データベースシステム　94−10
（１９９３．７．２１）

Lessons from Working in a Foreign Culture : A Bridge between Two Research Areas

平賀 瑠美

日本 アイ・ビー・エム 株式会社　東京基礎研究所

リレーショナル・データベース・マネジメント・システムの内部データを視覚化するグラフィカル・ユーザ・インタフェース作成の経験に基づき、データベースとヒューマン・コンピュータ・インタラクションという二つの研究分野の交流についての考察を述べる。作成されたグラフィカル・ユーザ・インタフェースのデザインは、リレーショナル・データベース・マネジメント・システム特有の表現を持つことがわかった。このことから、今後、ユーザビリティの高いグラフィカル・ユーザ・インタフェースを容易に作成するためのモデルを提唱する。

Lessons from Working in a Foreign Culture : A Bridge between Two Research Areas

Rumi Hiraga

Tokyo Research Laboratory, IBM Japan Ltd.

Based on the experience of working in a different culture, we describe how cultural context influences the design of a graphical user interface. Here, we use the word culture to refer to a research area in computer science. The two areas in concern are human computer interactions and database management systems. The work that bridged these two cultures was to make a graphical user interface for a database management system for professional users. With the resulted design, we learned some features in designing graphical user interface. We propose a model which will bridge the two research areas. And we conclude that for elaborating and augmenting the new intercultural research area, researchers from the two research areas should evenly participate.

1 Introduction

Opportunities to work in a foreign culture are becoming more common these days as a result of broader overlaps between research areas and the growth of multinational corporations. The term "an unfamiliar culture" could be replaced by many other expressions. Many books have compared cultures along lines of nationality (e.g. Christopher [4]). A short survey of human-computer interaction (HCI) based on region has been presented by Karat [8].

Recently several conferences have focused on the intercultural relationship of two different research areas. Advanced Visual Interfaces (AVI '94) and the summer workshop of the SIGDB of the Information Processing Society Japan are examples. The ACM SIGCHI's annual conference is also an international conference focusing on interdisciplinary research.

Some new areas have been established by researchers from different research areas; an example is Computer-Supported Cooperative Work (CSCW). A list of research areas involved in CSCW has been published [7]. An important feature of the new interdisciplinary research is the equality of the participating research areas.

In this paper, cultural difference refers to computer science research areas, specifically those of HCI and database management systems (DBMS).

The use of query languages as user interfaces was surveyed in the early '80s by Reisner [12]. With the wide use of window systems, several graphical user interfaces for RDBMSs have been made. Generally, graphical user interfaces (GUIs) in a relational database management system (RDBMS) are used to define tables, to execute queries, and to show the results of queries. Users of these systems are not necessarily database professionals. Outside the academic arena, there are several commercial RDBMS products that support GUIs. Gupta's SQLWindows [6] is an example of such a GUI.

AERIAL [3] and GRAQULA [14] are examples that resulted from academic research. The quarterly review of ACM SIGMOD [1] publishes user interfaces built for database management systems. At the SIGMOD '92 conference, a paper on a GUI of a database was presented [5]. It was a formal description of the model and language, rather than an actual implementation involving users. At a recent SIGCHI conference, a GUI for dynamic queries was described by Ahlberg [2]. An application on a product DBMS was built by Rieman [13]. The target users of these systems were not database professionals.

Generally, DBMS researchers rarely attend SIGCHI conferences and HCI researchers rarely attend SIGMOD conferences. Though attending conferences is not the only way for people to get to know each other, we may be able to say that currently DBMS and HCI researchers have few shared interests.

In this paper, the design and implementation of a GUI for an RDBMS is described from a cultural perspective. The familiar culture is HCI, while the foreign culture is DBMS. The final design was greatly affected by the cultural context.

Through the experience, we learned some design features that could be useful for future construction of GUIs. Some of them seem to be especially applicable to GUIs for professional users. We found some discussable issues that will require researchers from both areas to work together and will help them to understand the other area better, and thus to provide more suitable GUIs for users.

First, we describe the Starburst RDBMS, the system on which our GUI was built. Next, we explain the GUI that was implemented and describe the impact of the cultural context on our design decisions. Finally, we explain the usefulness and importance of interdisciplinary research on DBMS and HCI by proposing a model.

2 Starburst RDBMS

The GUI was built on top of the Starburst extensible RDBMS, which has been developed at IBM's Almaden Research Center. Close to one hundred people have worked on Starburst if visiting scientists from several countries and summer students are included. A paper by Lohman [10] summarizes the latest extensible features of Starburst.

