SCHEMA SQL

P - 2263

K C T ★ LIBRARY Coimbatore

By

S.JEYAKANNAN
Registration Number: 71205621014

Of

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements

for the award of the degree

Of

MASTER OF COMPUTER APPLICATION

ANNA UNIVERSITY
CHENNAI 600 025

June 2008

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE-641006

BONAFIDE CERTIFICATE

Certified that this project report titled **"SCHEMA SQL"** is the bonafide work of **"Mr.S.JEYAKANNAN"** (Registration Number: **71205621014)** who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

[illegible]
Supervisor

[illegible]
Head of the Department

Submitted to Project and Viva Examination held on 01.07.08

[illegible]
Internal Examiner

[illegible] 01/07/08
External Examiner

COMPREHENSIVE BUSINESS SOLUTION

Project Completion Certificate

This is to certify that Mr. **S. Jeyakannan** (Register No: 71205621014) of Kumaragu College of Technology, had done his project at Comprehensive Business Solutions, Chennai with t project title **"Schema SQL"**, under the guidance of Mr. Christopher Theo Samuel. B from D 2007 to Jun 2008. During the project, he has successfully covered all the areas required for h project.

We wish him all success in his career.

For Comprehensive Business Solutions

[illegible]

Christopher Theo Samuel. B
HR - Manager

ABSTRACT

The project "**SQL SCHEMA**" is developed for the company **COMPREHENSIVE BUSINESS SOLUTIONS,** Chennai. The integration of data, especially from heterogeneous sources, is a hard and widely studied problem. One particularly challenging issue is the integration of sources that are semantically equivalent but schematically heterogeneous. While two such data sources may represent the same information, one may store the information inside tuples (data) while the other may store it in attribute or relation names (schema). The Schema SQL query language is a recent solution to this problem powerful enough to restructure such sources into each other without the loss of information.

In this project we develop an application which implements the well recent Schema SQL .The Schema SQL could get the input from the user as the SQL language and could even manage any type of databases such as Access or Oracle or even MS-SQL. All the data bases could be handled with a single query which is in SQL.

ACKNOWLEDGEMENT

First and foremost I thank God for his good will and blessings showered on me throughout the project. The success of this project needs cooperation and encouragement from different quarters. Words are inadequate to express my profound and deep sense of gratitude to those who helped me in bringing out this project successfully.

I wish to express my deep unfathomable feeling of gratitude and indebtedness to **Dr. Joseph V. Thanikal, Ph.D**, Principal – Kumaraguru College of Technology, Coimbatore for the successful completion of the project work.

I am very gladly taking this opportunity to express a special word of thanks to **Dr. M. Gururajan M.Sc., Ph.D,** Head of the Department, Kumaraguru College of Technology, Coimbatore for encouraging me to do this work.

I am very much indebted to **Mr. A. Muthukumar M.Sc., M.C.A.**, Assistant professor Kumaraguru College of Technology, Coimbatore for his complete assistance, guidance and support given to me throughout my project.

It's always a pleasure and privileges to be associated with a prestigious outstanding esteemed organization **"Comprehensive Business Solution"**, Chennai. I am very happy and grateful to be a part of this Company.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

SQL	Structure Query language
CBS	Comprehensive Business Solutions
.NET	Visual Basic .net
VIT	Variable Instantiation Table
UI	User Interface
FST	Federation System Table
WWW	World Wide Web.

CHAPTER 1

INTRODUCTION

1.1 About the Project

Introduction to Schema SQL

A single organization contains large number of database designed, created and maintained by number of users on different location; these databases can be on different operating system, different platform and different data models. There should be some mechanism for interoperability among databases, for this we provide a principled extension of SQL, called **"Schema SQL"**, that offers the capability of uniform manipulation of data and meta-data in relational multi-database systems. We develop a precise syntax and semantics of Schema SQL in a manner that extends traditional SQL syntax and semantics.

Schema SQL retains the flavor of SQL while supporting querying both data and meta-data. It can be used to define and create "restructuring views", views that represent data in a database in a structure substantially different from original database, in which data and meta-data may be interchanged. Schema SQL provides a great facility for interoperability and data/meta-data management in relational multi-database systems.

Introduction to Project

In recent years, there has been a tremendous propagation of databases in the work place, dominated by relational database systems. An emerging need for sharing data and programs across the different databases has motivated the need for **"Schema SQL"**, sometimes also referred to as heterogeneous database systems. Systems capable of operating over a distributed network and encompassing a heterogeneous mix of computers, operating systems, communication links, and local database systems have become highly desirable, and commercial products are slowly appearing on the market.

One of the fundamental requirements in a multi-database system is interoperability, which is the ability to uniformly share, interpret, and manipulate information in component databases. Almost all factors of heterogeneity in a Schema SQL pose challenges for interoperability. These factors can be classified into semantics issues (e.g., interpreting and cross-relating information in different local databases), syntactic issues (e.g., heterogeneity in database schemas, data models, and in query processing, etc.), and systems issues (e.g. operating systems, communication protocols, consistency management, security management, etc). We consider the problem of interoperability among a number of component relational databases storing semantically similar information in structurally dissimilar ways.

1.2 About the company

Comprehensive Business Solutions Technologies has been in forefront of developing and delivering enterprise asset management, parts & catalog management, inventory & materials management, work order scheduling and work force management, fleets management, facilities and various business solutions. Comprehensive Business Solution's strengths are based upon industry centric management expertise, consulting services, software & technology products development, and process automations by listening to their clients, identifying their challenges from day to day operational issues to strategic management goals, and implementation of business solutions to solve their challenges.

Comprehensive Business Solutions has been providing information technology and database management consulting services for more than a decade. They have successful database management consulting and execution experience with fortune 2000 clients all over the world. Comprehensive Business Solutions has also been developing industry solutions based on Oracle and Microsoft database platforms with customers all over the world. Their expertise includes database architecture, data modeling, security & privacy, migration, database administration and maintenance services.

Comprehensive Business Solutions, based at Chennai, is a part of a group involved in IT development, IT services, Plant Maintenance Consulting & Software, Materials consulting & Software, Payroll and HR Solution , Biometric based IT solutions, Data preparation etc., They have good experience in successfully executing very large projects for Government of India Organizations, Private Organizations for both National and International customers.

Comprehensive Business Solutions has implemented its existing software solutions (ERP, plant , equipment maintenance, Spare parts and materials management , Material Cataloging, Biometric based IT solutions etc.), at multiple customer locations, providing interfaces to SAP, Oracle financial etc.

CBS is a technology oriented company promoted by professionals with rich experience and expertise in the industry. The company is focusing on providing technical consultancy and solutions in the automation industry for various applications. CBS also looks at providing the necessary business computing systems for optimizing the total systems integration, with the help of its vast product range.

CBS offers total solutions on Automatic Identification Systems for Transaction Automation through the much sought after Barcode / Radio Frequency and Biometrics Technology. In general all the IT products and solutions brought to you by one trusted source.

We are located at Chennai and is well equipped with the necessary infrastructure, employing experienced professionals who have put in years in this field offering consultations and solutions in any application area. Our support personnel are fully trained to meet any requirements – Technical / Software and are able to clarify doubts instantly. They increase their clients' competitiveness by rolling out industry solutions leveraging our proven software products, information technology solutions and business Process optimization services.

APPLICATIONS

- Attendance / access control.
- Work-in-process tracking
- Inventory management – raw materials / finished goods.
- Warehousing / distribution logistics.
- Asset management.
- Library management.
- Product identification / labelling.
- Baggage tracking / ticketing
- Vehicle parking.
- Event management.
- Compliance labelling – exports.
- Blood banks.
- Document management.
- Patient / medical records tracking.
- Retail sales – pos activity

- Vehicle parking.
- Event management.
- Compliance labelling – exports.
- Blood banks.
- Document management.
- Patient / medical records tracking.
- Retail sales – pos activity

CHAPTER 2
SYSTEM ANALYSIS

2.1 Existing System

Today every organization contains a large number of database designed, created at different places and these databases can be of different format. This database can be on different systems over the network. It is not possible to query data from different database using Ordinary SQL Query. Sql queries are used only to extract data from single database having same data model and same format. A general problem that is common to all of the above approaches is the resolution of heterogeneities caused by the autonomous, distributed, heterogeneous data sources. Heterogeneities occur at several levels - Semantic level and schema level heterogeneity: This occurs with the same real-world objects and concepts being represented in different databases using multitude of data models and user perspectives; Database, platform and network level heterogeneity

2.2 Proposed system

Today every single organization has database spread over in different places and these database can be of different format. The user wants data from this different database, in order to generate reports. This mechanism can be implemented by a language called Schema SQL which is the extension of SQL.

We use interoperability among databases, which has the ability to uniformly share, interpret, and manipulate information in component databases in a Schema SQL. In this system we design queries to combine the data from different tables which are distributed in different location and this database can be of different format and different data models, the user can write queries to retrieve the data and to combine the tables into single table.

2.3 FEASIBILITY STUDY

2.3.1 Technical Feasibility

Technical Feasibility focuses on technology related issues, Practicality of available technical solution, risks involved and resources available. The man power required is a single person. It also accesses that whether the available technology is mature enough to meet the system needs.

2.3.2 Operational Feasibility

Operational Feasibility is people oriented and focuses on evaluating whether a system will work properly in the organization as well as the feedback of the end users about the problem. All the issues like performance, efficiency, providing information and services to the users, security etc. of the system are covered by operational feasibility.

2.3.3 Economic feasibility

Economic feasibility of the system is the measure of cost effectiveness of the system and includes cost benefit analysis, cost involved, income generated etc. while doing cost benefit analysis, fixed costs (cost of developing system) and cost for operating the system are also taken into account.

CHAPTER 3

DEVELOPMENT ENVIRONMENT

3.1 Hardware requirements

Operating System	:	Windows XP
Processor	:	Intel Pentium IV, 2.6 GHz
RAM	:	256 GB
Hard disk capacity	:	40 GB

3.2 Software requirements

Front End	:	Visual Basic.Net
Back End	:	SQL Server, MS-Access, Oracle

3.3 Software Description

Visual Basic.NET

- **Working With Forms and Controls**

Forms allow us to work visually with controls and other items from the tool box. In VB.NET forms are based on the System.Windows.Forms namespace and the form class is System.Windows.Forms.Form.

A control is an object that can be drawn on to the form to enable or enhance user interaction with the application.

- **Working With Menus**

In this project, we used to control for three main menus namely Database Selection, Connection, and Query builder.

Database Selection menu has the operation of selecting the database through the check box. Connection Wizard performs the operations of selecting the provider, giving the provider with Id, password and Data Source. Then finally testing the connection between two databases. Likewise all other menus are tied up with their corresponding forms through sub menus.

Many people have looked at VB.NET and grumbled about the changes. There are significant changes to the language: a new optional error handling structure, namespaces, true inheritance, free threading, and many others. If you took a Visual Basic 1.0 developer and showed him an n-tier application with an ASP front end, a VB COM component middle tier, and a SQL Server back end full of stored procedures, it would look quite alien to him. Yet, over the past few years, the vast majority of developers have been using Visual Basic to create COM components, and they have become quite versed in ADO as well. The needs for reusability and centralization (a way to avoid distributing components to the desktop) have driven this move to the n-tier model. The move to the Web revealed some problems.

Scalability was an issue, but more complex applications had other requirements, such as transactions that spanned multiple Components, multiple databases, or both. To address these issues, Microsoft created Microsoft Transaction Services (MTS) and COM+ Component Services. MTS (in Windows NT 4) and Component Services (an updated MTS in Windows 2000) acted as an object-hosting environment, allowing you to gain scalability and distributed transactions with relative ease.

However, VB components could not take full advantage of all that Component Services had to offer, such as object pooling, because VB did not support free threading. In the ASP/VB6 model, Microsoft had developers building a component and then calling it via an

ASP. Microsoft realized that it would be a good idea to make the component directly callable over HTTP, so that an application anywhere in the world could use that component. Microsoft threw their support behind SOAP, Simple Object Access Protocol, which allows developers to call a component over HTTP using an XML string, with the data returning via HTTP in an XML string.

Components sport URLs, making them as easy to access as any other Web item. SOAP has the advantage of having been a cross-industry standard, and not just a Microsoft creation. At this point, you might be tempted to think that SOAP is all you need, and that you can just stick with VB6. Therefore it is important to understand what VB.NET gives you, and why it makes sense for you, and many other developers, to upgrade to .NET.

For example, you create components and want them to be callable via SOAP, but how do you let people know that those components exist? .NET includes a discovery mechanism that allows you to find components that are available to you. “Building Web Services with VB.NET.” .NET also provides many other features, such as garbage collection for freeing up resources, true inheritance for the first time ,debugging that works across languages and against running applications, and the ability to create Windows services and console applications.

Before proceeding, it’s important to understand a little bit more about what is meant by “.NET”. There are many “.NETs” here. There is VB.NET, which is the new version of Visual Basic. There is Visual Studio.NET, an Integrated Development Environment that hosts VB.NET, C#, and C++.NET. Underlying all this is the .NET Framework and its core execution engine, the Common Language Runtime. In the .NET model, you write applications that target the .NET Framework.

SQL Server

Internet Integration

The SQL Server database engine includes integrated XML support. It also has the scalability, availability, and security features required to operate as the data storage component of the largest Web sites. The SQL Server programming model is integrated with the Windows DNA architecture for developing Web applications, and SQL Server supports features such as English Query and the Microsoft Search Service to incorporate user-friendly queries and powerful search capabilities in Web applications.

Scalability and Availability

The same database engine can be used across platforms ranging from laptop computers running Microsoft Windows 98 through large, multiprocessor servers running Microsoft Windows 2000 Data Center Edition. SQL Server 2000 Enterprise Edition supports features such as federated servers, indexed views, and large memory support that allow it to scale to the performance levels required by the largest Web sites.

Enterprise-Level Database Features

The SQL Server relational database engine supports the features required to support demanding data processing environments. The database engine protects data integrity while minimizing the overhead of managing thousands of users concurrently modifying the database. SQL Server distributed queries allow you to reference data from multiple sources as if it were a part of a SQL Server 2000 database, while at the same time, the distributed transaction support protects the integrity of any updates of the distributed data. Replication allows you to also maintain multiple copies of data, while ensuring that the separate copies remain synchronized. You can replicate a set of data to multiple, mobile, disconnected users, have them work autonomously, and then merge their modifications back to the publisher

K C T
LIBRARY
Coimbatore - 641 006.
P-2263

Ease of Installation, deployment and use

SQL Server includes a set of administrative and development tools that improve upon the process of installing, deploying, managing, and using SQL Server across several sites. SQL Server also supports a standards-based programming model integrated with the Windows DNA, making the use of SQL Server databases and data warehouses a seamless part of building powerful and scalable systems. These features allow you to rapidly deliver SQL Server applications that customers can implement with a minimum of installation and administrative overhead.

Data Warehousing

SQL Server includes tools for extracting and analyzing summary data for online analytical processing. SQL Server also includes tools for visually designing databases and analyzing data using English-based questions.

Oracle

Portability

Oracle is ported to more platforms than any of its competition, running on more than 100 hardware platforms and 20 networking protocols. This makes writing an Oracle application fairly safe from changes of direction in hardware and operating system. One caveat, however, is that applications using some constructs may have to be reworked when porting them to a block mode environment. You can also develop a fairly fully featured application with little knowledge of the underlying OS.

