table definitions from the source database to the target database, the tool has to then transfer the data present, from the source table/tables selected to the target table/tables that were newly created. To do this the tool generates a query which queries the source table and extracts all the data present in the table as a result set. The tool then takes this result set and generates insert statements according to the definition of the table. Once the entire script is generated it is transferred to the MySQL side and executed to populate the tables created.

Figure 2 shows the functional flow diagram of the design.
Figure 2 Functional Flow Diagram

1. Source/Target Connection
2. Select the objects like tables
3. Extract the definition of objects selected
4. Setup database table & column mappings
5. Generate Scripts to be run on the target database after the mapping & execute them
6. Query the source database for data of the objects selected & transfer the data to the target
7. Target schema created & data transferred
Chapter 5

IMPLEMENTATION

This chapter discusses the implementation details of the system. It explains how each part of the design was implemented and integrated to form a complete system. This chapter also discusses the various challenges and issues faced during the implementation and how they were resolved.

5.1 Source/Target Database Connections

The first thing that had to be done before starting the implementation was to set up the technologies that were finalized. The three main applications that had to be taken care of were MS Visual Studio 2008 with .NET Framework 3.5, MS SQL Server 2008 and MySQL5.0. All the three applications had extensive documentation on installing them correctly and hence there was little trouble while installing them. After these three applications were set up, the next step was to see if all the three could communicate with each other. In other words, we have to test if we can connect to the 2 databases from the development environment which is the MS Visual Studio 2008.

To connect to SQL Server from Visual Studio 2008, the .NET framework offers a data provider for SQL Server which allows us to use a class called SqlConnection. This SqlConnection class accepts a valid connection string which supplies necessary authentication information to connect to a specific data source. To use this SqlConnection
class we have to import the namespace System.Data.SqlClient into our code. Figure 3 is a code snippet from the project that will help in better understanding what has been explained above.

```vbnet
Imports System.Data.SqlClient

Dim connectionString As String
connectionString = SServerConnStringBuilder(dsourceText.Text, "master", unametxt.Text, passtext.Text)
Using conn As New SqlConnection(connectionString)
    conn.Open()
End Using
```

**Figure 3 Code Snippet to Connect to MS SQL Server**

In the above code we see that the namespace System.Data.SqlClient is first imported. A function called SServerConnStringBuilder is then called with a few parameters that will help it build the connection string to connect to MS SQL Server. The value from the function (which is the connection string) is then assigned to a string variable called connectionString, which is given to the SqlConnection object. Now when we say conn.Open(), the connection is established between the SQL Server database and the code. To close this connection we will have to say conn.Close(). An example connection string to connect to MS SQL Server database is shown in Figure 4.

```
connectionString="Data Source=DEEPAK-PC;Initial Catalog=Deepak;User ID=sa;Password=admin"
```

**Figure 4 Example of a Connection String**
In the connection string above Data Source identifies the server used. It could have the name of the local machine or domain name of any other machine. Initial Catalog points to the database in the server you would want to connect to. User ID is the name of the user configured in the SQL Server and the Password points to the corresponding password set for that User ID.

Now to connect to the MySQL database we cannot use the same SqlConnection object neither can we use the connection string used to connect to the MS SQL Server. To connect to MySQL database from the .NET Framework we will first have to download and install a connector called MySQL Connector/Net. The version that is used in this project is the MySQL Connector/Net 6.0.4. After we are done installing this connector we will need to add it as a reference into the project code. Once we add the connector as a reference into the project we can then access the series of classes provided by MySQL in the MySQL Connector/Net. These classes can then be used in the code by importing the namespace MySql.Data. The class that will aid in connecting to the MySQL database is MySqlConnection which is defined in the namespace MySql.Data.MySqlClient. Figure 5 shows a code snippet from the project to connect to the MySQL database.
Here, as we can see a string variable called mySqlConString is used to build the connection string required to connect to a MySQL database. This string variable is then sent as an argument to the class MySqlConnection, which will then open the connection for the code to contact the MySQL database. A sample connection string to connect to MySQL database is as shown in Figure 6.

```
connectionString = "Server=localhost;Port=3307;Database=Deepak;Uid=root;Pwd=admin"
```

5.2 User Interface

Windows Forms are used to build the user interface in the project. The first form is a configuration page with many fields. This form accepts values from the users like data source, username, password, etc. and helps build the connection strings for both the databases, with which the application can then connect to both the MS SQL Server database and the MySQL database. The figure 7 shows how the config page looks.
SplitContainer Control is used to split the windows form into two parts as shown in the figure above. The left hand side of the windows form contains fields that receive inputs from the user to connect to the MS SQL Server database and the right hand side of the form contains fields that receive inputs from the user to connect to the MySQL database. As we already discussed, in the design of this report, MS SQL Server has two authentication modes namely Windows Authentication and Sql Server Authentication to connect to the database. If the Sql Server Authentication checkbox is ticked, as in figure 7 two textboxes appear that take the inputs for the username and password whereas, if the Windows Authentication checkbox is checked, the windows login account is considered to login to the database automatically, we do not need the two textboxes for username and password and hence these will be disabled. There are two other buttons that are disabled by default. The ‘Go’ button and the ‘Select the Tables’ button. The ‘Go’ button will be enabled only after SQL Server ‘Data Source’ is filled with some value, whereas
the ‘Select the Tables’ button is enabled only after filling all the fields present in the
form. The source code of the Config page can be found in Appendix A.

