COMPUTER ASSISTED QUERY SYSTEM

Technical Report 94-12

Sanjay Kapoor

Department of Computer Science and Engineering
Auburn University
Auburn University, AL 36849-5347

August 26, 1994

PROJECT REPORT

COMPUTER ASSISTED QUERY SYSTEM

Submitted
by
Sanjay Kapoor

as a partial requirement of fulfilling
master of
Computer Science and Engineering

Advisor
Dr. W. H. Carlisle

Committee members
Dr. James Cross
Dr. Kai Chang

Abstract

With the growing concern for making database management systems (DBMS) easier to use, efforts are being made to design a query language interface that takes advantage of the capabilities of graphical workstations. The trend is to isolate the user from hassle of complexity of SQL statements and from having to know how the data is stored; that is, to allow a user to make queries without knowing SQL, or some other database language. For the administrator of the database, another important requirement of the interface is to insert data, delete data and be able to create new tables in the database. My project is an exploration of replacing a database command language by an appropriate user interface and of software required to support such an interface.

INTRODUCTION

Over the past 20 years database systems have been used extensively throughout the world for information retrieval. The techniques for retrieving data from databases have grown and matured, but have not changed very much from their original form. Recent research in user interface has added a new perspective for dealing with database systems. These investigations have shown the importance of including a user profile and the domain knowledge in the user interface. Users of information retrieval benefit from a user interface that uses built in domain knowledge.

The user interface is where users formulate their request and transmit them to the system, and where the system conveys its responses and sends the messages back to the users. User interface design includes how the users interact with the system, how the system expresses its responses, and how the responses are shown on the terminal. There are three basic types of interaction between human and computers, namely command language, menu selection, and direct manipulation.

Studies have shown that a user of a database system must understand two different aspects of searching: mechanical and conceptual [Brogman, 1984]. Mechanical knowledge includes the syntax and semantics of entering a search expression, structuring a search, and negotiating through the system. Mechanical problems include the typographical errors or misspellings, unrecognizable commands, incorrect formatting of commands, and invalid

item numbers. Conceptual knowledge includes when to use an access point, how to narrow or broaden search results and why the result of search is zero or too much information to handle. Conceptual problems include misunderstanding of database scope, system features, misuse of boolean operators, and termination symbols. Users encounter problems when either of these aspects is not fully understood. For the nonexpert, a graphical user interface has the advantage of being easier to use. All the options are presented on screen. The user is led through the system step by step. There are fewer keystrokes, which reduces the chance of mistakes. By comparison, the advantage of a command language is the power for handling complex queries. However with a command language, the user must know the syntax rules and initiative actions of the query language.

A well-designed graphical user interface should make visible on the screen everything relevant to a database search. When relevant information is visible, objects can be understood in terms of their visible characteristics and in terms of their effect on the screen. Visibility also relieves the user from the burden of memorizing all commands and options. When all applicable options are shown on the screen, the user needs only to recognize and choose an action, which is easier than generating a command and issuing it according to syntax rules.

This research is an exploration of replacing a database command language by an appropriate user interface and of software required to support such an interface.

PROBLEM STATEMENT

With the growing concern for making database management systems (DBMS) easier to use, efforts are being made to design a query language interface that takes advantage of the capabilities of graphical workstations. The trend is to isolate the user from hassle of complexity of SQL statements and from having to know how the data is stored; that is, to allow a user to make queries without knowing SQL, or some other database language. For the administrator of the database, another important requirement of the interface is to insert data, delete data and be able to create new tables in the database. For our project ORACLE was chosen as the database system. The reason for selecting ORACLE was to provide a working prototype DBMS as part of the system's design.

This project was developed using client/server programming as shown in fig. 2.1. The front end client component communicates with the user and server. A specified user query is sent to the interface between the front end and the back end. This interface in turn sends a query to a back end database server. The back end server executes the user query and sends the result back to the interface between the back end and front end. The front end gets the results from the interface and displays the results in some user friendly manner.

