FRONT END
FOR
QUERY TOOL

submitted by
Ajay K. Yelamanchili

in partial fulfillment of the requirements for the
Degree of MCSE

Advisor:
Dr. W. Homer Carlisle

Committee:
Dr. Kai Chang
Dr. James Cross

Auburn University, Auburn, AL.
February 23, 1996
Acknowledgments

I would like to thank my advisor Dr. Carlisle for all his guidance and suggestions throughout this work. I would also like to thank my committee members, Dr. Chang and Dr. Cross for their cooperation. I would like to thank Vasudev Yendapally, who worked on the back-end of the tool for his help in integrating the components of the tool. Finally, I would like to thank Auburn University and Department of Computer Science for the excellent computing facilities.
# Table of Contents

1. Introduction .................................................................................................................. 1  
   Fig 1 ..............................................................................................................................

2. Information Retrieval .................................................................................................... 3  
   2.1 Information retrieval from mainframe system ....................................................... 3  
   2.1.1 Benefits and Limitations of command languages ............................................. 4  
   2.2 Information retrieval with Client/Server Computing ........................................... 4  
   2.2.2 Benefits and Limitations of Client/Server front-end GUI .................................. 4  
   Fig 2 ..............................................................................................................................

3. Detailed Description of Project ...................................................................................... 6  
   3.1 Problem Statement: ............................................................................................... 6  
   3.2 Project Implementation Overview ........................................................................ 6  
   3.3 Project Overview .................................................................................................... 9  
   3.4 Implementation details .......................................................................................... 11  
   Fig 3 ..............................................................................................................................  
   Fig 4 - Fig 15 ................................................................................................................

4. Project Results .............................................................................................................. 14  
   4.1 Tests performed on the system ............................................................................. 14  
   4.2 An example test .................................................................................................... 14  
   Fig 16. ........................................................................................................................  
   Fig 17. ........................................................................................................................  
   Fig 18. ........................................................................................................................  
   Fig 19. ........................................................................................................................

5. An evaluation of the interface ....................................................................................... 16  

6. Conclusions .................................................................................................................. 17  
   5.1 Benefits: ................................................................................................................. 17  
   5.2 Limitations: ............................................................................................................ 18  

7. References .................................................................................................................... 19
1. Introduction

It is no great surprise that the enterprise is changing. Where we once had centralized mainframe systems, we now have distributed systems consisting of a mixture of different components - different hardware platforms, operating systems, network protocols and database management systems. Although, the mainframe systems have not gone away completely, slowly, and surely the plugs on the mainframe systems will be pulled. Even though the enterprise is changing, one thing has not: the need for storing and retrieving data and making effective use of it in business decisions.

While the enterprise was refining solutions for data management in their mainframe systems. The computer industry came up with an effective alternate solution - Client/Server computing. Though it's history is short, Client/Server computing offers useful innovations for information management such as tools to manage databases remotely, to pull information from disparate databases, and replicate corporate data across a number of different locations [9].

1.1 Overview of Client/Server System

In a client/server system there are three distinct components, each focusing on a specific task:

a) a server (back-end) that provides it's services to a number of clients,

b) a client application (front-end) that seeks services from server, and

c) a network that supports the communication between the client and the server [8].

Fig 1. illustrates this client/server architecture and data transfer.

The server, focuses on efficiently handling the information. It's primary job is to manage
its resources optimally among the multiple clients that concurrently request information from the
server. The server must also manage any synchronization of services and communications once a
request has been initiated. Servers concentrate on tasks such as:

a) managing information among many concurrent users,
b) controlling access and other security requirements,
c) protecting information with backup and auto-recovery features in case of failures,
d) enforcing data integrity rules across all client applications.

The client, interfaces users with data on the server side. It is the process that requests
services from a server. The client normally initiates the conversation with the server. The client
in a client/server system focus on tasks such as:

a) presenting the user with an interface to accomplish work,
b) managing presentation logic such as pop-up lists on a data entry form or bar graphs in a
   graphical data presentation tool,
c) performing application logic, such as calculating fields in a data entry form,
d) validating data entry,
e) finding, requesting and receiving information from servers.