Market Presence

Oracle is by far the largest RDBMS Vendor, and spends more on R&D than most of its competitors earn in total revenue. Oracle has the largest independent RDBMS market share in VMS, UNIX and OS/2 Server fields. This market clout means that you are unlikely to be left in the lurch by Oracle and there are always lots of third party interfaces supported and also, proficient staffs are relatively easy to get.

Backup and Recovery

Oracle provides industrial strength support for on-line backup and recovery and good software fault tolerance to disk failure. You can also do point-in-time recovery. Of course, you need the archive mechanisms and storage space to do this, but Oracle supports continuous archiving to tape devices spanning multiple volumes.

Performance

Speed of a tuned Oracle database and application is quite good, even with large databases. The performance is not only "raw", but includes consideration of performance with locking and transaction control.

SQL Dialect

The dialect of SQL offered by Oracle is in my opinion superior to the others in the extensions it offers over ANSI-2, which is very much a lowest common denominator. Constructs such as the absolute function and decode keyword are very powerful Oracle additions to the standard SQL.

CHAPTER 4

SYSTEM DESIGN

4.1 FUNCTIONAL REQUIREMENTS

The Functional Specification was verified that the product requirements are being met and to help plan resource requirements. Therefore, it includes visual aspects, behavioral characteristics, and other UI matters. The Functional Specification also contains details about all menu items, keyboard shortcuts, and/or command buttons that are used to implement the desired functionality, as well as gestures and combinations of gestures that perform simple and complex operations. It also contains illustrations of any windows produced (including dialog boxes) and the wording of prompts and other messages that users may see.

4.2 NON-FUNCTIONAL REQUIREMENTS:

4.2.1 Usability

In order to make Schema SQL more usable, developing an easy-to-use User Interface (UI) is very high-priority. However, there are many factors, such as the access experience and technical expertise, to consider while making sure that the UI layout is intuitive and well-executed. The greatest challenge of designing an easy-to-use UI, then, is to figure out how to accommodate both beginning and advanced users.]

Schema SQL will overcome this challenge by limiting the number of features available initially, and allowing advanced users to make use of them when they feel ready. Schema SQL will also follow typical UI standards for naming menus, buttons, and dialog boxes whenever possible in order to make the environment more familiar to users.

4.2.2 Reliability

Because Schema SQL is a network-based, a lot of the reliability falls under the issue of performance. In addition, much of it falls under security as well (such as the compromising of user accounts). However, the following attributes of reliability which do not relate directly to performance and security deserve mention:

- User account data should not be corrupted as a result of a server crash.
- Neither the client nor the server should crash due to receipt of invalid data.

4.2.3 Security

Security measures will be implemented to allow some level of security within the system. These security measures include:

Access will be controlled with usernames and passwords only; no personal information will be collected from users. This has the side-effect of disabling simple password recovery, since users will have to contact system administrators in order to have their accounts unlocked.

Because the passwords are stored on the server, the client should send an encrypted form of the password so that the server administrator should not be able to easily see the password simply by reading incoming packets. This precaution exists because many users have the tendency to use the same password on multiple service accounts.

4.2.4 Performance

Because Schema SQL is a multi-database system used over a network, the performance of Schema SQL will largely be determined by network performance. Therefore, an efficient use of bandwidth will be paramount to the success of this project. Another factor to consider is server scalability; since it is intended to have no hard cap on the number of users that can be connected to any individual server, it is vital that server continues to behave reliably as the number of users increases.

4.2.5 Maintainability and Upgradeability

We will concentrate on following good design methods and maintain good documentation to allow anyone who is interested to work on this project without having to sift through code in order to determine what is going on with the project.

4.3 Architectural Design

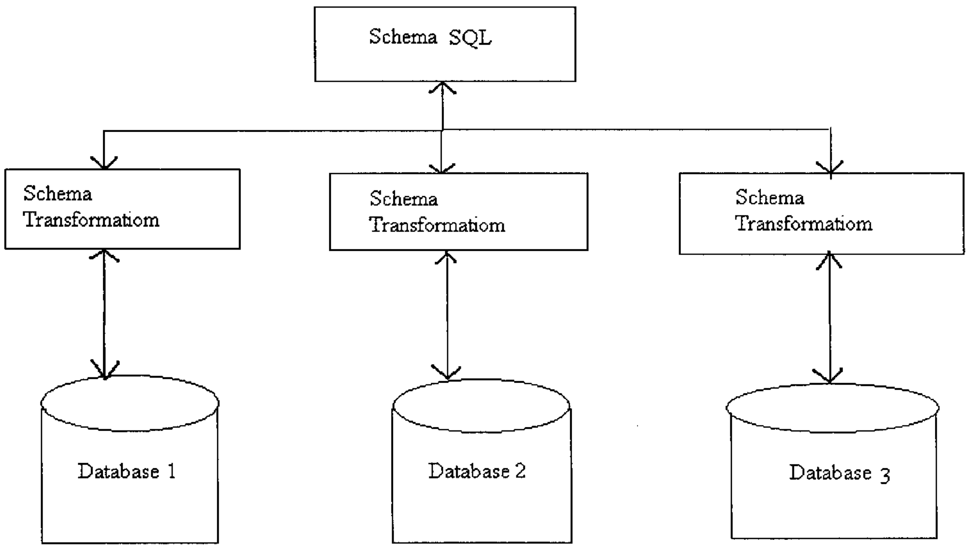
In this section we describe the architecture of a system for implementing a multi-database querying and restructuring facility based on SCHEMA SQL. The architecture consists of a SCHEMA SQL server that communicates with the local databases in the federation.

We assume that the meta-information comprising of component database names, names of the relations in each database, names of the attributes in each relation, and possibly other useful information (such as statistical information on the component databases useful for query optimization) are stored in the SCHEMA SQL server in the form of a relation called Federation System Table (FST).

In this architecture, global SCHEMA SQL queries are submitted to the SCHEMA SQL server, which determines a series of local SQL queries and submits them to the local databases. The SCHEMA SQL server then collects the answers from local databases, and, using its own resident SQL engine, executes a final series of SQL queries to produce the answer to the global query.

Query processing in a SCHEMA SQL environment consists of two major phases. In the first phase, tables called VIT's (Variable Instantiation Table) corresponding to the variable declaration in the FROM clause of a SCHEMA SQL statement are generated. The schema of a VIT consists of all the variables in one or more variable declarations in i.e. from clause and its contents correspond to instantiations of these variables. VIT's are materialized by executing appropriate SQL queries on the FST and/or component databases.

In the second phase, the SCHEMA SQL query is rewritten into an equivalent SQL query on the VIT's and the generated answer is appropriately presented to the users.



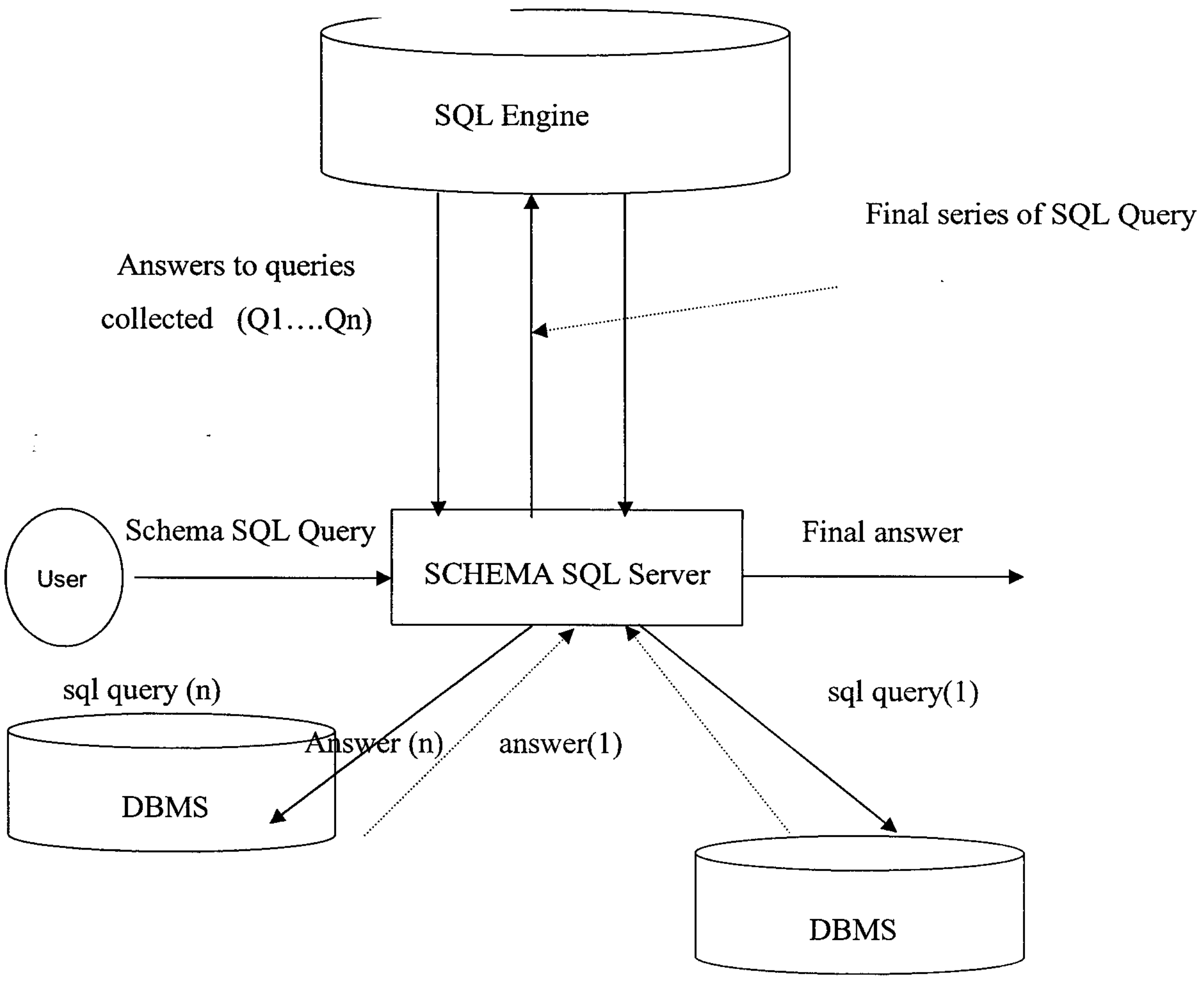


Figure 4.1 Architecture of Schema SQL

System Flow Diagram

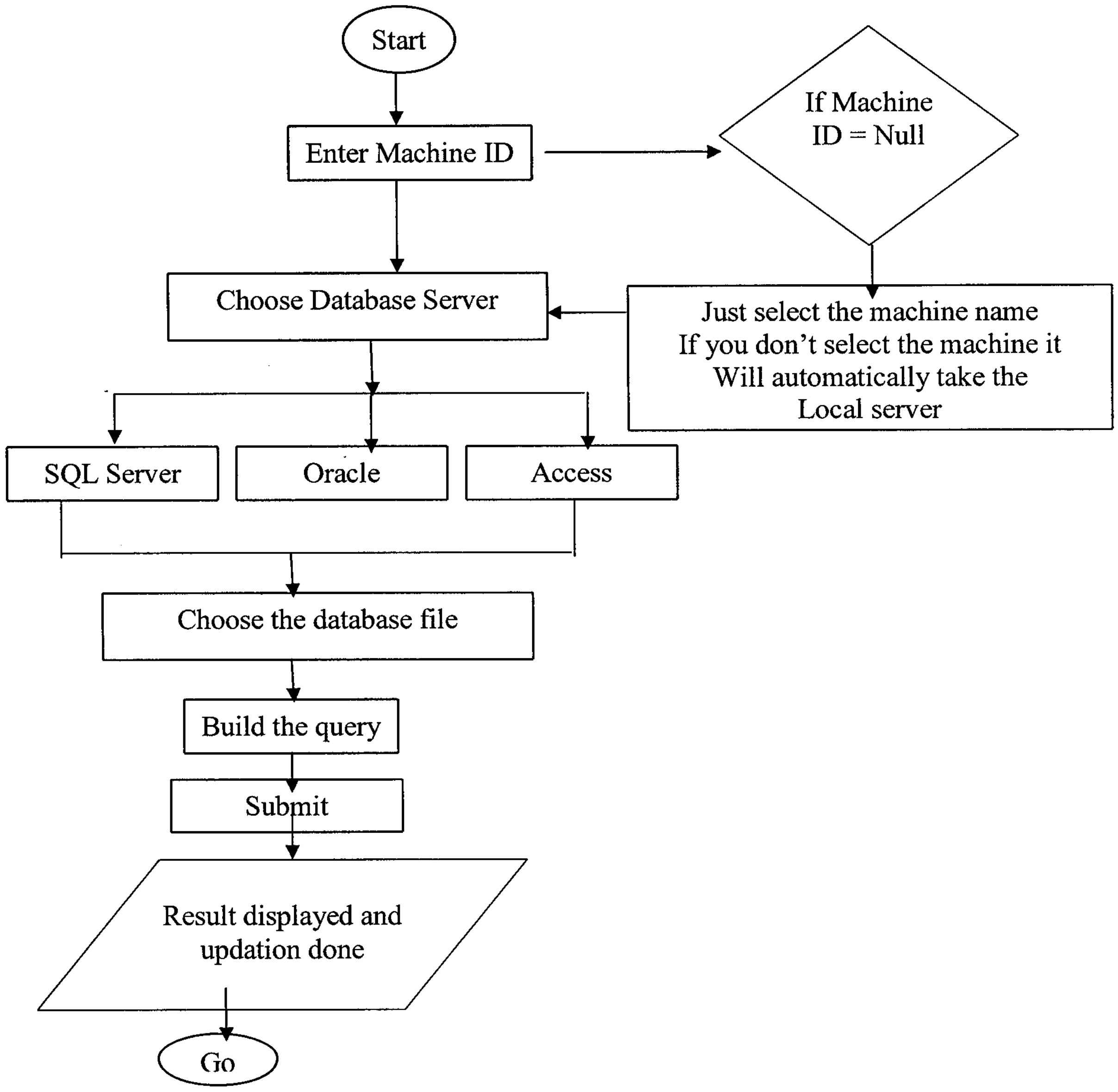

Start
Enter Machine ID
If Machine ID = Null
Choose Database Server
Just select the machine name If you don't select the machine it Will automatically take the Local server
SQL Server
Oracle
Access
Choose the database file
Build the query
Submit
Result displayed and updation done
Go

