DATABASE QUERYING TOOL

Technical Report 95-05

Vasudev Yendapally

Department of Computer Science & Engineering

Auburn University

Auburn University, Alabama 36849-5347

May 3, 1995

1.INTRODUCTION

Occasionally, computer technology makes a break with the past. The relational model of database management, with its simple, tabular data structures and powerful data manipulation operations, was one such revolution. Another revolution in computing technology, client/server computing, took place in the last decade with the spread of minicomputers and microcomputers and a network to support intra machine communication. These highly cost-effective and flexible open systems have made client/server computing possible. Client/server computing delivers the benefits of the network computing model along with the shared data access and high performance characteristics of the host-based computing model.

Clients and servers are characterized by endless access to each other's resources. They provide advanced communications and raw computing power to handle demanding applications, as well as graphic user interfaces (GUIs).

In a client/server database system there are three distinct components, each focusing on a specific job required to provide a user's environment:

- a database server that manages the database among a number of clients,
- a client application that desires services of the database, and
- a network that supports the communication between the client, (front end) and the server that deals with the network and the database system (back end).

One server, the so called back-end, focuses on efficiently managing the information. A back-end server's primary job is to manage its resource optimally among the multiple clients that concurrently request the server for the resource. Generally the server does not send information to the requester until the client process tells it to do so. The server must also manage the synchronization of services as well as communications once a request has been initiated. Database servers concentrate on tasks such as:

- managing a single database of information among many concurrent users,
- controlling database access and other security requirements,
- protecting database information with backup and recovery features,
- centrally enforcing global data integrity rules across all client applications.

A client application, the front-end, is the part of the system that users employ to interact with data. It is the process that requests services from a server. The client normally initiates the conversation with the server. The client is therefore the more active partner in this association, by requesting specific functions, accepting corresponding results from the server, and acknowledging the completion of services. The client, however, does not manage the synchronization of services and associated communications. Many clients may share a single server and the same client may access many servers. The client applications in a client/server database system focus on jobs such as:

- presenting an interface a user can interact with to accomplish work,
- managing presentation logic such as pop-up lists on a data entry form or bar graphs in a graphical data presentation tool,

- performing application logic, such as calculating fields in a data entry form,
- validating data entry,
- requesting and receiving information from a database server.

Finally, the network communication hardware and software provide the means to transmit the data between the clients and the server in a system. Both the clients and the server run communication software that allows them to talk across a network.

