An interface for novice and infrequent database management system users

by JAMES A. LARSON and JENNIFER B. WALLICK

Honeywell, Inc.
Bloomington, Minnesota

ABSTRACT

Special interfaces are needed for novice and infrequent users of database management systems. Such interfaces must remind users of the structure and names of database objects as they guide users in formulating syntactically valid database commands. A prototype system developed at the Honeywell Corporate Technology Center provides such an interface by integrating schema displays depicting the contents and structure of the database and syntax diagrams representing the valid syntactic options of a database query language. By traversing these graphs, novice and infrequent database management system users can easily formulate syntactically valid database management system commands while learning the formal syntax of the database management system command language.
INTRODUCTION

The database management system (DBMS) presents formidable problems to users. New users of a DBMS must be trained to formulate commands acceptable to the DBMS. New users must also learn the structure of the database, the names and relationships of database objects, and which commands to use to access the various objects. Infrequent DBMS users may need to refresh their memories about both what data are in the database and how to formulate commands to access those data objects. An interface to the DBMS is needed for these users that will aid them in learning and relearning how to use the DBMS.

At the Honeywell Corporate Computer Sciences Center we have been investigating approaches to make DBMSs more friendly to novice users. The remainder of this paper describes one such interface under investigation. We first describe the requirements of an interface for novice and infrequent DBMS users and then give an overview of our system.

INTERFACE REQUIREMENTS FOR NOVICE AND INFREQUENT DBMS USERS

Novice and infrequent DBMS users need an interface to a DBMS that guides them in formulating database requests. Requirements of such an interface include the following:

1. Display valid options. When formulating database requests, users should not be required to know or remember the contents of the database, the structure of the database, or the formal syntax of a query language. A system for novice and infrequent users should be designed so that users formulate requests by choosing from a set of syntactically and semantically valid options.

2. Break the problem into subproblems. Novice and infrequent users need assistance in knowing how to tell the computer what information they want from the database. A system for these users should allow piecemeal formulation of database requests so that a user may concentrate on one subproblem at a time.

3. Display current status. Novice and infrequent users should be reminded of what they have accomplished. The system should display what portions of a database request the user has formulated so that the user can decide what options to choose next.

4. Allow users to change their minds. Users should be able to back up to any previous state of command formulation and resume entering options from the state to which they backed up.

5. Permit only syntactically valid commands. A system should be designed so that it minimizes command input errors and permits the user to enter only syntactically valid commands.

6. Provide online help facilities. The system should aid users with additional instructions when users are not sure about a particular option of the system. These instructions should be designed to communicate the meaning of an option with respect to what users have previously accomplished, thereby aiding them in deciding what option to choose next.

7. Move novice users to more advanced interfaces. A system for novice users should provide mechanisms to move users to more advanced and expedient interfaces. In particular, the system should help users learn the syntax of a database language as well as the contents and structure of the database.

8. Control access to a database. Database interfaces should have a mechanism whereby the database administrator can prohibit classes of users from performing various types of operations on selected database objects.

Forms and menus are two database interfaces often used for novice and infrequent users. Unlike forms and menus, the interface developed at Honeywell is an easy-to-use graphical facility for building database commands that trains users to learn the linear keyword form of the language.

SYSTEM OVERVIEW

Syntax diagrams have been used successfully in programming language manuals to illustrate visually the structure of programming languages. A syntax diagram (Figure 1) is a directed graph representing the syntactic structure of a formal language. Any path from the start node to a finish node constitutes a valid statement in the language. Our system uses syntax diagrams to guide users through DBMS commands as well as to teach the more frequent user the syntax of the DBMS language.

A syntax diagram of a query language is a directed graph that contains six types of nodes: (1) start nodes, (2) literal nodes, (3) value nodes, (4) database item nodes, (5) subgraph nodes, and (6) finish nodes.

For implementation purposes these nodes can be distinguished by color and shape; but novice users need not visually distinguish the nodes, because they will be prompted with appropriate instructions and information as each node is selected. Visual distinction of the nodes might be useful for helping users learn the command language.
Start Nodes

Beginning with this node, the user selects nodes, one at a time, along a path in the graph. A complete path specifies a valid request.

Literal Nodes

The names of these nodes correspond to the keywords of the query language. Each time the user selects a literal node, its name is appended to the request being formulated.

Value Nodes

When the user selects a value node, the user is prompted to enter an integer or character string from the keyboard. Value nodes in a query language are used for specifying conditions on attributes of records to be retrieved. The value specified is appended to the request being formulated.

Database Item Nodes

When the user selects a database item node from the syntax graph, the database schema is displayed and the user is asked to select a database object name from the schema. The selected database object name is appended to the request being formulated.

Subgraph Nodes

When the user selects this node, the current syntax graph is placed on a stack, and a syntax graph corresponding to the name of the subgraph node is displayed. This is useful for languages with complex syntax graphs, because it allows syntax graphs to be broken down into a series of subgraph displays.

