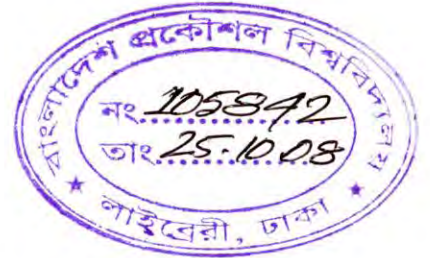


# Development of a Back-End Tool for Loading Data into Data Warehouse using Open-Source Technology



by  
Md. Masum Billah

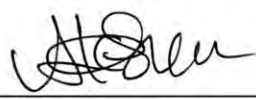
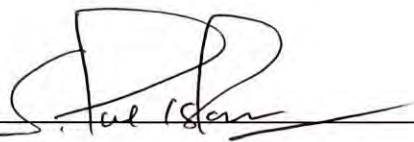
MASTER OF ENGINEERING IN INFORMATION AND COMMUNICATION  
TECHNOLOGY



Institute of Information and Communication Technology  
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY  
August 2008

The project titled "Development of a Back-End Tool for Loading Data into Data Warehouse using Open-Source Technology" submitted by Md. Masum Billah, Roll No: M04053124, (Session: April 2005) has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Master of Engineering (ICT) held on 23 August, 2008.

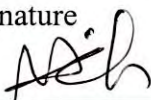
### BOARD OF EXAMINERS

1.   
\_\_\_\_\_
- Dr. Md. Liakot Ali  
Assistant Professor  
Institute of Information and Communication Technology  
BUET, Dhaka-1000. Chairman
  
2.   
\_\_\_\_\_
- Dr. Md. Abul Kashem Mia  
Professor  
Institute of Information and Communication Technology  
BUET, Dhaka-1000. Member
  
3.   
\_\_\_\_\_
- Dr. Md. Saiful Islam  
Assistant Professor  
Institute of Information and Communication Technology  
BUET, Dhaka-1000. Member

## Candidate's Declaration

It is hereby declared that this project or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Signature



---

Md. Masum Billah

## Acknowledgement

My sincere thanks to my project supervisor, Dr. Md. Liakot Ali, for involving me in such a challenging field of advanced database technology. I am grateful to do my project under his supervision. Without his ever helping personalities, this project would not have been completed.

I gratefully acknowledge the support and advice from Professor Dr. S. M. Lutful Kabir, Director, IICT.

I would like to convey my thanks to Dr. Abu Sayed Md. Latiful Hoque for his valuable suggestion about the project.

I thank Md. Ashik ali Khan and Mohammad Ahdabul Islam for their restless encouragement while I was struggling to debug some problems during implementation.

My special thanks to all the teachers, students and staffs of IICT, BUET.

The completion of this project would not have been possible without encouragement of the members of my family and friends. Thank you all.

## Table of Contents

	<b>Page No</b>
Board of Examiners	i
Candidate's Declaration	ii
Acknowledgement	iii
Table of Contents	iv
List of Figures	vi
List of Tables	vii
List of Acronyms	viii
Abstract	1
<b>Chapter 1: Introduction</b>	<b>2</b>
1.1 Overview	2
1.2 Background and Present State of the Problem	2
1.3 Objectives	3
1.4 Organization of the Report	3
<b>Chapter 2: Literature Review</b>	<b>4</b>
2.1 Data Warehouse	4
2.2 Data Warehouse Related Terms and Definition	5
2.2.1 Measure Attribute, Dimension Attribute and Multidimensional Data	5
2.2.2 Fact Table, Star Schema and Data Mart	5
2.3 Operational Database versus Data Warehouse	6
2.4 Data Warehousing Activities	6
2.5 Back-End Tools and Utilities	7
2.5.1 Data Cleaning	7
2.5.2 Data Loading	8
2.5.3 Data Refreshing	9
2.6 Vendor Provided Back-End Tools and Utilities	9



2.7 Open-source Technology	10
2.7.1 Open Source Database Management System	10
<b>Chapter 3: Analysis and Design of the Back-End Tool</b>	<b>11</b>
3.1 Introduction	11
3.2 Data warehouse Architecture	11
3.3 Database Design Methodology for Data Warehouse	12
3.4 Critical Issues for Data Loading	14
3.5 Design of the Back-End Tool	15
3.5.1 Data Sourcing and Data Profiling	16
3.5.2 Target Schema Design	16
3.5.3 Data Mapping	17
3.5.4 Data Extraction	17
3.5.5 SQL Script Generation	18
3.5.5 Data Loading	18
<b>Chapter 4: Back-End Tool Implimentation and Testing</b>	<b>19</b>
4.1 System Infrastructure of Data warehouse	19
4.2 Implimentation of the Back-End Tool	20
4.3 Features of the Back-End Tool	21
4.4 Operation of the Back-End Tool	22
4.3 Test Case Design Showing Data loading into Data Warehouse	27
4.4 Results	28
<b>Chapter 5: Conclusion and Future work</b>	<b>31</b>
5.1 Conclusion	31
5.2 Scope of Future Work	32
References:	33
Appendix A: Program Codes of the Developed Back-End Tool	34

## List of Figures

<b>Figure No.</b>	<b>Caption</b>	<b>Page No</b>
Fig. 3.1	Data Warehouse Architecture	11
Fig. 3.2	Operational Database Schema for Sales Information	13
Fig. 3.3	Data Warehouse Schema for this Project	14
Fig. 3.4	Data Warehouse with Data Loading	15
Fig. 3.5	Data Mapping between <i>Sales</i> and <i>Sales_Fact</i> Table	17
Fig. 4.1	System Infrastructure of Data Warehouse	19
Fig. 4.2	Tool's Process Diagram	20
Fig. 4.3	Snapshot of Back-End Tool	22
Fig. 4.4	A Snapshot of SQL Script Generation	23
Fig. 4.5	A Snapshot of Data Loading into Data Warehouse	24
Fig. 4.6	A Snapshot of SQL Script Execution	25
Fig. 4.7	A Snapshot of PSTN Network Connection Establishment	26

## List of Tables

<b>Table No.</b>	<b>Title</b>	<b>Page No</b>
Table 4.1	Data Set of Site 1	27
Table 3.2	Data Set of Site 2	27
Table 3.3	Data Set of Site 3	28
Table 3.4	Loaded Data into the Data Warehouse	30



## List of Acronyms

<b>Acronym</b>	<b>Full Name</b>
API	Application Programming Interface
BSD	Berkeley Software Distribution
CD	Compact Disc
CSV	Comma Separated Values
DB	Database
DBF	Database Foxpro
DBMS	Database Management System
DVD	Digital Versatile/Video Disc
ER	Entity Relationship
GPL	General Public License
LAN	Local Area Network
LGPL	Lesser General Public License
MDB	Microsoft Database
MSSQL Server	Microsoft Structured Query Language Server
ODBC	Open Database Connectivity
ODI	Oracle Data Integrator
OLAP	On-Line Analytical Processing
OLTP	On-Line Transaction Processing
ORDBMS	Object-Relational Database Management System
OSS	Open-Source Software
PSTN	Public Switched Telephone Network
RDBMS	Relational Database Management System
SQL	Structured Query Language
SSIS	SQL Server Integration Services
TCO	Total Cost of Ownership
VPN	Virtual Private Network

## Abstract

Almost every enterprise uses operational Database Management System (DBMS) for its information management. Many companies have multiple branches in many places, each of which generates significant amount of data. These data may be a potential resource for the knowledge worker (executive, manager and analyst) to make better and faster complex business decision for the organization. For this purpose, historical and subject-oriented selective data are collected from different data sources and these data are stored under a unified schema in a data warehouse. It enables the higher executive to view summarized report over historical data of an organization.

Vendor provided Structured Query Language (SQL) engines like Oracle, MSSQL Server, DB IBM2 provide integrated data warehousing tools. But small and medium enterprises do not take the advantages of the vendor provided DBMS due to its high Total Cost of Ownership (TCO). Although open-source SQL engine is available for free of cost, it does not provide data warehousing tools. As a remedy of this problem, we have developed a back-end tool which extracts data from many sources and load into data warehouse using open-source DBMS MySQL. This tool also periodically refreshes data warehouse on updated data of operational DBMS connected through computer network. It contains an integrated PSTN network connectivity component that establishes dial up connection easily.

# Chapter 1

## Introduction



### 1.1 Overview

Computer technology has brought an evolution in the field of Database Management System (DBMS). Almost every enterprise is developing operational DBMS for its information management. Many companies have multiple branches in many places, each of which generates significant amount of data. These data may be a potential resource for the knowledge worker (executive, manager and analyst) to make better and faster decision for the organization. Operational database systems are unable to meet this need for a variety of reasons. Firstly it contains detail information and many of them have no relevance to management and analysis. Secondly operational databases of an organization are distributed and many of them maintain separate operational database. Thirdly the data processing load for summarized report decreases the performance of the operational database. As a result, data warehouse is specially designed to support management information and analysis. Historical and subject-oriented selective data, collected from multiple sources are stored under a unified schema in a data warehouse at a single site. It enables the higher executive to view summarized report over historical data of an organization.

### 1.2 Background and Present State of the Problem

To fulfill the objectives of data warehouse, a number of front-end tools and back-end tools are required. Front-end tools include analysis, OLAP queries/reporting, data mining etc. while back-end tools include data cleaning, data loading and Refreshing etc. [2]. Vendor specific Structured Query Language (SQL) engines like Oracle, MS SQL Server, IBM DB2 provide data warehousing facility [3-4]. However, small and medium



enterprise cannot take the advantages of the vendor provided DBMS due to its high Total Cost of Ownership (TCO) [7-8]. As a result open-source SQL engines are becoming popular nowadays. But open-source SQL engines have still no generalized data warehousing facility [5-6]. Recently OLAP queries tool is developed using open source DBMS [7]. It is necessary to carry out research project to develop different back-end tools using open-source technology.

### **1.3 Objectives**

The objectives of this project are

- (a) To develop a back end tool for loading data into data warehouse from many sources using open source Structured Query Language (SQL) engine MySQL.
- (b) To test data loading into data warehouse using the developed back-end tool where data sources are multiple operational MySQL databases connected through computer network or backup databases stored in non-volatile media like CD, VCD, pen drive etc.

### **1.4 Organization of the Report**

This report is organized with five chapters. Chapter one presents background of data warehouse, objectives of the project and lay out of the report. Chapter two describes literature review of data warehousing activities, different types of vendor provided tools and utilities, and open-source technology. Chapter three highlights data warehouse architecture, schema design methodology and step by step back-end tool design. Chapter four narrates back-end tool implementation details with Data warehouse infrastructure, back-end tool's process diagram, tool's features, test case design and result. Chapter five mentions conclusion and scope of the future work. This report ends with an appendix that contains the source codes of the project.

## Chapter 2

### Literature Review

#### 2.1 Data Warehouse

A data warehouse can be defined as a “subject-oriented, integrated, time-varying, non-volatile collection of data that is used primarily in organizational decision making” [8]. The explanation of different terms with this definition is given below:

**Subject orientation:** Data warehouse does not contain information that will not be used for informational or analytical processing. It is organized and optimized around major subject areas to provide answers of queries coming from diverse functional areas within a company. On the other hand operational database contain detailed data to satisfy processing requirements of data related to daily operation of a company.

**Integration and transformation:** The data within the warehouse are integrated. This means that there is consistency among naming conventions, measurements of variables, encoding structures, physical attributes and other silent data characteristics. An example of this integration is the treatment of codes such as gender codes. Within a corporation, various applications may represent gender codes in different ways: male versus female, m versus f, 1 versus 0 etc. In the data warehouse gender code is always represented in a consistent way, regardless of how many ways by which it may be encoded in the source data.

**Time Variance:** The data within the warehouse represent the flow of data through time. The data are accurate as of some moment in time, providing a historical perspective. Once the data, collected from different sources are loaded into the data warehouse, it cannot be updated. It is refreshed on a periodic basis, as determined by the business need.



**Non-Volatile:** Data in the warehouse are static not dynamic. Once the data is loaded into the data warehouse, it cannot be updated or deleted. For this reason, the physical design of a data warehouse optimizes the access of data, rather than focusing the requirements of data update and delete.