Figure 4.2 System Flow Diagram for Query Optimization

Block Diagram

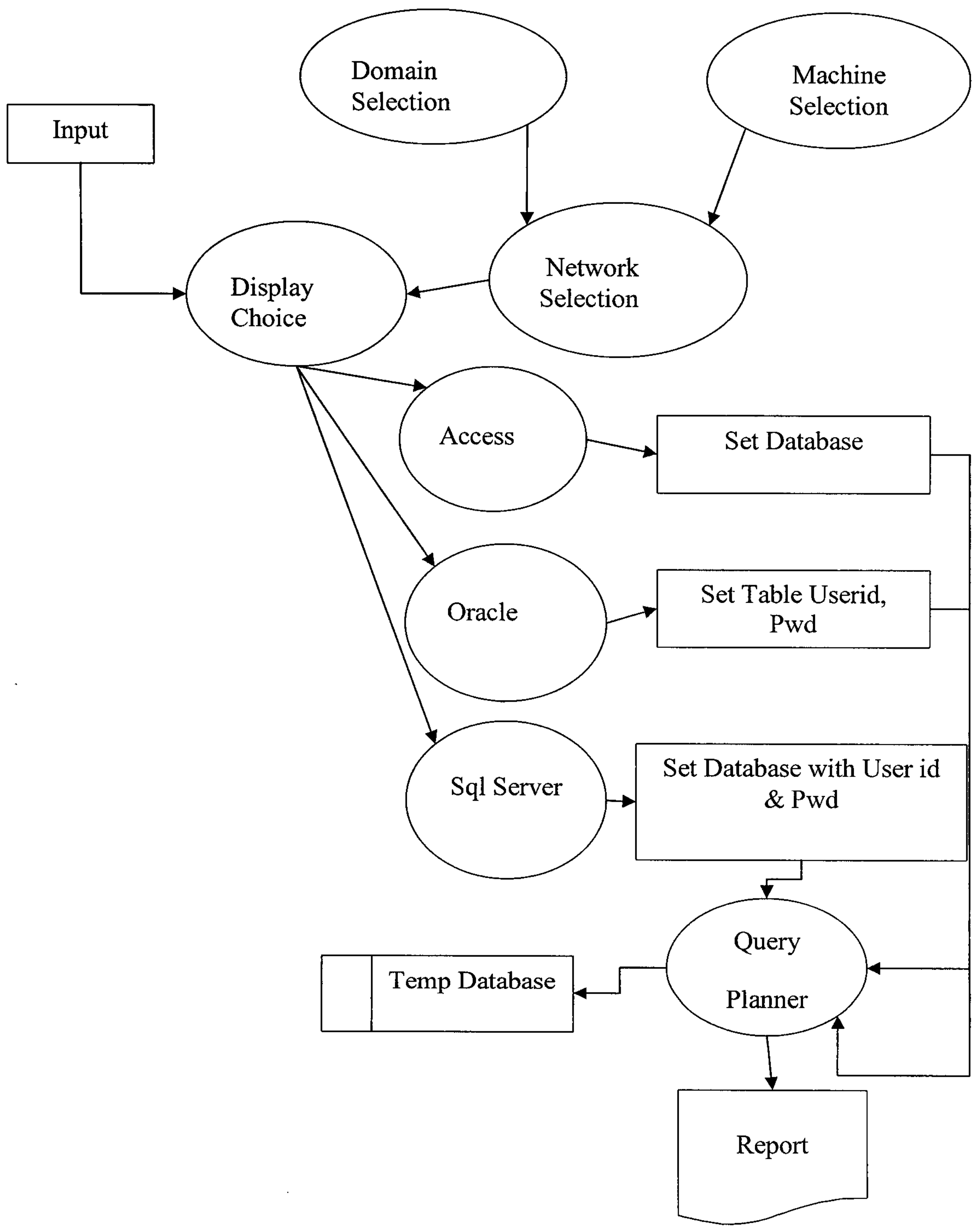


Figure 4.3 Block Diagram For Schema Sql

Data Flow diagram

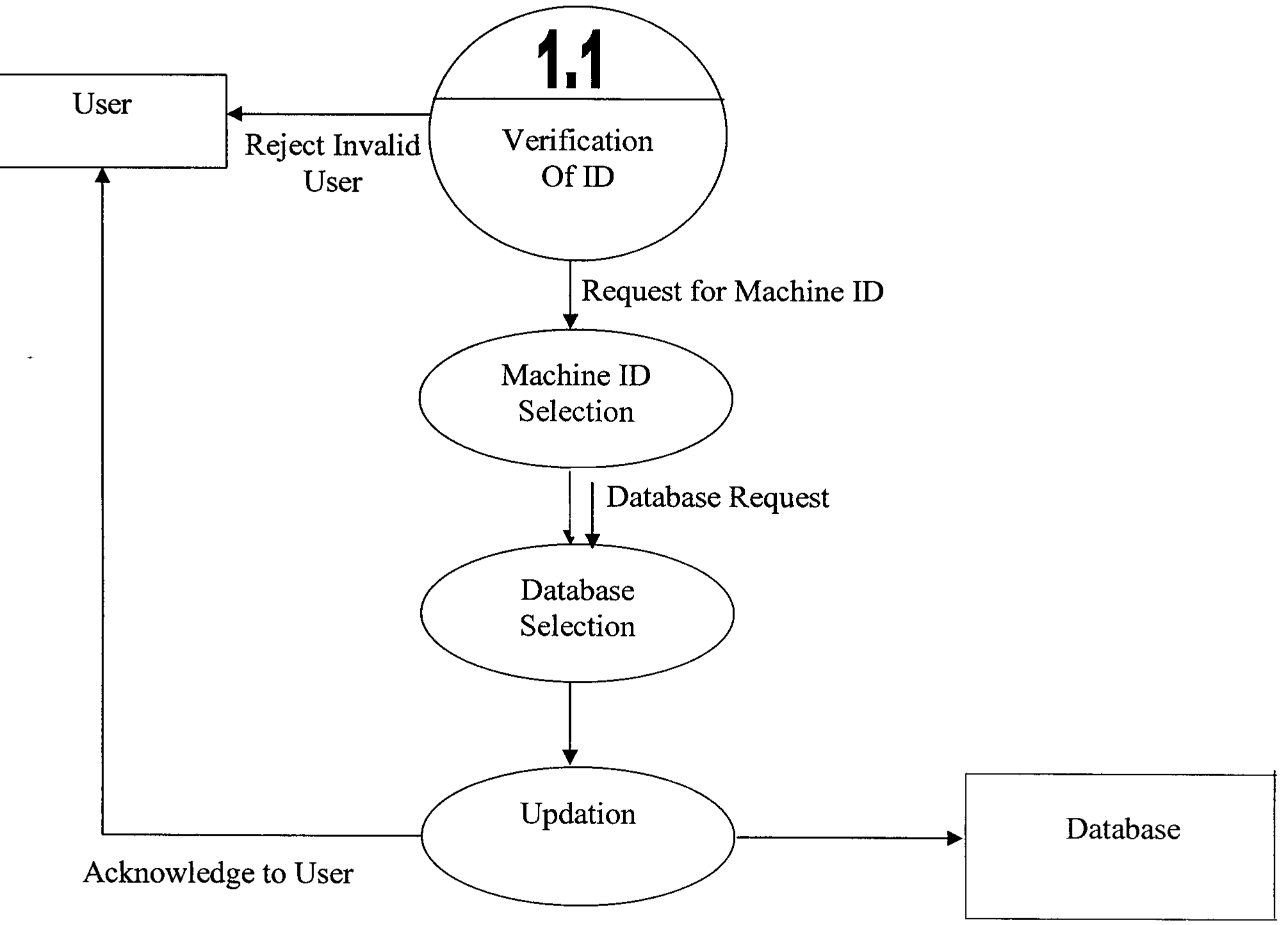


Figure 4.4 Data Flow Diagram for Schema SQL

CHAPTER 5

SYSTEM DEVELOPMENT

5.1 Introduction

System development is a series of operations performed to manipulate data to produce output from computer system. This aim at translating the design of the system produced during the design phase into code in user programming language. A modular approach is used for the development of the software.

The development phase for the project was created from the specifications created during the design phase. A principal activity of the development phase is coding and testing the computer program that make up the computer program component of the overall system. Other important activities include implementation, planning, equipment acquisition and system testing. The development phase concludes with the report and review.

5.2 MODULES

This project has three modules, they are

- ➢ Database selection Module.
- ➢ Connection Module.
- ➢ Query Builder Module.

5.2.1 Database Selection Module

This module is used to select various database servers. Here we use three database management systems MS-Access, Oracle and SQL Server. In this module the user have to select the database for access and SQL server and also have to select the table for oracle. Moreover the user has to select the provider for the various database systems.

5.2.2 Connection Module

Domain Selection

In this form, the user has to choose the domain like main, workgroup. This combo box list out all domain names and the user may choose any one domain form it.

5.2.3 Query Builder Module

This module is used to build the query using create, update, delete, select, insert commands. This form has buttons to create all these types of queries and also to execute queries. It also contains list boxes to display the Views and Tables from the selected databases

.

The last button named "mixed" is used to combine two tables from various databases and to display the result. This creates a new table and stored the combined results in to a temporary table. All the results are displayed in the Data grid.

We write the query on the text box. And this form is so flexible to create the query, because the user no need to know syntax for the command used in query. The syntax for all the commands is given in the help menu.

Sample Queries

Access and Oracle

```
select A::table1.name, A::table1.street,A::table1.city, O::table3.desig from A::table1, O::table3
where A::table1.name=O::table3.name
```

Access and SQL Server

```
select A::table1.name, A::table1.street,A::table1.city, S::table4.desig from A::table1, S::table4
where A::table1.name=S::table4.name
```

SQL Server and Oracle

```
select S::table4.name, S::table4.street,S::table4.city, O::table3.desig from S::table4, O::table3
where S::table4.name=O::table3.name
```

```
select A::table1.name, A::table1.street,A::table1.city, S::table4.desig from A::table1, S::table4
where A::table1.name=S::table4.name group by name
```

5.4 SYSTEM SECURITY

In the networking process, the authentication is a important one. It checks whether the client as well as server are access by the authorized person are not. By using this process, we can secure the file in the database and then also quit the unauthorized access of the database. Following is a list of requirements for database security.

5.4.1 Physical Database Integrity

The data of a database are immune to physical problems, such as power failures, and someone can reconstruct the database if it is destroyed through a catastrophe.

5.4.2 Logical Database Integrity

The structure of the database is preserved. With logical integrity of a database, a modification to the value of one field does not affect other fields.

5.4.3 Access Control

A user is allowed to access only authorized data, and different users can be restricted to different modes of access.

5.4.4 User Authentication

Every user is positively identified, both for the audit trail and for permission to access certain data.

CHAPTER 6

SYSTEM TESTING

Testing is a process of executing a program with the intention of finding an error. A good test case is one that has a high probability of finding an as-yet undiscovered error. During the development of a project, errors of various types can occur at any stage. At each phase, different techniques are used to detect the errors. However, some error such as those occur while collecting requirements and some design errors have also to be removed and the system tested for the successful working of any project

6.1 Verification and Validation

System Verification

System verification answers the question "Am I building the product right?" It includes the review of interim work steps and interim deliverables during a project to ensure they are acceptable. Verification also determines if the system is consistent, adheres to standards, uses reliable techniques and prudent practices, and performs the selected functions in the correct manner. In data access, it verifies whether the right data is being accessed, in terms of the right place and in the right way.

For example, the stations names gather from database, so each station names should be verified whether they are bound to the correct database field. It is done during development of the key artifacts. Verification is a demonstration of consistency, completeness, and correctness of the software at each stage and between each stage of the development life cycle. In result analysis, verification is done during the development itself. Each database bindings are verified after binding to test whether the control is bound to the right data field.

System Validation

Validation answers the question "Am I building the right product?" This checks whether the developer is moving towards the right product, whether the development is moving the actual intended product that was agreed upon in the beginning. Validation also determines if the system compiles with the requirements and performs functions for which it is intended and meets the organization's goal and user needs. It is traditional and is performed at the end of the project. In data access, it checks whether we are accessing the right data, in terms of data required to satisfy the requirement.

Validation is performed after a work product is produced against established criteria ensuring that the product integrates correctly into the environment. It determines the correctness of the final software product by a development project with respect to the user needs and requirements.

Functional validation is done in the Railway Reservation checking System to check whether each of the functions is done correctly as expected in every module.

The system is validated following cases:

- Check the authentication of users to prevent illegal access
- Number of validation
- Character validation
- Validations to check the special characters
- Validations to check the proper number entry of the text in the input field
- Validations to check for proper display of error messages when the input fields are kept empty.

6.2 Unit Testing

Unit testing focuses verification effort on the smallest unit of software unit of software design, the module. Using the procedural design description as a guide, important control paths are tested to uncover errors within the boundary of the module.

Test 1:

Procedure

The mandatory fields have to be filled before proceeding to next process.

Solution

The alert message has to be displayed to fill the mandatory details.

Test 2:

Procedure

The reservation process involves adding airline detail, passenger detail and the payment detail involving various operations.

Solution

This problem is solved by using session tracking.

Test 3:

Procedure

When a reservation is in progress and if a user tries to create a new reservation.

Solution

The prompt is made that the reservation is in progress and want to continue with the current reservation or the previous one.

6.3 Integration Testing

Integration testing is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with interfacing. The objective is to take unit tested modules and build a program structure that has been dictated by design. Modules are integrated by moving download through the control hierarchy, beginning with the main control module. Modules subordinate to the main control module. Modules subordinate to the main control module are incorporated into the structure in either a depth first or breadth first manner. As integration testing is conducted, the tester should identify critical modules.

User Acceptance testing

User acceptance of the system is a key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required.

The acceptance testing will be covered as listed by the acceptance criteria mentioned below:

- Functionalities
- Interfaces
- Interfaces with other software
- Performance Criteria
- Development and testing criteria
- Testing automation
- Quality criteria

CHAPTER 7

IMPLEMENTATION AND FUTURE ENHANCEMENTS

IMPLEMENTATION

Implementation means the process of converting a new or a revised system design into an operational one. It is the most crucial stage in achieving a new successful system and in giving confidence on the new system for the users that it will work efficiently and effectively. In this phase, we can build the components either from scratch or by composition. Given the architecture document from the design phase and requirement document from the analysis phase, we can build exactly what has been requested.

This phase deals with issues of quality, performance, baselines, libraries and debugging. The end deliverable is the product itself. There are three types of implementation:

1. Implementation of a computer system to replace a manual system
2. Implementation of new computer system to replace an existing one.
3. Implementation of a modified application to replace an existing one, using the same computer.

Implementation of "Schema SQL" comes under Third category. At the end of the specific period, the system performance and the reliability are tested. Implementation is the key stage in achieving a successful new system because it involves

FUTURE ENHANCEMENTS

It is necessary to keep up with changing user needs and the operational environment. Normally software fails because of improper cumulative maintenance, wear and tear. The system can be handled separately with out affecting other parts of the system. Thus, future enhancements are very easy in this system. Since the system is developed using modularized design, it can be upgraded without much modification.

In future, we will consider exploiting knowledge in the knowledge bases for optimizing query optimization in a network database environment. Extending knowledge bases to automate the process of managing semantic knowledge in network based systems.

CHAPTER 8

CONCLUSION

We introduced "Schema SQL", a principled extension of SQL for relational multi-database systems. In Schema SQL data and meta-data, that is, database instance and its schema, are treated uniformly, thus making it possible to query both the contents and the structure of a database. In a multi-database environment, Schema SQL provides the means for handling schematic (structural) heterogeneity, that is, similar information represented in different structures.

View definition in Schema SQL makes it possible to define restructuring views, namely, views that can change the structure of input data in a manner that exploits the data/meta-data interplay, while preserving the information content. Data (i.e., attribute values) in one representation can play the role of meta-data objects (database name, relation name, and attribute name) in a restructured view. SCHEMA SQL also provides novel aggregation capabilities. In addition to the usual SQL column aggregations, in Schema SQL it is possible to aggregate on a set of columns, individually or collectively, whether the columns are in one relation or in several. This can be done using a single query.