Once the parameters to connect to the databases are inputted and tested to see that
the connection parameters used are valid and the connections were successful, then the
combobox with the label “Select the Database” is populated with the databases that are
present in the MS SQL Server. To populate the combobox with the databases present in
the SQL Server, the following query is used:

\[ \text{Select } \text{name from master.dbo.sysdatabases order by name} \rightarrow \text{query 1} \]

This query will give all the databases present in the SQL Server as a result set sorted in
the ascending order.

Now, after selecting a database from the list shown in the combobox, we hit the
‘Select the tables’ button to open our next important form which displays all the tables
present in the particular database selected.
As we can see in figure 8, the Table Selection form displays all the tables that have been created in a particular database selected. The form has a grid which is shown in the figure with two columns. One column displays the name of the table and the other displays a checkbox corresponding to the entry of the table name. It also has buttons which will aid in selecting the checkboxes. ‘Select All’ button selects all the checkboxes whereas the ‘Clear All’ button clears all the checkboxes. The ‘Cancel’ button cancels the operation whereas the ‘Ok’ button takes it to the next step. The next step is to then call the various functions which handles the main logic of converting the definition of the tables selected in the above form from MS SQL Server’s format to MySQL’s format and
then transferring the data present in the tables from SQL Server to the newly created MySQL tables.

The query that is used to pull all the tables, present in the particular database is:

\[
\text{select \ TABLE\_NAME \ from \ INFORMATION\_SCHEMA\_TABLES \ where \ TABLE\_TYPE = 'BASE TABLE'} \rightarrow \text{query 2}
\]

INFORMATION_SCHEMA.TABLES contains the following fields or columns as shown in figure 9.

![Table of Data Present in the Table INFORMATION_SCHEMA.TABLES](image)

Figure 9 Data Present in the Table INFORMATION_SCHEMA.TABLES

Now since what is required is a list of the table names, the query used is

INFORMATION_SCHEMA.TABLES just for the TABLE_NAME.

5.3 Main Functions

After a table or set of tables are selected from the table selection form, the application enters the core logic, which is to extract the table definitions of the tables selected from MS SQL Server and convert these generated table definitions to a script that can be run on MySQL to replicate the same table on the MySQL database. There are
four principal functions that handle this main logic. The function names and the parameters they accept are as shown below:

getListOfTables(sqlConnString)
getSqlServerSchema(sqlConnString, listOfTables)
convertSchema(sqlSchema, dbName, mysqlconstring)
transferSqlData(sqlSchema, dbName, sqlConnString, mysqlconstring)

These four functions along with a series of other functions which these functions call in turn form the core or the business logic of the application. The source code of these functions is included in Appendix B.

5.3.1 getListOfTables(sqlConnString)

As the name suggests this function gets the names of all the tables from a database selected by the user from the form and stores them in a list. This function takes the SQL Servers connection string as a parameter and connects to the database in the connection string. Once connected to the database, query 2 is run to get a list of all the tables present in that particular database. This list is then populated to the Table Selection form which is displayed to the user. The user then selects a table or tables for conversion and this selected list is collected and returned back to the calling function. The flowchart for this function is shown in Figure 10.
Before moving further and exploring the next important function, there are a couple of data structures which need to be understood. To store all the data of a table like the table name, column names, primary key, foreign key and indexes a class called `tableDefn` has been defined. This class is as shown below.

**Public Class tableDefn**

- `Public name As String`
- `Public columns As List(Of columnDefn)`
Public pKey As List(Of String)
Public fKeys As List(Of fKeyDefn)
Public Indexes As List(Of indexDefn)

End Class

Since the entire member variables of this class are defined as Public, these variables can be accessed from anywhere in the program by just creating an instance of the class. The variable ‘name’ is basically used to store the name of the table. The variable ‘columns’ is a list of type columnDefn. columnDefn now is another class that is declared separately that encompasses details of columns like column names, column types, column lengths etc. pKey is a string list which will hold the primary keys if any defined on the table. fKey is again a list of type fKeyDefn, which is another class holding foreign key specific variables. Finally, Indexes is a list of type indexDefn, which will encapsulate index specific details. The class declarations are as shown below.

Public Class ColumnSchema
    Public ColumnName As String
    Public ColumnType As String
    Public ColumnLength As Integer
    Public IsNullable As Boolean
    Public DefaultValue As String
    Public IsIdentity As Boolean
End Class
Public Class fKeyDefn

    Public name As String
    Public ColumnName As String
    Public ForeignTableName As String
    Public ForeignColumnName As String
    Public CascadeOnDelete As Boolean
    Public IsNullable As Boolean

End Class

Public Class IndexSchema

    Public IndexName As String
    Public IsUnique As Boolean
    Public Columns As List(Of IndexColumn)

End Class

5.3.2 getSqlServerSchema(sqlConnString, tableList)

This function takes the connection string to connect to SQL Server and the list of selected tables by the user as the parameters. The flowchart for this function is as shown in Figure 11.
This function traverses through each table selected by the user one after the other and extracts all the information regarding the table definition and stores it in the tableDefn class, so that in the next step it can generate a script for creating a table in the MySQL
database by extracting the information stored in the tableDefn class. This function calls CreateTableSchema function whose flowchart is shown in Figure 12.