## 2.2 Data Warehouse Related Terms and Definition

### 2.2.1 Measure Attribute, Dimension Attribute and Multidimensional Data

Suppose we have a relation "sales" with the schema *sales (item-name, color, size, number)*. For data analysis, we can identify some of its attributes as measure attributes, since they measure some value, and can be aggregated upon. For instance, the attribute *number* of the *sales* relation is a measure attribute, since it measures the number of units sold.

Some of the other attributes of the relation are identified as dimension attributes, since they define the dimensions on which measure attributes are viewed, e.g. *item-name*, *color* and *size* are dimension attributes.

Data that can be modeled as dimension attributes and measure attributes are called multidimensional data.

### 2.2.2 Fact Table, Star Schema and Data Mart

In data warehousing, a *fact table* consists of the measurements, metrics or facts of a business process. It is often located at the centre of a star schema, surrounded by dimension tables.

The star schema consists of a few "fact tables" referencing any number of "dimension tables". The fact tables hold the main data, while the dimension tables describe each value of a dimension and can be joined to fact tables as needed.

A *data mart* is a subset of an organizational data store, usually oriented to a specific purpose i.e. Data warehouse contains total information of an organisation whereas *data mart* contains information of one or more department.

### **2.3 Operational Database versus Data Warehouse**

Operational database typically perform clerical data processing tasks such as order entry and banking transaction that are daily operations of an organization. These tasks are structured and repetitive. They consist of short, atomic and isolated transactions. The transactions require detailed, up-to-date data and read or update a few records accessed on their primary keys. Operational databases tend to be hundreds of megabytes to gigabytes in size. Consistency and recoverability of the databases are critical. Its key objective is to maximize transaction throughput rather than query throughput.

Data warehouse, in contrast is targeted for decision support for an organization. Historical, summarized and consolidated data are more important than detailed, individual records. Since data warehouse contains consolidated data, collected from several operational databases over long period of time, its size is larger than operational databases and tend to be hundreds of gigabytes to terabyte in size. In case of data warehouse, query throughput and response time are more important than transaction throughput.

### **2.4 Data Warehousing Activities**

The primary objective of data warehouse is to bring information together from different sources and put the information into a format that is suitable for making business decision. This requires a set of activities that are far more complex than just collecting data and reporting against it. The followings present the data warehousing activities [9]:

- Define the architecture, do capacity planning, select the storage servers, database, OLAP servers, and tools.
- Integrate the servers, storage, and client tools.
- Design the warehouse schema and views.
- Define the physical warehouse organization, data placement, partitioning, and access methods.
- Connect the sources using gateways, ODBC drivers, or other wrappers.
- Design and implement scripts for data extraction, cleaning, transformation, load and refresh.
- Populate the repository with the schema and view definitions, scripts, and other metadata.
- Design and implement end-user applications.
- Roll out the warehouse and applications.

## **2.5 Back-End Tools and Utilities**

Data warehousing uses a variety of data extraction and cleaning tools, and loading and refreshing utilities for populating warehouses [2]. These tools are known as back-end tools and utilities. Descriptions of these tools are given in the following sub-section.

### **2.5.1 Data Cleaning**

Since a data warehouse is used for decision making, it is important that the data in the warehouse be correct. However, since large volumes of data from multiple sources are involved, there is a high probability of errors and anomalies in the data. Therefore, tools that help to detect data anomalies and correct them can have a high payoff. Some examples where data cleaning becomes necessary are: inconsistent field lengths, inconsistent descriptions, inconsistent value assignments, missing entries and violation of integrity constraints. There are three related, but somewhat different, classes of data



cleaning tools [2]. Data migration tools allow simple transformation rules to be specified; e.g., “replace the string gender by sex”. Warehouse Manager from Prism is an example of a popular tool of this kind. Data scrubbing tools use domain-specific knowledge (e.g., postal addresses) to do the scrubbing of data. Data auditing tools make it possible to discover rules and relationships (or to signal violation of stated rules) by scanning data. Thus, such tools may be considered variants of data mining tools.

### **2.5.2 Data Loading**

After extracting, cleaning and transforming of data, data must be loaded into the warehouse. Additional preprocessing may still be required: checking integrity constraints, sorting, summarization, aggregation and other computation to build the derived tables stored in the warehouse. Typically batch load utilities are used for this purpose. In addition to populating the warehouse, a load utility must allow the system administrator to monitor status, to cancel, suspend and resume a load, and to restart after failure with no loss of data integrity. The load utilities for data warehouses have to deal with much larger data volumes than for operational databases. There is only a small time window (usually at night) when the warehouse can be taken offline to refresh it. Sequential loads can take a very long time, e.g., loading a terabyte of data can take weeks and months! Hence, pipelined and partitioned parallelism is typically exploited. Doing a full load has the advantage that it can be treated as a long batch transaction that builds up a new database. While it is in progress, the current database can still support queries. When the load transaction commits, the current database is replaced with the new one. Using periodic checkpoints ensures that if a failure occurs during the load, the process can restart from the last checkpoint. However, even using parallelism, a full load may still take too long.

### 2.5.3 Data Refreshing

Refreshing a warehouse includes propagating updates on source data to correspondingly update the base data and derived data stored in the warehouse. There are two sets of issues to consider: when to refresh, and how to refresh. Usually, the warehouse is refreshed periodically (e.g., daily or weekly). The refresh policy is set by the warehouse administrator, depending on user needs and traffic, and may be different for different sources.

Refresh techniques may also depend on the characteristics of the source and the capabilities of the database servers. Extracting an entire source file or database is usually too expensive, but may be the only choice for legacy data sources. Most contemporary database systems provide replication servers that support incremental techniques for propagating updates from a primary database to one or more replicas. Such replication servers can be used to incrementally refresh a warehouse when the sources change.

## 2.6 Vendor Provided Back-End Tools and Utilities

Vendor provided RDBMS MSSQL Server and Oracle contain back-end tools. *Oracle Data Integrator* (ODI) is a back-end tool used in Oracle database management system. This tool is used for data extraction, loading and transformation. It allows heterogeneous access from a wide range of data sources such as SQL Server, MySQL, DB2, CSV, XML and many other databases and format. *SQL Server Integration Services* (SSIS) is also a back-end tool used in SQL Server database management system. It not only allows data extraction, loading and transformation but also provides excellent data cleaning facilities.



## **2.7 Open-Source Technology**

Open-Source is a development methodology which offers practical accessibility to a product's source (goods and knowledge). Some consider open-source as one of various possible design approaches, while others consider it a critical strategic element of their operations. Before open-source became widely adopted, developers and producers used a variety of phrases to describe the concept; the term open-source gained popularity with the rise of the Internet, which provided access to diverse production models, communication paths, and interactive communities.

Open-Source software (OSS) began as a marketing campaign for free software. OSS can be defined as computer software for which the human-readable source code is made available under a copyright license that meets the Open-Source Definition. This permits users to use, change, improve and redistribute the software in modified or unmodified form. It is very often developed in a public, collaborative manner.

### **2.7.1 Open-Source DBMSs**

Available open-source DBMSs are MySQL, PostgreSQL and mSQL. MySQL is the most popular and best known open-source DBMS. It is distributed under General Public License (GPL) and Lesser General Public License (LGPL). This permits users to use, change, modify, and redistribute it without payment. PostgreSQL is also an object-relational database management system (ORDBMS). It is released under a BSD-style (Berkeley Software Distribution) license and is thus free software. As with many other open-source programs, PostgreSQL is not controlled by any single company, but relies on a global community of developers and companies to develop it.

## Chapter 3

### Analysis and Design of the Back-End Tool

#### 3.1 Introduction

Data warehouse is an essential element of decision support system. Its demand is growing rapidly for complex business decision and analysis. As a result, it has increasingly become a focus of database industry. In this chapter we have discussed data warehouse architecture and its different component, data warehouse schema for this project and step by step back-end tool design.

#### 3.2 Data Warehouse Architecture

Fig. 3.1 shows the data warehouse architecture.

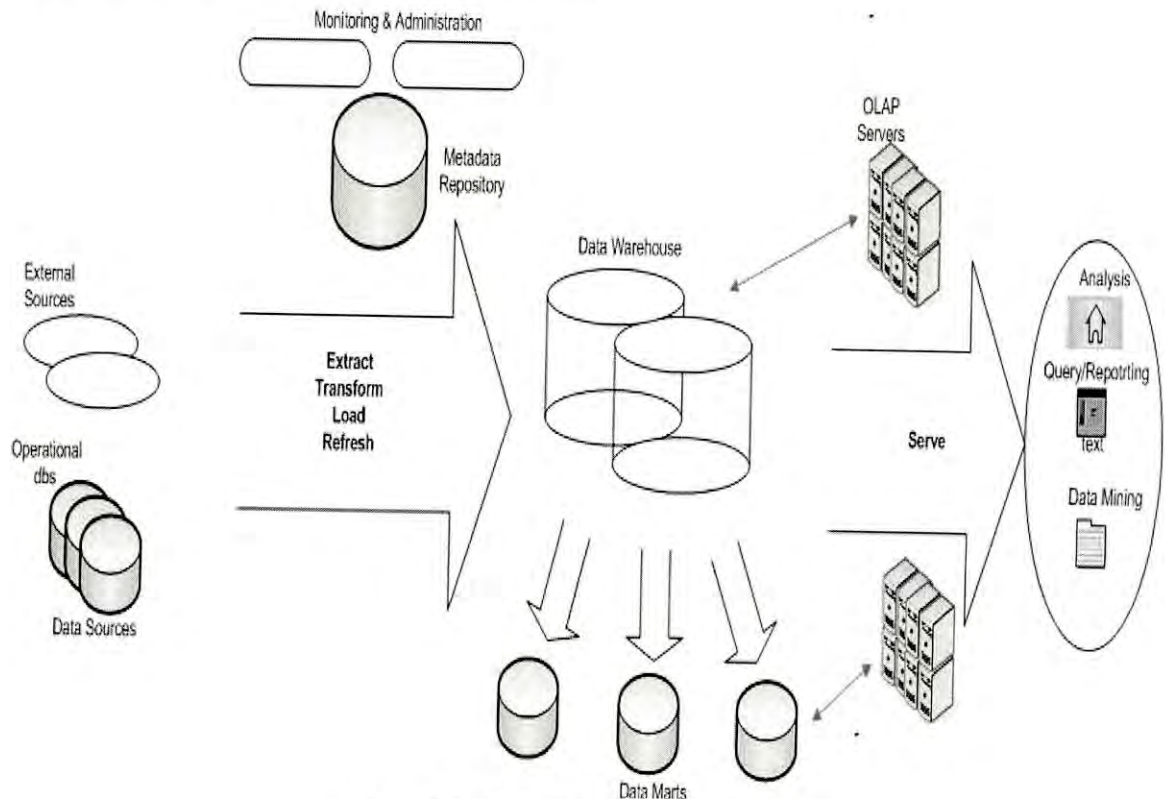


Fig. 3.1: Data Warehouse Architecture



Its data extracting component extracts data from a range of different sources. Its data transforming component removes data inconsistency and data are represented in a consistent way. After completion of data extraction and transformation, data are integrated and loaded into data warehouse using data loading tool. Refreshing tool periodically update data warehouse on the source data. In addition to the main warehouse, there may be several departmental data marts. Data in the warehouse and data marts is stored and managed by one or more warehouse servers. These server present multidimensional views of data through a variety of front end tools. It includes query tools, report writers, analysis tools and data mining tools. Finally, there is a repository for storing and managing metadata and tools for monitoring and administering the warehousing system.

### **3.3 Database Design Methodology for Data Warehouse**

Entity Relationship diagrams and normalization techniques are popularly used for database design in On-line Transaction Processing (OLTP) environments. However, the database designs recommended by ER diagrams are inappropriate for decision support systems. Because the major concern of decision support system is faster query throughput and data loading (including incremental loads) rather than transaction throughput. Most of the data warehouses use a star schema to represent the multidimensional data model. The database consists of a single fact table and a single table for each dimension. Each tuple in the fact table consists of a pointer (foreign key – often uses a generated key for efficiency) to each of the dimensions that provide its multidimensional coordinates, and stores the numeric measures for those coordinates. Each dimension table consists of columns that correspond to attributes of the dimension.