Finally, the network communication hardware and software provide the means to transmit
the data between the client and the server in a system. Both the client and the server run communi-
cation software that allows them to talk across a network.
Fig. 1 Client/Server architecture.

Note:
The links from Server to and from database represent data transfer.
2. Information retrieval

Information retrieval, concerned with retrieving stored information of interest to the user, deals with two types of representations: statements of the user's information needs, and the information stored.

2.1 Information retrieval from mainframe systems

In the past, when organization relied on mainframes, information retrieval was basically done through command languages from terminals connected to the mainframe [3]. Fig.2 depicts information retrieval from mainframe systems through a terminal. The command languages originated from operating system commands and used abbreviations or mnemonics for their command names. The user issued a command and waited for results from the mainframe. If the result was correct, the next command was issued. If not correct, the user had to adopt some other strategy. When dealing with command languages, the user had to know all the commands and the arguments to be passed whereever needed.

When using command line languages, users often make mistakes. The users invoke commands that are not available, type menu selection choices that are not permitted, request things that do not exist, or make typographical errors [3]. Error messages are often brief or vague, and in order to correct the problem users have to invest a substantial amount of time in figuring out the problem. Moreover, as commands, abbreviations, and symbols are made up of a series of keystrokes from a keyboard, there is a substantial chance of a user making an error when entering the sequence of key presses. Some key pressing sequences are more error-prone than others, especially those that require holding down a combination of keys.
2.1.1 Benefits and Limitations of command languages

A command language is an advantage to a user who is well versed with the system. It gives the user the power to handle very complex queries. Interactions with the commands can be much faster than with other methods such as menus, and users feel much more in control with command languages [3]. But a naive user will be totally lost in a command based system. The user must remember the syntax rules and series of actions that need to be taken in order to accomplish a task. If the user makes a mistake, it will be a very slow laborious process to recover.

2.2 Information retrieval with Client/Server computing:

A goal of Client/Server database graphical user interface (GUI) research is a presentation that demands less mental effort from its users, is easier to learn and easier to use [5]. Instead of concentrating on the learning and remembering some command syntax, the user can focus efforts on finding and choosing desired information. With options available on the screen, effects of all user actions are immediately visible, and with fewer commands to remember, a direct GUI interface is less demanding on users. At the same time, these direct-manipulation interfaces can accommodate as many functions as a command-language interface and also can give the user a sense of control.

Fig. 1 depicts information retrieval in a Client/Server system. Unlike command languages, user interfaces can be designed to eliminate the need to learn the syntax and semantics of entering search terms, structuring the search, and navigating through the system [4].

2.2.1 Benefits and Limitations of Client/Server front-end GUI:

In general, GUI systems have the advantage of being easier to use for naive users. GUI
Fig. 2 Mainframe systems.
menus eliminate rigorous training and memorization of complex command sequences. Options are presented on the screen, and a user will be led through the system step-by-step [7]. All user actions can be understood in terms of their effect on the screen. There are fewer key strokes, which reduce the chances of making mistake. Some commercial GUI systems have options to graphically display database schema and dependencies.

In GUI systems, there is a trade-off between system speed and a user’s accuracy. As the response times grow longer, users may become more anxious because the penalty for an error increases. This causes users to slow down in their work. As response time grows shorter and display rates increase, users pick up the pace of the system and may fail to comprehend the presented material. This can result in incorrect solution plans, and may cause execution errors [6]. A GUI for Client/Server systems should meet the general GUI standards of Microsoft. If not, users will have tough time using the GUI.
3. Detailed Description of Project

3.1 Problem Statement

In a client/server system the processing load of an application is split between the client and the server. The server manages data among a number of clients, controls concurrency and minimizes network traffic [10]. The client application sends request to the server for data, analyzes the data it receives and manages presentation logic.