Figure 12 Flowchart of CreateTableSchema (Part 1)
CreateTableSchema function basically fills all the variables of the class tableDefn by extracting values from the result set of executing a series of system stored procedures run on that particular table selected. The first system stored procedure that is executed in this function is:

`Exec sp_columns 'tableName'` → query 3
This system stored procedure returns all the properties regarding the columns of the table specified by the ‘tableName’. All the values that are required are then extracted and saved into the member variables of the class columnDefn. The next stored procedure that is executed is

*Exec sp_pkeys ‘tableName’ → query 4*

This stored procedure returns the details of any primary key that is defined on the table specified by the tableName. Again, the column name on which the primary key is defined is extracted from the result and given to the pKey variable of the class tableDefn.

The third system stored procedure executed in this function is

*Exec sp_helpindex ‘tableName’ → query 5*

This stored procedure returns details about any indexes defined on the table like the index name, the description of the index and on which key the index is defined. The required fields are then extracted from the result and stored in the IndexSchema class.

The source code of the above three system stored procedures can be found in the ‘System Stored Procedures’ folder under any database in MS SQL Server Management Studio.

5.3.3 convertSchema(sqlSchema,dbName,mysqlconstring)

This function basically creates or generates the script for creating the table on MySQL database by re-extracting all the data that has been stored in the classes and building the script using a string builder. The flow chart for this function is as shown in Figure 14. This function in turn calls BuildCreateTableQuery function whose flowchart is shown in Figure 15.
5.3.4 transferSqlData(sqlSchema, dbName, sqlConnString, mysqlconstring)

Once the tables are created on MySQL database, the data has to be transferred from the source database tables to target database tables. This function basically does that. Using the sqlConnString the application connects to the SQL Server database and queries the table for all the data. The resultant set of the select query is then used again to create the insert query which is executed on the MySQL database to populate the tables that were newly created.
Start
BuildCreateTableQuery(tableDefn, bdName, mysqlcornstring)

Create a new string builder sb.

for i=0 to tableDefn.columns.Count - 1

YES

Extract the values in tableDefn.columns[i] and append them to the stringbuilder to form a column line.

NO

If tableDefn.pkey.Count > 0

YES

Until tableDefn.pkeyCount = 0

YES

Sb Append pkey

NO

Until tableDefn.pkeyCount = 0

YES

Sb Append fKey

NO

Until tableDefn.fkeyCount = 0

NO

If tableDefn.Indexes is nothing

YES

Sb Append Indexes

NO

Execute the sb in Mysql

Return

Figure 15 Flowchart of BuildCreateTableQuery
Chapter 6

RESULTS

This chapter explains the execution of the application step by step and shows the results produced by the application after every step.

The first screen of the application is as shown in the Figure 16. The user is expected to input the fields shown, which will be used to construct the connection strings to connect to the databases. The fields required to connect to the SQL Server database are the data source, username and password if ‘Sql Server Authentication’ is chosen. The user doesn’t need to enter the username and password if ‘Windows Authentication’ is chosen as the database by default would consider the credentials used to log into the windows account.

![Figure 16 Configuration Page Screen](image-url)
The second half of the screen consists of fields required to connect to the MySQL database. The fields required are the data source of the MySQL database, the port number, username and the password. The ‘test’ button is used to test and check if the values entered are valid and if it connects to the database. The ‘Go’ button on the SQL Server side is used to connect to the database and populate the ‘Select the Database’ combobox.

The ‘Select the tables’ button is used to open the next form which is as shown in Figure 17. The form displays all the tables present in the particular database selected and the user is allowed to select the tables to be transferred by checking the checkboxes corresponding to the name of the table. The ‘Clear All’ button is used to clear all the selections previously made and the ‘Select All’ button is used to select all the tables in the list.

![Figure 17 Table Selection Screen](image)
So as we can see from the Figure 17 there are three tables that are selected from the ‘Deepak’ database – Books, Authors and AuthorBook. The table definitions of these three tables are as shown in Figures 18, 19, 20. BookID is the primary key in the table Books whereas AuthID is defined as the primary key in the table Author. The third table AuthorBook has two columns BookID and AuthID which refer to the columns with the same names in the tables Books and Author respectively. These table definitions in Figures 18, 19, 20 are generated by MS SQL Server Management Studio when you right-click on the table names and say ‘Script Table as’ and then click on ‘CREATE To’.

```sql
CREATE TABLE [dbo].[Books](
    [BookID] [smallint] NOT NULL,
    [BookTitle] [varchar](60) NOT NULL,
    [Copyright] [datetime] NOT NULL,
PRIMARY KEY CLUSTERED (
    [BookID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
)
```

**Figure 18 Table Definition of the Table Books**

```sql
CREATE TABLE [dbo].[Author](
    [AuthID] [smallint] NOT NULL,
    [AuthFN] [varchar](20) NULL,
    [AuthMN] [varchar](20) NULL,
    [AuthLN] [varchar](20) NULL,
PRIMARY KEY CLUSTERED (
    [AuthID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
)
```

**Figure 19 Table Definition of the Table Author**
CREATE TABLE [dbo].[AuthorBook]
(  [AuthID] [smallint] NOT NULL,
  [BookID] [smallint] NOT NULL,
  PRIMARY KEY CLUSTERED
    (  [AuthID] ASC,
        [BookID] ASC
     ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[AuthorBook] WITH CHECK ADD FOREIGN KEY ([AuthID]) REFERENCES [dbo].[Author] ([AuthID])
GO