Distributed sales information of an imaginary organization is selected for the test case of this project. Sales related to four normalised table are shown in Fig. 3.2. The tables

are *Sales*, *Customer*, *Site* and *Product*. The *sales* table of the operational database contain sales related detail information in normalized form. Sales information include *Site\_ID*, *Customer\_ID*, *Product\_ID*, *Date*, *Quantity*, *adjustment*, *vat*, *waiver*, *total\_amount* and *remarks*. The *sales* table contain three foreign keys. These are *Site\_ID*, *Customer\_ID* and *Product\_ID* coming from *Site*, *Customer* and *Product* table respectively. *Site* table contains *Site\_ID*, *Site\_name* and *Site\_address* information. *Customer* table contains *Customer\_ID*, *Customer\_name*, *Customer\_address* and *Phone\_number* information, while *Product* table contains *product\_ID*, *Product\_name* and *Sales\_rate* information.

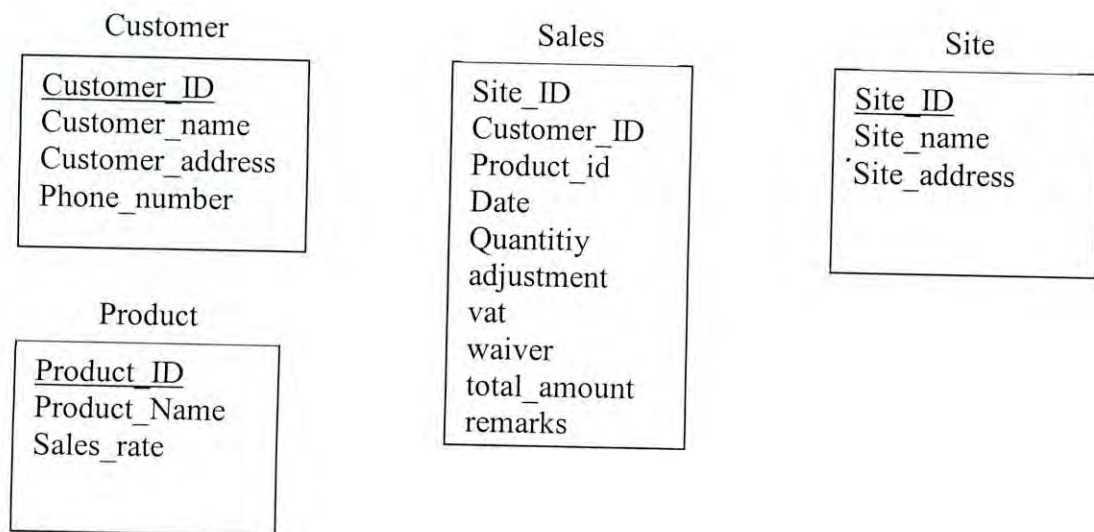


Fig. 3.2: Operational Database Schema for sales information

But all the sales information are not selected for data warehouse. Data warehouse schema includes only important measure attributes and dimensional attributes. Fig. 3.3 shows data warehouse schema for this project. This schema contains five tables : one fact table and four dimensional tables. The fact table is *sales\_Fact* table and the dimensional tables are *Customer*, *Site*, *Product* and *Date*. The field of *Sales\_Fact* table is selected from *Sales* table of the operational database. It contains a measure attribute



*Quantity* and a set of foreign key : *Site\_ID*, *Customer\_ID*, *Product\_ID* and *Date* from each dimension table.

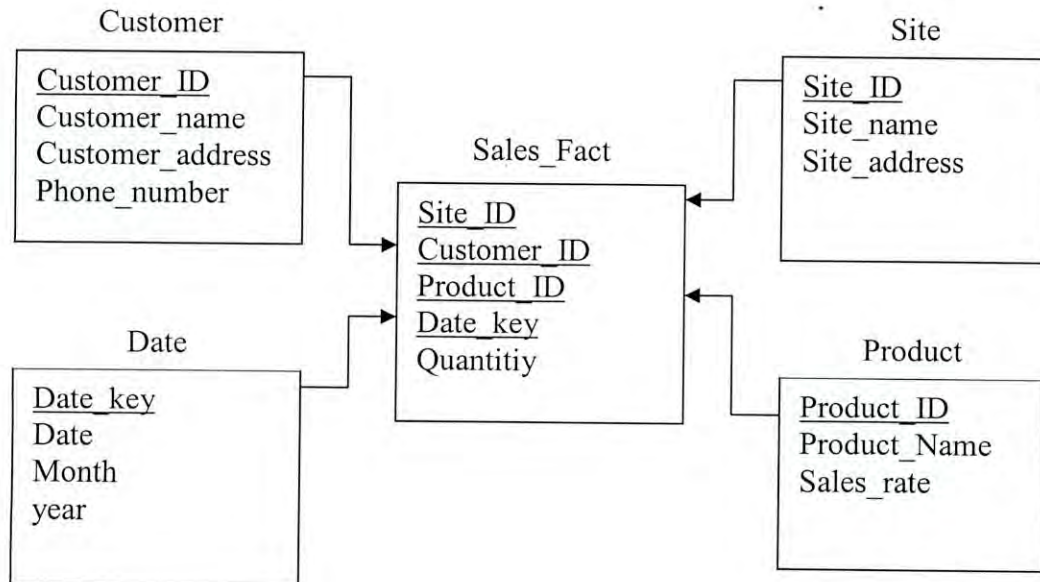


Fig. 3.3: Data Warehouse Schema for this Project

### 3.4 Critical Issues for Data Loading

The loading utilities for data warehouses have to deal with much larger data volumes than that of operational database. Two issues are critical for data loading. First issue is which measures will be taken if a failure occurs during the load. This issue has been solved using periodic checkpoints. It ensures that if a failure occurs during the load, the process can restart from the last checkpoint. The second issue is how to refresh the data warehouse. Extracting an entire database is too expensive. So we have followed incremental technique for propagating updates from the source database to the data warehouse. It incrementally refreshes a warehouse when the sources changes i.e. it only incorporates change information into data warehouse.



### 3.4 Design of the Back-End Tool

Fig. 3.5: shows data warehouse architecture with data loading. There are two types of warehouse architecture in terms of data loading. These are:

- **Source driven architecture:** data sources transmit new information to warehouse, either continuously or periodically (e.g. at night).
- **Destination driven architecture:** warehouse periodically requests new information from data sources.

The project tool supports both of the architecture.

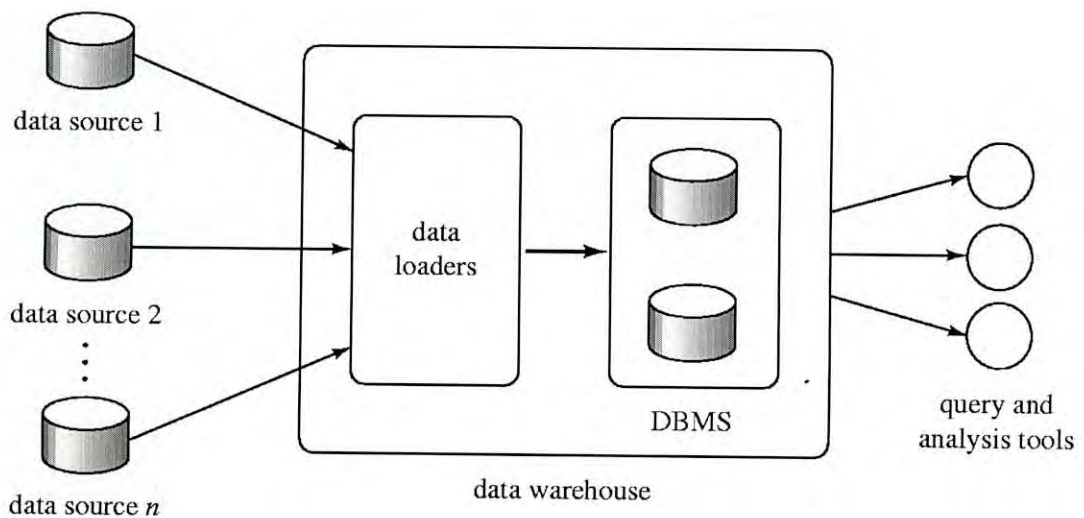


Fig. 3.4: Data Warehouse with Data Loading

In order to design data loading tool for data warehouse, the following steps are followed sequentially.

- (i) Data sourcing and data profiling
- (ii) Target schema design
- (iii) Data mapping
- (iv) Data extraction
- (v) Script generation
- (vi) Data loading

### **3.4.1 Data Sourcing and Data Profiling**

Digital data may come from many sources and different format. It may be offline or online. It may be stored in main frame computer, database server, mini computer, personal computer or any other storage device. It may be stored as XLS, MDB, DBF, CSV, XML, WSDL, SQL script or text. Whatever its source is, it is identified first.

After identification of data source, data profiling activities are initiated. Major data profiling activities are to examine data structure, content, format, functional dependency, anomalies and derive data rules that will later be used within data warehouse.

### **3.4.2 Target Schema Design**

In this phase, data warehouse shema is defined. Constraints, relations, checkings and data rules that are extracted from data source during data profiling are incorporated in data warehouse schema. It significantly ensures quality data loading from a range of different data sources. Star Schema is widely adopted data warehouse schema. This database design methodology is used in this project. The project scema has been shown in Fig. 3.3.

### 3.4.3 Data Mapping

Once data source identification and schema design are completed then data mapping between source schema and warehouse schema are determined. Operational data sources contain detail transaction information. So selective fields from operational database table schema are mapped with data warehouse schema. Data mapping between *sales* table and *sales\_fact* table is shown in Fig. 3.5.

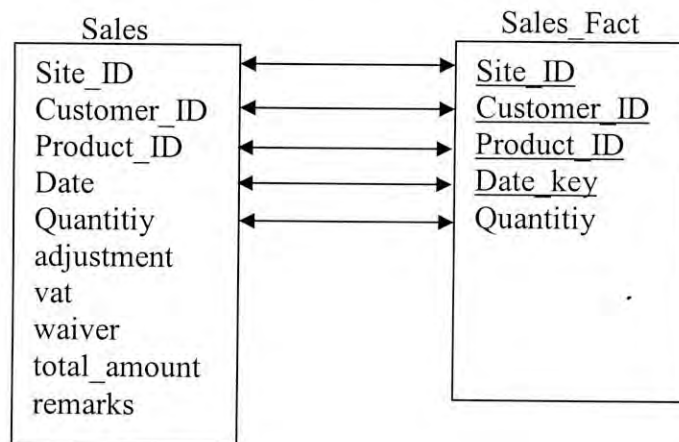


Fig. 3.5: Data mapping between *Sales* and *Sales\_Fact* table

### 3.4.4 Data Extraction

In this phase data are extracted from data source in accordance with previously defined data mapping. For example, the peer mapping between the two tables as shown in Fig. 3.5 is: {(Site\_ID, 1), (Customer\_ID, 3274), (Product\_ID, 3), (Date\_key, 23-10-2007), (Quantity, 10)}. Here, the values are extracted from source in accordance with data mapping between source and target schema.

### **3.4.5 SQL Script Generation**

In this phase extracted data, mapped with corresponding schema are represented using SQL syntax. In this case, data are converted and formatted according to data type. A sample SQL script is: “insert into sales\_fact(Site\_ID, Customer\_ID, Product\_ID, Date\_key, Quantity)values ('1', '3274', '3', format(23-10-2007,'yyyy-mm-dd'),10)”

### **3.4.6 Data Loading**

In this phase Generated SQL scripts are executed into data warehouse SQL engine. If SQL engine does not find any error after parsing the SQL script, it will be executed and data are stored in data warehouse schema. When SQL scripts are generated, each SQL command is terminated with a delimiter. So by marking delimiter, each SQL command is identified and executed into data warehouse.