As other research projects on RDBMS, Starburst also focused its research issues on the following:

- Performance of query execution
- Concurrency control
- Distribution of data and control
- Expressiveness of query languages.

The usability is the secondary issue. For example, although a query language is the most common interface for users of a DBMS, the emphasis in the query language research has not been its ease of use.

2.1 SQL Compiler

Users of Starburst can interactively submit queries. The submitted queries are compiled to execute accessing data. In the compilation process, there are several internal structures. After a query has been parsed, it is stored in a structure called the Query Graph Model (QGM). A detailed description of the QGM is given by Pirahesh [11]. The query optimizer of Starburst then refers to the QGM to transform the submitted query into an *execution plan*, which is then used to retrieve data. The query optimizer examines many alternative execution plans and picks the most efficient one for execution by estimating the cost.

The internal structure of an execution plan is a tree whose nodes are made up of primitive database operators, such as scan, sort, and join. Arcs from child nodes to a parent node represent input stream relationships. All nodes have certain pieces of information in common, such as quantifiers (tables), columns, predicates, and costs. The inputs for each primitive database operator differ.

There is a command *print_plan* in Starburst to display the execution plan chosen by the query optimizer in the line mode.

For the following query, seven operators are generated for the execution plan.

$$\begin{aligned}&select\ lastname, firstname, deptname\\ &\quad from\ employee\ e, department\ d\\ &\quad where\ e.empno = d.mgrno\\ &\quad and\ e.workdept = d.deptno \qquad (1)\end{aligned}$$

The tree structure is shown with indentation. The execution plan is interpreted from 115 lines of the output from the print_plan command. On the assumption that users are all developers of Starburst. who know the inside of the Starburst implementation well, some information such as quantifier or columns is provided with numbers corresponding to the internal data values. A more detailed description of the Starburst optimizer is given by Lohman [9]

3 Graphical User Interface

An interactive graphical user interface was built for Starburst with X11/Motif on AIX 3.2. The purpose of the GUI is to show the internal structure of a QGM and an execution plan for a user-submitted query. The GUI is intended to be used by both developers and end users who are RDBMS professionals. They design an RDBMS, implement it, write papers about it, or use it for applications with special requirements. Unless explicitly differentiated, the term "users" refers in this paper to developers and professional end users.

Figure 1 is a graphical representation of the execution plan for the query in the query 1. Basically the GUI for QGM and that for the execution plan have a similar look and feel. Each GUI has two areas. The upper area is for displaying the structure of a QGM or a plan generated by the query optimizer. The lower is a set of buttons for controlling displayed information. In the following, we will describe the details of a GUI for the query optimizer.

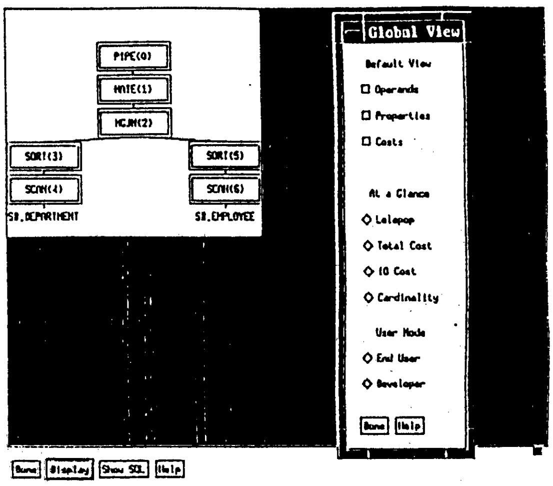


Figure 1: A GUI for an optimized plan

A pop-up window labeled "Global View" in Figure 1 appears when the Display button in the control area is clicked. Information contained in primitive operators (the nodes in a tree) is classified into three parts: operands, output properties, and costs. The Default View selection in the pop-up window lets users specify information to be displayed. The view of the operands shows the arguments of an operator. The view of properties and costs show the estimated results of an operator's execution by the query optimizer. The selection affects all nodes, and users do not need to go through the selection process every time they inspect a node.

Specific information on each node in a tree can be seen by pressing the At a Glance buttons. The operator's name is presented on each node of the tree in Figure 1. Each node can be labeled with the estimated total cost, the I/O cost, or the cardinality.

Different types of user can see the same information on operands and properties in different formats, specified in the User Mode buttons. The information pop-up window in Figure 2 appears according to the selections in the Default View and the User Mode when the user clicks a node in the tree. In this pop-up window, all the information is shown when the developer mode is selected.