ALTER TABLE [dbo].[AuthorBook] WITH CHECK ADD FOREIGN KEY ([BookID]) REFERENCES [dbo].[Books] ([BookID])
GO

Figure 20 Table Definition of the Table AuthorBook

Figure 21 Index Definitions of the Three Tables
The index definitions of the three tables are as shown in the Figure 21. The screenshot above is taken after running the sp_helpindex system stored procedures on MS SQL Server Management Studio. The source code of this stored procedure can be found under ‘System Stored Procedures’ folder of any database in SQL Server Management Studio. Unique, clustered indexes are automatically created on primary keys. There is also a non clustered index AUTHLN_Index on the column AuthLN. The data present in the three tables are as shown in the Figure 22.

![Figure 22 Data Present in the Three Tables](image-url)
Now after the conversion and transfer of data from MS SQL Server to MySQL has happened. The table definitions of the three tables transferred to the MySQL side are as shown in the Figure 23. Figure 23 is a screenshot taken from the MySQL Query Browser.

```
1) DROP TABLE IF EXISTS 'deepak'.'books';
2) CREATE TABLE 'deepak'.'books' (...
3) PRIMARY KEY ('BookID'),...
4) ENGIN=InnoDB DEFAULT CHARSET=latin1;
5) DROP TABLE IF EXISTS 'deepak'.'author';
6) CREATE TABLE 'deepak'.'author' (...
7) PRIMARY KEY ('AuthID'),...
8) ENGIN=InnoDB DEFAULT CHARSET=latin1;
9) DROP TABLE IF EXISTS 'deepak'.'authorbook';
10) CREATE TABLE 'deepak'.'authorbook' (...
11) PRIMARY KEY ('AuthID', 'BookID'),...
12) FOREIGN KEY ('AuthID') REFERENCES 'author' ('AuthID'),...
13) FOREIGN KEY ('BookID') REFERENCES 'books' ('BookID')...
14) ENGIN=InnoDB DEFAULT CHARSET=latin1;
```

**Figure 23 Table Definitions After Transfer in the MySQL Database**

As you can see the table definitions of the three tables are an exact replica of the tables created on MS SQL Server. The database name, table name, column names remain the same without any changes. Also the other definitions like primary keys, foreign keys, indexes are all transferred to the tables in MySQL in the same way they were defined in MS SQL Server.
Figures 24, 25, 26 show that the data has been transferred correctly into the tables created in MySQL. The Figures 24, 25 and 26 are all screenshots of the result sets got after querying the MySQL Query Browser.

Figures 24, 25, 26 show that the data has been transferred correctly into the tables created in MySQL. The Figures 24, 25 and 26 are all screenshots of the result sets got after querying the MySQL Query Browser.

![SQL Query Area](image1)

**Figure 24 Data Present in the Table Books**

<table>
<thead>
<tr>
<th>BookID</th>
<th>BookTitle</th>
<th>Copyright</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Letters</td>
<td>1905-04-19 00:00:00</td>
</tr>
<tr>
<td>2</td>
<td>Ohio</td>
<td>1905-04-04 00:00:00</td>
</tr>
<tr>
<td>3</td>
<td>Angels</td>
<td>1905-05-21 00:00:00</td>
</tr>
<tr>
<td>4</td>
<td>Speaks</td>
<td>1905-04-17 00:00:00</td>
</tr>
<tr>
<td>5</td>
<td>Man</td>
<td>1905-05-20 00:00:00</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>1905-05-04 00:00:00</td>
</tr>
<tr>
<td>7</td>
<td>Card</td>
<td>1905-05-16 00:00:00</td>
</tr>
<tr>
<td>8</td>
<td>The</td>
<td>1905-05-17 00:00:00</td>
</tr>
</tbody>
</table>

![SQL Query Area](image2)

**Figure 25 Data Present in the Table Author**

<table>
<thead>
<tr>
<th>AuthID</th>
<th>AuthFN</th>
<th>AuthMN</th>
<th>AuthLN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Henry</td>
<td>S.</td>
<td>Thompson</td>
</tr>
<tr>
<td>2</td>
<td>Jack</td>
<td>Card</td>
<td>Gates</td>
</tr>
<tr>
<td>3</td>
<td>Fred</td>
<td></td>
<td>Ek</td>
</tr>
<tr>
<td>4</td>
<td>White</td>
<td>Maria</td>
<td>Pike</td>
</tr>
<tr>
<td>5</td>
<td>Anne</td>
<td>Kennedy</td>
<td>Tode</td>
</tr>
<tr>
<td>6</td>
<td>Jane</td>
<td>G.</td>
<td>Nelson</td>
</tr>
<tr>
<td>7</td>
<td>Jane</td>
<td></td>
<td>Yin</td>
</tr>
<tr>
<td>8</td>
<td>Alan</td>
<td></td>
<td>Wang</td>
</tr>
</tbody>
</table>
Figure 26 Data Present in the Table AuthorBook
In this project we saw and discussed how the tool implemented can communicate with two heterogeneous databases and transfer the table definitions and also the data present in those tables from a source database to a target database. The two heterogeneous databases used in this project are Microsoft’s SQL Server 2008 and MySQL 5.0. The language used to implement the tool is VB.NET and the .Net Framework used is 3.5.