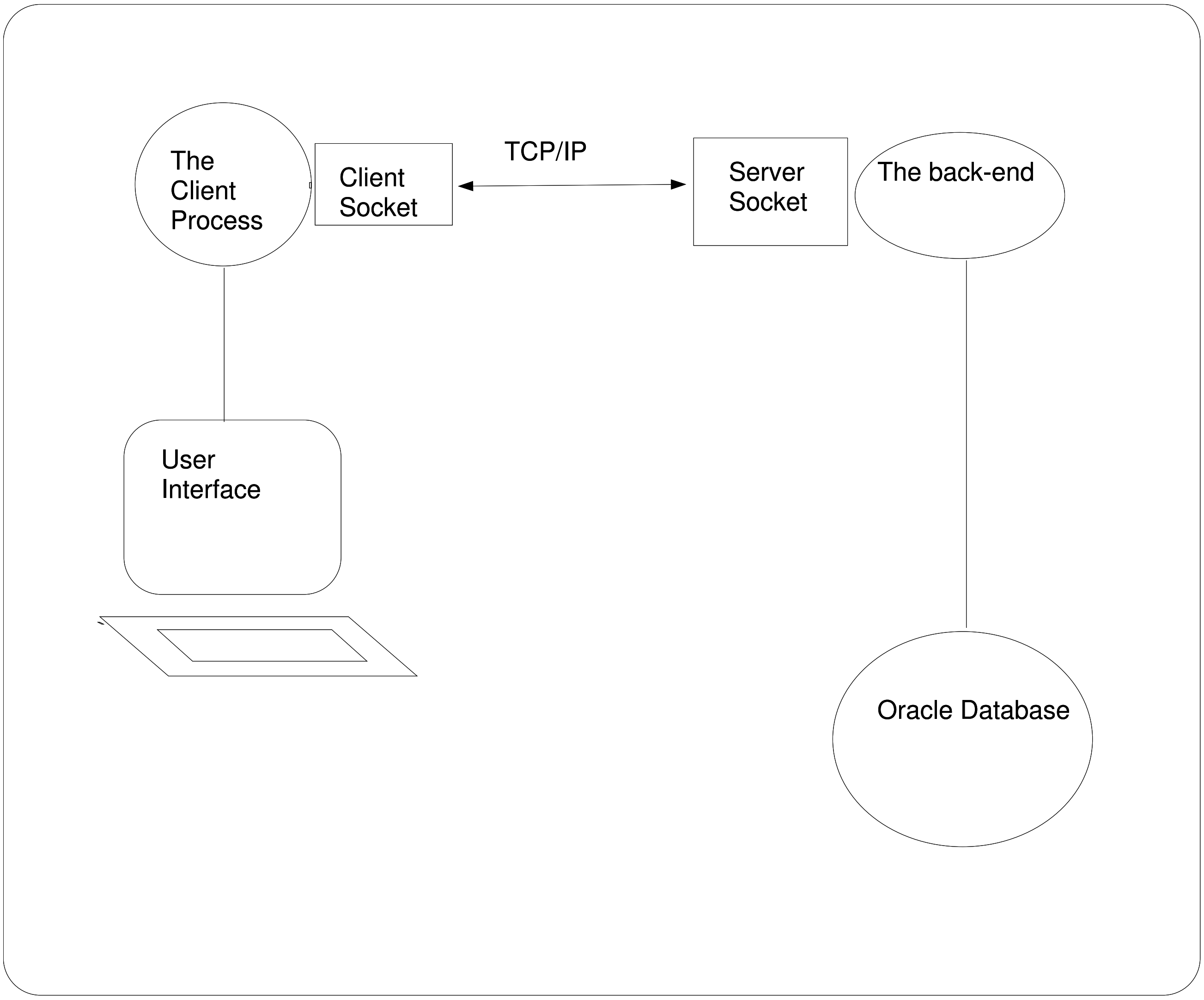


Fig 1. Client /Server model

2.LITERATURE REVIEW

Database Gateways.

Database gateways act as interface translators that move data, SQL commands, and applications from one type of database to another. A client, gateway, and server can reside on the same platform or different ones. This allows gateways to connect clients and servers running on dissimilar networks. Database gateways are ideally suited to bridge network protocols, such as TCP/IP, OSI, and SNA.

Database gateways have the following functions:

- accept statements specified by a well-defined grammar (usually SQL) from a client application,
- translate the statements to a specific database format,
- send the statements to be executed to a database program,
- translate the results back into a well-defined format,
- return the data and status information to the client.

To appear transparent to the user, a database gateway must also include login validation, a cancel/interrupt/error handler and datatype conversions. The following are examples of commercial database gateways.

EDA/SQL

Enterprise Data Access/SQL (EDA/SQL), from Information Builders, Inc.

(IBI), is a family of client/server products, based on an open architecture, that provides SQL-based access to relational and non-relational data sources on a networked multivendor system. EDA/SQL provides a uniform, relational view of data, regardless of its storage structure. At the core of the EDA architecture is the multi-threaded EDA/SQL server, a host component, which manages the flow of data; handles the interactions with communication protocols and target databases.

Database Gateway

Database Gateway from Micro Decision ware, Inc. (MDI) resides on an OS/2 server platform and accepts SQL transactions from client stations. The SQL transactions are redirected to DB2, SQL/DS databases for processing. The client stations can be running DOS, Windows, or OS/2. The Database Gateway Solution includes two components. One component runs on the LAN under the OS/2 communication manager and handles communications to and from the client applications and the host. The second component is the host potion that links the gateway to DB2, SQL/DS.

Database Gateway for SQL/DS interfaces with SQL/DS-VSE Access server on the host to provide access to SQL/DS databases. Communication between the front-end and the gateway is achieved using named pipes, the interprocess communication mechanism used in the SQL server. The gateway will therefore run on any network that can support named pipes. Communication between the gateway server and the IBM host is then achieved via IBM's APPC protocol.

SQL Bridge

SQL Bridge from Microsoft Corp. is a protocol gateway for building distributed client/server systems in Windows, DOS and OS/2 environments. It allows applications to access SQL server data through one protocol, rather than several, providing seamless integration of SQL server applications, databases, and gateways, independent of operating systems and network protocols. It routes client/server requests across networks having different protocols, simplifying network configuration, and increasing interoperability.