Considering the benefits of the client/server systems and the growing concern for making database systems easier to use, the goal of the project was to design a query tool that takes advantage of the graphical workstations in the networked environment to support a naive user's access to the information provided by a database server.

In this project we tried to develop an easy-to-use, X/Motif based query tool to access data from the ORACLE DBMS. Our aim was to build an operational and simple to use front-end GUI that will be especially suited for a naive computer user. This will help a naive user to access the information in the database, without ever worrying about the commands required to retrieve the information. Anyone with a knowledge of windows, menus and buttons, mouse and keyboard should be able to use the tool.

3.2 Project Implementation Overview

The main strength of this system is its ability to link to an ORACLE database server and access the information in the database. The tool takes care of connecting to the source of data and building a query to extract the necessary information. ORACLE database servers support queries
with SQL command statements, and clients can hide the complexity of SQL statements required to retrieve information. Consequently the user can make queries without knowing SQL, or any database command language.

A fully functional system consists of three components. They are:

a) the front-end client

b) communication software that supports an interface between client and server

c) the back-end database server.

Fig. 3 shows the three components of such a system.

The front-end client is the X/Motif GUI with which the user interacts to query the database system. The front-end client in turn communicates with the back-end server through the communication software. The communication software is a module written in PRO*C, C, and using UNIX sockets. The back-end server is the ORACLE DBMS in the UNIX operating system which is the datastore for the project.

Each of the above mentioned components has been assigned specific responsibilities to ensure that the system works in client/server manner. These responsibilities will be discussed in the following sections.

a) The Front-End's Responsibility:

The front-end is responsible for providing access to the database information, building SQL queries, retrieving responses of the server and analyzing results and displaying them to the user. The following are some of the tasks the front-end handles:
1. Security:

In any client/server system security is the most important issue. Only authorized users should be allowed to retrieve information from the database system. To ensure this security, users should not be allowed to make any sort of queries until they are authenticated by the database system, generally with a user-id and password. The security system on the server plays an important role in validating the user-id and password. The security system should also check the privileges that the users have when they try to gain access to a database.

2. Database & Table listing:

If the user passes a security check, the user should be provided with a listing of databases that the user has privileges to access. Once the user selects database, all the tables in that particular database should be listed. This will help the users in picking the tables they are interested in and formulating query on the tables.

3. Attribute listing:

Once the user selects a set of tables, the attributes/fields of all these tables must be displayed. The user can then select the fields for which he/she needs the information.

4. SQL clauses:

No interface for databases is complete without the SQL clauses. The user must be presented with facility to refine queries, using the WHERE, ORDER BY, GROUP BY, HAVING clauses. This helps in limiting the number of rows of information retrieved from the database system [9].
5. Query Execution:

Once the user is through with all the steps for making a valid query, the user must be allowed to execute the query and see the response of the database system for the query.

6. Display results:

The database system’s response to the user’s query should be read from the communications interface, decrypted and then the response is displayed to the user. Each attribute of an information object should be in separate columns. This helps the users to quickly get to the information of their interest.

b) Communication software’s responsibility:

It is the responsibility of the communication software to ensure that an appropriate client/server communication takes place between the client and the server. The response time should be optimal, otherwise the user will get frustrated. If network security is required, the request for information must be encrypted. A protocol using UNIX sockets can be used for communication between a client and server. The following are some of the tasks the communication software is responsible for:

1. Client messages:

The client sends messages to the server through the communication software. The communication software processes all the messages from the client and sends them to the server.

2. Server messages:
The server sends all its responses to the client’s queries back to the client through the communication software. The communication software sends all server responses to the client without loss of any data.

c) Server’s responsibility:

The server should provide access to information to only the authorized users. It needs to check the privileges of users, when they try to gain access to database. The following are some of the tasks, the server is responsible for:

1. Connect authorized user:

The server should connect only an authorized user and provide access to the databases which the user is authorized to access.