The latter case is basically aggregation is determined dynamically, at execution time, and is dependent on the database instance. Horizontal (i.e., row) aggregation is also possible in Schema SQL. After extracting data from all the heterogeneous databases and storing this data in centralized server, OLAP reports can be generated

APPENDIX 1

SAMPLE CODE

Query Builder form Load procedure

```
Private Sub fclsquery_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    Try
      'status bar values
      Dim datetime As New DateTime
      SBarPanel3.Text = datetime.Now
      SBarPanel2.Text = ""
      Lbviews.Items.Clear()
      Lbviews.Enabled = True
      lbfields.Items.Clear()
      lbfields.Enabled = True
      lbtables.Items.Clear()
      lbtables.Enabled = True
      Cbgroup.Items.Clear()
      cborder.Items.Clear()
      Cbgroup.Text = "none"
      cborder.Text = "none"
      txtbuilder.Text = ""
      rba.Checked = False
      rbd.Checked = False
      cboperator.SelectedItem = "none"
      txtvalue.Text = "value"

      If dbms.ckbaccess.Checked = True And dbms.ckbsql.Checked = True Then  '
      lbdbengine.Items.Clear()
        lbdbengine.Items.Add("A::Access")
        lbdbengine.Items.Add("S::SQL Server")
        butaccess.Enabled = True
        Butsqlserver.Enabled = True
        butmixed.Enabled = True
        o = New ADODB.Connection
        rs = New ADODB.Recordset
        o.ConnectionString = db.Labaccess.Text
        o.Open()
        Call Show_tables(o, 1)
        o1 = New ADODB.Connection
        rs1 = New ADODB.Recordset
```

```
     o1.ConnectionString = db.Labsqlserver.Text
     o1.Open()

     Call Show_tables(o1, 2)
  ElseIf dbms.ckbaccess.Checked = True And dbms.ckboracle.Checked = True then
     lbdbengine.Items.Clear()
     lbdbengine.Items.Add("A::Access")
     lbdbengine.Items.Add("O::Oracle")
          ElseIf dbms.ckbsql.Checked = True And            dbms.ckboracle.Checked
  = True Then
     lbdbengine.Items.Clear()
     lbdbengine.Items.Add("S::SQL Server")
     lbdbengine.Items.Add("O::Oracle")
   ElseIf dbms.ckbaccess.Checked = True Then
     lbdbengine.Items.Clear()
     lbdbengine.Items.Add("A::Access")
     butaccess.Enabled = True
     o = New ADODB.Connection
     rs = New ADODB.Recordset
     o.ConnectionString = db.Labaccess.Text
     o.Open()
     Call Show_tables(o, 1)
          ElseIf dbms.ckbsql.Checked = True Then
          lbdbengine.Items.Clear()
     lbdbengine.Items.Add("S::SQL Server")
     Butsqlserver.Enabled = True
     o1 = New ADODB.Connection
     rs1 = New ADODB.Recordset
     o1.ConnectionString = db.Labsqlserver.Text
     o1.Open(db.str)
     Call Show_tables(o1, 2)

   ElseIf dbms.ckboracle.Checked = True Then
     lbdbengine.Items.Clear()
     lbdbengine.Items.Add("O::Oracle")
   End If

   Exit Sub
Catch e1 As Exception
   MsgBox(Err.Description, vbCritical + vbOKOnly, "form load")
   a1 = True
Catch e2 As DataException
   MsgBox(Err.Description, vbCritical + vbOKOnly, "form load")
   a1 = True
End Try
If a1 = True Then
```

```
        MsgBox("Exception occured, cant open Query builder")
        query.Close()
      End If
    End Sub
```

Show tables procedure

```
Public Sub Show_tables(ByVal a As ADODB.Connection, ByVal flag As Integer)
        On Error GoTo handle_Renamed

        If Lbviews.Items.Contains("NO VIEWS ") = True Then
          Lbviews.Items.Clear()
          Lbviews.Enabled = True
        ElseIf lbtables.Items.Contains("NO TABLES ") = True Then
          lbtables.Items.Clear()
          lbtables.Enabled = True
        End If

        If flag = 1 Then
          rs = a.OpenSchema((ADODB.SchemaEnum.adSchemaTables), New Object()
 {Nothing, Nothing, Nothing, "TABLE"})
          Do Until rs.EOF
            lbtables.Items.Add("A::" & rs.Fields("TABLE_NAME").Value)
            rs.MoveNext()
          Loop
          rs.Close()
          'views
          rs = a.OpenSchema((ADODB.SchemaEnum.adSchemaTables), New Object()
 {Nothing, Nothing, Nothing, "VIEW"})
          Do Until rs.EOF
            Lbviews.Items.Add("A::" & rs.Fields("TABLE_NAME").Value)
            rs.MoveNext()
          Loop
          rs.Close()
        ElseIf flag = 2 Then ' for sql server connection
          'tables

          rs = a.OpenSchema(ADODB.SchemaEnum.adSchemaTables, New Object() {Nothing,
 Nothing, Nothing, "TABLE"})
          Do Until rs.EOF
            lbtables.Items.Add("S::" & rs.Fields("TABLE_NAME").Value)
            rs.MoveNext()
          Loop
          rs.Close()
          'views
        rs = a.OpenSchema(ADODB.SchemaEnum.adSchemaTables, New Object() {Nothing,
Nothing, Nothing, "VIEW"})
        Do Until rs.EOF
          Lbviews.Items.Add("S::" & rs.Fields("TABLE_NAME").Value)
          rs.MoveNext()
        Loop
        rs.Close()
      End If

      If Lbviews.Items.Count = 0 Then  '  no views
        Lbviews.Items.Add("NO VIEWS ")
        Lbviews.Items.Add("in this database")
        Lbviews.Enabled = False
      ElseIf lbtables.Items.Count = 0 Then   ' no tables
        lbtables.Items.Add("NO TABLES")
        lbtables.Items.Add("in this database")
        lbtables.Enabled = False
      End If
      'to delete the temperory table
      If lbtables.Items.Contains("S::tempschemasql") Then
        'MsgBox("entering")
        Dim r1  As New ADODB.Recordset
        Dim s As String = "drop table tempschemasql"
        r1.Open(s, o1, ADODB.CursorTypeEnum.adOpenDynamic,
ADODB.LockTypeEnum.adLockOptimistic)
        lbtables.Items.Remove("S::tempschemasql")
      End If

      Exit Sub
handle_Renamed:
      MsgBox(Err.Description, vbCritical + vbOKOnly, "Show tables")
   End Sub
```

Show Fields Procedure

```
Public Sub show_fields(ByVal sql As String, ByVal a As Integer)
      Dim n As Short
      Dim e As ADODB.Connection
      If a = 1 Then
         e = o
      ElseIf a = 2 Then
         e = o1
      End If
      rs.Open(sql, e, ADODB.CursorTypeEnum.adOpenDynamic,
ADODB.LockTypeEnum.adLockOptimistic)
      'listbox------ fields
      lbfields.Items.Clear()

      For n = 0 To rs.Fields.Count - 1
         lbfields.Items.Add(rs.Fields(n).Name)
      Next
      Cbgroup.Items.Clear()
      Cbgroup.Items.Add("none")
      For n = 0 To rs.Fields.Count - 1
         Cbgroup.Items.Add(rs.Fields(n).Name)
      Next
      cborder.Items.Clear()
      cborder.Items.Add("none")
      For n = 0 To rs.Fields.Count - 1
         cborder.Items.Add(rs.Fields(n).Name)
      Next
      rs.Close()
      If lbfields.Items.Count = 0 Then
         lbfields.Items.Add("NO FIELDS")
         lbfields.Items.Add("in this table")
         lbfields.Enabled = False
      End If
   End Sub
```

Access Connection Procedure

```
Public Sub access_only(ByVal st As String)
    Dim str_renamed As String = ""
    Dim tem As String = ""
    Dim tem1 As String = ""
    Dim t As String = ""
    Dim n, a As Integer
    On Error GoTo handle_error
    '-------------------for status bar display---------------------
    Dim i As Integer
    Dim g, g1 As String
    i = Trim(st).IndexOf(" ")
    g = Trim(st).Substring(0, i)
    g1 = Trim(st).Substring(i)
    If UCase(g) = "CREATE" Then
       Dim j As Integer
       Dim f As String
       j = Trim(g1).IndexOf(" ")
       f = g1.Substring(0, j + 1)
       g = g + f
    ElseIf UCase(g) = "DROP" Then
       Dim j As Integer
       Dim f As String
       j = Trim(g1).IndexOf(" ")
       f = g1.Substring(0, j + 1)
       g = g + f
    ElseIf UCase(g) = "ALTER" Then
       Dim j As Integer
       Dim f As String
       j = Trim(g1).IndexOf(" ")
       f = g1.Substring(0, j + 1)
       g = g + f
    End If

      While InStr(st, "A::")
       a = 1
       n = InStr(st, "A::")
       tem1 = tem1 + st.Substring(0, (n - 1))
       tem = tem + st.Substring((n + 2))
       st = tem1 + tem
       str_renamed = tem1 + tem
       tem1 = ""
```

```
        tem = ""
      End While
      ' MsgBox("final " + str_renamed)
      If a = 1 Then
        t = str_renamed
      Else
        t = st
      End If
      If t.Length >= 6 Then          'InStr(t, "select") Then   'InStr(t, UCase("select")) O
        If UCase(t.Substring(0, 6)) = "SELECT" Then
          Call show_rows(t, 1)
          GoTo e
        End If
      End If
      rs = New ADODB.Recordset
      rs.Open(t, o, ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockOptimistic)

      status_display(g)          'calling status display functon
E:
      Exit Sub


handle_error:
      m.mes(Err.Description)

    End Sub
```

SQLServer Connection Procedure

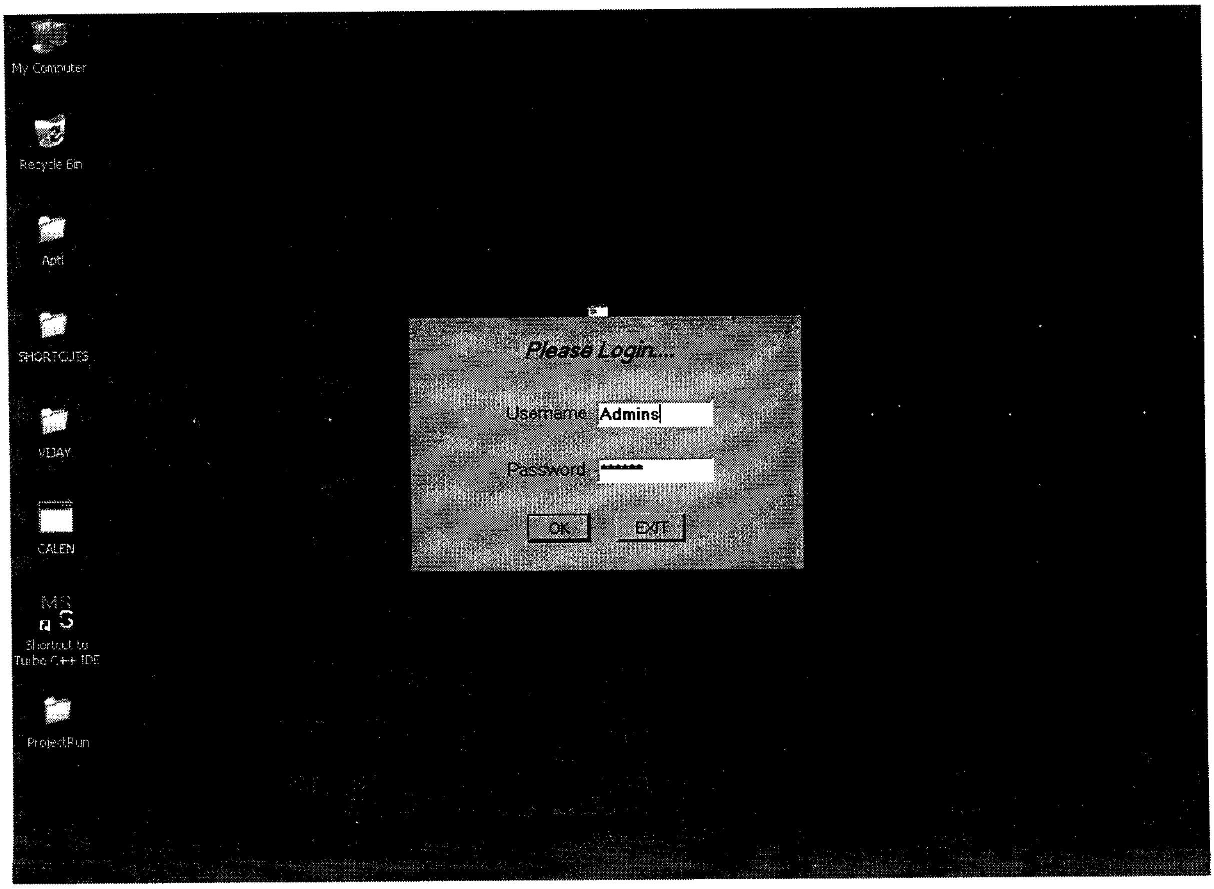
```
Public Sub SQlServer_only(ByVal st As String)
    Dim str_renamed As String = ""
    Dim tem As String = ""
    Dim tem1 As String = ""
    Dim t As String = ""
    Dim n, a As Integer
    On Error GoTo handle_error
    'MsgBox(st)
    While InStr(st, "S::")
        a = 1
        n = InStr(st, "S::")
        tem1 = tem1 + st.Substring(0, (n - 1))
        tem = tem + st.Substring((n + 2))
        st = tem1 + tem
        str_renamed = tem1 + tem
        tem1 = ""
        tem = ""
    End While
    'MsgBox("final " + str_renamed)
    If a = 1 Then
        t = str_renamed
    Else
        t = st
    End If
    If t.Length >= 6 Then       'InStr(t, "select") Then   'InStr(t, UCase("select")) O
        If UCase(t.Substring(0, 6)) = "SELECT" Then
            Call show_rows(t, 2)
            GoTo e
        End If
    End If
    rs = New ADODB.Recordset
    rs.Open(t, o1, ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockOptimistic)
E:
    Exit Sub


handle_error:
    m.mes(Err.Description)

  End Sub
```

Show rows Procedure

```
Public Sub show_rows(ByVal sql As String, ByVal a As Integer)
    On Error GoTo handle_error
    Dim t As Integer  ' status bar display  -   no of rows

    If a = 1 Then
       dt8 = New DataTable
       c8.ConnectionString = db.Labaccess.Text
       c8.Open()
       d8 = New OleDb.OleDbDataAdapter(sql, c8)
       d8.Fill(dt8)
       dgquery.DataSource = dt8
       dgquery.ColumnHeadersVisible = True
       t = dt8.Rows.Count
    ElseIf a = 2 Then
       dt8 = New DataTable
       c8.ConnectionString = db.Labsqlserver.Text
       c8.Open()
       d8 = New OleDb.OleDbDataAdapter(sql, c8)
       d8.Fill(dt8)
       dgquery.DataSource = dt8
       dgquery.ColumnHeadersVisible = True
       t = dt8.Rows.Count
    End If
    updatestring = sql
    If t <> 0 Then
       SBarPanel2.Text = "Row(s) affected = " + t.ToString
    Else
       SBarPanel2.Text = "No Row(s) affected "
    End If

    Exit Sub

handle_error:
    m.mes(Err.Description)

  End Sub
```