So as a conclusion to this project report, we can say that this project has accomplished its primary goals as discussed in the scope section of Chapter 2. The main features of the tool are to transfer data between MS SQL Server and MySQL as an automated process with little manual intervention. It replicates the table definitions along with the primary key, foreign key and index definitions on the tables from MS SQL Server to MySQL in one single process. The tool provides an interactive GUI which will allow the users to connect to each of the databases and select a table or multiple tables from MS SQL Server to be transferred to MySQL. The GUI will also return messages for successful operations or errors.
7.1 Limitations and Future Enhancements

There are certain limitations in this project which can be worked upon in the future to further the capabilities of this tool. The first limitation is processing clauses concerning constraints. The foreign key definition can specify clauses such as “ON DELETE CASCADE” which will govern the behavior of an SQL delete statement. Such processing clauses have not been taken care off in this project and are a topic which can be added as a future enhancement to the tool. Another limitation is the datatype mapping between the two databases. Although I have tried to map most of the datatypes in MS SQL Server to equivalent datatypes in MySQL there are certain datatypes in MS SQL Server especially the new ones introduced in SQL Server 2008 like the spatial datatypes – Geography and Geometry and another datatype called Hierarchyid which have not been mapped. Hierarchyid for example is a datatype introduced in MS SQL Server 2008 which represents the position of a particular node in a hierarchy. It basically encodes data about a particular node in a hierarchy structure by encoding the path from the root of the structure to the node selected. Now after reading a little more about this particular datatype it was hard to think of an equivalent datatype in MySQL which can be mapped to this datatype.

One other limitation of the tool is that the transfer of data is single directional from MS SQL Server to MySQL. A future enhancement of this tool could be to make this tool bidirectional. That is the tool should support transferring data from MS SQL Server to MySQL and vice versa.
The source code of Config.vb page is below:

```vbnet
Imports System
Imports System.Collections.Generic
Imports System.ComponentModel
Imports System.Data
Imports System.Drawing
Imports System.Linq
Imports System.Text
Imports System.Windows.Forms
Imports System.Data.SqlClient
Imports System.IO
Imports MySql.Data.MySqlClient

Partial Public Class Config
    Inherits Form
    Public Sub New()
        InitializeComponent()
    End Sub

    Private Sub gobtn_Click(ByVal sender As Object, ByVal e As EventArgs) Handles gobtn.Click
        Try
            Dim connectionString As String
            If wincbx.Checked Then
                connectionString = SServerConnStringBuilder(dsourceText.Text, "master")
            Else
                connectionString = SServerConnStringBuilder(dsourceText.Text, "master", username.Text, password.Text)
            End If
            Using conn As New SqlConnection(connectionString)
                conn.Open()
                Dim listDatabases As New SqlCommand("select name from master.dbo.sysdatabases order by name", conn)
                Using reader As SqlDataReader = listDatabases.ExecuteReader()
                    dbcombo.Items.Clear()
                    While reader.Read()
                        dbcombo.Items.Add(DirectCast(reader(0), String))
                    End While
                    If dbcombo.Items.Count > 0 Then
                        dbcombo.SelectedIndex = 0
                    End If
                End Using
            End Using
        End Try
    End Sub
End Class
```
```vbscript
End If
End Using
dbc当地人.combo.Enabled = True
End Using
Catch ex As Exception
    MessageBox.Show(Me, ex.Message, "Failed To Connect", MessageBoxButtons.OK,
                    MessageBoxIcon.Error)
    End Try
End Sub

Private Sub wincbx_CheckedChanged(ByVal sender As Object, ByVal e As EventArgs)
Handles wincbx.CheckedChanged
If wincbx.Checked Then
gobtn.Enabled = dsourcetext.Text.Trim().Length > 0
sqlcbx.Checked = False
unamelbl.Visible = False
unametext.Visible = False
passlbl.Visible = False
passtext.Visible = False
ElseIf sqlcbx.Checked Then
go
tn.Enabled = dsourcetext.Text.Trim().Length > 0
wincbx.Checked = False
unamelbl.Visible = True
unametext.Visible = True
passlbl.Visible = True
passtext.Visible = True
End If
End Sub

Private Sub sqlcbx_CheckedChanged(ByVal sender As Object, ByVal e As EventArgs)
Handles sqlcbx.CheckedChanged
If sqlcbx.Checked Then
go
tn.Enabled = dsourcetext.Text.Trim().Length > 0
wincbx.Checked = False
unamelbl.Visible = True
unametext.Visible = True
passlbl.Visible = True
passtext.Visible = True
ElseIf wincbx.Checked Then
go
tn.Enabled = dsourcetext.Text.Trim().Length > 0
sqlcbx.Checked = False
unamelbl.Visible = False
unametext.Visible = False
passlbl.Visible = False
passtext.Visible = False
End If
End Sub
```
Private Sub dsourcetext_TextChanged(ByVal sender As Object, ByVal e As EventArgs)
Handles dsourcetext.TextChanged
    gobtn.Enabled = dsourcetext.Text.Trim().Length > 0
End Sub
Private Sub Mypasstext_TextChanged(ByVal sender As Object, ByVal e As EventArgs)
Handles Mypasstext.TextChanged
    test.Enabled = Mypasstext.Text.Trim().Length > 0
End Sub