2. Messages from intermediate server:

The server must be able to receive messages from the communication software and process them. Once it finishes processing them, it has to send the results back to the communication software.

3. Respond to client’s queries:

The server should respond to the client’s queries. It should process the client queries and send back its responses to the client through the communication software.

3.3 Project overview

This project is concerned with providing an interface to retrieve information from an
Fig. 3 Project overall picture.
Fig. 4 Initial screen.
Fig. 5 Main screen.
Fig. 6 Login popup window.
Fig. 7 Main screen after user passes security check.
Fig. 8 Popup window for listing Databases.
Fig. 9 Popup window for listing tables.
Fig. 10 Main screen after user selects tables.
Fig. 11  Main screen after user selects attributes from tables.
Fig. 12 GROUP BY SQL clause popup window.
Fig. 13 WHERE SQL clause popup window.
Fig. 14 ORDER BY SQL clause popup window.
Fig. 15 HAVING SQL clause popup window.
ORACLE server on a UNIX platform. The network communication hardware and software provide the means to transmit data between the client and server in the system. Both the clients and the server run communication software that allows them to talk across a network. The tool that has been designed and developed allows clients to access the ORACLE server using the application programmers interfaces (API's): dynamic SQL and embedded SQL.

The goal of the project is to develop easy to use GUI to access the data from an ORACLE database. To support this goal requires a software back-end server that allows the user to connect to the database only if they are authorized to connect and then display the list of the databases they have access to. The back end must safely and efficiently manage the data between multiple users. From the users point of view, when the user selects a database, the tables in that particular database should be listed for the user to make a selection, and proceed with the querying. Once the user selects a table, the attributes of the table should be displayed. Now, the user can make queries on the fields/attributes of the table by selection. The user can be provided with the option of making sub-queries. Whatever requests the user makes requires the back-end to accept and send the request to the database server where the request is processed and then returned to the client through the communication network.

Oracle was chosen as the database server for the project, because it is available on the College of Engineering network, and is the most commercially accepted database tool for the UNIX environment. In developing the client X/MOTIF and 'C' were used.

3.4 Implementation details

The front-end client in this client/server system was developed using X/Motif. It provides
the interface to query the database system and to display the database system's responses to the queries. The client uses sockets to communicate with the communication software. The client writes the query to the socket, which will be read by the communication software and it executes the query to retrieve the data from ORACLE server. The results of the execution of the query will be picked up by the communication software which writes the results to the socket. The front-end will be waiting for the results of it's query. Once the clients picks up the results, it displays them for the user to view. The front-end was developed to work with communication software developed by another member of the team.