The Mafia Project

This project is concerned with tools, similar to the above commercial systems but with the main difference being that this project is targeted to the UNIX platform, and communication achieved using TCP/IP sockets. The tool allows allows applications to access ORACLE server using the application programmers interfaces (API's) : dynamic SQL and embedded SQL.

3.DETAILED DESCRIPTION OF RESEARCH

3.1 Problem Statement:

A client/server system can deliver better performance than a file server system because a client application and a database server work together to split the processing load of an application. The server manages the database among a number of clients, while the clients send, request, and analyze the data they receive from the server. In a client/server application, the client application works with small specific data sets, such as rows in a table - not files, as in the file server system. A database server is selective, locking and returning only information a client requests, or adding information as specified. This ensures concurrency, minimizes network traffic, and improves system performance.

Just as the computing models have evolved, so have the user interfaces. GUIs make user more productive for several reasons. Instead of requiring a user to know application commands, a GUI uses an intuitive, easy-to-use shell with a pointing device (such as mouse), as well as application windows, menus, dialog boxes, buttons, and so on, that hide the complexity of commands from the user. This enables the user to become more productive. Another reason is that GUIs can typically display multiple applications in different windows at the same time. This way, users can see what is happening in one application while working with another, or transfer information between different applications with cut-and-paste operations. Finally, well-defined GUIs have specifications with which all applications must confirm to make the look and feel of every application consistent. This way, users can easily learn new applications because they recognize certain functions from application to application.

Looking at the benefits of the client/server computing model and the ease of use of GUIs has led us into thinking about a querying tool that can take advantage of the client/server methodology and attractive GUIs. The idea is to facilitate the naive user to use the database and query the required information without having to learn and memorize a database language.

The goal of the project is to investigate easy to use GUIs to access the data from a database. To support this goal requires a software back-end server that allows the user to connect to the database only if he is authorized to connect and then displaying the list of the databases he has 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 selection and proceed with the querying. Once the user selects on a table the attributes of the table should be displayed. Now the user can make queries on the fields/attributes 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.

3.2 Project Implementation Overview

The Database Server

Oracle was chosen as the database server for the project, because it is locally available in the UNIX environment, is the most commercially accepted database tool for this environment, and Oracle provides a set of tools to support the project design, most importantly an application development toolset.

The Back-End

An application development tool is an application that developers use to create custom client applications. Application development tools come in two varieties: 3GLs and 4GLs. A developer can create a client application with procedural programming techniques by using a third-generation language (3GL) development tool along with a third-generation programming language such as C, Ada, or COBOL. Alternatively, a developer can use a fourth generation language (4GL) application development tool to develop a client application with nonprocedural programming techniques such as visual form layout and screen painting. To develop a 3GL client database application, the source code of the application is programmed using the procedural commands of a 3GL compiler along with embedded SQL statements that make calls to a database server. For this project pro*c, which is a 3GL application development tool was used.

The Oracle precompilers work with the C language compiler to develop custom applications that have embedded SQL statements. The precompiler is used to do a first pass on the source program to identify and translate all embedded SQL statements and generate a modified source program that the C compiler can understand. Then the standard language compiler is used to compile and link the program to its finished and executable form.

As mentioned earlier, embedded SQL refers to the use of SQL statements embedded within a procedural programming language. It is a collection of all SQL commands and the flow control commands that integrate the SQL commands within a procedural language. Each embedded SQL statement is prefixed by the keywords EXEC SQL to distinguish it from the host language.

The Network

The connection and the communication between the client and the server were designed to use TCP/IP sockets. Given an internet connection, the roles played by the sockets at the two communication end points are not the same. The socket created first is called a "server," the one created later is called a "client." When the client process (the process using the client socket) exits, no effect is generated to the server socket. In this case, although the complete connection is gone, the server socket is not thrown away. Therefore the server process can still accept calls from other clients. In contrast, if the server process quits, the connection and the client socket are both discarded. Thus the server socket supports the back-end and the client socket is for the client application (front-end).