Private Shared Function SServerConnStringBuilder(ByVal source As String, ByVal dbName As String) As String
    Dim conStrBuild As String = ("Data Source=") & source.Trim() & ";Initial Catalog=") + dbName.Trim() & ";Integrated Security=SSPI;"
    Return conStrBuild
End Function

Private Shared Function SServerConnStringBuilder(ByVal source As String, ByVal dbName As String, ByVal user As String, ByVal pass As String) As String
    Dim conStrBuild As String = ("Data Source=") & source.Trim() & ";Initial Catalog=") + dbName.Trim() & ";User ID=") + user.Trim() & ";Password=") + pass.Trim()
    Return conStrBuild
End Function

Private Sub test_Click(ByVal sender As Object, ByVal e As EventArgs) Handles test.Click
    lblmsg.Visible = False
    Dim myServer As String = Mydsourcetext.Text
    Dim myPort As Integer = myporttext.Text
    Dim myUname As String = Myunametext.Text
    Dim myPass As String = Mypasstext.Text
    Dim mySqlConString As String = "server = " + myServer + ";port = " + myPort.ToString() + ";uid = " + myUname + ";password = " + myPass
    If dbcombo.SelectedItem IsNot Nothing Then
        Dim connection As New MySqlConnection(mySqlConString)
        Try
            connection.Open()
            MessageBox.Show("Connection successful")
            tableselection.Enabled = True
        Catch ex As Exception
            MessageBox.Show(Me, ex.Message, "Failed To Connect", MessageBoxButtons.OK, MessageBoxIcon.[Error])
        End Try
        Dim Query As String = "CREATE DATABASE IF NOT EXISTS " & dbcombo.SelectedItem
        Dim addUser As New MySqlCommand(Query, connection)
        Try
            addUser.ExecuteNonQuery()
        Catch ex As Exception
            MessageBox.Show(Me, ex.Message, "Failed To Connect", MessageBoxButtons.OK, MessageBoxIcon.[Error])
        End Try
    End If
End Sub
```csharp
    addUser.ExecuteNonQuery()
    Catch ex As Exception
        MessageBox.Show(Me, ex.Message, "Failed To Connect", MessageBoxButtons.OK, MessageBoxIcon.[Error])
    End Try
    "close the connection
    connection.Close()
    Else
        lblmsg.Visible = True
        lblmsg.Text = "*Please select a database from the SQL Server databases list and click test again"
    End If

End Sub

Private Sub tableselection_Click(ByVal sender As Object, ByVal e As EventArgs) Handles tableselection.Click
    Dim sqlServerConstr As String
    If wincbx.Checked Then
        sqlServerConstr = SServerConnStringBuilder(dsourcetext.Text, DirectCast(dbcombo.SelectedItem, String))
    Else
        sqlServerConstr = SServerConnStringBuilder(dsourcetext.Text, DirectCast(dbcombo.SelectedItem, String), unametext.Text, passtext.Text)
    End If
    Dim dbName As String = dbcombo.SelectedItem
    Dim mySqlConString As String = "server = " + Mydsourcetext.Text + ";port = " + myporttext.Text.ToString() + ";database = " + dbName + ";uid = " + Myunametext.Text + ";password = " + Mypasstext.Text + ";charset = utf8"
    MainConverter.convertSqlServerToMySQL(sqlServerConstr, dbName, mySqlConString)
End Sub
End Class
```
APPENDIX B

The source code of the main functions along with other important functions it calls is given below.

```
Public Shared Sub convertSqlServerToMySQL(ByVal sqlConnString As String, ByVal dbName As String, ByVal mysqlconstring As String)
    Dim listOfTables As List(Of String) = getListOfTables(sqlConnString)
    If listOfTables IsNot Nothing Then
        Dim sql
        Schema As List(Of tableDefn) = getSqlServerSchema(sqlConnString, listOfTables)
        convertSchema(sqlSchema, dbName, mysqlconstring)
        transferSqlData(sqlSchema, dbName, sqlConnString, mysqlconstring)
    End If
End Sub

Public Shared Function getListOfTables(ByVal connString As String) As List(Of String)
    Using conn As New SqlConnection(connString)
        conn.Open()
        Dim tableNameList As New List(Of String) ()
        ' This command will read the names of all tables in the database
        Dim cmd As New SqlCommand("select TABLE_NAME from INFORMATION_SCHEMA.TABLES where TABLE_TYPE = 'BASE TABLE' order by TABLE_NAME", conn)
        Using reader As SqlDataReader = cmd.ExecuteReader()
            While reader.Read()
                tableNameList.Add(DirectCast(reader("TABLE_NAME"), String))
            End While
        End Using
        Dim selectionHandler As SqlServerTableSelection
        selectionHandler = New SqlServerTableSelection(AddressOf tableList)
        Dim updated As List(Of String) = selectionHandler.Invoke(tableNameList)
        Return updated
    End Using
End Function

Private Shared Function getSqlServerSchema(ByVal connString As String, ByVal tableList As List(Of String)) As List(Of tableDefn)
    Dim tables As New List(Of tableDefn) ()
    Try
```
Using conn As New SqlConnection(connString)
  conn.Open()
  Dim count As Integer = 0
  For Each table As String In tableList
    Dim ts As tableDefn = createTableSchema(conn, table)
    createForeignKeySchema(conn, ts)
    tables.Add(ts)
    count += 1
  Next
End Using
Catch ex As Exception
  MessageBox.Show(ex.Message, "Operation failed while extracting the table definition", MessageBoxButtons.OK, MessageBoxIcon.[Error])
End Try
Return tables
End Function