As shown in Fig.4 when the application starts, the user is presented with a screen containing two menu options, Continue and Quit. Quit will exit the application, and Continue takes the user to the main screen illustrated in Fig. 5. Although the main screen has many options for the user to choose, all options except Connect are disabled when the user is initially presented with the main screen. The user has to choose the Connect option to connect/logon to the database server. When the user chooses the Connect option, a dialog box pops up to get a user-id and password of the user. This is shown in Fig. 6. If the user-id and password are valid the server lets the user logon. After successful login, all the valid options on the main screen will be enabled as shown in Fig. 7. The user can now choose the Databases options which will give him/her a listing of databases the user has privileges to access. An example screen is shown in Fig. 8. Once the user selects a database, a listing of all the tables in the database will be displayed as in Fig. 9. The user can select tables from which he/she needs data. The attributes of the tables that the user selects will be displayed at the bottom of the screen, as shown in Fig. 10. The user can select the attributes to view their values. All the choices that the user has made until now will be displayed in a tree like structure, as illustrated by Fig. 11. The user can use SQL clauses like GROUP BY,
Fig. 16 Main screen after selecting product and price table.
Fig. 17 Main screen after selecting attributes from product table and price table.
Fig. 18 WHERE SQL clause window with criteria.
<table>
<thead>
<tr>
<th>PRODID</th>
<th>DESCRIP</th>
<th>STDPRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>100860</td>
<td>ACE TENNIS RACKET I</td>
<td>4.8</td>
</tr>
<tr>
<td>100861</td>
<td>ACE TENNIS RACKET II</td>
<td>4.8</td>
</tr>
<tr>
<td>100860</td>
<td>ACE TENNIS RACKET I</td>
<td>58</td>
</tr>
<tr>
<td>100861</td>
<td>ACE TENNIS RACKET II</td>
<td>58</td>
</tr>
<tr>
<td>100860</td>
<td>ACE TENNIS RACKET I</td>
<td>54</td>
</tr>
<tr>
<td>100861</td>
<td>ACE TENNIS RACKET II</td>
<td>54</td>
</tr>
<tr>
<td>100860</td>
<td>ACE TENNIS RACKET I</td>
<td>35</td>
</tr>
<tr>
<td>100861</td>
<td>ACE TENNIS RACKET II</td>
<td>35</td>
</tr>
<tr>
<td>100860</td>
<td>ACE TENNIS RACKET I</td>
<td>32</td>
</tr>
<tr>
<td>100861</td>
<td>ACE TENNIS RACKET II</td>
<td>32</td>
</tr>
<tr>
<td>100860</td>
<td>ACE TENNIS RACKET I</td>
<td>30</td>
</tr>
<tr>
<td>100861</td>
<td>ACE TENNIS RACKET II</td>
<td>30</td>
</tr>
<tr>
<td>100860</td>
<td>ACE TENNIS RACKET I</td>
<td>45</td>
</tr>
<tr>
<td>100861</td>
<td>ACE TENNIS RACKET II</td>
<td>45</td>
</tr>
<tr>
<td>100860</td>
<td>ACE TENNIS RACKET I</td>
<td>42</td>
</tr>
<tr>
<td>100861</td>
<td>ACE TENNIS RACKET II</td>
<td>42</td>
</tr>
<tr>
<td>100860</td>
<td>ACE TENNIS RACKET I</td>
<td>39</td>
</tr>
<tr>
<td>100861</td>
<td>ACE TENNIS RACKET II</td>
<td>39</td>
</tr>
<tr>
<td>100860</td>
<td>ACE TENNIS RACKET I</td>
<td>2.8</td>
</tr>
<tr>
<td>100861</td>
<td>ACE TENNIS RACKET II</td>
<td>2.8</td>
</tr>
<tr>
<td>100860</td>
<td>ACE TENNIS RACKET I</td>
<td>2.4</td>
</tr>
<tr>
<td>100861</td>
<td>ACE TENNIS RACKET II</td>
<td>2.4</td>
</tr>
</tbody>
</table>

Fig. 19 Results window.
WHERE, ORDER BY, HAVING etc. to limit the number of rows retrieved by the server. This helps in viewing only that information of interest to the user. Popup windows are provided for the SQL clauses so that the user can formulate an expression to limit the amount of information retrieved. Fig 12 - Fig. 18 illustrates this usage, and a discussion of the query is given in the next section. After the user has finished building the query, the query is executed by selecting the Execute option. The results of the query execution by the server are displayed in a results window. Fig. 19 shows a results window. The user can quit the results window by pressing the OK button. If the users commit errors while building a query, they can undo the query and start over with a new one by choosing the New Query option.
4. Project Results

The querying tool was successfully implemented, and tested with intermediate communication software developed by another team member. The tool was used to build valid SQL queries and sub-queries. These queries were then passed to the intermediate communication software which passed them for execution to an ORACLE V6 server on a SUN platform for execution. The server provided results, and the error messages were displayed in the results window. In order to execute any query the intermediate communication server should be running. So, for the system to work properly, it is important to see that the intermediate server is started before the client application.

4.1 Tests performed on the system and example usage:

The application was tested as follows:

A sample database was used for the purpose of testing the software. The tool was successfully tested using a variety of SQL queries and sub-queries. All valid SQL query/sub-query provided correct results, and the error messages were displayed if there is an error in the query.