Through the Display button from the information pop-up window, a pop-up window labeled "Local View" appears. There are the Default View buttons in the Local View pop-up window. This selection pop-up window was provided to allow users to see different types of information after a particular node has been chosen. A selection made through this window is valid locally for the node.

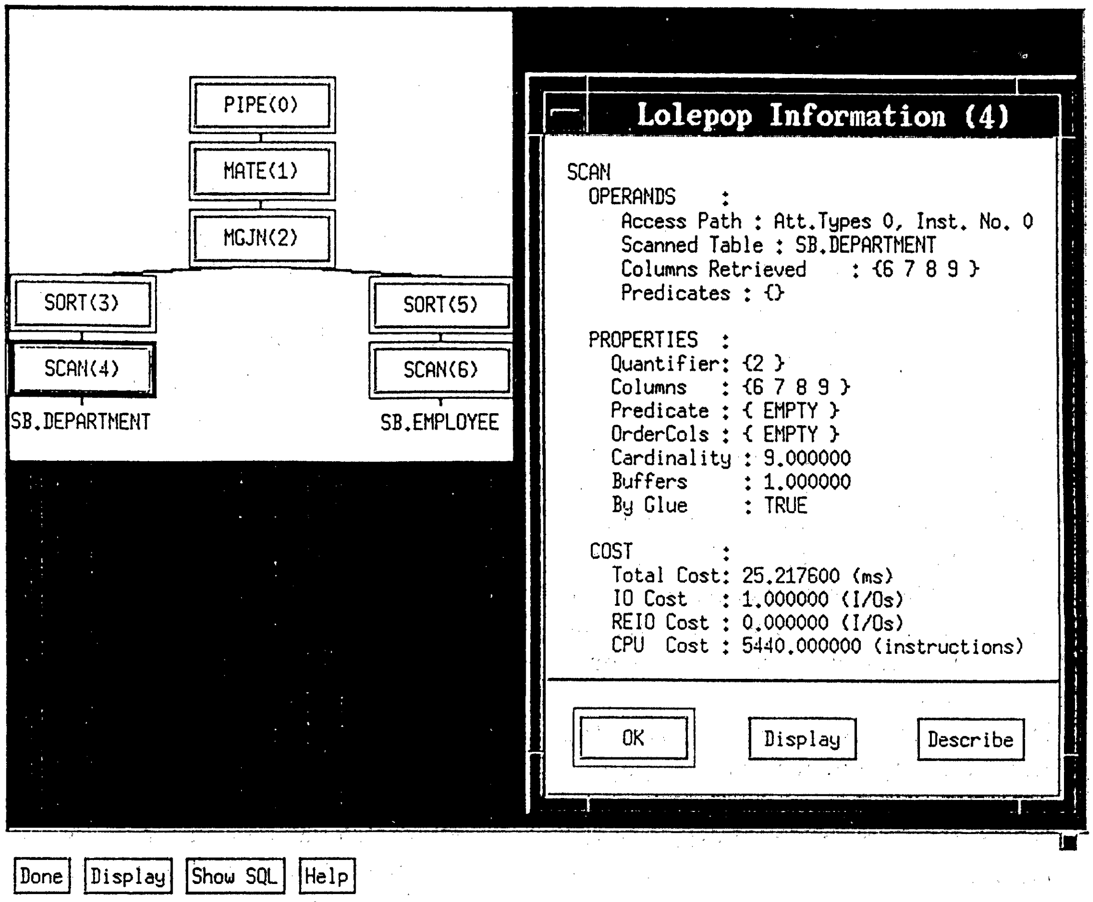
Some operators in the estimated execution plan have two input streams that can be seen as two children. Since each input stream has a specific meaning, the placement of a child node on the right or left child becomes important and imparts a different meaning in each case.

For another submitted query, the execution plan for it consists of 33 nodes all told. This in turn would cause the print_plan command to generate 498 lines of output, making it very cumbersome. In contrast, the GUI is much more usable:

- It reduces the users' effort and the time taken to locate the information for a specific operator. A node in a tree is easier to find in a graphical format than an operator name in a list. In general, the same operator name can appear several times in a query.
- It improves users' understanding of the overall plan structure and relationships between operators.
- It allows users to obtain a specific type of information for all nodes at once.
- It allows users to obtain selective information. Also the two levels of selection (Global View and Local View) gives users the control of the display with less actions.
- It can display the same information in different formats depending on the type of user.