APPENDIX 2

SCREEN SHOTS



Username Admins

Password ******

Figure A 2.1 Login Form

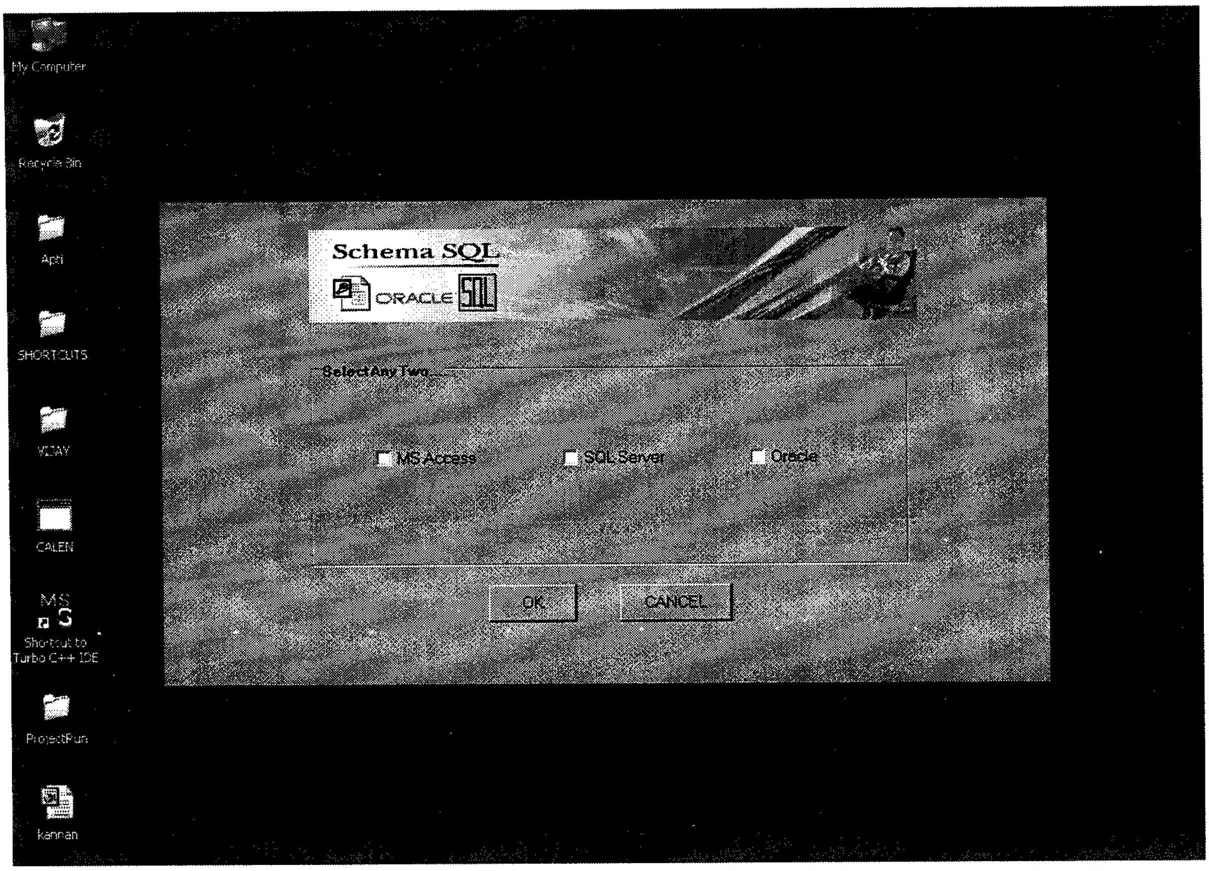
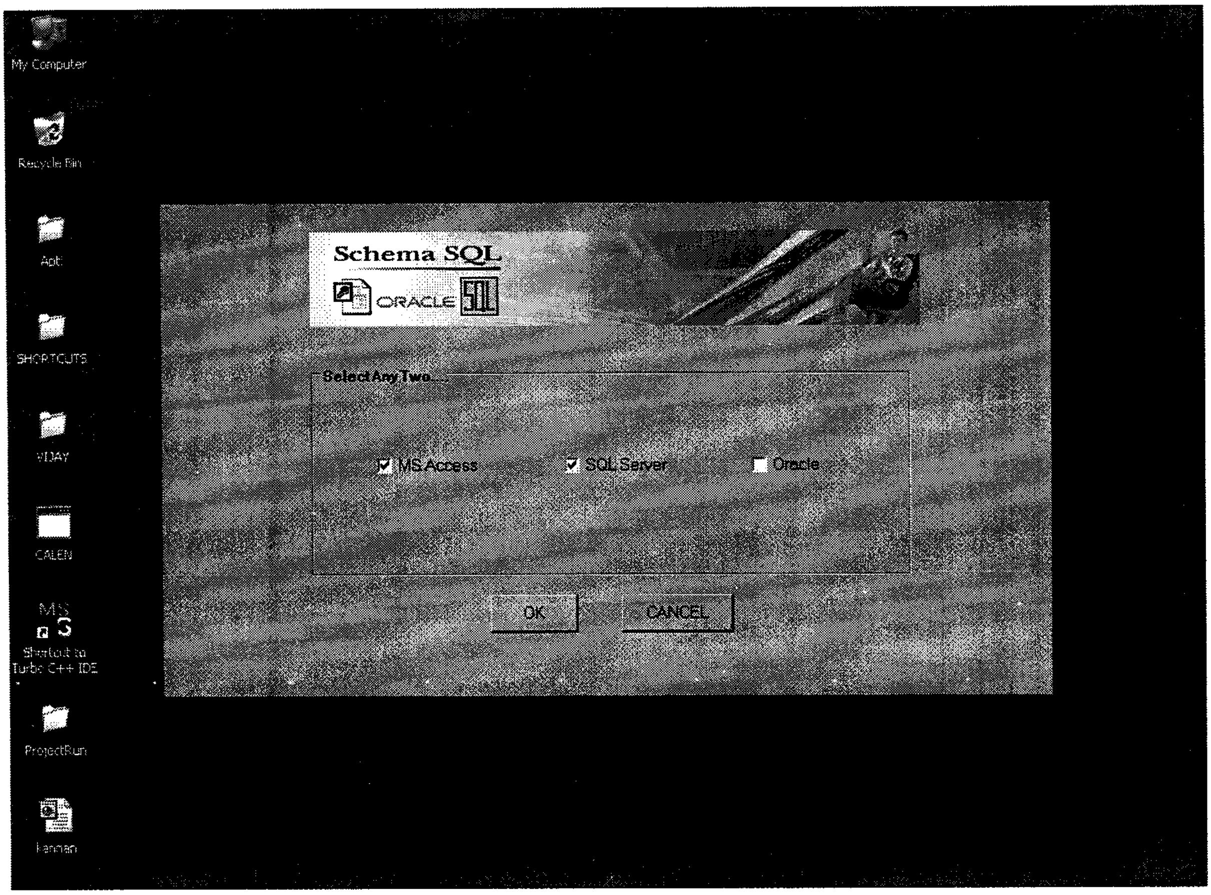


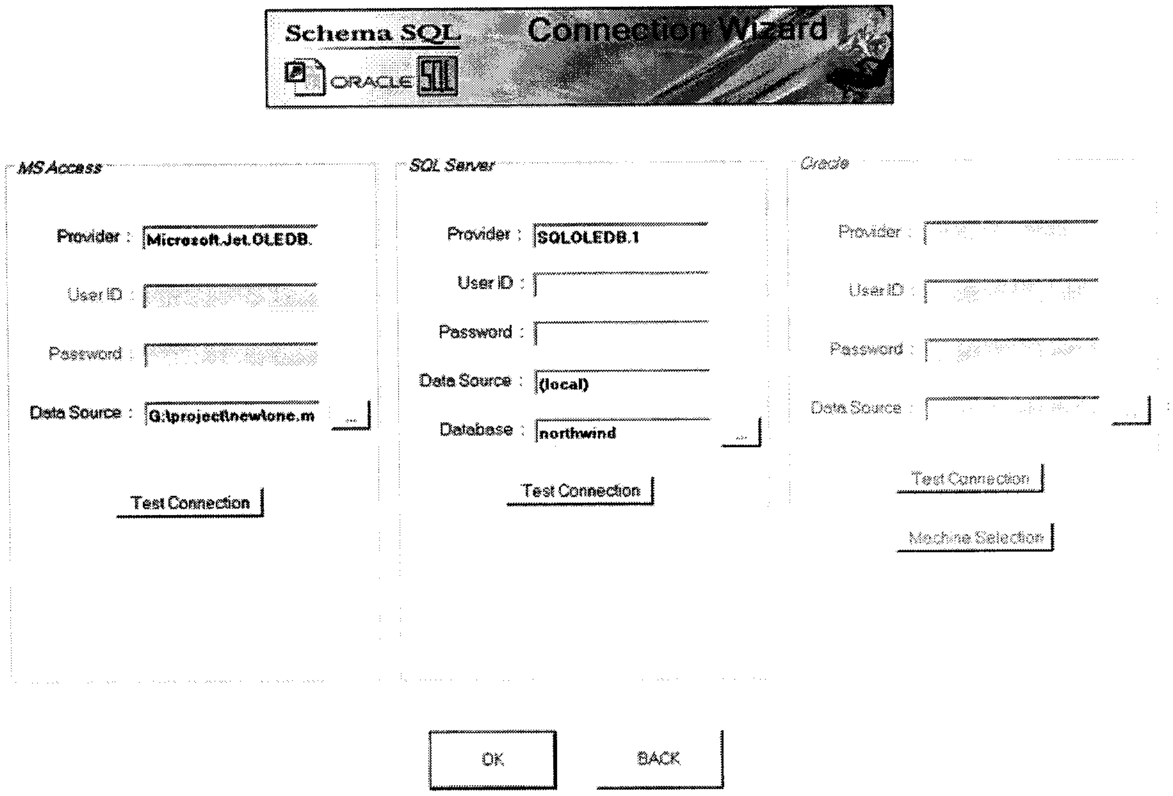
Figure A 2.2 Home Screen



Select Any Two...

[✓ MS Access] [✓ SQL Server] Oracle

Figure A 2.3 Database Selection



MS Access

Provider : Microsoft.Jet.OLEDB.

User ID :

Password :

Data Source : G:\project\new\one.m

SQL Server

Provider : SQLOLEDB.1

User ID :

Password :

Data Source : (local)

Database : northwind

Oracle

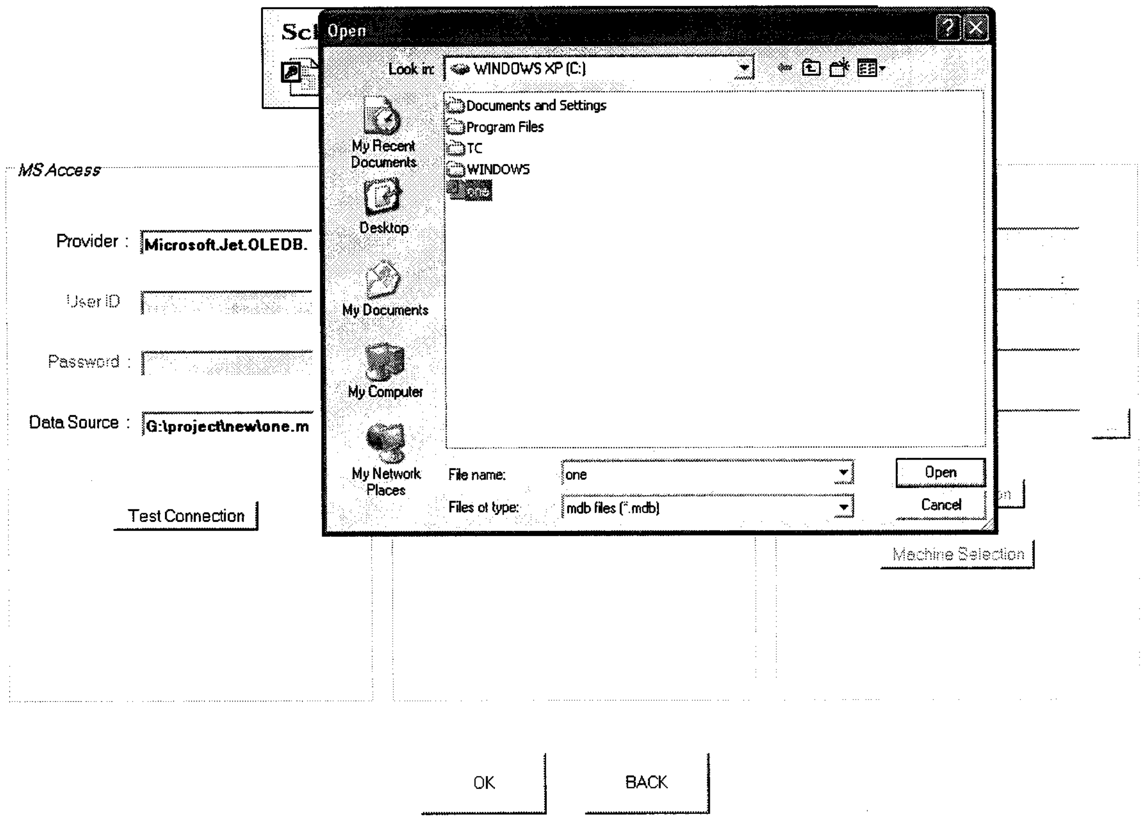
Provider :

User ID :

Password :

Data Source :

Figure A 2.4 Connection Wizard



Provider : Microsoft.Jet.OLEDB.