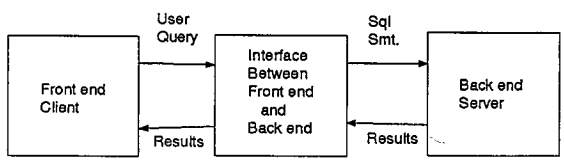

Fig. 2.1

With this separation of responsibilities, the client's objective is to present the user with a means to make queries. This front end will provide the user a convenient way to specify the database queries. The following are some problems that must be addressed in designing the client:

1. only authorized users can access the database, so the client must provide the user with a way to connect to the system;
2. the client must provide the user with a "view", i.e. the names of tables which can be accessed;
3. the client must convert the user's query into a message for the server and handle the response.

As previously stated, the main objective of the client is to assist the user in specifying the correct query without any knowledge of SQL commands and to present the results in a user friendly form. The user can specify the query by clicking on widgets. It is important to handle both the mechanical and conceptual knowledge in order to assist the user in specifying a correct query, converting that query into a valid SQL statement, and finally presenting the results in a user friendly manner.

To achieve an appropriate client/server communication, a protocol should be developed using UNIX sockets. The following are some problems that must be addressed in designing a protocol:

1. receive the message sent by the user interface, process the received message and send the message to the back end server;
2. receive the messages send by back end server and save them in an appropriate manner to be accessed by the user interface.

The major responsibilities of the back end server that must be addressed are:

1. connecting the user with the database system;
2. establishing a connection with the interface and receiving messages from the interface;
3. processing the SQL query and sending the results and error messages back to the interface.

A complete back end, a communication protocol, and a prototype front end to demonstrate the protocol and functions of the back end was written for this masters project. The back end was designed using PRO*C and the C language, and the prototype front end was designed using Motif widgets and the C language. The prototype user interface provides the user with following functions.

- logon to database.
- buttons to see available tables.
- way to specify queries.

SOLUTION

As stated in the previous section this project is developed using client/server programming. The design of the three main software components is explained below.

Server:

The server receives the client query, gets the results corresponding to client query, and sends back the results to front end client. SQL is required in order to access and manipulate the Oracle data. SQL can be used in two ways, either interactively or by placing the SQL statement in an application program, this is called embedded SQL. For embedding a SQL statement in an application a special tool is required. An ORACLE precompiler is the tool that allows to embed SQL statement in a high level source program. For our project we are using the C Oracle precomplier which is also called as PRO*C precompiler. Embedded SQL statements can be either static or dynamic. In our back end server design, we are using PRO*C precompiler with dynamic embedded SQL statement.

Pro*C opens a work area called a context area to process an embedded SQL statement. The context area stores information needed to execute the SQL statement. An identifier called a cursor accesses the information in the context area and control its processing.

Select-list items are the columns or expressions following the keyword "select" in

an SQL query. Placeholders are dummy bind variables that holds the places in a SQL statement for actual bind variables. Bind variables are the used by the application program and Oracle to exchange data and status information.

The PRO*C precompiler generates calls to ORACLE for all executable SQL statements. For executing a dynamic SQL statement Oracle needs the following information

1. the number of bind variables and select-list items;
2. the length of each variable;
3. the data type of each bind variable and select-list item;
4. the address of each bind variable, and of the output variable that will receive each select-list item

All the information Oracle needs about the select-list items or place holders for bind variables (except their values) is stored in a data structure called the SQL descriptor area (SQLDA). The values of select-list items are stored in output variables. The values of bind variables are stored in input variables. Addresses of these variables are stored in the SQLDA. So SQLDA provides the addresses where Oracle writes output values and reads input values. Output values are fetched using cursor, and input values are typically filled in by the program.

Software for the back end server is divided into the following functions:

1. Connection_to_Oracle

2. Contact_server

3 Query_process

Connection_To_Oracle: This function connects the client to the Oracle database. The function receives the user name and pass word sent by the client, confirms the user name and pass word, and connects to the Oracle database.

Contact_server is a gateway to the server process. When the client process contacts the Contact_server, it spawns a Query_process and then returns to its original state waiting for the another client request. This process acts as a server waiting for a service request from the client process.

