USING NATURAL LANGUAGE DESCRIPTIONS TO IMPROVE THE USABILITY OF DATABASES

Carole D. Hafner
John D. Joyce

Computer Science Department
General Motors Research Laboratories
Warren, MI 48090

ABSTRACT

This paper describes the REGIS extended command language, a relational data language that allows users to name and describe database objects using natural language phrases. REGIS accepts multiple-word phrases as the names of tables and columns (unlike most systems, which restrict these names to a few characters). An extended command parser uses a network-structured dictionary to recognize multi-word names, even if some of the words are missing or out of order, and to prompt the user if an ambiguous name is entered. REGIS also provides facilities for attaching descriptive text to database objects, which can be displayed on-line or included in printed reports. Initial data from a few databases indicate that users choose to take advantage of the naturalness of multi-word descriptions when this option is available.

I INTRODUCTION

The REGIS extended command language is a relational data language that allows users to name and describe database objects using natural language phrases. REGIS [4] is an interactive data management system that has been in use at General Motors since 1975. The system is designed to be easy for non-programmers to understand, and it has given many people their first hands-on experience with computers.

A REGIS database consists of a hierarchical structure of named objects: one or more files, each containing zero or more tables, each composed of zero or more columns of data. REGIS users can create, query, or modify database objects interactively, using simple keyword-based relational commands, such as the following:

```
BLUES = SUBSET TABLE1 WHERE COLOR = BLUE
  (creates a new table from selected rows
   of an existing one)
RESULTS = PROJECTION BLUES TYPE ITEM COST
  (creates a new table from specified
   columns of an existing one)
LIST RESULTS
  (lists a table at the terminal)
```

Future research directions for REGIS are aimed at developing natural language capabilities that will make computer databases easier to understand and access.

The usability of database query languages has been recognized as an important problem (Codd [1], Greenblatt and Waxman [2], Welty and Stemple [5]); however, a closely related issue that has not been addressed is the usability of the data itself. In order to interact with a database effectively, users must be able to understand and refer to the objects in the database. Current database systems restrict the names of database objects to a few characters, which can lead to cryptic abbreviations that are difficult to understand and remember. Documentation facilities (if they exist at all) are not designed to be accessed interactively. The need to refer to external sources for descriptive information, and the need to remember cryptic abbreviations, are obstacles to usability that are especially disruptive to the new or occasional user of a database.

To provide a more supportive environment for data management, a new command interface has been added to REGIS, which accepts multiple-word phrases as the names of tables and columns, and which also provides on-line documentation capabilities. Multiple-word names can be up to 40 characters long, instead of the previous REGIS limit of 8 characters. "Comment" data consisting of descriptive text can be attached to files, tables, or columns. Users can display the comments for parts of the database: e.g., for all the tables in a file, for a particular table, for a table and all of its columns, or for a particular column. Table names, column names, and comments can be created, queried, and changed interactively.

II A FLEXIBLE PROCEDURE FOR NAME RECOGNITION

A straightforward implementation of multi-word names for database objects would not be practical, since it would significantly increase the amount of typing required during command input. Commands would become much longer, leading to slow and tedious interaction, and increasing the number of typing errors. To solve this problem, a flexible recognition procedure is used in REGIS, which recognizes multi-word names even if some of the

Data from Original Version of REGIS

DESIRE	SIXMONTH	TIMEKEEP	MAKESERS	SZPURCH	PLNTOVST
5	5	15	128	18	5
3	3	17	59	13	5

Data from Extended Version of REGIS

DESIRE A CAR?	PROBABILITY OF BUYING WITHIN SIX MONTHS	HOW LONG DO YOU KEEP A NEW CAR?	MAKE AND SERIES YOU PLAN TO PURCHASE	SIZE OF CAR YOU PLAN TO PURCHASE	PLAN VISIT TO DEALER IN NEXT TWO WEEKS?
5	5	15	128	18	5
3	3	17	59	13	5

Figure 1. Comparison of Single-word and Multi-Word Column Names.

words are missing or out of order. Users are able to refer to database objects by specifying any part of the name: for example, if the name of an object is "RESULTS OF FIRST TEST", the user can enter "FIRST TEST", "TEST RESULTS", "FIRST RESULTS", or just "RESULTS", and the object will be located. If an ambiguous name is entered, the user is prompted with a list of choices and asked to select one.