User ID :

Password :

Data Source : G:\project\new\one.m

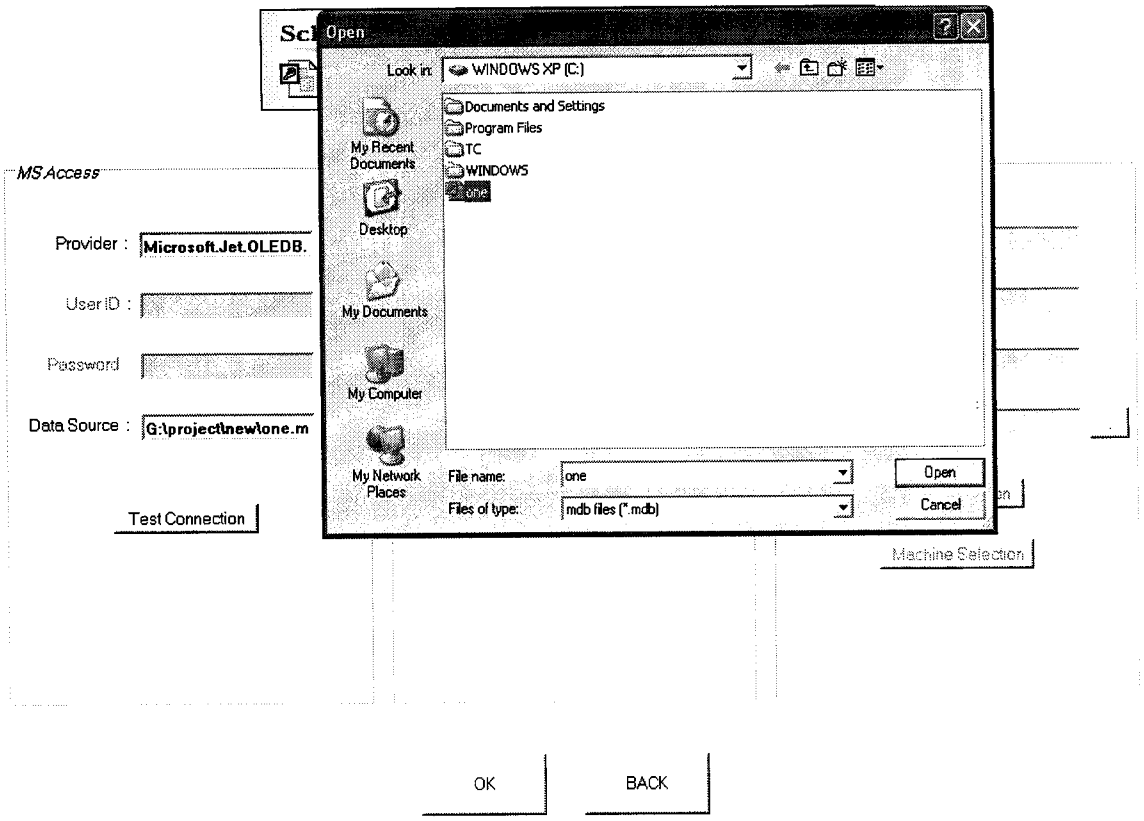
Look in: WINDOWS XP (C:)

Documents and Settings
Program Files
TC
WINDOWS
one

File name: one

Files of type: mdb files (*.mdb)

Figure A 2.5 Connection Wizard - Access database Selection



Provider : Microsoft.Jet.OLEDB.

User ID :

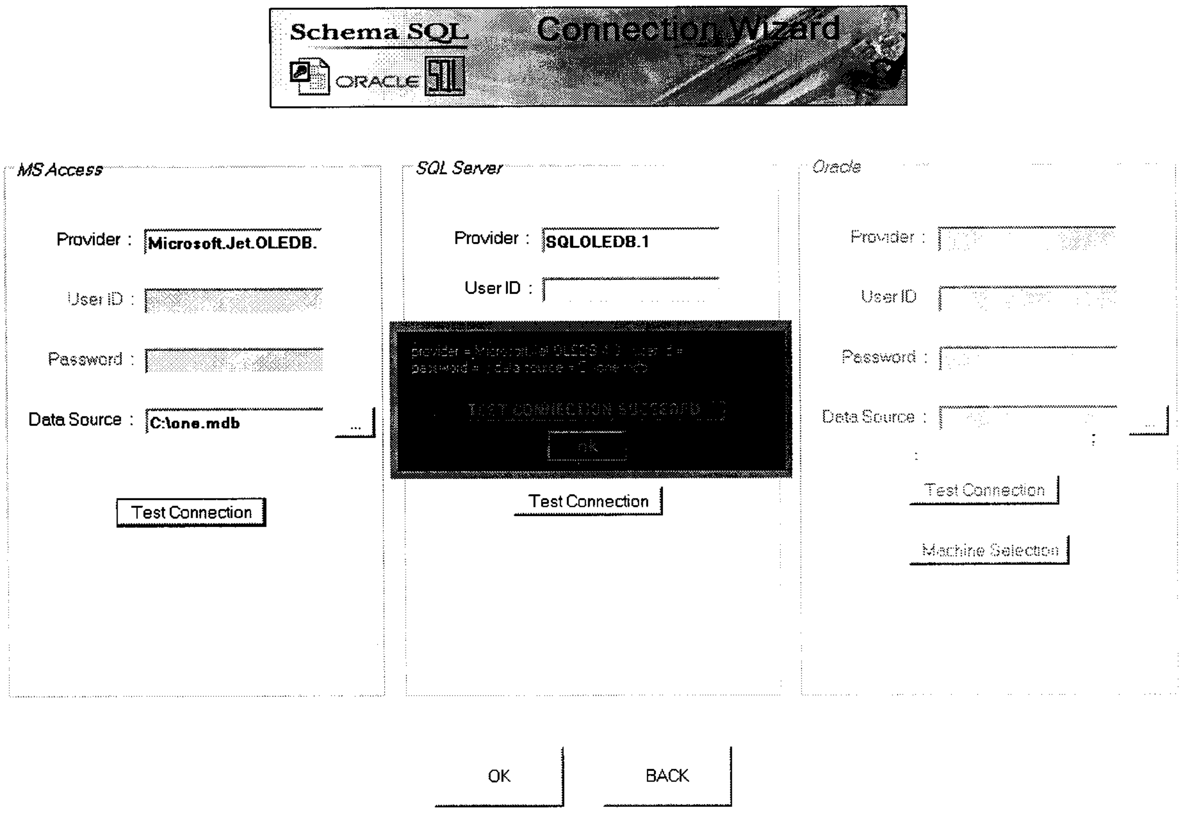
Password

Data Source : G:\project\new\one.m

File name: one

Files of type: mdb files (*.mdb)

Figure A 2.6 Connection Wizard - Database Selection


Schema SQL Connection Wizard

MS Access

Provider : Microsoft.Jet.OLEDB.

User ID :

Password :

Data Source : C:\one.mdb

SQL Server

Provider : SQLOLEDB.1

User ID :

provider = Microsoft.Jet.OLEDB.4.0 ; user id =
password = ; data source = C:\one.mdb

TEST CONNECTION SUCCEEDED !!

Oracle

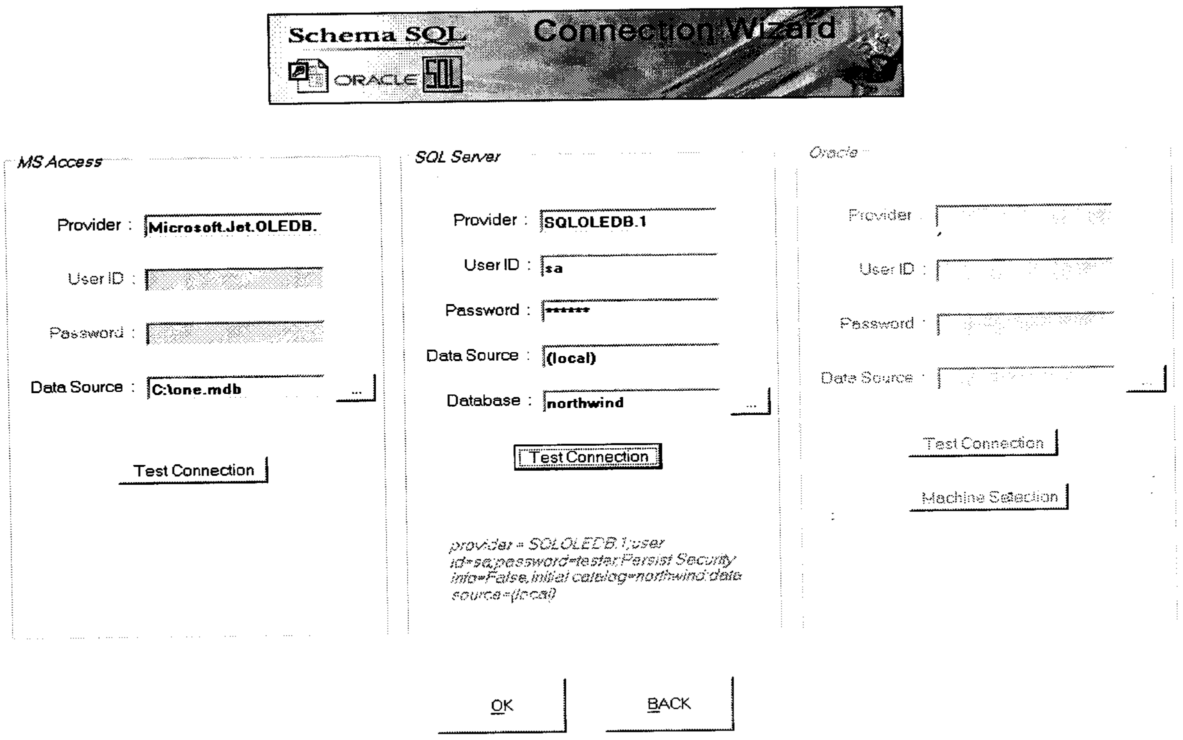
Provider :

User ID

Password :

Data Source :

Figure A 2.7 Connection Wizard - Access database Test Connection



MS Access

Provider : Microsoft.Jet.OLEDB.

User ID :

Password :

Data Source : C:\one.mdb

SQL Server

Provider : SQLOLEDB.1

User ID : sa

Password : ******

Data Source : (local)

Database : northwind

provider = SQLOLEDB.1;user id=sa;password=tester;Persist Security info=False;initial catalog=northwind;data source=(local)

Oracle

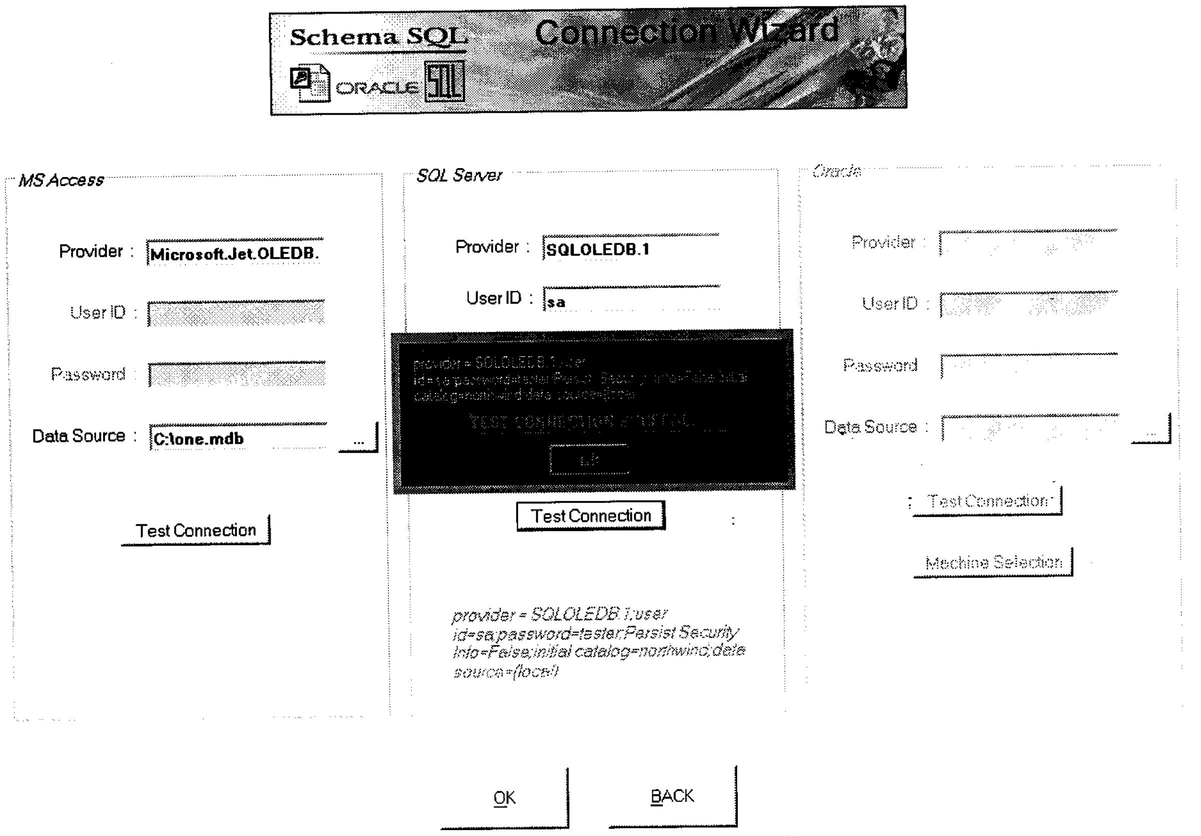
Provider :

User ID :

Password :

Data Source :

Figure A 2.8 Connection Wizard - SQL database Selection


Schema SQL Connection Wizard

MS Access

Provider : Microsoft.Jet.OLEDB.

User ID :

Password :

Data Source : C:\one.mdb

SQL Server

Provider : SQLOLEDB.1

User ID : sa

provider = SQLOLEDB.1;user id=sa;password=tester;Persist Security Info=False;initial catalog=northwind;data source=(local)

TEST CONNECTION [illegible]

provider = SQLOLEDB.1;user id=sa;password=tester;Persist Security Info=False;initial catalog=northwind;data source=(local)

Oracle

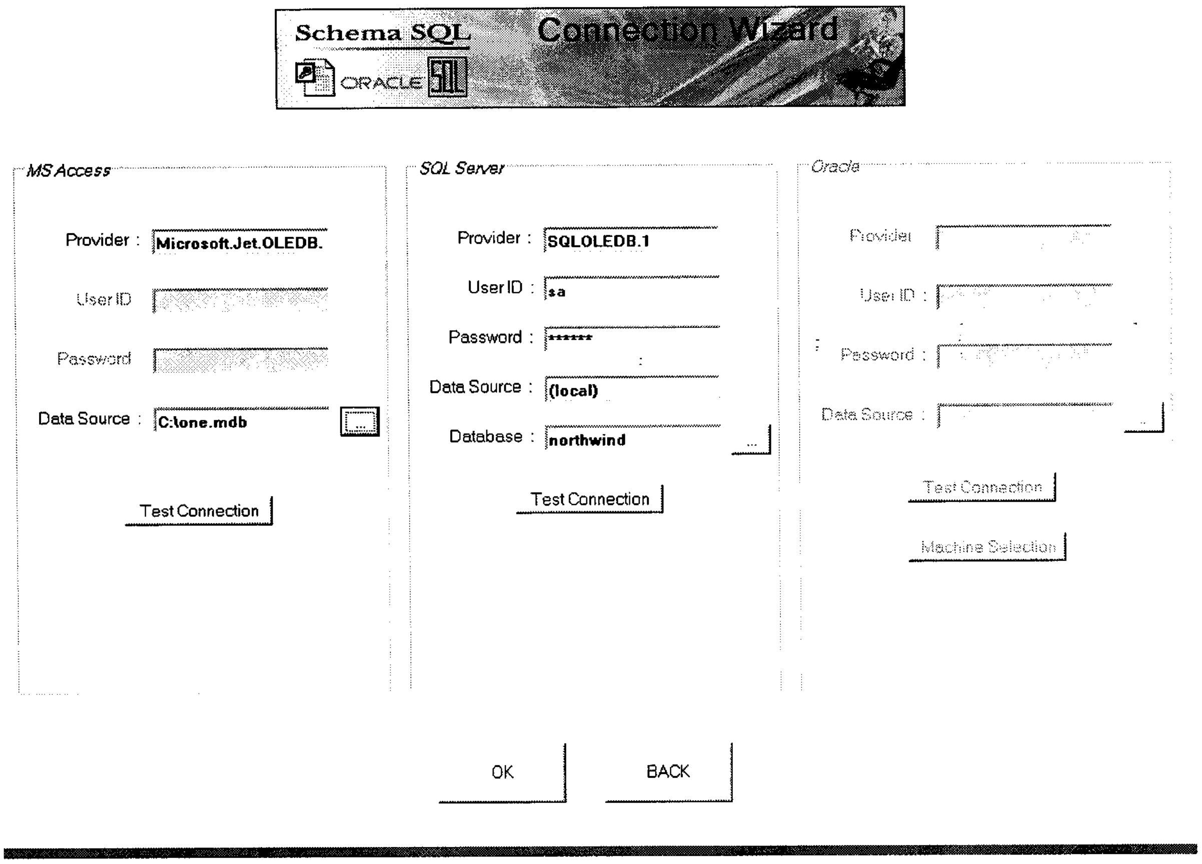
Provider :

User ID :

Password :

Data Source :

Figure A 2.9 Connection Wizard - SQL database Test Connection



Provider : Microsoft.Jet.OLEDB.