3.1 Design Alternatives and Design Decisions

In this section, design features of the GUI are described along with some of the alternatives that were considered. The context associated with working in an RDBMS greatly influenced the design decisions.



```
SCAN
  OPERANDS   :
     Access Path : Att.Types 0, Inst. No. 0
     Scanned Table : SB.DEPARTMENT
     Columns Retrieved    : {6 7 8 9 }
     Predicates : {}

  PROPERTIES  :
    Quantifier: {2 }
    Columns   : {6 7 8 9 }
    Predicate : { EMPTY }
    OrderCols : { EMPTY }
    Cardinality : 9.000000
    Buffers     : 1.000000
    By Glue     : TRUE

  COST        :
    Total Cost: 25.217600 (ms)
    IO Cost   : 1.000000 (I/Os)
    REIO Cost : 0.000000 (I/Os)
    CPU  Cost : 5440.000000 (instructions)
```

Figure 2: An Information Pop-up Window

3.1.1 Accessing Information

The selection in the Display is globally valid, and users see the same type of information for all nodes. Another set of selection buttons for each node is also provided.

Interaction using only global selection buttons is simpler than interaction using local selection buttons as regards the dialog structure. On the other hand, local selection buttons allow users to concentrate on the values of each operator without the interruption of lengthy mouse movements to global selection buttons if they want to see other information that appeared in the information pop-up window. The particular way of accessing information is determined by the working style of database users.

3.1.2 Category of Displayed Information

Only a small portion of an execution plan's internal structure can practically be displayed. The information selected for display was the same as that produced by the print_plan command. There can be more than ten different pieces of information associated with each operator in an execution plan. Since users are not typically interested in seeing all the information for a node at once, the type of information had to be classified in order to keep the GUI usable.

There were various design alternatives concerning how to classify them in the GUI. If many classes were defined, the displayed information could be very small in quantity and compact, but more interaction with the GUI would then be required of users when more detailed information was desired. In contrast, with fewer classes, users would get more information with less interaction, but the quantity of information could become overwhelming.

Classification of information was influenced by the way in which it is typically used and its meaning. Three categories were chosen as for the Default View selection buttons. It was important to separate out the cost information, since the primary objective of an optimizer is to select an efficient, low-cost execution plan. Note that, depending on the type of a node, some information may appear in both the operands view and the properties view. As shown in Figure 2, for example, column information appears twice.

3.1.3 Tree Direction

As shown in Figure 1, execution plans are displayed vertically. In the original design, they were displayed horizontally, but RDBMS users are used to seeing execution plans displayed vertically in publications, so a vertical display was considered more readable. As explained previously, the placement of a child node at the right or left is also important for users to understand a plan intuitively.

3.1.4 Information for Two Types of User

The information desired by both developers and end users is more or less the same. There is, however, a difference in the way that information is used. Therefore, a selection for different types of user was provided. Depending on the type of user, certain pieces of information are displayed differently. For example,column names and table names are displayed for end users. When a user is a developer, however, column numbers are displayed instead in the information pop-up window. There could be another type of users, such as database administrators. According to the number of user types, the GUI design is modified.

3.2 Lessons

In this paper, we described a GUI that was designed to display query execution plans in the Starburst RDBMS. We showed that the cultural context greatly influenced the design of our GUI.

Through our work on the design, we learned some features of the design of graphical user interfaces that could have general applications or that could be useful in systems for professional users:

- **Providing selection buttons two levels of effectiveness: valid locally and valid globally.** This feature is usable when the same operations are performed on several displayed objects in a window. Locally effective selection buttons must be attached to an object. By freeing users from the need for lengthy mouse movements, this allows them to concentrate on a displayed object.

- **Showing duplicated information for the purpose of completeness of information.** If some phenomena have more than one interpretation, even outputs with the same format can be shown in the same window.

- **Displaying information in different formats from the same source of information.** This can be used to describe one thing at different levels of abstraction. Though the sources of information are different, some debuggers that show both a high-level language and an assembly language are examples of this feature.

We hope that these observations will be helpful in the design and construction of graphical user interfaces, especially for professional users.