## Chapter 4

### Back-End Tool Implementation and Testing

#### 4.1 System Infrastructure of Data Warehouse

The following Fig. 4.1 shows the system infrastructure of data warehouse. It includes

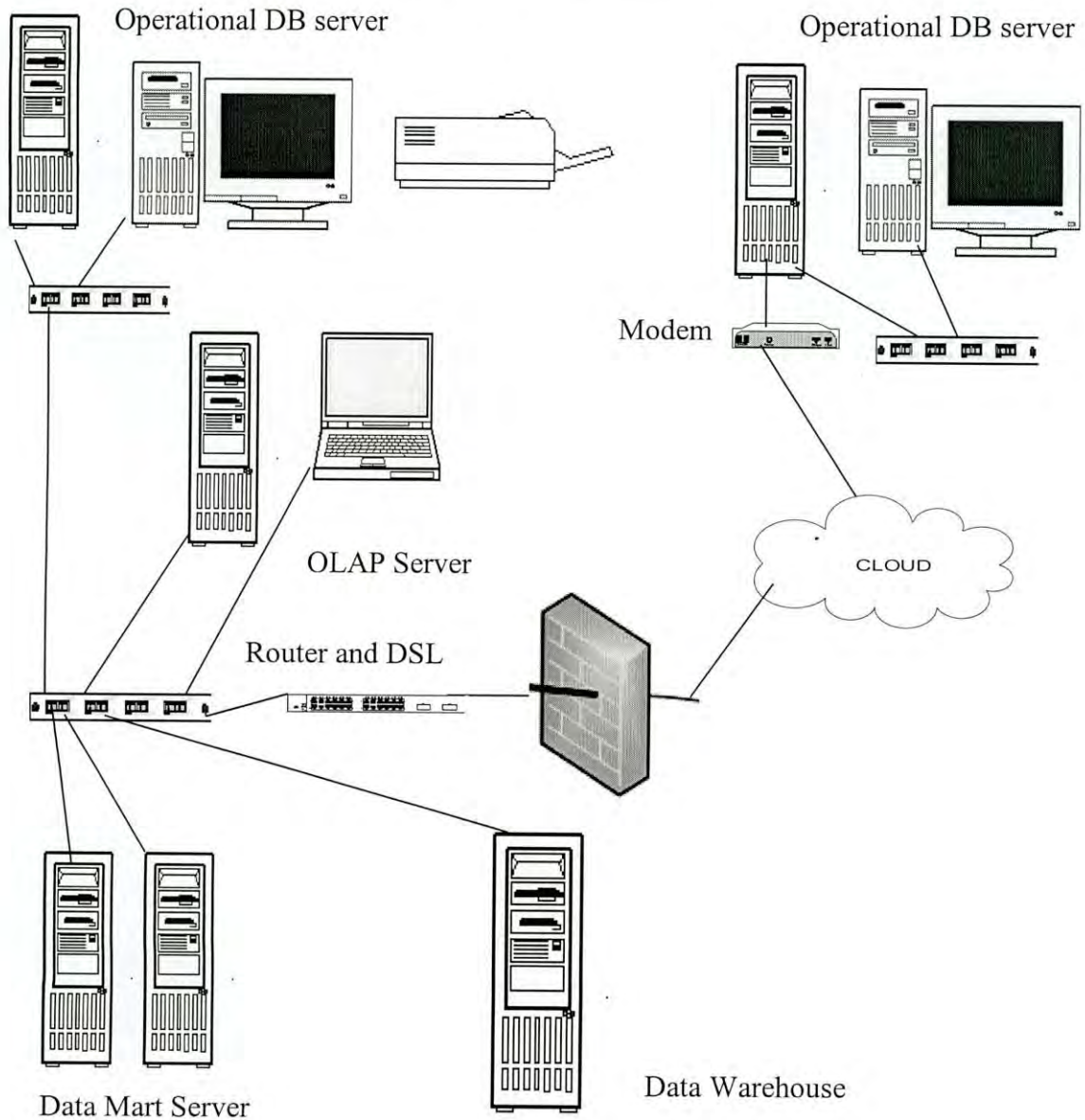


Fig. 4.1: System Infrastructure of Data warehouse



distributed operational database servers, data warehouse servers, OLAP servers, data mart servers, work stations and client laptops which are connected through networks. Distributed operational database are the data sources of the data warehouse and they are connected through LAN, MAN, VPN or PSTN network. Data warehouse server contains information of all department of a organization whereas data mart server only contains information of one or more departments. Decision making queries, analysis and reporting tools access data mart and OLAP server through client computer.

## 4.2 Implementation of the Back-End Tool

Figure 4.2 shows back-end tool's process diagram.

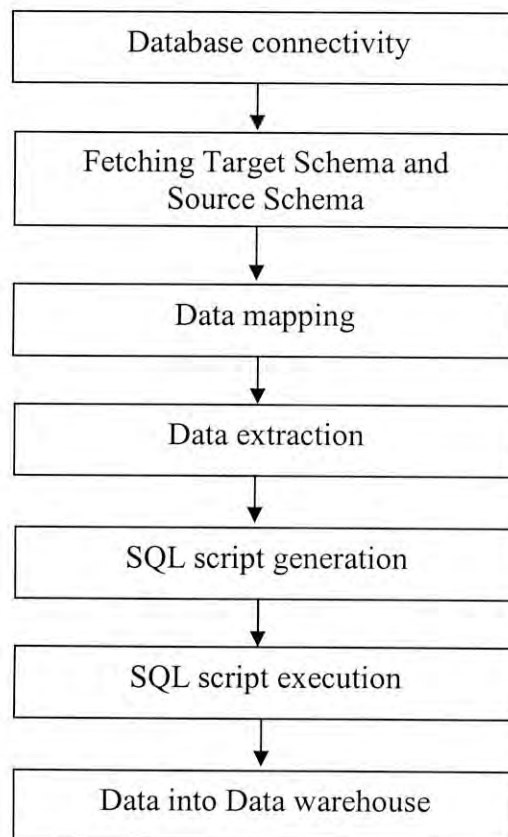


Fig. 4.2: Tool's process diagram

In order to load data from operational source to data warehouse, a number of software modules has been developed and integrated. For database connectivity a module has been developed that connect source database and data warehouse using Open Database Connectivity (ODBC). It allows database users to access database through the tool. The next module has been developed for accessing meta data information of both source database and data warehouse. These meta data information are fed as input for data mapping module. Data mapping module maps field of data warehouse schema with fields of source database schema. Now data extraction module extracts data from source database in accordance with mapped fields. After then extracted data are represented as SQL format through SQL generation module. Finally generated SQL commands are executed by SQL execution module and data are loaded into data warehouse.

### **4.3 Features of the Back-End Tool**

The back-end tool contains the following features:

- The tool is user-friendly.
- The back-end tool can upload data into data warehouse from both operational database and backup database stored in hard drive, CD, DVD or any other media.
- This tool also periodically refreshes data warehouse on updated source data of operational DBMS connected through computer network.
- It uses record by record commit. So if any failure occurs during the data load from operational database, the process can restart from the last committed record. In case of data loading from SQL script, it uses periodic checkpoints. So if a failure occurs during the load, the process can restart from the last checkpoint. Hence, the tool loads data without loss of data integrity.
- It supports both source driven and destination driven data loading.
- This tool contains a automated PSTN network connectivity component.



#### 4.4 Operation of the Back-End Tool

Figure 4.3 shows the initial screen of the backend tool. It has three Menus: *Connection*, *Database* and *About*. *Connection* menu has four sub menus: *Dail*, *hangup*, *login*, *logout*, while *Database* menu contain three sub menus: *Generate File*, *Upload* and *Execute*. The operation of this menu and submenu are discussed respectively.

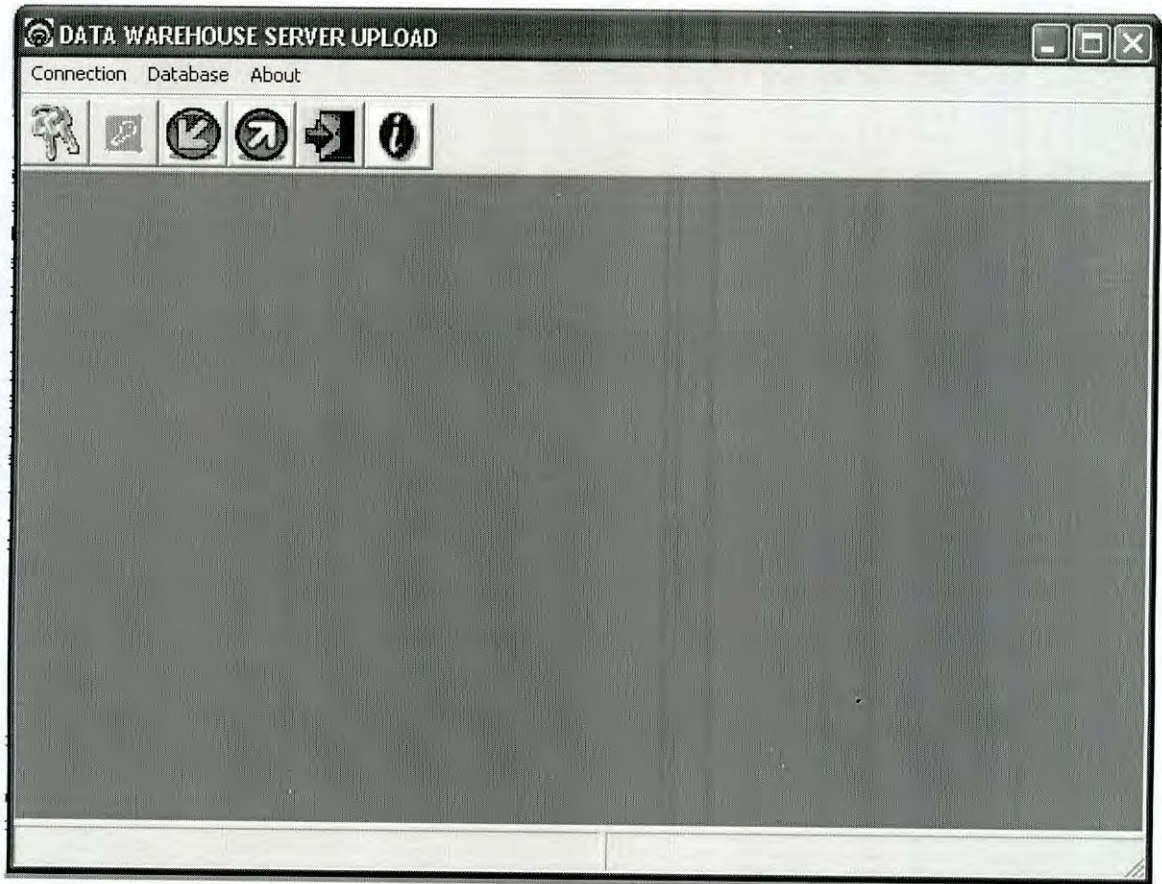


Fig. 4.3: Snapshot of back-end tool



*Generate File* menu is used to generate SQL script for the source database. Figure 4.4 shows a snapshot of SQL script generation from operational database. The SQL script is generated by clicking the button *Generate SQL File*.