Data Source : C:\one.mdb

Provider : SQLOLEDB.1

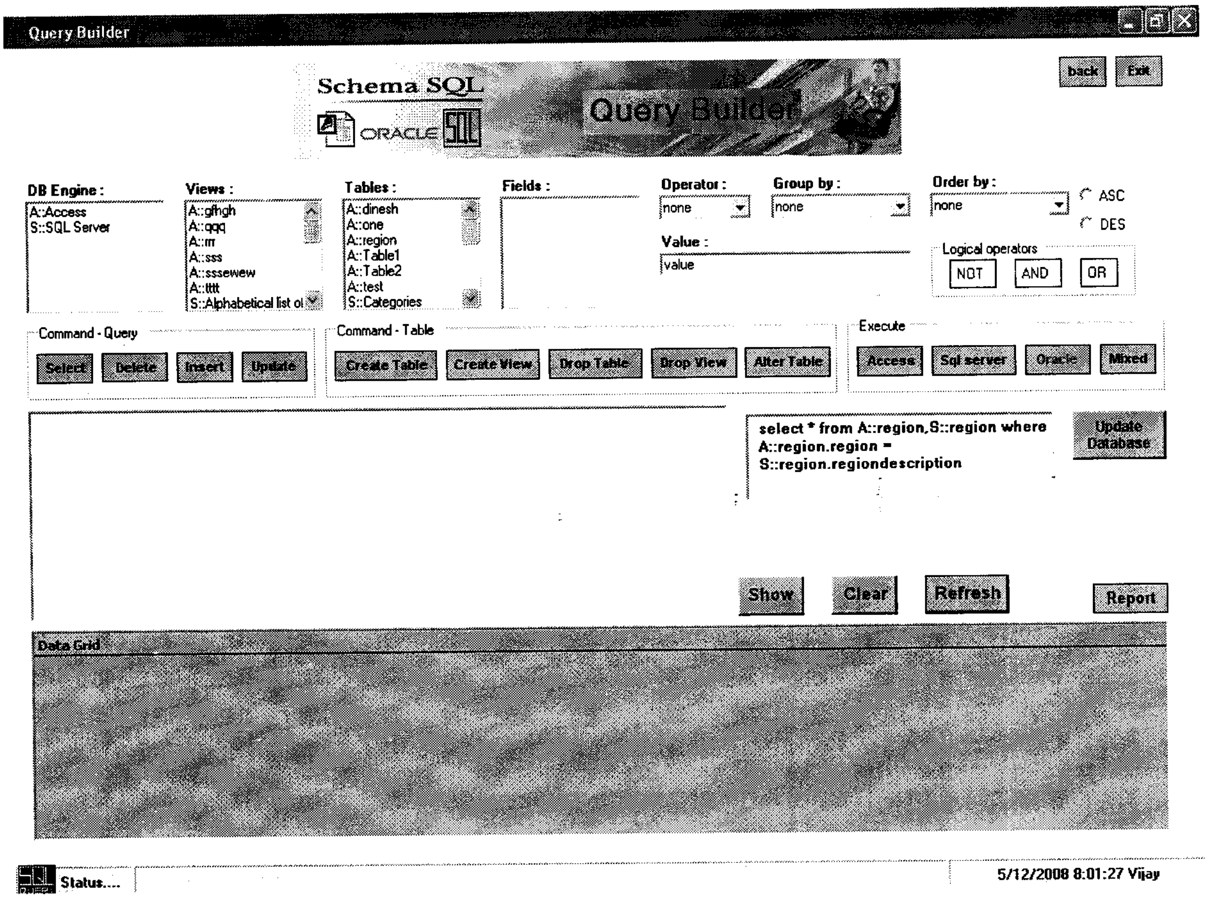
User ID : sa

Password : ******

Data Source : (local)

Database : northwind

Figure A 2.10 Connection Wizard - Accepting



DB Engine: A::Access, S::SQL Server

Views: A::gfhgh, A::qqq, A::rrr, A::sss, A::sssewew, A::tttt, S::Alphabetical list of

Tables: A::dinesh, A::one, A::region, A::Table1, A::Table2, A::test, S::Categories

Operator: none

Group by: none

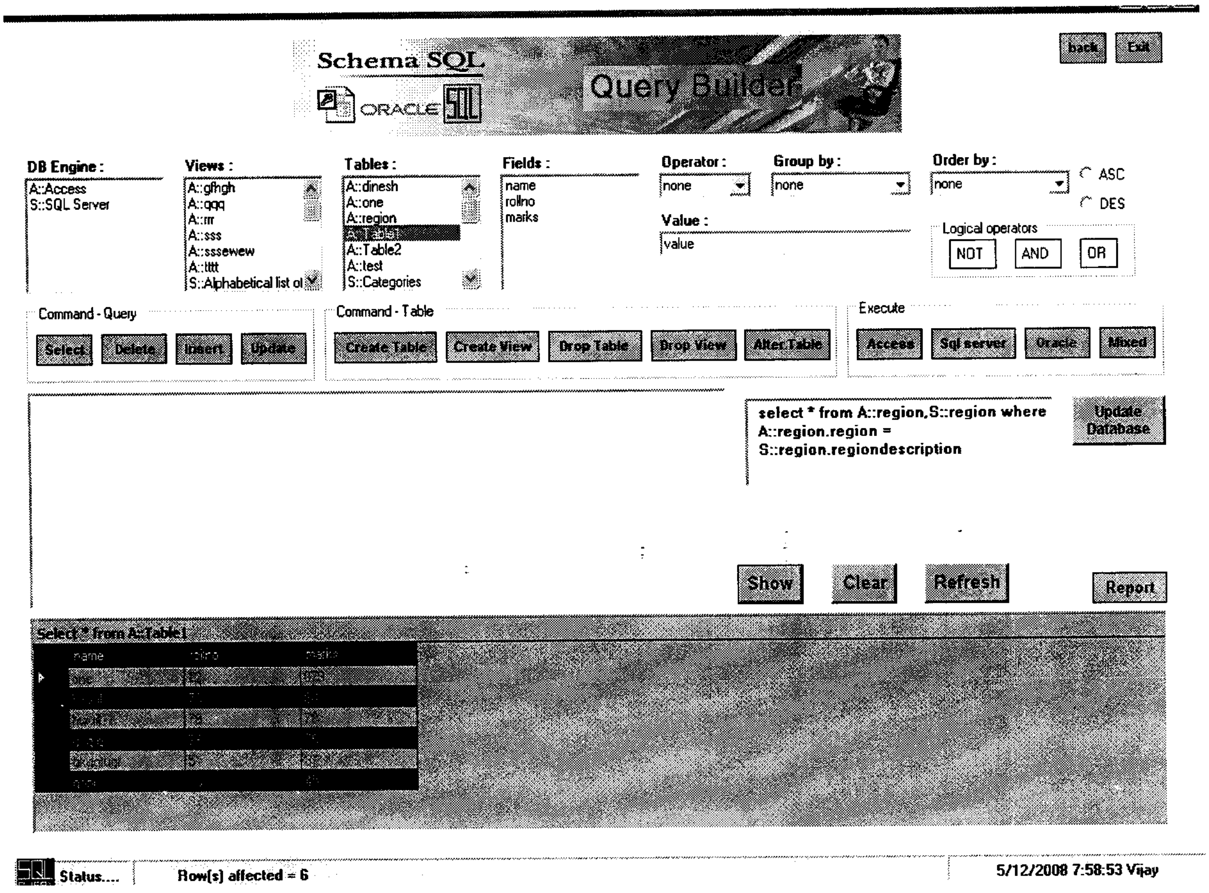
Order by: none

Value: value

```
select * from A::region,S::region where
A::region.region =
S::region.regiondescription
```

5/12/2008 8:01:27 Vijay

Figure A 2.11 Query Builder Wizard



DB Engine :
A::Access
S::SQL Server

Views :
A::gfhgh
A::qqq
A::rrr
A::sss
A::sssewew
A::tttt
S::Alphabetical list of

Tables :
A::dinesh
A::one
A::region
A::Table1
A::Table2
A::test
S::Categories

Fields :
name
rollno
marks

Operator : none

Group by : none

Order by : none

ASC
DES

Value : value

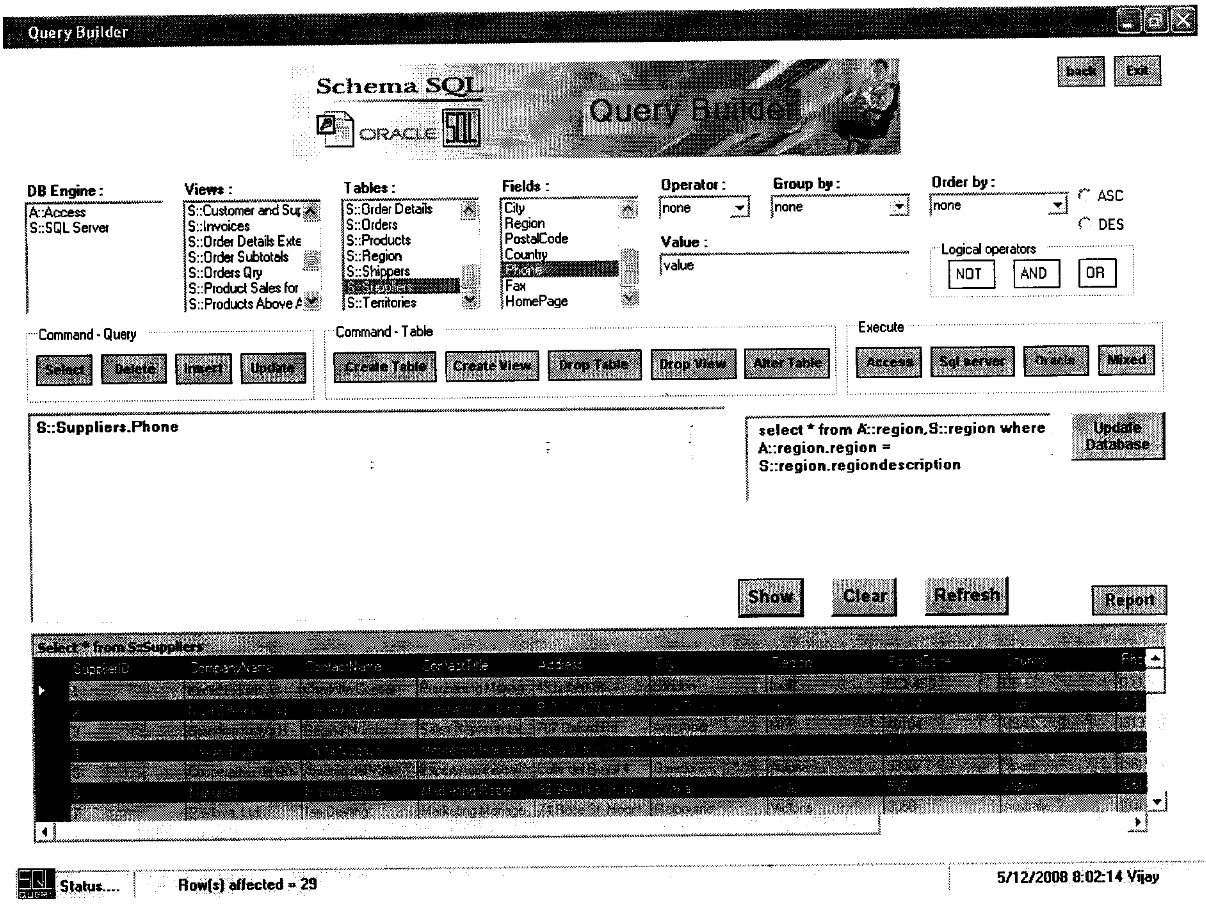
```
select * from A::region,S::region where
A::region.region =
S::region.regiondescription
```

Select * from A::Table1

name	rollno	marks
one	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	78	78
[illegible]	[illegible]	[illegible]
[illegible]	5	[illegible]
[illegible]	[illegible]	[illegible]

Status.... Row(s) affected = 6 5/12/2008 7:58:53 Vijay

Figure A 2.12 Query Builder Wizard - Display Table Values



DB Engine :
A::Access
S::SQL Server

Views :
S::Customer and Su
S::Invoices
S::Order Details Exte
S::Order Subtotals
S::Orders Qry
S::Product Sales for
S::Products Above A

Tables :
S::Order Details
S::Orders
S::Products
S::Region
S::Shippers
S::Suppliers
S::Territories

Fields :
City
Region
PostalCode
Country
Phone
Fax
HomePage

Operator : none
Group by : none
Order by : none
Value : value
Logical operators

Command - Query
Command - Table
Execute

S::Suppliers.Phone

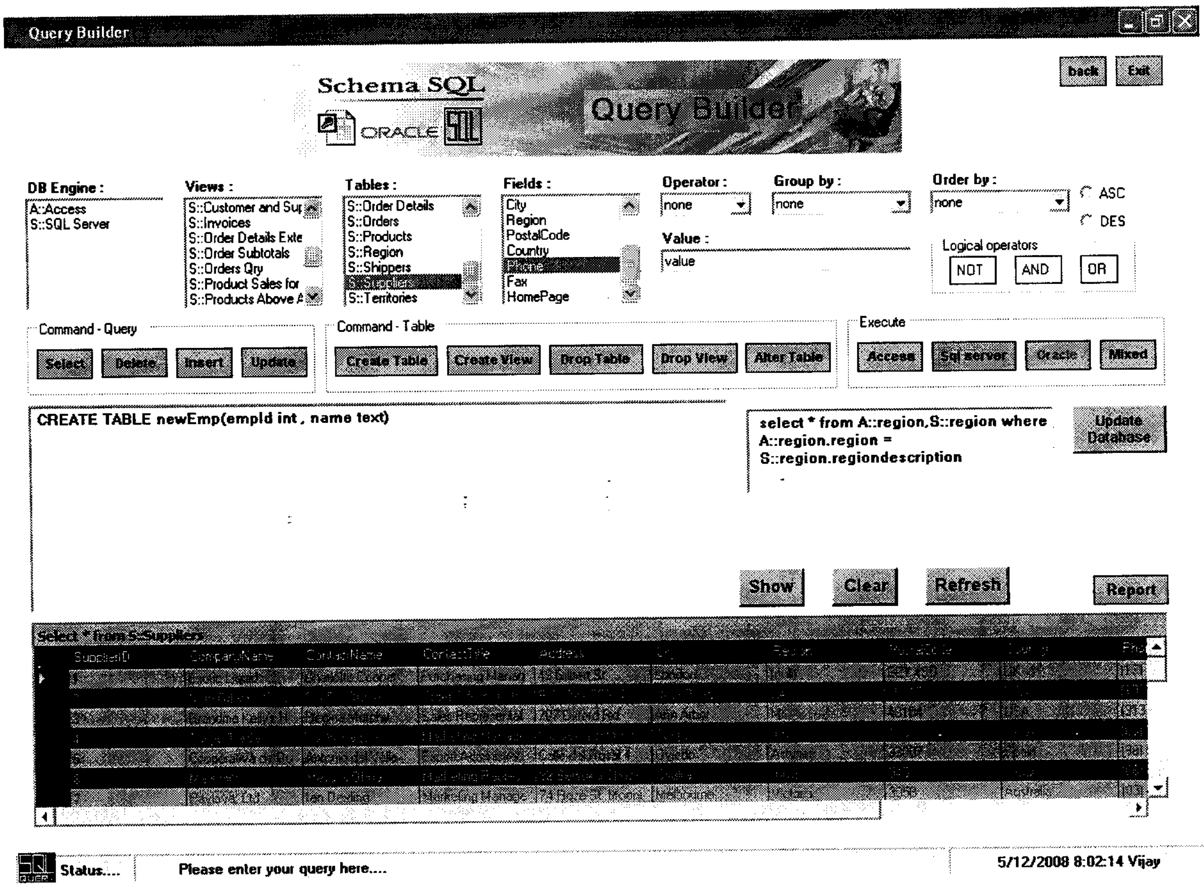
select * from A::region,S::region where A::region.region = S::region.regiondescription