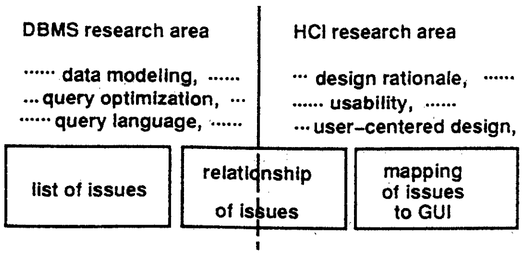

DBMS research area
...... data modeling, ...
... query optimization, ...
...... query language, ...
HCI research area
... design rationale, ...
...... usability, ...
... user-centered design,
list of issues
relationship of issues
mapping of issues to GUI

Figure 3: Meta-Modeller Model

4 A Proposal

Finally we describe a proposal for interdisciplinary research on DBMS and HCI to obtain more sophisticated and useful tools. We built a model based on the implementation of two graphical user interfaces for DBMS specialists. The model is for modeling the image that DBMS specialists expect to see in the graphical user interface and could be called a Meta-Modeller Model (MMM).

MMM consists of three parts (Figure 3). One is for describing the expectations or concept of the users to a GUI. The concept is described as a list of issues. The second is for describing the relationship of each issue, and the third is for mapping the concept to a user interface design. To get an actual GUI, each component of MMM should be instantiated.

The issues vary according to the research areas. For DBMS, for example, they could be

- Visualizing a model. For example, the model of the optimized plan should be drawn as a binary tree.
- User level
- Information retrieval. It is necessary to consider from where the information should be obtained.
- Information filtering. It is necessary to consider how to show the information.

The relationship part is to define the relationship between the issues listed above. The following points should be considered:

- Which issues are orthogonal to other issues. In the GUI for the optimizer, the user level affects the information filtering.
- Nested relationships in an issue. In the above GUI, the information filtering could be specified at two levels, locally and globally.

The mapping part is to decide:

- The layout of a window
- The type of link which should display attached information
- The dialog of a user interface.

Issues will be listed mainly by DBMS researchers and mapping will be done mainly by HCI researchers. Researchers from the two areas may be required to work together on the relationship part, which will affect the usability. There could be many combinations of research tasks and the instantiation of MMM could be different for each combination.

There is a need to provide a graphical user interface for the optimizer and QGM for a broader range of users. We can assume that instantiation of MMM is common through GUIs for DBMS. Thus, once MMM has been instantiated for a combination of DBMS and HCI, it should be useful when another implementation of a graphical user interface for DBMS is tried. MMM is on the way to completion and it could be completed through the collaboration of DBMS researchers and HCI researchers. When MMM is working, many other useful graphical user interfaces for DBMS research could easily be made.

5 Acknowledgement

I would like to thank the Starburst group, especially Guy Lohman, at the Almaden Research Center for much useful advice on building the graphical user interface. Thanks are also due to the management of Tokyo Research Laboratory, especially to Kazuya Kosaka, for supporting this research.

References

[1] Special issue: Advanced user interfaces for database systems. *SIGMOD Record*, 21(1), March 1992.

[2] C. Ahlberg et al. Dynamic queries for information exploration: An implementation and evaluation. In *Proceedings of CHI '92*, pages 619–626, May 1992.

[3] L.M. Burns et al. Aerial: Ad hoc entity-relationship investigation and learning. In *Proceedings IEEE Intl. Conf. Syst., Man, and Cybernetics*, pages 1151–1159, October 1991.

[4] R.C. Christopher. *The Japanese Mind: The Goliath Explained.* Linden Press/Simon & Schuster, 1983.

[5] I. F. Cruz. Doodle: A visual language for object-oriented databases. In *Proceedings of SIGMOD '92*, pages 71–80, June 1992.

[6] Guputa Technologies,Inc. *SQLWindows 3.0*, 1992.

[7] F. Halasz. From the program chair. In *Proceedings of CSCW '90*, pages vii–viii, October 1990.

[8] J. Karat. International perspectives: Some thoughts on differences between north american and european hci. *SIGCHI Bulletin*, 23(4):9–10, October 1991.

[9] G.M. Lohman. Grammar-like functional rules for representing query optimization alternatives. In *Proceedings of SIGMOD '88*, pages 18–27, May 1988.

[10] G.M. Lohman et al. Extensions to starburst: Objects, types, functions, and rules. *Communications of the ACM*, 34(10):94–109, October 1991.

[11] H. Pirahesh et al. Extensible/rule based query rewrite optimization in starburst. In *Proceedings of SIGMOD '92*, pages 39–48, June 1992.

[12] P. Reisener. Query languages. 13(1):13–31, March 1981.

[13] Davies S. Rieman. J. and J. Roberts. A visit to a very small database: Lessons from managing the review of papers submitted for chi '91. In *Proceedings of CHI '92*, pages 471–478, May 1992.

[14] G.H. Sockut et al. Graqula: A graphical query language for entity-relationship or relational databases. IBM T.J. Watson Research Center. Research Report RC 16877, March 1991.