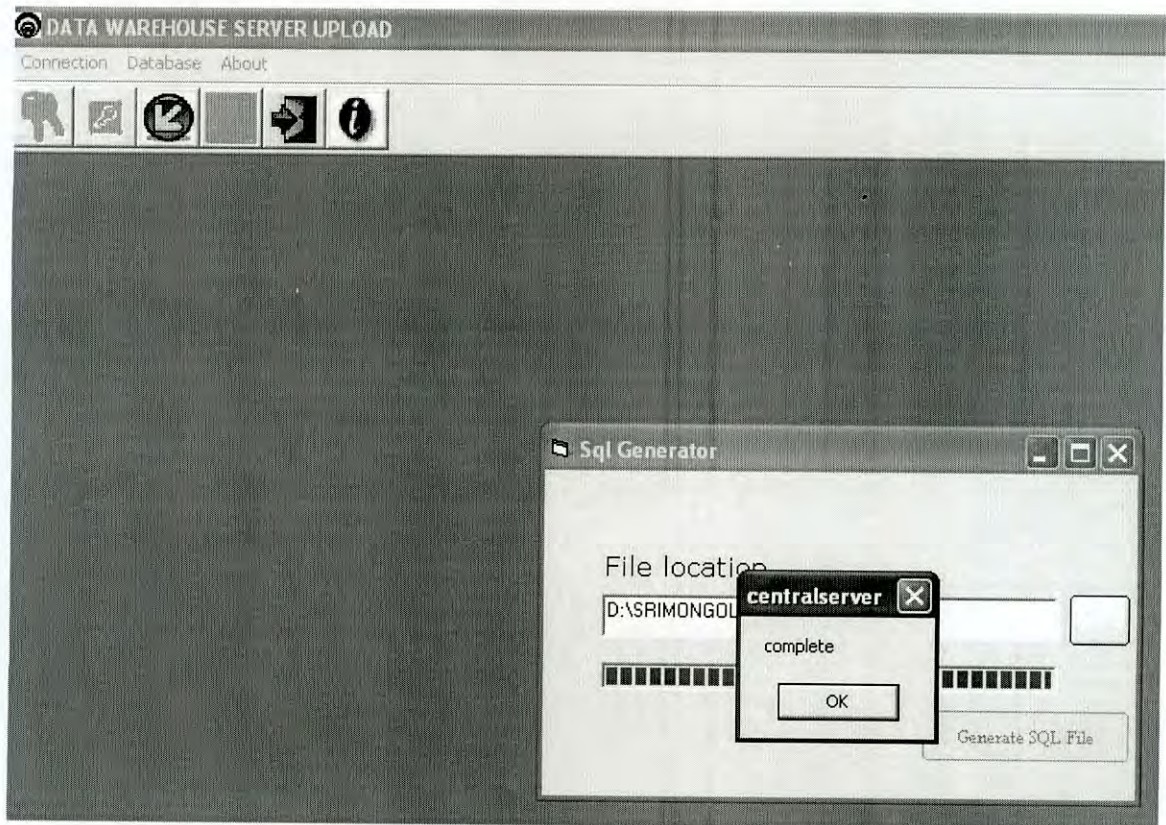


Fig. 4.4: A Snapshot of SQL Script Generation



Figure 4.5 shows a snapshot of data loading into data warehouse from operational database. Here operational database is configured with Data warehouse. The loading process is started by clicking the button *upload now*.

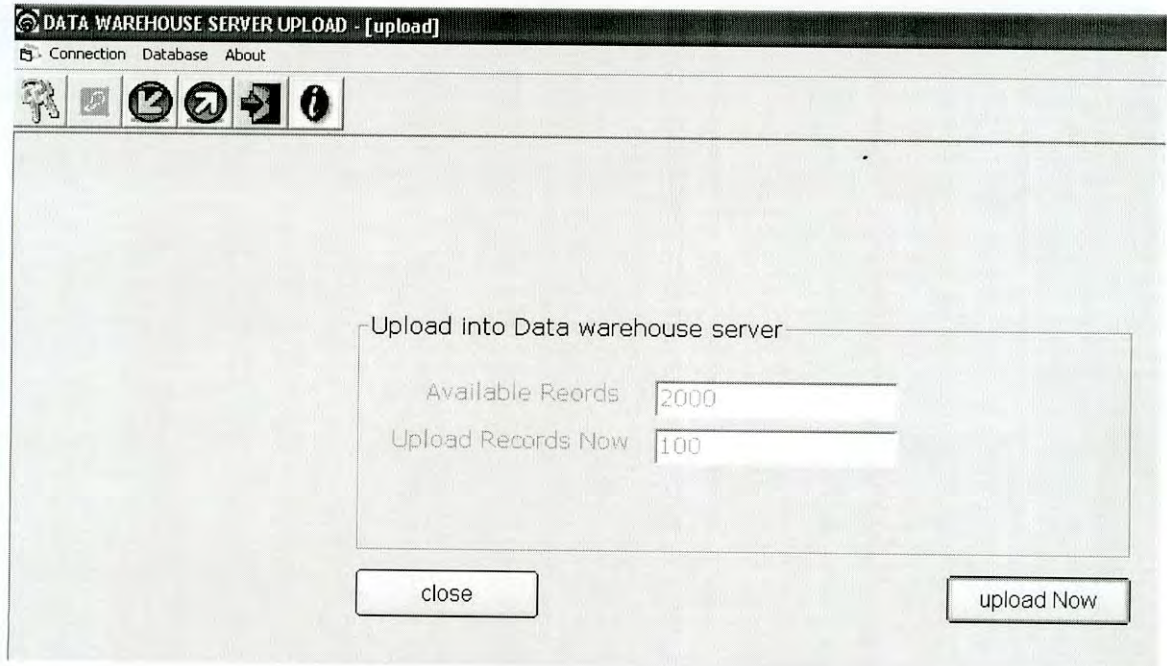


Fig. 4.5: A Snapshot of Data Loading into Data Warehouse

Figure 4.4 shows a snapshot of SQL script execution into data warehouse. In this form generated SQL file is loaded and is executed by clicking the button *Execute SQL file*

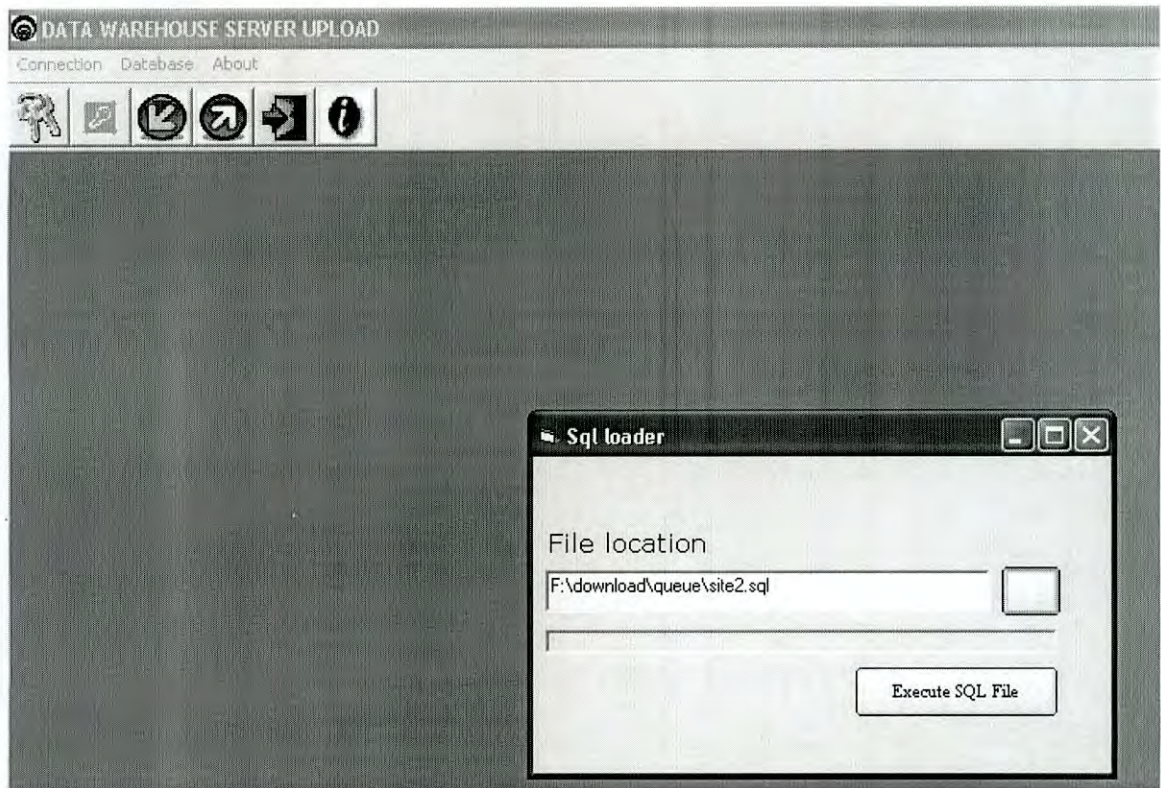


Fig. 4.6: A Snapshot of SQL Script Execution



For PSTN network connectivity, a Application Programming Interface (API) is used and is tested its functionality. Figure 4.6 shows a snapshot of PSTN network connection establishment. Here, dial up connection is initiated by clicking the sub menu *Dial* from *Connection* menu. If the connection is successful, it shows the message “Connection is established” otherwise it shows the message “Connection is failed, Redial Again”.

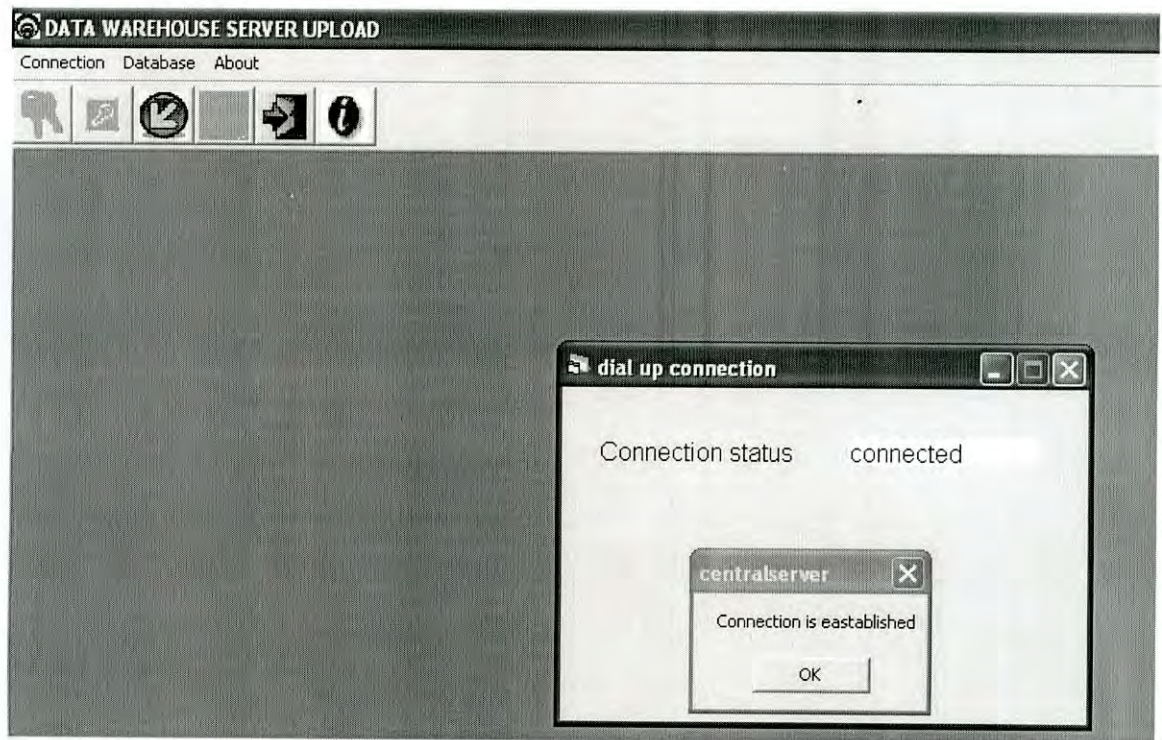


Fig. 4.7: A Snapshot of PSTN Network Connection Establishment

## 4.5 Test Case Design Showing Data Loading into Warehouse

We have considered three operational databases as a test case to load data into data warehouse. The test data sets are as follows:

**Table 4.1: Data Set of Site 1**

site_id	customer_id	product_id	date_key	quantity	value	adjustment	vat	waiver	total_amount	remarks
1	10974	1	2008-08-11	7753	775300	0	233	388	775145	(NULL)
1	11121	2	2008-08-12	3509	350900	175	105	175	351005	(NULL)
1	11221	2	2008-08-04	9485	948500	0	285	474	948311	(NULL)
1	11707	2	2008-08-19	8402	840200	0	252	420	840032	(NULL)
1	11938	2	2008-08-05	1766	176600	0	53	88	176565	(NULL)
1	12522	2	2008-08-04	4305	430500	0	129	215	430414	(NULL)
1	12802	2	2008-08-30	5089	508900	0	153	254	508799	(NULL)
1	13169	2	2008-08-21	9635	963500	0	289	482	963307	(NULL)
1	13542	2	2008-08-20	2813	281300	0	84	141	281243	(NULL)
1	13769	3	2008-08-12	8089	808900	0	243	404	808739	(NULL)