4.2 An example test:

To illustrate usage of the tool we trace the action of a sample query. Suppose the user wishes to query two tables, product and price that are in the database. As soon as the user logs on he can see all the tables present in the database by selecting the tables icon. As described earlier, when the user selects on the table icon the client sends a signal to the back-end indicating this request. The server now makes an appropriate request to the Oracle database which fetches the results. The user then sees a list of tables in the database. The user highlights the tables product
and price and selects OK. Now the tables product and price are displayed, and the attributes of the
tables are displayed at the bottom of the main screen. This is shown in Fig. 16. If the user wants
to query the product id and the product description in the product table and the standard price from
the price table where prodid is less than 100870, the user selects the prodid and the description
attributes from product table and stdprice from price table. Fig. 17 shows the main screen after
these selections. Further, to use WHERE clause on prodid table the user can select the WHERE
clause icon and specify the criteria in the popup window for WHERE clause. In the popup
window for the WHERE clause, the user selects product.prodid as the Expression1 (field or
combination of fields on which criteria is to be applied) , "<" for Operator, and Expression2 (the
limiting value ) will be 100870. Fig. 18 shows criteria in the WHERE clause window. Once the
criteria is entered, the user can press OK and return to main screen to execute the query. Now the
query is ready to be executed. The user can press the Execute button to send the query to the
server. The query is processed and the results are sent back to the client as previously explained.
The results will be displayed in a popup results window as shown in Fig. 19.
5. An evaluation of the interface

The goal of the project was to develop an easy-to-use, X/Motif based query tool suited for a naive computer user to access data from the ORACLE DBMS without worrying about SQL commands and their syntax. The GUI developed provides an option to list names of all the tables in the database. Once, the user selects the tables to query, the selected table attributes are shown at the bottom of the screen. This helps the user in deciding which attributes to query upon. After selecting attributes, the user can go ahead and execute the query. All the selections the user makes till the point of execution will be shown in a tree like hierarchy in the main window. This helps in identifying a table and its selected attributes. If the user is not satisfied with the selections, the query can be undone by pressing the New Query option. Options to include WHERE, ORDER BY, HAVING, GROUP BY clauses are provided.

The tool currently serves as query tool only. Tables cannot be updated with this tool. When using SQL clauses like WHERE, ORDER BY etc. to make sub-queries, some knowledge of SQL and databases is assumed. Database schema and dependencies are not shown, making it difficult for the users when they try to make joins.
6. Conclusions

The examples given in the previous section show that a GUI can provide a very powerful means of communication with a database, supports connection to the database, access to authorized information in the database, and queries from the database.

The client and the server are independent programs. Network software resides in both the programs to allow them to communicate with each other. By having the client and server separated, each functional component in the system can be specialized to do something the best way it can. For example, to develop a client application a programmer concentrates on the presentation of the data and an attractive user interface, whereas the back-end developer concentrates on retrieving and distributing the data between the clients. Thus a division of processing load is achieved by compiling and running the client and the server separately.

The clients connect to the server and the intermediate server concurrently responds to the multiple clients that request information. The intermediate server thus plays the role of the oracle database client. In reality the user requests are sent to the ORACLE database by the intermediate server and the results are obtained and passed back to the client. Thus the objective of servicing multiple clients is achieved.

We summarize the advantages and also give the limitations of the current system:

6.1 Benefits:

The end-users have better access to data. This is partly due to the basic architecture of the client/server system and partly due to the ease of use of the system.
The client/server based applications can be developed in less time than the mainframe-based applications. Since front-end and back-end processes are separate, they can be developed and maintained separately thus increasing the reliability. Processing load is split between the client application and a database server.

### 6.2 Limitations:

This tool currently serves only as a querying tool. Tables cannot be updated with this tool. Although the naive user can perform any simple query to the database, some amount of knowledge of the oracle database is assumed when performing sub-queries.
7. References