The asynchronous communication protocol used between the front-end and the back-end can be said to be conversational because both components remain active during a conversation. The communication is complete only if the client or server passes a message and gets back an acknowledgment. Any request the user makes is sent to the back-end with the help of the client socket. The server forks a child and delegates the work to the child. The child sends the query to the ORACLE server and gets back the results. The results are then sent to the client through the server socket. The server sends the next packet only after getting an acknowledgment for the previous packet. This avoids confusion between the client and server and avoids mix-up of message packets. The child exits after performing the work delegated by the parent. For every request made the server forks a new child, this way requests from different clients can be taken by the server at the same time, processed, and sent back to the respective clients.

3.3 Program Description:

The code for the backend was written in pro*c, which is an embedded SQL language. Pro*c permits the application developer to write the code in the C programming language with embedded SQL statements. The modules that were designed to implement data retrieval will be explained in detail.

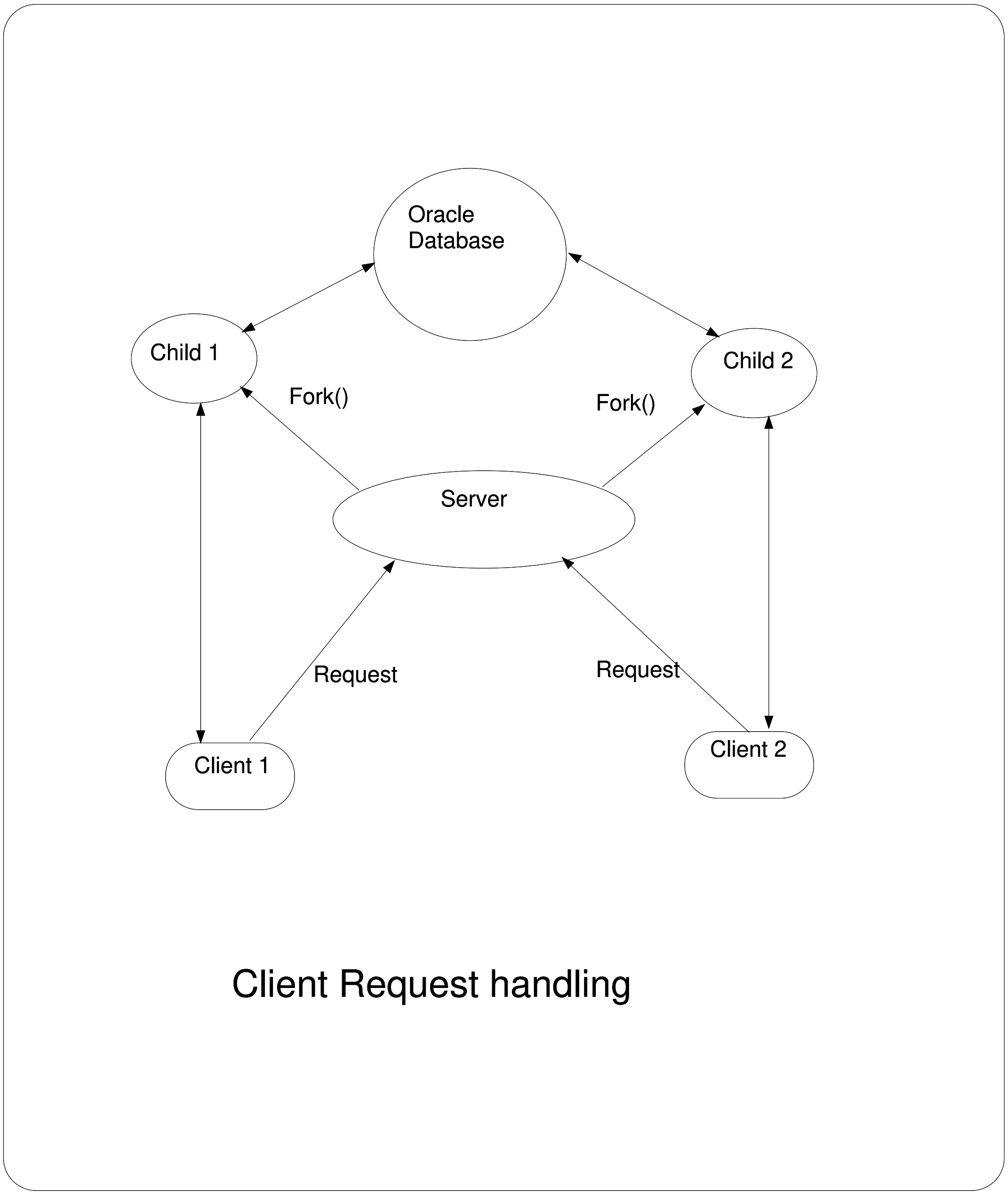
The code for the back-end is special purpose, as it was written to support a front-end being developed by other investigators. However with some minor changes it can be adapted to front-ends that request information using the communication protocol that was designed. Code for the major components of the back end can be found in the appendix. This section is intended to assist the reader understand the code found there.