**Table 4.2: Data Set of Site 2**

site_id	customer_id	product_id	date_key	quantity	value	adjustment	vat	waiver	total_amount	remarks
2	10	3	2008-08-07	9504	950400	0	285	475	950210	(NULL)
2	10079	1	2008-08-06	8393	839300	0	252	420	839132	(NULL)
2	10089	3	2008-08-04	7335	733500	0	220	367	733353	(NULL)
2	1022	2	2008-08-02	9635	963500	0	289	482	963307	(NULL)
2	10508	2	2008-08-04	4670	467000	0	140	234	466906	(NULL)
2	10645	3	2008-08-06	9496	949600	0	285	475	949410	(NULL)
2	10668	3	2008-08-15	7767	776700	0	233	388	776545	(NULL)
2	10681	1	2008-08-21	699	69900	35	21	35	69921	(NULL)
2	10711	2	2008-08-17	5835	583500	0	175	292	583383	(NULL)
2	10729	2	2008-08-10	1512	151200	0	45	76	151169	(NULL)



**Table 4.3: Data Set of Site 3**

site_id	customer_id	product_id	date_key	quantity	value	adjustment	vat	waiver	total_amount	remarks
3	17097	2	2008-08-23	9977	997700	0	299	499	997500	(NULL)
3	28617	2	2008-08-07	9789	978900	0	294	489	978705	(NULL)
3	33054	2	2008-08-13	9973	997300	0	299	499	997100	(NULL)
3	48133	1	2008-08-25	9813	981300	0	294	491	981103	(NULL)
3	6314	3	2008-08-30	9858	985800	-493	296	493	985110	(NULL)
3	658	2	2008-08-21	100000	10000000	0	3000	5000	9998000	(NULL)
3	80649	1	2008-08-29	9855	985500	0	296	493	985303	(NULL)
3	82616	2	2008-08-06	9845	984500	0	295	492	984303	(NULL)
3	87489	2	2008-08-06	9934	993400	0	298	497	993201	(NULL)
3	9723	2	2008-08-07	9865	986500	0	296	493	986303	(NULL)

## 4.6 Results

For each data set, the generated SQL script is given bellow. These scripts are executed into data warehouse. After execution of the generated SQL scripts of test case, data warehouse shows the data of Table 4.4.

### Generated SQL for Data set of Site 1:

```
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('1','10974','1', '2008-08-11',7753);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('1','11121','2', '2008-08-12',3509);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('1','11221','2', '2008-08-04',9485);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('1','11707','2', '2008-08-19',8402);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('1','11938','2', '2008-08-05',1766);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('1','12522','2', '2008-08-04',4305);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('1','12802','2', '2008-08-30',5089);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('1','13169','2', '2008-08-21',9635);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('1','13542','2', '2008-08-20',2813);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('1','13769','3', '2008-08-12',8089);
```

**Generated SQL for Data set of Site 2:**

```
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('2','10','3', '2008-08-07',9504);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('2','10079','1', '2008-08-06',8393);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('2','10089','3', '2008-08-04',7335);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('2','1022','2', '2008-08-02',9635);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('2','10508','2', '2008-08-04',4670);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('2','10645','3', '2008-08-06',9496);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('2','10668','3', '2008-08-15',7767);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('2','10681','1', '2008-08-21',699);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('2','10711','2', '2008-08-17',5835);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('2','10729','2', '2008-08-10',1512);
```

**Generated SQL for Data set of Site 3:**

```
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('3','17097','2', '2008-08-23',9977);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('3','28617','2', '2008-08-07',9789);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('3','33054','2', '2008-08-13',9973);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('3','48133','1', '2008-08-25',9813);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('3','6314','3', '2008-08-30',9858);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('3','658','2', '2008-08-21',100000);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('3','80649','1', '2008-08-29',9855);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('3','82616','2', '2008-08-06',9845);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('3','87489','2', '2008-08-06',9934);
insert into sales_fact(site_id,customer_id,product_id,date_key,quantity)values('3','9723','2', '2008-08-07',9865);
```

**Table 4.2: loaded data into the data warehouse**

site_id	customer_id	product_id	date_key	quantity
1	10974	1	2008-08-11	7753
1	11121	2	2008-08-12	3509
1	11221	2	2008-08-04	9485
1	11707	2	2008-08-19	8402
1	11938	2	2008-08-05	1766
1	12522	2	2008-08-04	4305
1	12802	2	2008-08-30	5089
1	13169	2	2008-08-21	9635
1	13542	2	2008-08-20	2813
1	13769	3	2008-08-12	8089
2	10	3	2008-08-07	9504
2	10079	1	2008-08-06	8393
2	10089	3	2008-08-04	7335
2	1022	2	2008-08-02	9635
2	10508	2	2008-08-04	4670
2	10645	3	2008-08-06	9496
2	10668	3	2008-08-15	7767
2	10681	1	2008-08-21	699
2	10711	2	2008-08-17	5835
2	10729	2	2008-08-10	1512
3	17097	2	2008-08-23	9977
3	28617	2	2008-08-07	9789
3	33054	2	2008-08-13	9973
3	48133	1	2008-08-25	9813
3	6314	3	2008-08-30	9858
3	658	2	2008-08-21	100000
3	80649	1	2008-08-29	9855
3	82616	2	2008-08-06	9845
3	87489	2	2008-08-06	9934
3	9723	2	2008-08-07	9865



## Chapter 5

### Conclusion and Future work

#### 5.1 Conclusion

Historical, summarized and consolidated data are important for making business decision and analysis. Data loading tool of data warehouse is able to bring in data from a range of different data sources and integrates the information in a single system.

The objective of this project is to develop a back-end tool using open-source technology for loading data into data warehouse from a number of data sources.

In order to develop the project, we have designed data warehouse schema using open-source DBMS MySQL. We have developed an SQL generator that represents data as SQL format. We have also developed data loader that execute generated SQL commands and load data into data warehouse. This tool also periodically refreshes data warehouse on updated source data of operational DBMS connected through computer network. It use incremental loading during refresh to reduce the volume of data that has to be incorporated into the warehouse. The tool contains an integrated PSTN network connectivity component that establishes dial up connection easily.

The developed tool has been tested for both operational database and backup database and it successfully worked for both cases. The test results show that the tool generates SQL script correctly and loads data into data warehouse without loss of data integrity.

## 5.2 Scope of Future Work

The suggested future works are as follows:

**(i) Development of Data Cleaning Tool:** Heterogeneous data drawn from multiple sources present a number of errors and inconsistencies. The inconsistencies and errors are removed by using data cleaning tools before loading data into data warehouse. So another project can be initiated to develop this tool.

**(ii) Incorporation of Pipelined and Parallelism Feature:** The developed tool uses sequential loading technique. So it can take a very long time in case of huge amount of data. Even it can take weeks and months to load terabyte of data. Hence, pipelined and parallelism feature can be incorporated with the developed tool to handle massive data loading into the warehouse.

**References:**

- [1] A. Sen and A.P. Sinha, "A Comparison of Data Warehousing Methodologies", Communications of the ACM, Vol. 48, No. 3, pp. 79-84, March 2005.
- [2] S. Chaudhury and U. Dayal, "An Overview of Data Warehousing and OLAP Technology", ACM SIGMOD, Vol. 26, No. 1, pp. 65-74, March 1997.
- [3] [http://www.oracle.com/solutions/business\\_intelligence/dw\\_home.html](http://www.oracle.com/solutions/business_intelligence/dw_home.html) (last access on 30-05-2007)
- [4] <http://www.microsoft.com/sql/solutions/dw/default.aspx> (last access on 30-05-2007)
- [5] <http://www.mysql.com> (last access on 30-08-2007)
- [6] <http://www.postgresql.com> (last access on 30-08-2007)
- [7] M.A. Islam , " Online Analytical Processing System for Student's Information of BUET using Open Source Technology", M.Engg. Project Report, ICT, BUET, November, 2006.
- [8] W.H. Inmon, "Building the Data Warehouse". John Wiley, 1992.
- [9] R. Kimball, "The Data Warehouse Toolkit". John Wiley, 1996.
- [10] A. Silberschatz, H. F. Korth and S. Sudarshan, "Database System Concepts", 5th ed., McGraw-Hill, 2005.



## Appendix A: Program Codes of the Developed Back-End Tool

### Source Code for SQL script generation

```

Private Sub data_transfer_insert(table_name As String)
    Dim table_properties(50, 2) As Byte
    Dim I As Integer

    Dim column_number As Integer
    Dim conditional_clause As String
    Dim fields(50) As String
    Dim tbl_prop_rs As ADODB.Recordset
    Dim sqlstr As String
    Dim table_field_str As String
    Set tbl_prop_rs = New ADODB.Recordset
    sqlstr = "show fields from " & table_name & ""
    tbl_prop_rs.Open sqlstr, local_adoconnection
    table_field_str = "("
    'column_number = tbl_prop_rs.RecordCount
    I = 0
    conditional_clause = ""

    While Not tbl_prop_rs.EOF
        'it holds tag of data type for each column
        table_properties(I, 0) = IIf(InStr(tbl_prop_rs!Type, "date"), 1, IIf(InStr(tbl_prop_rs!Type, "char"),
        2, IIf(InStr(tbl_prop_rs!Type, "enum"), 2, IIf(InStr(tbl_prop_rs!Type, "blob"),
        4, IIf(InStr(tbl_prop_rs!Type, "timestamp"), 5, 3))))
        'it holds tag of each column whether it is part of primary key or not
        table_properties(I, 1) = IIf(InStr(tbl_prop_rs!Key, "PRI"), 1, 0)
        'it holds each column name of a table
        fields(I) = tbl_prop_rs!Field
        table_field_str = table_field_str & tbl_prop_rs!Field
        tbl_prop_rs.MoveNext
        If Not tbl_prop_rs.EOF Then
            table_field_str = table_field_str & ","
        Else
            table_field_str = table_field_str & ")"
        End If
        I = I + 1
    Wend
    table_field_str = Replace(table_field_str, ".g_id,o_id", "")
    'Next i
    column_number = I
    'select records from the table for insert
    sqlstr = "select * from " & table_name & " where o_id='1'"
    Dim upload_rs As ADODB.Recordset
    Set upload_rs = New ADODB.Recordset
    upload_rs.Open sqlstr, local_adoconnection, 3, 3

    While (Not upload_rs.EOF And record_count < Val(Me.Txt_available_record))
        'initialisation for insert query

```

```

sqlstr = "insert into " & table_name & table_field_str & "values("
'initialisation for conditional clause
  conditional_clause = ""
For I = 0 To column_number - 4
'it generates sql for both the records to be inserted and conditional clause
Select Case table_properties(I, 0)
  Case 1
    'it is for date type column
    sqlstr = sqlstr + " to_date(" & upload_rs(I) & "','mm-dd-yyyy'),'
    conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
    IIf(conditional_clause = "", "where " & fields(I) & "= to_date(" & upload_rs(I) & "','
    'mm-dd-yyyy'),' " and " & fields(I) & "=to_date(" & upload_rs(I) & "','mm-dd-yyyy'),''), "")
  Case 2
    'it is for string type column
    sqlstr = sqlstr & "" & upload_rs(I) & ""
    conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
    IIf(conditional_clause = "", "where " & fields(I) & "= " & upload_rs(I) & ""', " and " &
    fields(I) & "= " & upload_rs(I) & ""'), "")
  Case 3
    'it is for number type column
    sqlstr = sqlstr & "" & upload_rs(I) & ""
    conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
    IIf(conditional_clause = "", "where " & fields(I) & "= " & upload_rs(I) & ""', " and " &
    fields(I) & "= " & upload_rs(I) & ""'), "")
  Case 4
    'it is for image type column
    sqlstr = sqlstr & "NULL, "
    conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
    IIf(conditional_clause = "", "where " & fields(I) & "= " & upload_rs(I) & ""', " and " &
    fields(I) & "= " & upload_rs(I) & ""'), "")
  Case 5
    sqlstr = sqlstr + " to_date(" & upload_rs(I) & "','mm-dd-yyyy HH:Mi:ss PM'),'
    conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
    IIf(conditional_clause = "", "where " & fields(I) & "= to_date(" & upload_rs(I) & "','
    'mm-dd-yyyy'),' " and " & fields(I) & "=to_date(" & upload_rs(I) & "','
    'mm-dd-yyyy HH:Mi:ss PM'),''), "")

End Select
Next I

Select Case table_properties(I, 0)
  Case 1
    sqlstr = sqlstr & " to_date(" & upload_rs(I) & "','mm-dd-yyyy'),'
    conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
    IIf(conditional_clause = "", "where " & fields(I) & "= to_date(" & upload_rs(I) & "','
    'mm-dd-yyyy'),' " and " & fields(I) & "=to_date(" & upload_rs(I) & "','mm-dd-yyyy'),''), "")
  Case 2
    sqlstr = sqlstr & "" & upload_rs(I) & ""
    conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
    IIf(conditional_clause = "", "where " & fields(I) & "= " & upload_rs(I) & ""', " and " &
    fields(I) & "= " & upload_rs(I) & ""'), "")
  Case 3
    sqlstr = sqlstr & "" & upload_rs(I) & ""