Private Shared Function createTableSchema(ByVal conn As SqlConnection, ByVal tableName As String) As tableDefn
  Dim res As New tableDefn()
  res.name = tableName
  res.columns = New List(Of ColumnSchema)()
  Dim cmd As New SqlCommand("EXEC sp_columns" & tableName & ",", conn)
  Try
    Using row As SqlDataReader = cmd.ExecuteReader()
      While row.Read()
        Dim chkColDetNull As Object = row("COLUMN_NAME")
        If TypeOf chkColDetNull Is DBNull Then
          Continue While
        End If
        Dim colName As String = DirectCast(row("COLUMN_NAME"), String)
        Dim colLength As String
        colLength = DirectCast(row("PRECISION"), Integer)
        chkColDetNull = row("COLUMN_DEF")
        Dim colDefault As String
        If TypeOf chkColDetNull Is DBNull Then
          colDefault = String.Empty
        Else
          colDefault = DirectCast(chkColDetNull, String)
        End If
        Dim colNullable As Boolean = DirectCast(row("IS_NULLABLE"), String) = "YES"
        Dim colDataType As String = DirectCast(row("TYPE_NAME"), String)
        Dim colIdentity As Boolean = False
        If row("TYPE_NAME").ToString().IndexOf("identity") <> -1 Then
If row("TYPE_NAME").ToString().IndexOf("int") <> -1 Then
    colIdentity = True
    coldatatype = "int"
End If
End If
If coldatatype = "int" Then
    coldatatype = "int"
ElseIf coldatatype = "bit" Then
    coldatatype = "bit"
ElseIf coldatatype = "real" Then
    coldatatype = "float"
ElseIf coldatatype = "float" Then
    coldatatype = "double"
ElseIf coldatatype = "datetime" OrElse coldatatype = "smalldatetime" Then
    coldatatype = "datetime"
ElseIf coldatatype = "decimal" OrElse coldatatype = "numeric" Then
    coldatatype = "decimal"
ElseIf coldatatype = "money" OrElse coldatatype = "smallmoney" Then
    coldatatype = "decimal"
ElseIf coldatatype = "char" Then
    coldatatype = "text"
ElseIf coldatatype = "varchar" Then
    coldatatype = "varchar"
ElseIf coldatatype = "text" Then
    coldatatype = "longtext"
ElseIf coldatatype = "nchar" Then
    coldatatype = "nchar"
ElseIf coldatatype = "nvarchar" Then
    coldatatype = "nvarchar"
ElseIf coldatatype = "varbinary(max)" OrElse coldatatype = "image" Then
    coldatatype = "longblob"
ElseIf coldatatype = "varbinary" Then
    coldatatype = "varbinary(" & colLength & ")"
ElseIf coldatatype = "smallint" Then
    coldatatype = "smallint"
ElseIf coldatatype = "tinyint" Then
    coldatatype = "tinyint"
ElseIf coldatatype = "bigint" Then
    coldatatype = "bigint"
ElseIf coldatatype = "sql_variant" Then
    coldatatype = "varchar"
ElseIf coldatatype = "uniqueidentifier" Then
    coldatatype = "varchar"
ElseIf coldatatype = "ntext" Then
    coldatatype = "ntext"
ElseIf coldatatype = "binary" Then
    coldatatype = "blob"
End If
If coldatatype = "bit" OrElse coldatatype = "int" Then
    If colDefault = "('False')" Then
        colDefault = "(0)"
    ElseIf colDefault = "('True')" Then
        colDefault = "(1)"
    End If
End If

Dim col As New ColumnSchema()
col.ColumnName = colName
col.ColumnType = coldatatype
col.ColumnLength = colLength
col.IsNullable = colNullable
col.IsIdentity = colIdentity
col.DefaultValue = colDefault
res.columns.Add(col)
End While

End Try

Dim cmd2 As New SqlCommand("EXEC sp_pkeys '" & tableName & "'", conn)
Using reader As SqlDataReader = cmd2.ExecuteReader()
    res.pKey = New List(Of String)()
    While reader.Read()
        Dim colName As String = DirectCast(reader("COLUMN_NAME"), String)
        res.pKey.Add(colName)
    End While
End Using

Try
    Dim cmd3 As New SqlCommand("exec sp_helpindex '" & tableName & "'", conn)
    Using reader As SqlDataReader = cmd3.ExecuteReader()
        res.Indexes = New List(Of IndexSchema)()
        While reader.Read()
            Dim indexName As String = DirectCast(reader("index_name"), String)
            Dim desc As String = DirectCast(reader("index_description"), String)
            Dim keys As String = DirectCast(reader("index_keys"), String)

            If desc.Contains("primary key") Then
                Continue While
            End If

            Dim index As IndexSchema = getIndexSchema(indexName, desc, keys)
            res.Indexes.Add(index)
        End While
    End Using
CATCH ex AS Exception
End Try
Private Shared Sub convertSchema(ByVal schema As List(Of tableDefn), ByVal dbName As String, ByVal mysqlconstring As String)
    For Each dt As tableDefn In schema
        Try
            getCreateTableQuery(dt, dbName, mysqlconstring)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Operation failed while converting the database", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    Next
End Sub