Figure 1 shows part of a REGIS table, for an application that was converted from the original version of REGIS to the extended command version. Each column in the table represents a question that was asked in a survey of consumer attitudes. The table illustrates both the difficulty of finding descriptive abbreviations for data in some applications, and the importance of the flexible recognition procedure to the success of the system (users would be unlikely to use long, descriptive names if they were not able to refer to them more briefly when typing commands).

Flexible recognition of names provides a user-friendly environment for data management, where a user is not required to know the exact names of database objects. If a REGIS user enters the command "LIST SURVEY" and there are several surveys in the database, the system will display the following:

```
"SURVEY" IS AMBIGUOUS.  PLEASE SELECT ONE
ALTERNATIVE:

1 - MARCH 1979 CONSUMER SURVEY
2 - SURVEY OF 100 CHEVROLET OWNERS
3 - JANUARY 1981 CONSUMER SURVEY
!!- CANCEL THIS COMMAND
```

The user may have forgotten the exact name of the table he or she wanted to use, or may not have realized that there were several surveys in the database. If the list of table names does not provide enough information to select the correct one, the user can cancel the command and examine the database further by displaying "comment" data. (See Section IV for a discussion of the comment feature.)

III PARSING COMMANDS WITH MULTI-WORD NAMES

The implementation of flexible name recognition in REGIS has required significant extension of both the relational database schema and the command parser. The schema has been extended to include a network-structured application dictionary, containing all of the words that occur in the user's table and column names. Each word has "TABLE" links connecting it to the tables it describes, and "COLUMN" links connecting it to the columns it describes. A name recognition algorithm (described in Hafner [3]) traverses these links to determine what object the user is referring to. When an ambiguous reference is entered, the algorithm returns a list of potential choices to be displayed.

There are two areas in which the REGIS command parser uses computational linguistic techniques to help it behave more intelligently: in segmenting command strings into distinct parameters; and in restricting the choices for an ambiguous reference. Both of these capabilities depend on the use of a command language grammar, which tells the parser what kind of object it is looking for at each point in the

```
LIST 1979 CONSUMER SURVEY, PERSONAL ECONOMY TODAY, COMMENT

   COLUMN: PERSONAL ECONOMY TODAY
           WHAT IS YOUR PERSONAL ECONOMIC SITUATION TODAY,
           COMPARED WITH WHAT IT WAS ONE YEAR AGO?
             1  MUCH BETTER OFF
             2  SOMEWHAT BETTER OFF
             3  ABOUT THE SAME
             4  SOMEWHAT WORSE OFF
             5  MUCH WORSE OFF
             6  DO NOT KNOW
```

Figure 2. Listing of Comment Data.

parsing process: a table name, a column name, a command name, a keyword parameter from a fixed set, or a numeric parameter. The command language grammar is also used to generate more explicit error feedback than was possible in the previous version of REGIS.

Knowledge of both the command language syntax and the extended database schema is required to determine how the input should be segmented. In ordinary database query languages, segmenting a command string into parameters is not a problem; each word or "token" represents one object. Using multi-word names, however, the system cannot use blanks as delimiters. (Requiring other delimiters, such as commas or semi-colons, was rejected as being too inconvenient for users.) When the command parser is looking for a table or column name, it invokes the name recognition algorithm; when the parser is looking for a REGIS keyword or other value, it reverts to the token processing mode.

In selecting choices for an ambiguous reference, REGIS uses knowledge about both the syntax and the semantics of the command language: in many REGIS commands, a table name appears in one place in the command, and column names from that table appear in other positions. When this occurs, the command parser knows that the column names should only be compared with other columns in the given table; it will not find ambiguities with columns from other tables.