Select * from S::Suppliers

	SupplierID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Pho
▸	1	Exotic Liquids	Charlotte Coope	Purchasing Manag	49 Gilbert St.	London	[illegible]	EC1 4SD	UK	(171
	2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	3	Grandma Kelly's H	Regina Murphy	Sales Representati	707 Oxford Rd.	Ann Arbor	MI	48104	USA	(313
	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	5	Cooperativa de Qu	Antonio del Valle	Export Administrat	Calle del Rosal 4	Oviedo	Asturias	33007	Spain	(98)
	6	Mayumi's	Mayumi Ohno	Marketing Repres	92 Setsuko Chuo	Osaka		545	Japan	(06)
	7	Pavlova, Ltd.	Ian Devling	Marketing Manage	74 Rose St. Moon	Melbourne	Victoria	3058	Australia	(03)

Status.... Row(s) affected = 29 5/12/2008 8:02:14 Vijay

Figure A 2.13 Query Builder Wizard - Display SQL Table Values



```
CREATE TABLE newEmp(empId int, name text)
```

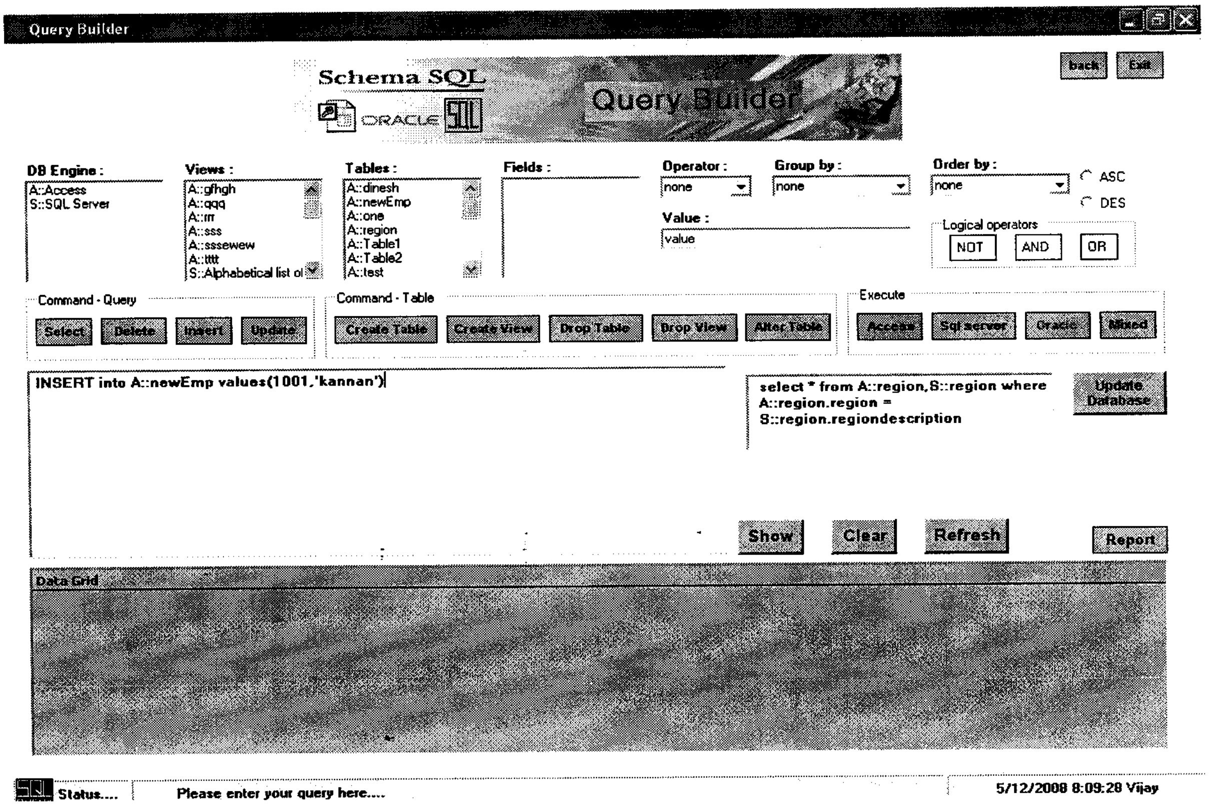
```
select * from A::region,S::region where
A::region.region =
S::region.regiondescription
```

Select * from S::Suppliers

SupplierID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Pho
1	Exotic Liquids	Charlotte Cooper	Purchasing Manag	49 Gilbert St.	London	[illegible]	EC1 4SD	UK	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
3	Grandma Kelly's H	Regina Murphy	Sales Representat	707 Oxford Rd.	Ann Arbor	MI	48104	USA	[illegible]
4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
5	Cooperativa de Q	Antonio del Valle	Export Administr	Calle del Rosal 4	Oviedo	Asturias	33007	Spain	[illegible]
6	[illegible]	[illegible]	Marketing Repres	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
7	Pavlova, Ltd.	Ian Devling	Marketing Manage	74 Rose St. Moonie	Melbourne	Victoria	3058	Australia	[illegible]

Status.... Please enter your query here.... 5/12/2008 8:02:14 Vijay

Figure A 2.14 Query Builder Wizard - Create Access Table



DB Engine: A::Access, S::SQL Server

Views: A::gfhgh, A::qqq, A::rrr, A::sss, A::sssewew, A::tttt, S::Alphabetical list of

Tables: A::dinesh, A::newEmp, A::one, A::region, A::Table1, A::Table2, A::test

Operator: none

Group by: none

Order by: none

Value: value

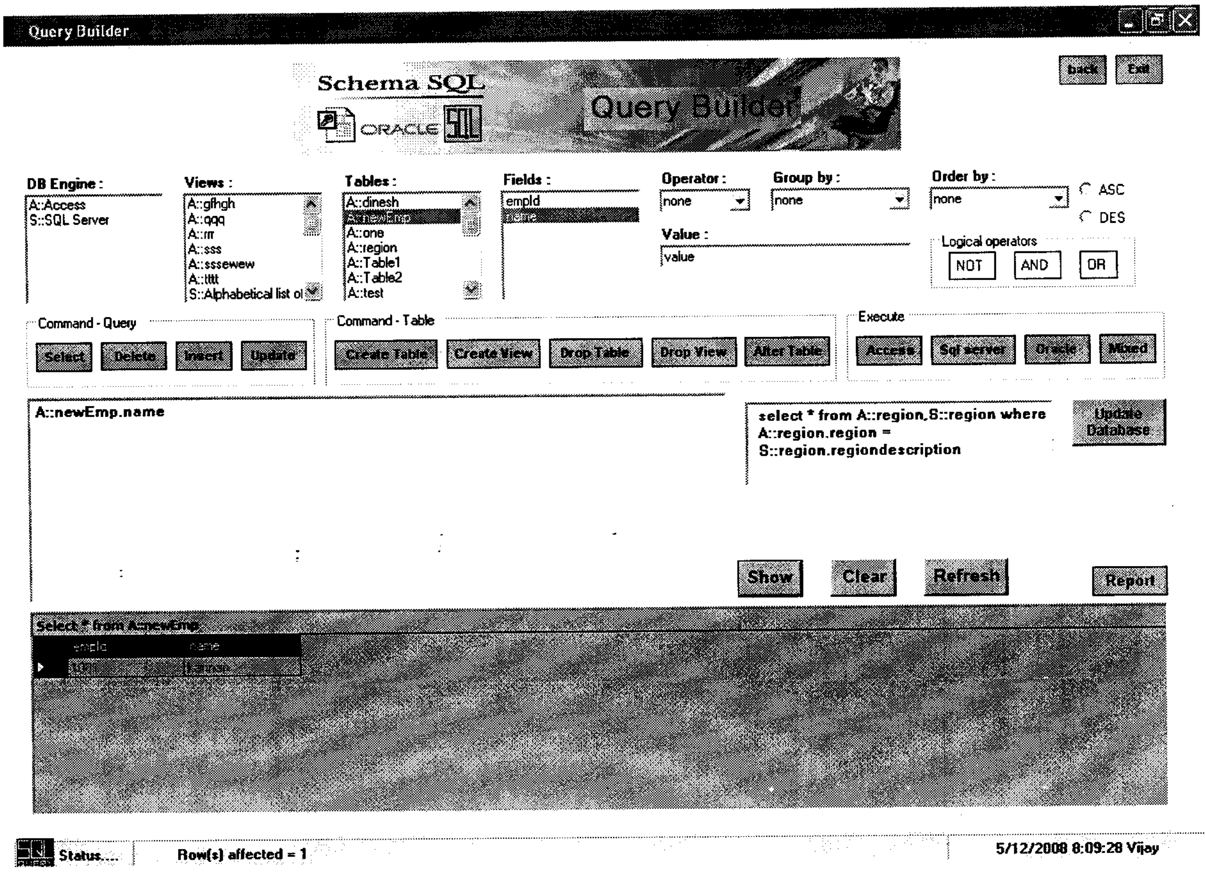
```
INSERT into A::newEmp values(1001,'kannan')
```

```
select * from A::region,S::region where
A::region.region =
S::region.regiondescription
```

Please enter your query here....

5/12/2008 8:09:28 Vijay

Figure A 2.15 Query Builder Wizard - Insert Access Table



DB Engine: A::Access, S::SQL Server

Views: A::gfhgh, A::qqq, A::rrr, A::sss, A::sssewew, A::tttt, S::Alphabetical list of

Tables: A::dinesh, A::newEmp, A::one, A::region, A::Table1, A::Table2, A::test

Fields: empId, name

Operator: none

Group by: none

Order by: none

Value: value

```
A::newEmp.name
```

```
select * from A::region,S::region where
A::region.region =
S::region.regiondescription
```

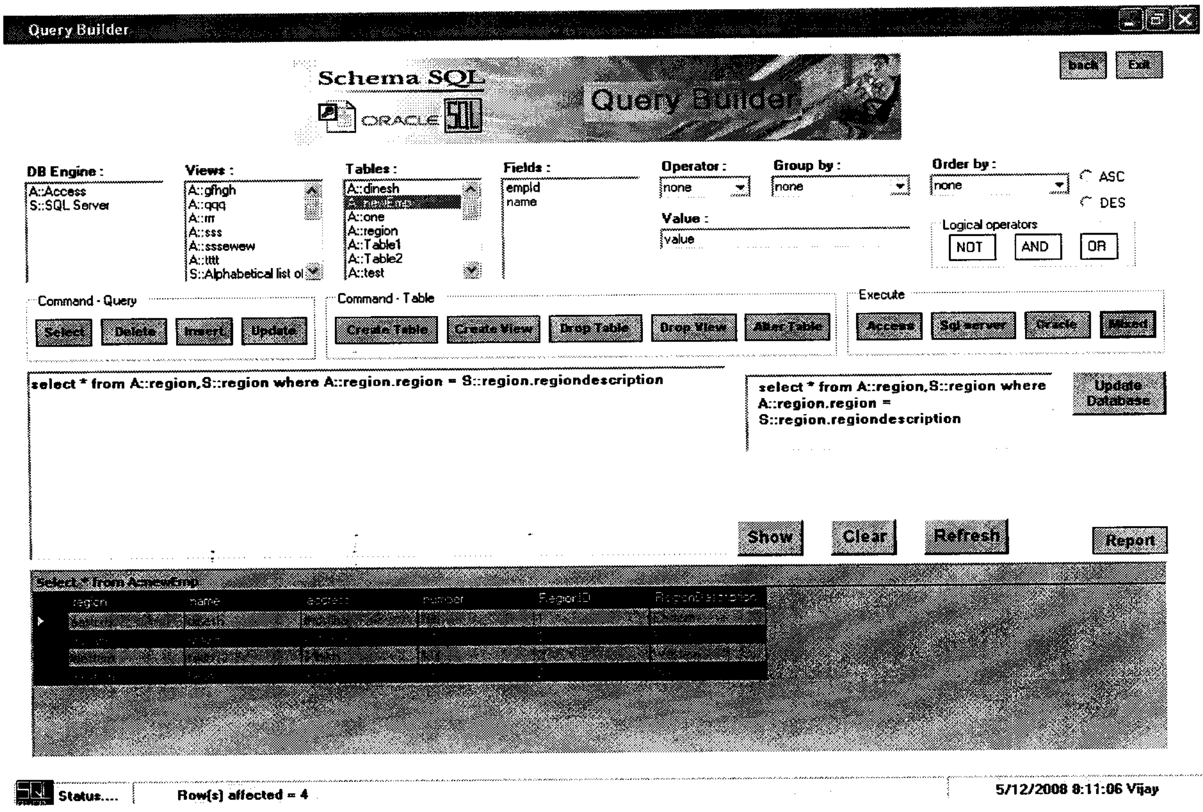
Select * from A::newEmp

	empId	name
▶	[illegible]	[illegible]

Row(s) affected = 1

5/12/2008 8:09:28 Vijay

Figure A 2.16 Query Builder Wizard - Selecting New Table Values



```
select * from A::region,S::region where A::region.region = S::region.regiondescription
```

```
select * from A::region,S::region where
A::region.region =
S::region.regiondescription
```

Select * from A:newEmp

region	name	address	number	RegionID	RegionDescription
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Status.... Row(s) affected = 4 5/12/2008 8:11:06 Vijay

Figure A 2.17 Query Builder Wizard - Selecting Values From Different database sources

REFERENCES

Books

- Rama Ramachandran, 'Professional VB.Net 2003', Wrox Publications, Second Edition, 2005.
- Hemlata, 'Sql Server 2000 References', Cyber-Tech Publications, First Edition, 2004.
- Herbert Schmidt, 'View On Sql Query Analyser', Tata McGraw-Hill publications, Fourth Edition, 2000.

Websites

- http://www.sourcecode.com/
- http://www.dotnetcoding.com/
- http://www.codeguru.com/
- http://www.codeproject.com/
- http://www.vbtraining.com/