Private Shared Function getCreateTableQuery(ByVal ts As tableDefn, ByVal dbName As String, ByVal mysqlconstring As String) As String
    Dim sb As New StringBuilder()
    sb.Append("CREATE TABLE " & ts.name & " (" & vbCrLf
    Dim pkey As Boolean = False
    For i As Integer = 0 To ts.columns.Count - 1
        Dim col As ColumnSchema = ts.columns(i)
        Dim cline As String = getColumnStatement(col, ts, pkey)
        sb.Append(cline)
        If i < ts.columns.Count - 1 Then
            sb.Append("," & vbCrLf)
        End If
    Next
    If ts.pKey IsNot Nothing AndAlso (ts.pKey.Count > 0 And Not pkey) Then
        sb.Append("," & vbCrLf
        sb.Append(" PRIMARY KEY (")
        For i As Integer = 0 To ts.pKey.Count - 1
            sb.Append(ts.pKey(i))
            If i < ts.pKey.Count - 1 Then
                sb.Append(",")
            End If
        Next
        sb.Append(")") & vbCrLf)
    Else
        sb.Append(vbrCl)
    End If
End Function
End If

If ts.fKeys.Count > 0 Then
    sb.Append"," & vbCrLf)
    For i As Integer = 0 To ts.fKeys.Count - 1
        Dim foreignKey As fKeyDefn = ts.fKeys(i)
        Dim stmt As String = String.Format("FOREIGN KEY ({0}) REFERENCES {1}({2})", foreignKey.ColumnName, foreignKey.ForeignTableName, foreignKey.ForeignColumnName)
        sb.Append(stmt)
        If i < ts.fKeys.Count - 1 Then
            sb.Append"," & vbCrLf)
        End If
    Next
End If

sb.Append(vbCrLf)
sb.Append(“);” & vbCrLf)

If ts.Indexes IsNot Nothing Then
    For i As Integer = 0 To ts.Indexes.Count - 1
        Dim stmt As String = getCreateIndex(ts.name, ts.Indexes(i))
        sb.Append(stmt & ”;” & vbCrLf)
        ' for
    Next
End If
Dim query As String = sb.ToString()
MySqlTableDef(query, dbName, mysqlconstring)
Return query
End Function

Public Shared Sub mySqlTableDef(ByVal query As String, ByVal dbName As String, ByVal mysqlconstring As String)
    Dim con As New MySqlConnection(mysqlconstring)
    con.Open()
    Try
        Dim queryAdd As New MySqlCommand(query, con)
        queryAdd.ExecuteNonQuery()
    Catch excp As Exception
        Dim myExcp As New Exception(”Could not add the table. Error: ” & excp.Message, excp)
        Throw (myExcp)
    End Try
    con.Close()
Private Shared Sub transferSqlData(ByVal schema As List(Of tableDefn), ByVal dbName As String, ByVal sqlConnString As String, ByVal mysqlconstring As String)
    Dim count As Integer = 0
    For Each dt As tableDefn In schema
        Try
            buildInsertQuery(dt, dbName, sqlConnString, mysqlconstring)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Operation failed while transferring data", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
        count += 1
    Next
    MessageBox.Show("Successfully finished converting the ", count & ") database and transferring data", "Success", MessageBoxButtons.OK, MessageBoxIcon.None)
End Sub

Private Shared Function buildInsertQuery(ByVal ts As tableDefn, ByVal dbName As String, ByVal sqlConnString As String, ByVal mysqlconstring As String)
    Using con As New SqlConnection(sqlConnString)
        con.Open()
        Dim com As New SqlCommand("Select * from " + ts.name, con)
        com.CommandType = CommandType.Text
        Dim dataAdap As New SqlDataAdapter(com)
        Dim datSet As New DataSet()
        dataAdap.Fill(datSet, "row")
        Dim insQuery As String = "insert into " & dbName & "." & ts.name & " values ("
        Dim query As String = ""
        Dim final As String = ""
        If datSet.Tables("row").Rows.Count.Equals(0) Then
            final = "insert into " & dbName & "." & ts.name & " values ();~"
        End If
        For Each datRow As DataRow In datSet.Tables("row").Rows
            query = insQuery
            For col As Integer = 0 To datSet.Tables("row").Columns.Count - 1
                query += "'
                If datRow(col).GetType() Is GetType(DateTime) Then
                    query += DirectCast(datRow(col), DateTime).ToString("yyyy-MM-dd")
                ElseIf datRow(col).GetType() Is GetType(System.Boolean) Then
                    query += If(CBool(datRow(col)), "1", "0")
                Else
                    query += datRow(col).ToString().Replace("'", "\"").Trim()
                End If
            Next
            query = query & ",")
            query = query.Substring(1, query.Length - 1) + "~"
            final += query
        Next
        MessageBox.Show(final, "Success", MessageBoxButtons.OK, MessageBoxIcon.None)
    End Using
End Function
End If
    query += ","
Next

query = query.Substring(0, query.Length - 1)
query += ");~"
final += query
Next
final = final.Remove(final.LastIndexOf("~"))
Dim finalString As String() = final.Split("~")
For Each insertRecord As String In finalString
    MySqlTableDef(insertRecord, dbName, mysqlconstring)
Next
End Using
End Function
BIBLIOGRAPHY


Available:


Available: http://dev.mysql.com/tech-resources/articles/visual-basic-datatypes.html