Query_process is a process that generates the results corresponding to a client query. This process allocates the memory, declares a cursor, describes the bind variables, opens a cursor, executes the statement, and calls a process_select_list function that gets the attribute values from the Oracle. If the statement is not a query, then the process_select_list function returns doing nothing. Then query_process finds out how many rows were processed and on completion releases the memory allocated and closes the cursor.

Protocol:

The communication between client and server is achieved using UNIX sockets. A simple communication protocol is designed which can be divided into two parts:

1. the messages send by client;
2. the messages send by server

Client Messages:

The client can send three types of messages:

1. request for connection;
2. SQL statement;
3. exit.

The format of a client message is very simple. It first sends a single character representing the type of message, then an integer representing the length of the message (string), and following that it sends the actual message string.

Server Messages:

The database systems maintain each table and information related to the table. Each table consists of different records and each record consists of different fields. The back end retrieves the relevant information corresponding to user query from a database table, and sends this information to the front end. In order to achieve a full communication, the server sends the following message types in response to client queries:

1. header of table which is actually requested attributes of table by client;

2. width of each column;

3. type of column;

4. actual records;

5. stop message;

6. error message.

The format of a server message is same as that of the client.

Client:

A prototype front end client was developed for the purpose of demonstrating the capabilities of the back end and testing the functionality of the back end. This prototype also allowed experimentation with ways to assist the user in making correct query and reduce the SQL statement errors. This front end also presents the results in a user friendly manner. The main function of this front end are:

1. connection - it connects the user with the Oracle database system by obtaining the user name and password, and sending these to the back end server.
2. Show tables - displays various tables which a user may access. Tables are shown in a scrolling list.
3. Show attributes - displays the attributes of the selected table in a scrolling list.
4. Showdate: displays all the columns present in a table in a form similar to spreadsheet. the user can scroll through this window to see records.
5. Make query - if user doesn't want to see all columns present in table then the desired attributes can be chosen. Also, a mechanism is provided to add conditions to a query.

The figure 3.1 illustrate the user interface. There are seven push buttons: Connect/Quit, ShowTables, ShowData, Select, Execute, Cancel and Exit as well as three scrolling lists and one scrolling window.

Quit Tables Show Data Select Execute Condition Cancel Ex

Table_Name

NEW_S
DEPT
EMP
BONUS
SALGR

Table_Attr

EMPNO
ENAME
JOB
MGR
HIRED

Table_Data

EMPNO	ENAME	JOB	MGR
1002	CROSS	PROFESSOR	5000
1001	CHANG	PROFESSOR	5000
1000	CARLISLE	PROFESSOR	5000

Query

```
SELECT EMPNO      ,ENAME      ,JOB      ,MGR      ,HIREDATE
```

Fig 3.1

EMPNO	ENAME	JOB	MGR
1000	CARLISLE	PROFESSOR	5000
1001	CHANG	PROFESSOR	5000
1002	CROSS	PROFESSOR	5000
7902	FORD	ANALYST	7566

Fig 3.2

A user begins a search by typing xcli on the command line. Only the connect button and exit button are sensitive, all others are insensitive. When the user is connected successfully to the server, the connect button will become insensitive and the showtables will become sensitive. When a user pushes on the showtable button, a list of all tables will be presented to the user in scrolling list. At this point the showtable button will become insensitive. When a user selects any table from the list, its attributes will be shown in the attribute list. The showdata and select buttons will become sensitive. A user can either see the whole table or tailor a query by clicking on attributes shown in the attributes list and then pressing the execute button. When the user presses the select button, the list of tables becomes insensitive and remains insensitive until the user executes or cancels the query. Conceptual knowledge and mechanical knowledge of a query is included by allowing sensitivity of only those widgets which can generate a valid query. A state diagram of the front end is given below.