In the code the INCLUDE statement is used. This include statement is not to be confused with the c directive #include. The SQLCA that is included is a record-like data structure, which provides for diagnostic checking and event handling. Its fields contain error, warning, and status information updated by ORACLE whenever a SQL statement is executed. Thus, the SQLCA always reflects the outcome of the most recent SQL operation. The SQLCA contains runtime information about the execution of SQL statements, such as ORACLE error codes, warning flags, event information, rows-processed count, parse error offset, and diagnostics. The ORACA included in the program is a similar data structure copied into the program to handle ORACLE-specific communications. To enable the ORACA, the ORACA precompiler option was set to YES.

The VARCHAR variable used in the declare section is an extended C type or predeclared struct. It contains the length of the array and the array itself as its members. An SQL query is assigned to the VARCHAR sql statement. Both the array and the length parts are properly set. The query contains one host-variable and one placeholder, for which an actual input host variable is supplied at the OPEN time. The PREPARED statement associates a statement name with a string containing a SELECT statement. The statement name is a SQL identifier, not a host variable, and therefore does not appear in the declare section. The DECLARE statement associates a cursor with a PREPAREd statement. The cursor name, like the statement name, does not appear in the declare section. The OPEN statement evaluates the active set of the PREPAREd query USING the specified input host variables, which are substituted positionally for placeholders in the PREPAREd query. For each occurrence of a placeholder in the statement there must be a variable in the USING clause, that is, if a placeholder occurs multiple times in the statement, the corresponding variable must appear multiple times in the USING clause. The USING clause can be omitted only if the statement contains no placeholders. OPEN places the cursor at the first row of the active set in preparation for a FETCH. A single DECLAREd cursor can be OPENed more than once, optionally USING different input host variables. The FETCH statement places the select list of the current row into the variables specified by the INTO clause, then advances the cursor to the next row. If there are more select-list fields than output host variables, the extra fields will not be returned. Specifying more output host variables than select-list fields results in an ORACLE error.

The main program makes a call to the function CallServer() which opens a socket and waits in an infinite loop. Whenever the user or the client requests for a connection the function ConnectDatabase() is called. The function connectdatabase takes the username and password each of which is a varchar and tries to connect to the database using the exec sql connect statement if the connection is successful it returns a 1 else it returns 0. Once the user is connected to the database the

Server on receipt of the tables request returns the tables in the database by calling the function DisplayAllTables(). This function prepares an SQL statement to get all the table names from the user tables and then sends them to the client through the socket. A user click on a particular table is an indication to the server that the attributes in the table need to be displayed, the function ShowAttributes() is called at this stage which gets the table name passed on to it. There is no specific SQL select statement that could get the attributes of a given table. So for this purpose an SQL statement that gets the attributes of all the user-tables is prepared. From the resulting attributes only the required attributes are picked by string comparison. The next function that is used is the ExecAnyStmt(). This function takes any statement from the userr, executes it, and gets the results back. The results are written back to the client for display. The utility functions ConvertToLower and ConvertToUpper are used to make the display look better, and to make proper string comparisons. The utility function ConnectServer() in the client is called whenever the client wants to make a request to the server.


Oracle Database
Child 1
Child 2
Fork()
Fork()
Server
Request
Request
Client 1
Client 2
Client Request handling

<u>Fig 2.</u>

4.PROJECT RESULTS

The querying tool was successfully implemented, and tested to confirm that it is capable of performing any query to any database to which the user has access. A prototype front end was used to test the back end software, and we briefly discuss this component. The main window presented to the user has continue and quit options (Fig.3). The user clicks on continue to have the next window popped up which gives the user the choice to connect or to quit. Once the user chooses to connect he is prompted to type a user name and the password. The username and the password are taken and then the user is logged onto the database if authorize, i.e. the username and the password are sent to the back-end which sends back a "yes" if the user is authorized to log on to the database otherwise it sends back a "no". The user cannot get past this window unless he is authorized to log on to the database.