```

```

conditional_clause = conditional_clause + IIf(table_properties(I, 1)= 1,
IIf(conditional_clause = "", "where " & fields(I) & "=" & upload_rs(I) & "", " and " &
fields(I) & "=" & upload_rs(I) & ""), "")
Case 4
sqlstr = sqlstr & "NULL) "
conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
IIf(conditional_clause = "", "where " & fields(I) & "=" & upload_rs(I) & "", " and " &
fields(I) & "=" & upload_rs(I) & ""), "")
Case 5
sqlstr = sqlstr + " to_date(" & upload_rs(I) & ", 'mm-dd-yyyy HH:Mi:ss PM'))"
conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
IIf(conditional_clause = "", "where " & fields(I) & "= to_date(" & upload_rs(I) & ",
'mm-dd-yyyy')", " and " & fields(I) & "=to_date(" & upload_rs(I) & ",
'mm-dd-yyyy HH:Mi:ss PM')"), "")
End Select
'On Error Resume Next
'it is for central server
'adoConnection.Execute (sqlstr)

Print #2, sqlstr & ";" & vbCrLf

'it is for local server
'On Error Resume Next

If Err.Number = 0 Then
sqlstr = "update " & table_name & " set o_id=0' " & conditional_clause & ""
On Error Resume Next
local_adoconnection.Execute (sqlstr)
End If
On Error GoTo 0

upload_rs.MoveNext
record_count = record_count + 1
ProgressBar1.value = ProgressBar1.value + 1
Wend

End Sub

Private Sub data_transfer_update(table_name As String)
Dim table_properties(50, 2) As Byte
Dim I As Integer

Dim column_number As Integer
Dim conditional_clause As String
Dim fields(50) As String
Dim tbl_prop_rs As ADODB.Recordset
Dim sqlstr As String
Set tbl_prop_rs = New ADODB.Recordset
sqlstr = "show fields from " & table_name & ""
tbl_prop_rs.Open sqlstr, local_adoconnection

```



```

'column_number = tbl_prop_rs.RecordCount
I = 0
conditional_clause = ""

While Not tbl_prop_rs.EOF
  'it holds tag of data type for each column
  table_properties(I, 0) = IIf(InStr(tbl_prop_rs!Type, "date"), 1, IIf(InStr(tbl_prop_rs!Type, "char"),
  2, IIf(InStr(tbl_prop_rs!Type, "enum"), 2, IIf(InStr(tbl_prop_rs!Type, "blob"), 4, 3))))
  'it holds tag of each column whether it is part of primary key or not
  table_properties(I, 1) = IIf(InStr(tbl_prop_rs!Key, "PRI"), 1, 0)
  'it holds each column name of a table
  fields(I) = tbl_prop_rs!Field
  tbl_prop_rs.MoveNext

  I = I + 1
Wend
'Next i
column_number = I
' select records from the table for update
sqlstr = "select * from " & table_name & " where o_id=2"
Dim upload_rs As ADODB.Recordset
Set upload_rs = New ADODB.Recordset
upload_rs.Open sqlstr, local_adoconnection, 3, 3

While (Not upload_rs.EOF And record_count < Val(Me.Txt_available_record))
  'initialisation for update query
  sqlstr = "update " & table_name & " set "
  'initialisation for conditional clause
  conditional_clause = ""
  For I = 0 To column_number - 4

    Select Case table_properties(I, 0)
      Case 1
        'it is for date type column
        sqlstr = sqlstr + " " & fields(I) & "=to_date(" & upload_rs(I) & "','mm-dd-yyyy'),'
        conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
        IIf(conditional_clause = "", "where " & fields(I) & "= to_date(" & upload_rs(I) & "','
        'mm-dd-yyyy'),' " and " & fields(I) & "=to_date(" & upload_rs(I) & "','mm-dd-yyyy'),'")', "")
      Case 2
        'it is for string type column
        sqlstr = sqlstr & "" & fields(I) & "=" & upload_rs(I) & ""
        conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
        IIf(conditional_clause = "", "where " & fields(I) & "=" & upload_rs(I) & ""', " and " &
        fields(I) & "=" & upload_rs(I) & ""', ""))
      Case 3
        'it is for number type column
        sqlstr = sqlstr & "" & fields(I) & "=" & upload_rs(I) & ""
        conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
        IIf(conditional_clause = "", "where " & fields(I) & "=" & upload_rs(I) & ""', " and " &
        fields(I) & "=" & upload_rs(I) & ""', ""))
      Case 4
        'it is for image type column
        sqlstr = sqlstr & "" & fields(I) & "=NULL, "
    End Select
  Next I
End While

```

```

        conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
        IIf(conditional_clause = "", "where " & fields(I) & "=" & upload_rs(I) & "", " and " &
        fields(I) & "=" & upload_rs(I) & ""), "")
    End Select
Next I

Select Case table_properties(I, 0)
    Case 1
        sqlstr = sqlstr & " " & fields(I) & "=to_date(" & upload_rs(I) & "','mm-dd-yyyy') "
        conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
        IIf(conditional_clause = "", "where " & fields(I) & "= to_date(" & upload_rs(I) & "','mm-dd-yyyy')", " and " & fields(I) & "=to_date(" & upload_rs(I) & "','mm-dd-yyyy')"), "")
    Case 2
        sqlstr = sqlstr & "" & fields(I) & "=" & upload_rs(I) & ""
        conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
        IIf(conditional_clause = "", "where " & fields(I) & "=" & upload_rs(I) & "", " and " &
        fields(I) & "=" & upload_rs(I) & ""), "")
    Case 3
        sqlstr = sqlstr & "" & fields(I) & "=" & upload_rs(I) & ""
        conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
        IIf(conditional_clause = "", "where " & fields(I) & "=" & upload_rs(I) & "", " and " &
        fields(I) & "=" & upload_rs(I) & ""), "")
    Case 4
        sqlstr = sqlstr & "" & fields(I) & "=NULL "
        conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
        IIf(conditional_clause = "", "where " & fields(I) & "=" & upload_rs(I) & "", " and " &
        fields(I) & "=" & upload_rs(I) & ""), "")

    End Select
    sqlstr = sqlstr & conditional_clause
'On Error Resume Next
'it is for central server
'adoConnection.Execute (sqlstr)
Print #2, sqlstr & ";" & vbCrLf
If Err.Number = 0 Then
    sqlstr = "update " & table_name & " set o_id='0' " & conditional_clause & ""
    'it is for local server
    On Error Resume Next
    local_adoconnection.Execute (sqlstr)
End If
On Error GoTo 0

upload_rs.MoveNext
record_count = record_count + 1
ProgressBar1.value = ProgressBar1.value + 1
Wend

End Sub

```

### Source Code for SQL script execution

```
Sub ExecuteSqlScript()
    Dim Script As String
    Dim FileNumber As Integer
    Dim Delimiter As String
    Dim aSubscript() As String
    Dim Subscript As String
    Dim I As Long
    Delimiter = ";"
    FileNumber = FreeFile
    Script = String(FileLen(CommonDialog_ulpload.FileName), vbNullChar)
    'Grab the scripts inside the file
    Open CommonDialog_ulpload.FileName For Binary As #FileNumber
    Get #FileNumber, , Script
    Close #FileNumber ' Put the scripts into an array
    aSubscript = Split(Script, Delimiter) ' Run each script in the array
    ProgressBar1.Max = UBound(aSubscript)
    For I = 0 To UBound(aSubscript) - 1
        aSubscript(I) = Trim(aSubscript(I))
        Subscript = aSubscript(I)
        Subscript = Replace(Subscript, vbCr, "")
        Subscript = Replace(Subscript, vbLf, "")
        ProgressBar1.value = I
    'On Error Resume Next
    adoConnection.Execute (Subscript)
    'CurrentProject.connection.Execute Subscript
    Next I
End Sub
```



## Source Code for PSTN network connectivity

```

Private Function Dial(ByVal connection As String, ByVal UserName As String, ByVal password As
String) As Boolean
    Dim rp As RASDIALPARAMS, h As Long, resp As Long
    rp.dwSize = Len(rp) + 6
    ChangeBytes connection, rp.szEntryName
    ChangeBytes "", rp.szPhoneNumber 'Phone number stored for the connection
    ChangeBytes "*", rp.szCallbackNumber 'Callback number stored for the connection
    ChangeBytes UserName, rp.szUserName
    ChangeBytes password, rp.szPassword
    ChangeBytes "*", rp.szDomain 'Domain stored for the connection
    'Dial
    resp = RasDial(ByVal 0&, vbNullString, rp, &HFFFF, Me.hWnd, h) 'AddressOf RasDialFunc
    Dial = (resp = 0)

End Function

Private Function ChangeToStringUni(Bytes() As Byte) As String
    'Changes an byte array to a Visual Basic unicode string
    Dim temp As String
    temp = StrConv(Bytes, vbUnicode)
    ChangeToStringUni = Left(temp, InStr(temp, Chr(0)) - 1)
End Function

Private Function ChangeBytes(ByVal str As String, Bytes() As Byte) As Boolean
    'Changes a Visual Basic unicode string to an byte array
    'Returns True if it truncates str
    Dim lenBs As Long 'length of the byte array
    Dim lenStr As Long 'length of the string
    lenBs = UBound(Bytes) - LBound(Bytes)
    lenStr = LenB(StrConv(str, vbFromUnicode))
    If lenBs > lenStr Then
        CopyMemory Bytes(0), str, lenStr
        ZeroMemory Bytes(lenStr), lenBs - lenStr
    ElseIf lenBs = lenStr Then
        CopyMemory Bytes(0), str, lenStr
    Else
        CopyMemory Bytes(0), str, lenBs 'Queda truncado
        ChangeBytes = True
    End If
End Function

Private Sub List1_Click()
    Dim rdp As RASDIALPARAMS, t As Long
    rdp.dwSize = Len(rdp) + 6

```

```

ChangeBytes List1.Text, rdp.szEntryName
'Get User name and password for the connection
t = RasGetEntryDialParams(List1.Text, rdp, 0)
If t = 0 Then
    user = ChangeToStringUni(rdp.szUserName)
    password = ChangeToStringUni(rdp.szPassword)
End If
End Sub

Private Sub Form_Load()
'example created by Daniel Kaufmann (daniel@i.com.uy)
'load the connections
timmer_flag = True
Timer1.Enabled = True
Cmdredial.Enabled = False
Cmdredial.Visible = False
cmdcancel.Enabled = False
cmdcancel.Visible = False
counter = 0
Me.Width = 15360
Me.Height = 11520
Me.Left = frmMDImain.Width / 2 - 2500
Me.Top = frmMDImain.Height / 2 - 3800
Me.Width = frmMDImain.Width / 3
Me.Height = frmMDImain.Height / 3
user = "test"
password = "test123"
Dim s As Long, l As Long, ln As Long, a$
ReDim r(255) As RASENTRYNAME95

r(0).dwSize = 264
s = 256 * r(0).dwSize
l = RasEnumEntries(vbNullString, vbNullString, r(0), s, ln)
For l = 0 To ln - 1
    a$ = StrConv(r(l).szEntryName(), vbUnicode)
    List1.AddItem Left$(a$, InStr(a$, Chr$(0)) - 1)
Next
If List1.ListCount > 0 Then
    List1.ListIndex = 0
    List1_Click
End If
flag = Dial(List1.Text, user, password)
End Sub