State Diagram for Query Assist

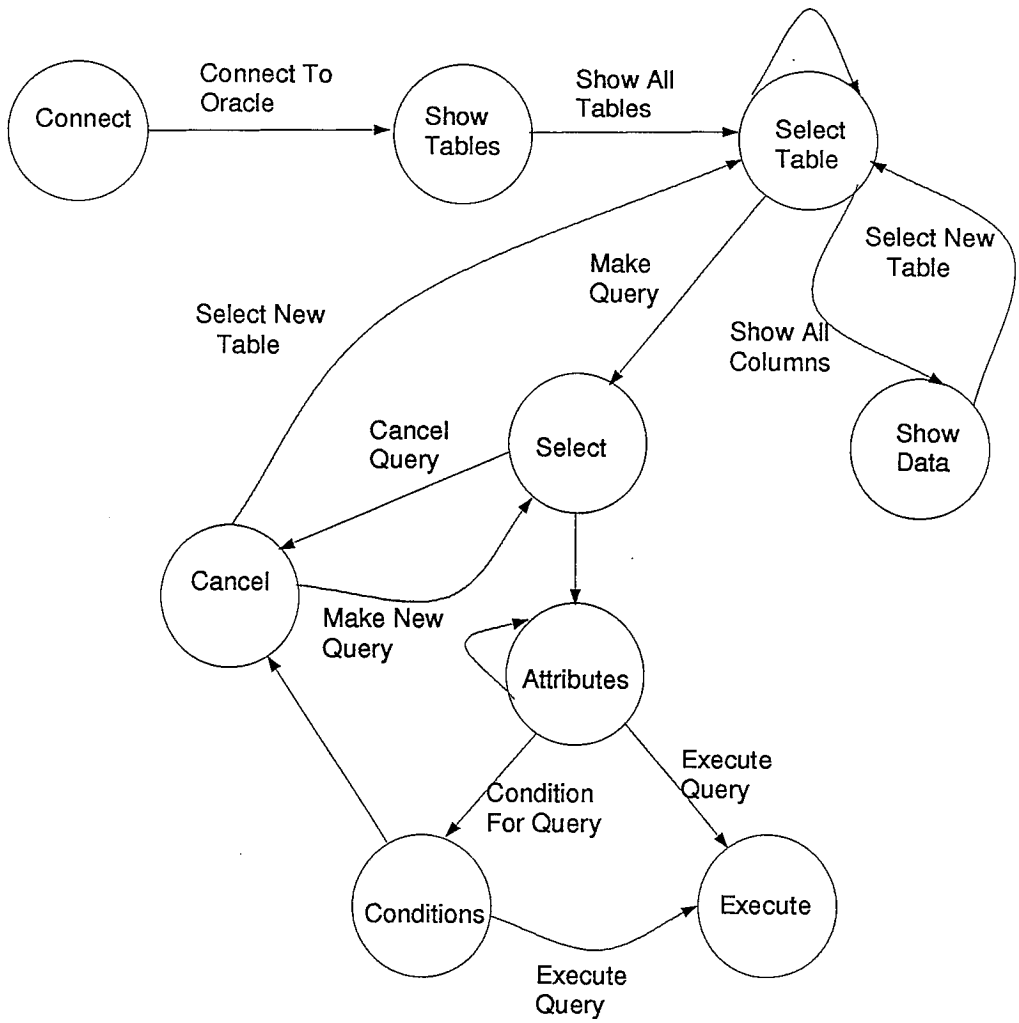

Fig. 3.3

Future directions:

The back end server is slow. A better protocol can be designed to enhance the speed of a back end server.

The front end user interface needs to be redeveloped to include most of the SQL commands, especially the joining of tables. The design of user interface should be intuitive so that user need not to spend lot of time learning how to use system.

REFERENCES

Dillon, Martin. 1991. *Interface for information Retrieval and Online Systems.* New York:Greenwood Press.

Borgman, Charistine L. 1986. "Why Are Online Catalogs Hard to Use? Lesson Learned from Information Retrieval Studies." *Journal of the American Society for Information science* 37: 387-400.

[ORACLE] "Programmers Guide To The Oracle Precompilers", version 1.4.

[ORACLE] "PRO*C Supplement to the Oracle Precompilers Guide", version 1.4.