Finish Nodes

This node indicates that the user has completely specified a path through the currently displayed graph. If the graph is a subgraph, then the parent graph is redisplayed and the user may continue to select nodes from that graph. If the graph was the original starting graph, then the command has been completely specified and is passed to a DBMS for processing.

When a database item node is selected from the syntax graph, a schema describing the classes of objects in the database is displayed on the screen. The schema objects may be displayed as a menu or alternatively as a graph of data objects such as that illustrated in Figure 2. The rectangles in the graph represent classes of entities, and the diamonds represent classes of relationships between entity classes. The ovals represent attributes of entity or relationship classes. This is a graphical representation of Peter Chen's Entity Relationship (ER) data model, a popular style for modeling data to be maintained by a DBMS. The user selects a class of objects from the database schema by positioning the cursor in the appropriate position on the screen or by typing in the name of the desired object at the keyboard.

As the user selects nodes from the syntax diagrams, a linear-keyword-oriented version of the command is constructed in a command window at the bottom of the screen. The user is able to view partially constructed commands as they are being formulated. The user may append keywords, database objects, or values to partially constructed commands either by typing in the keyword, database object name, or value at the keyboard or by positioning the cursor to the desired node on the syntax diagram or database schema display. It is expected that some users will prefer using the cursor and others will prefer entering options at the keyboard. More advanced users will eventually abandon this interface in favor of the more traditional keyboard-only interface.
Previously selected node in the syntax diagram (or backspace to a database object name, keyword, or value in the partially formulated command in the command window) and all database object names, keywords, or values following it will be erased from the command window. Thus a user can undo decisions and back up to any previous state, including the start state. The user may then continue formulating the command, possibly choosing options different from the ones previously entered.

If the user does not understand the meaning of any node or cannot decide which path to take on the graph, the user may move the cursor to the node or edge and press the HELP button. Additional information and messages to help the user will be displayed on the screen.

**EXAMPLE**

Suppose that the user wishes to formulate the database command "GET STUDENT WHERE SECTION.NUMBER 1 = 2's 3 AND COURSE.Number 1 = 2's 177". Further suppose that the user has already formulated the first part of the request, "GET STUDENT WHERE." The syntax diagrams of Figure 1 and the ER graph of Figure 2 are used to aid the user to complete the query formulation. This is illustrated in Figures 3–11. The user may move the cursor only in the "top" (most recently displayed) syntax diagram or database schema. Alternatively, the user may move the cursor in the command window and enter database object names, keywords, and values via the keyboard.
Get student where section number

Figure 6—User selects "section.number" option from database graph

Get student where section number

Figure 7—User selects no more options from arithmetic expression syntax diagram

Get student where section number = 3

Figure 9—User selects "value" option and types "3"

Get student where section number = 3 and

Figure 10—User selects "and" option
Boolean Expression

IMPLEMENTATION

A prototype of this system has been implemented at the Honeywell Corporate Computer Sciences Center in FORTRAN on a Honeywell Level 6 minicomputer interfaced with a Megatek Graphics Terminal. The system supports an interactive mode for building and storing arbitrary syntax diagrams as well as a mode for traversing syntax diagrams with a joystick to construct commands. Nodes can be varying sizes, with a separate color for each type of node.

FURTHER RESEARCH

In this section we investigate ways this system could be used to control access, issues in displaying ER graphs of database schemas, and the automatic creation of syntax diagrams.

Controlling Access

This system could be used to control access to a database by (1) displaying only the part of the schema that describes data the user is allowed to access and (2) modifying the syntax diagrams so that the user is prohibited from executing certain operations (such as delete or modify). We have not investigated a dynamic syntax graph, which would prohibit users from specifying certain selected operations on one part of the schema and other selected operations on another part of the schema.

Displaying ER Graphs

Scrolling both up-down and left-right can partially solve the problem of displaying a schema with a large number of entities in the form of an ER graph. Several other approaches to displaying portions of an ER graph should be evaluated:

1. Optionally turning off the visibility of attributes so that the entity sets and relationships sets can be displayed more densely.
2. Positioning the ER graph nodes so that the graph can be displayed in little space while minimizing the number of arcs that cross each other.
3. Partitioning the ER graph into subgraphs such that the objects in each subgraph are related. This clustering can be based on semantics or on statistical clustering methods.

Automatic Creation of Syntax Diagrams

The syntax diagrams and the Bachus-Naur form (BNF) of the command language are closely related. We feel that it is possible to build software that would convert BNF into equivalent syntax diagrams. However, we feel that such software needs human guidance so that (1) the resulting syntax diagrams are not too large to fit on a screen and not as small as a three-choice menu, (2) each syntax diagram corresponds to a single language concept, and (3) the nodes of the syntax diagram are positioned in an aesthetically pleasing layout.

REFERENCES