```

## Source Code for Data loading

```

Private Sub data_transfer_insert(table_name As String)
  Dim table_properties(50, 2) As Byte
  Dim I As Integer

  Dim column_number As Integer
  Dim conditional_clause As String
  Dim fields(50) As String
  Dim tbl_prop_rs As ADODB.Recordset
  Dim sqlstr As String
  Dim table_field_str As String
  Set tbl_prop_rs = New ADODB.Recordset
  sqlstr = "show fields from " & table_name & ""
  tbl_prop_rs.Open sqlstr, local_adoconnection
  'column_number = tbl_prop_rs.RecordCount
  I = 0
  conditional_clause = ""

  While Not tbl_prop_rs.EOF
    'it holds tag of data type for each column
    table_properties(I, 0) = IIf(InStr(tbl_prop_rs!Type, "date"), 1, IIf(InStr(tbl_prop_rs!Type, "char"),
    2, IIf(InStr(tbl_prop_rs!Type, "enum"), 2, IIf(InStr(tbl_prop_rs!Type, "blob"), 4,
    IIf(InStr(tbl_prop_rs!Type, "timestamp"), 5, 3))))))
    'it holds tag of each column whether it is part of primary key or not
    table_properties(I, 1) = IIf(InStr(tbl_prop_rs!Key, "PRI"), 1, 0)
    'it holds each column name of a table
    fields(I) = tbl_prop_rs!Field
    table_field_str = table_field_str & tbl_prop_rs!Field
    tbl_prop_rs.MoveNext
    If Not tbl_prop_rs.EOF Then
      table_field_str = table_field_str & ","
    Else
      table_field_str = table_field_str & ")"
    End If
    I = I + 1
  Wend
  table_field_str = Replace(table_field_str, ",g_id,o_id", "")
  'Next i
  column_number = I
  'select records from the table for insert
  sqlstr = "select * from " & table_name & " where o_id='1'"
  Dim upload_rs As ADODB.Recordset
  Set upload_rs = New ADODB.Recordset
  upload_rs.Open sqlstr, local_adoconnection, 3, 3

  While (Not upload_rs.EOF And record_count < Val(Me.Txt_upload_record))
    'initialisation for insert query
    sqlstr = "insert into " & table_name & table_field_str & "values("
    'initialisation for conditional clause
    conditional_clause = ""
    For I = 0 To column_number - 4
      'it generates sql for both the records to be inserted and conditional clause

```



```
Select Case table_properties(I, 0)
```

```
Case 1
```

```
'it is for date type column
```

```
sqlstr = sqlstr + " to_date(" & upload_rs(I) & "','mm-dd-yyyy'),'"
```

```
conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
```

```
IIf(conditional_clause = "", "where " & fields(I) & " = to_date(" & upload_rs(I) & "','
```

```
'mm-dd-yyyy'),' " and " & fields(I) & " = to_date(" & upload_rs(I) & "','mm-dd-yyyy'),'"), ""))
```

```
Case 2
```

```
'it is for string type column
```

```
sqlstr = sqlstr & "" & upload_rs(I) & "','"
```

```
conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
```

```
IIf(conditional_clause = "", "where " & fields(I) & " = " & upload_rs(I) & ""', " and " &
```

```
fields(I) & " = " & upload_rs(I) & ""'), ""))
```

```
Case 3
```

```
'it is for number type column
```

```
sqlstr = sqlstr & "" & upload_rs(I) & "','"
```

```
conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
```

```
IIf(conditional_clause = "", "where " & fields(I) & " = " & upload_rs(I) & ""', " and " &
```

```
fields(I) & " = " & upload_rs(I) & ""'), ""))
```

```
Case 4
```

```
'it is for image type column
```

```
sqlstr = sqlstr & "NULL, "
```

```
conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
```

```
IIf(conditional_clause = "", "where " & fields(I) & " = " & upload_rs(I) & ""', " and " &
```

```
fields(I) & " = " & upload_rs(I) & ""'), ""))
```

```
Case 5
```

```
sqlstr = sqlstr + " to_date(" & upload_rs(I) & "','mm-dd-yyyy'),'"
```

```
conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
```

```
IIf(conditional_clause = "", "where " & fields(I) & " = to_date(" & upload_rs(I) & "','
```

```
'mm-dd-yyyy'),' " and " & fields(I) & " = to_date(" & upload_rs(I) & "','
```

```
'mm-dd-yyyy HH:Mi:ss PM'),'"), ""))
```

```
End Select
```

```
Next I
```

```
Select Case table_properties(I, 0)
```

```
Case 1
```

```
sqlstr = sqlstr & " to_date(" & upload_rs(I) & "','mm-dd-yyyy'),'"
```

```
conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
```

```
IIf(conditional_clause = "", "where " & fields(I) & " = to_date(" & upload_rs(I) & "','
```

```
'mm-dd-yyyy'),' " and " & fields(I) & " = to_date(" & upload_rs(I) & "','mm-dd-yyyy'),'"), ""))
```

```
Case 2
```

```
sqlstr = sqlstr & "" & upload_rs(I) & ""'
```

```
conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
```

```
IIf(conditional_clause = "", "where " & fields(I) & " = " & upload_rs(I) & ""', " and " &
```

```
fields(I) & " = " & upload_rs(I) & ""'), ""))
```

```
Case 3
```

```
sqlstr = sqlstr & "" & upload_rs(I) & ""'
```

```
conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
```

```
IIf(conditional_clause = "", "where " & fields(I) & " = " & upload_rs(I) & ""', " and " &
```

```
fields(I) & " = " & upload_rs(I) & ""'), ""))
```

```
Case 4
```

```
sqlstr = sqlstr & "NULL) "
```

```
conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
```

```

        IIf(conditional_clause = "", "where " & fields(I) & " = " & upload_rs(I) & "", " and " &
        Fields(I) & " = " & upload_rs(I) & ""), "")
    Case 5
        sqlstr = sqlstr + " to_date(" & upload_rs(I) & "',mm-dd-yyyy)"
        conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
        IIf(conditional_clause = "", "where " & fields(I) & " = to_date(" & upload_rs(I) & "',
        'mm-dd-yyyy)', " and " & fields(I) & " = to_date(" & upload_rs(I) & "',
        'mm-dd-yyyy HH:Mi:ss PM)'), "")
    End Select
    On Error Resume Next
    'it is for central server
    adoConnection.Execute (sqlstr)
    If Err.Number = 0 Then
        sqlstr = "update " & table_name & " set o_id=0' " & conditional_clause & ""
        'it is for local server
        local_adoconnection.Execute (sqlstr)
    End If
    upload_rs.MoveNext
    record_count = record_count + 1
    ProgressBar1.value = ProgressBar1.value + 1
Wend

End Sub

Private Sub data_transfer_update(table_name As String)
    Dim table_properties(50, 2) As Byte
    Dim I As Integer

    Dim column_number As Integer
    Dim conditional_clause As String
    Dim fields(50) As String
    Dim tbl_prop_rs As ADODB.Recordset
    Dim sqlstr As String
    Set tbl_prop_rs = New ADODB.Recordset
    sqlstr = "show fields from " & table_name & ""
    tbl_prop_rs.Open sqlstr, local_adoconnection
    'column_number = tbl_prop_rs.RecordCount
    I = 0
    conditional_clause = ""

    While Not tbl_prop_rs.EOF
        'it holds tag of data type for each column
        table_properties(I, 0) = IIf(InStr(tbl_prop_rs!Type, "date"), 1, IIf(InStr(tbl_prop_rs!Type, "char"),
        2, IIf(InStr(tbl_prop_rs!Type, "enum"), 2, IIf(InStr(tbl_prop_rs!Type, "blob"), 4, 3))))
        'it holds tag of each column wether it is part of primary ke or not
        table_properties(I, 1) = IIf(InStr(tbl_prop_rs!Key, "PRI"), 1, 0)
        'it holds each column name of a table
        fields(I) = tbl_prop_rs!Field
        tbl_prop_rs.MoveNext

        I = I + 1
    End While
End Sub

```

```

Wend
Next i
column_number = I
' select records from the table for update
sqlstr = "select * from " & table_name & " where o_id='2'"
Dim upload_rs As ADODB.Recordset
Set upload_rs = New ADODB.Recordset
upload_rs.Open sqlstr, local_adoconnection, 3, 3

While (Not upload_rs.EOF And record_count < Val(Me.Txt_upload_record))
  'initialisation for update query
  sqlstr = "update " & table_name & " set "
  'initialisation for conditional clause
  conditional_clause = ""
  For I = 0 To column_number - 4

    Select Case table_properties(I, 0)
      Case 1
        'it is for date type column
        sqlstr = sqlstr + " " & fields(I) & "=to_date(" & upload_rs(I) & "','mm-dd-yyyy'),'
        conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
        IIf(conditional_clause = "", "where " & fields(I) & "= to_date(" & upload_rs(I) & "','
        'mm-dd-yyyy'),' " and " & fields(I) & "=to_date(" & upload_rs(I) & "','mm-dd-yyyy'),'"), "")
      Case 2
        'it is for string type column
        sqlstr = sqlstr & "" & fields(I) & "=" & upload_rs(I) & ","
        conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
        IIf(conditional_clause = "", "where " & fields(I) & "=" & upload_rs(I) & """, " and " &
        fields(I) & "=" & upload_rs(I) & """), "")
      Case 3
        'it is for number type column
        sqlstr = sqlstr & "" & fields(I) & "=" & upload_rs(I) & ","
        conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
        IIf(conditional_clause = "", "where " & fields(I) & "=" & upload_rs(I) & """, " and " &
        fields(I) & "=" & upload_rs(I) & """), "")
      Case 4
        'it is for image type column
        sqlstr = sqlstr & "" & fields(I) & "=NULL, "
        conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
        IIf(conditional_clause = "", "where " & fields(I) & "=" & upload_rs(I) & """, " and " &
        fields(I) & "=" & upload_rs(I) & """), "")
    End Select
  Next I

  Select Case table_properties(I, 0)
    Case 1
      sqlstr = sqlstr + " " & fields(I) & "=to_date(" & upload_rs(I) & "','mm-dd-yyyy)' "
      conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,
      IIf(conditional_clause = "", "where " & fields(I) & "= to_date(" & upload_rs(I) & "','
      'mm-dd-yyyy'),' " and " & fields(I) & "=to_date(" & upload_rs(I) & "','mm-dd-yyyy'),'"), "")
    Case 2
      sqlstr = sqlstr & "" & fields(I) & "=" & upload_rs(I) & "" "
      conditional_clause = conditional_clause + IIf(table_properties(I, 1) = 1,

```



```

        If(conditional_clause = "", "where " & fields(I) & "=" & upload_rs(I) & "", " and " &
        fields(I) & "=" & upload_rs(I) & ""), "")
    Case 3
        sqlstr = sqlstr & "" & fields(I) & "=" & upload_rs(I) & " "
        conditional_clause = conditional_clause + If(table_properties(I, 1) = 1,
        If(conditional_clause = "", "where " & fields(I) & "=" & upload_rs(I) & "", " and " &
        fields(I) & "=" & upload_rs(I) & ""), "")
    Case 4
        sqlstr = sqlstr & "" & fields(I) & "=NULL "
        conditional_clause = conditional_clause + If(table_properties(I, 1) = 1,
        If(conditional_clause = "", "where " & fields(I) & "=" & upload_rs(I) & "", " and " &
        fields(I) & "=" & upload_rs(I) & ""), "")

    End Select
    sqlstr = sqlstr & conditional_clause
    On Error Resume Next
    'it is for central server
    adoConnection.Execute (sqlstr)
    If Err.Number = 0 Then
        sqlstr = "update " & table_name & " set o_id='0' " & conditional_clause & ""
        'it is for local server
        local_adoconnection.Execute (sqlstr)
    End If
    upload_rs.MoveNext
    record_count = record_count + 1 ProgressBar1.value = ProgressBar1.value + 1
Wend

End Sub

```