Once the user logs on to the database the databases, query, newquery,execute and quit buttons on the querywin are enabled and the connect button is disabled since he is already into the database. Once the user chooses to query the selected database, icons are displayed to the left hand side of the tool as shown in Fig.4. The user can click on the icons and make the appropriate query. The first icon shows a table and clicking on that will display all the tables that the user has access to in the current database. Once the user chooses the tables that he wants to query, only those tables are displayed. At the bottom of the window the attributes in each of the selected tables are also displayed. This is shown in Fig.4. The user can highlight on the attributes that he wants to query, and clicking the execute button at this stage is equivalent to the sql query "select the highlighted attributes from the tables". The user is provided with the options of group by, order by, where and having. The user can click on these choices to construct an appropriate query.

Once the user query is performed the results are displayed on a pop-up window as shown in

Fig. 5. The user can quit this window by clicking on the OK button at the bottom of the window. The user can perform another query by clicking again on the new_query button and thus keep querying the database interactively until he is done with the querying.

4.1 Tests performed on the system and example usage:

The software was tested as follows:

A sample database was made for the purpose of testing the software. The tool was successfully tested using all possible SQL queries and sub-quesries. Any legitimate SQL query/ subquery provided desired results, and the error messages were displayed if there is an error in the query.

Now to understand the tool better we trace the action of a sample query. Suppose the user wishes to query two tables, storea and w_countrya that are in the database. As soon as the user logs on he can see all the tables present in the database by selecting on 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 back-end now makes an appropriate request to the oracle database and fetches the results, and the user sees a list of tables in the database. He then highlights the tables storea and w_countrya and selects OK. Now the tables storea and w_countrya are displayed also the attributes of the tables are displayed at the bottom of the querywin. This is shown in Fig.6. If the user wants to query the name of the store and the country names in the w_countrya and he wants to order the items by the store name, he selects the store name and the country name and selects the orderby icon. A window for the order by selection is popped up and the user is prompted to select the attribute he wants to order by. The user chooses the store name by which he wants to order his results. Now the query is constructed and sent to the back-end. The query is processed and the results are sent back to the front-end as explained earlier. The user click on "execute" displays the results on the popup window as shown in Fig. 7.

5.CONCLUSION

From the example that was given in the previous section it is evident that pictures provide a very powerful means of communication with a database, and that the user can connect to the database and access authorized information in the database and perform the queries on the tables without actually knowing the SQL syntax or the database language.

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

The clients connect to the server and the server concurrently services the multiple clients that request information. The server thus plays the role of the oracle database and the user thinks he is communicating directly with the oracle database. In reality the user requests are sent to the oracle database and the results obtained by the back-end. Thus the objective of servicing multiple clients was achieved.

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

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 modularity of the client/server architecture has enabled the package to be developed to facilitate the naive user to use the database and query the required information without having to learn and memorize a database language.

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 productivity.

The client/server system can deliver better performance than a file server system because a client application and a database server work together to split the processing load of the application.

Limitations:

This tool currently serves only as a querying tool . Tables cannot be updated through this.

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.

6.References

1. Berson, A., *Client/Server Architecture* , Newyork, McGraw-Hill, 1992.

2. Black U.D., *Data Communications and distributed networks*, second edition, Englewod cliffs, NJ. Yourdon press, Prentice Hall, 1992

3. Barbara Bochenski , *Implementing production-quality client/server systems.*, John Wiley & Soncs Inc. 1994.

4. Setrag Khoshafian, Arvola Chan, Anna Wong, Harry K.T. Wong, *A guide to developing Client/Server SQL applications,* Morgan Kaufmann publishers, inc. 1992

5. Steven M. Bobrowski, *Mastering Oracle 7 & Client/Server computing,* Sybex inc. USA.

6. Dawna Travis Dewire, *Client/Server Computing,* McGraw-Hill inc. 1993.

7. Dimitris N. Chorafas, *Beyond LANs,* McGraw-Hill inc. 1994. .

8. Jerry Cashin, *Client/Server Networking protocols,*.first edition, Computer Technology Research Corporation