IV CREATING AND DISPLAYING COMMENT DATA

The comment feature of REGIS allows descriptive text to be incorporated into a database and displayed on request. Comments are created and attached to a database object by entering the command that is normally used to create the object, followed by the keyword COMMENT, followed by an unrestricted amount of text. The commands shown below would cause the text following the keyword COMMENT to be attached to a file, a table, and a column, respectively:

```
DEFINE FILE1 COMMENT . . . .
TABLE CHEVY OWNERS COMMENT . . .
COLUMN CHEVY OWNERS, DATE OF LAST
  PURCHASE, COMMENT . . .
```

To display the comments for a database object, the LIST command is used. The commands:

```
LIST FILES
LIST CHEVY OWNERS COMMENT
LIST CHEVY OWNERS, LAST PURCHASE
  DATE, COMMENT
```

would display the comments created by the previous commands. (File comments are listed by default.) Figure 2 shows the comment for one column of the survey database described in Section II. The comment tells exactly what question was asked of the respondents, and shows how their answers were encoded in the database.

V USE OF THE SYSTEM

Both the original version of REGIS and the extended command version are in production use at General Motors. Initial data from a few databases indicate that users choose to take advantage of the naturalness of multi-word descriptions when this option is available. In a sample of applications running on the original version of REGIS, we found that only 35% of the column names were English words, as compared with 93% for the extended version. The average number of words per column name in the extended version was 2.4. (This result may

be biased in favor of English words, since the users of the new version were aware that they were part of an experiment.)

Informal contact with users indicates that the ability to incorporate descriptive comments into a database is a useful feature which contributes to the overall task of information management. Several users of the original version of REGIS have decided to change over to the new version in order to take advantage of the on-line documentation capability.

We expected that the potential for ambiguous references would cause some difficulties (and perhaps objections) on the part of users; however, these difficulties have not occurred. Referring to a database object by a subset of the words in its name is a concept that users understand and are able to manipulate (sometimes rather ingeniously) to create applications that are responsive to their needs.

VI CONCLUSIONS

The REGIS extended command language incorporates natural language descriptions into a user's database in a flexible and easy-to-use manner. The recognition of partly-specified names and the ability to recover from ambiguity are features that are not found in other data management systems.

REGIS does not have the power of a natural language understanding system; syntactic variants of object names will only be recognized if they contain the same words as the original name, and syntactic variants of commands are not supported at all. However, on the positive side, REGIS does not require a linguist or database administrator to explicitly create an application dictionary; the dictionary is created automatically by the system, and is updated dynamically when users add, delete, or rename objects.

The REGIS extended command language required approximately two work-years of effort to develop, much of it devoted to integrating the extended capabilities into the REGIS production environment. The project's goal, to deliver a limited capability for English language description directly into the hands of users, has been accomplished. Future studies of the use of this facility in the production environment will provide feedback on the linguistic habits and priorities of database users.

VII ACKNOWLEDGEMENTS

The REGIS extended command language was originally proposed by William S. Mark, and he contributed substantially to the design of the system.

VIII REFERENCES

1. Codd, E. F., "Seven Steps to Rendezvous with the Casual User." Research Report RJ 1333, IBM Thomas J. Watson Research Center, Yorktown Heights, NY (1971).

2. Greenblatt, D. and Waxman, J., "A Study of Three Database Query Languages." In Databases: Improving Usability and Res 77-97. Edited by B. Schneiderman. Academic Press, NY (1978).

3. Hafner, C., "Incorporating English Descriptions into a Relational Database." Information Systems, Vol. 7 No. 2. (1982).

4. Joyce, J. D. and Oliver, N. N., "REGIS - A Relational Information System with Graphics and Statistics." In Proceedings of the National Computer Conference, pp. 839-844. AFIPS Press (1976).

5. Welty, C. and Stemple, D. W., "A Human Factors Comparison of a Procedural and a Nonprocedural Query Language." Research Report TR 78-24, Computer and Information Sciences Deparment, University of Massachusetts, Amherst MA (1978).