Artificial Intelligence Techniques for Modeling Database User Behavior

Steve Tanner and Dr. Sara J. Graves
Department of Computer Science
The University of Alabama in Huntsville
Huntsville, Alabama 35899

Abstract

This paper describes the design and development of the Adaptive Modeling System. This system models how a user accesses a relational database management system in order to improve its performance by discovering use access patterns. In the current system, these patterns are used to improve the user interface and may be used to speed data retrieval, support query optimization and support a more flexible data representation. The system models both syntactic and semantic information about the user's access and employs both procedural and rule-based logic to manipulate the model.

Introduction

The users of a database management system (DBMS) often repeat particular patterns of usage. If these patterns are known during the design phase of the database, they can be used to structure the data in an efficient manner. Some DBMSs allow users to manually incorporate information about these patterns into the database (e.g. create new views, indexes, etc.). However, very few systems are able to recognize and maintain a model of these patterns for the individual user's benefit.

The Adaptive Modeling System (AMS) is a tool that creates and maintains a model of a user's queries to a DBMS and the relationships between and within those queries. The AMS changes the model to constantly adapt to the way in which the DBMS is currently being accessed by a particular user. As the use of the database changes over time, the AMS is able to monitor and model these changes automatically. Furthermore, intelligent use of the knowledge stored in the model enables the system to recognize patterns and trends. This information can be used by both the users and the administrator of the DBMS.

This paper describes a system that models a user's accesses to a database. This knowledge of the user is maintained in the model's knowledge base (KB) to enhance the user's access to the DBMS. As a user's access to the data changes over time, the system adapts its knowledge base to reflect those changes. The system learns the types of actions that a user is performing and the relationships between the data in the database that those actions imply. The knowledge base models both semantic and syntactic information in a flexible manner. It is manipulated by both procedural and rule-based logic. This use of conventional methods and AI techniques (DBMS and

Knowledge-Based Information and procedural and rule-based logic) creates a powerful combination for extending a database. The system is designed to enhance the use of a SQL-based query language to access a relational database system.

Related Research

The system described in this paper integrates research from several areas including machine learning, object-oriented database design, and entity relationship design.

Much of the research in machine learning has centered around the idea that a system can deduce rules from examples [Michalski 1983] [Michalski 1986] [Winston 1984]. Since the AMS operates while users query a database, it has a ready source of example queries. However, one of the problems with many current learning systems is that the examples must be correct and static (i.e. if it's correct now, it will always be correct). The AMS acquires information that is dynamic and at times incorrect with regard to future information.

Intelligent tutoring systems is an area of adaptive learning that has been explored by the authors. In many of these tutoring systems a changing model of user behavior is established such that as incorrect information is given to them, they must respond accordingly [Visetti 1987] [Woolf 1987]. The tutoring systems also expect the model of the user to change, as the user becomes more skilled at the task being tutored.

The integration of object-oriented techniques and relational systems [Blaha 1988], and using modified types of object-oriented structures in the building of database systems [Chen 1976] are under consideration for the AMS.

Attempts to improve the speed of retrieval of information from databases through the use of knowledge base and semantic query optimization are also of interest to the designers of the AMS. Some current efforts in these areas are being undertaken by [Brunner 1988], [Chakravarthy 1987] and [Malley 1987].

The Basis for the AMS

The AMS is based on the following assumptions:

- When a user accesses data in a database, he does so in a manner that often leaves specific traceable patterns. These patterns represent relationships in the data that are important to the user.
- These patterns can be identified and retained by an intelligent modeling system.
- The AMS can use these patterns to enhance the performance of database systems in a number of ways.

The AMS is aimed specifically at improving the user interface by making query predictions of which the user can take advantage when accessing the database. This

and some of the other improvements that can be realized by using recognized patterns are listed below:

- Improve The User Interface: Knowledge as to what the user is probably about to do can be used to give the user meaningful prompts. This will eliminate keystrokes as well as help the user organize his thoughts.

- Speed Data Retrieval: If the system can accurately predict what data the user is likely to access next, the data can be retrieved before the user needs it.

- Support Query Optimization: Over time, the system can recognize ways in which users access the same information in multiple ways. This information could be used by a query optimization mechanism.

- Allow Flexible Data Representation: The system may be able to reorganize the structure of the data based upon relationships discovered between data. This would allow the system to adapt to new unforeseen uses without explicit user or administrator intervention.

- Support Trend Analysis: Trend analysis is based to a large extent upon recognizing patterns of behavior in a system. Any recognition of these patterns would be of great interest for trend analysis.

When a given user is accessing data in a database, there are often links between the individual data items that form patterns. These data patterns are used extensively when the initial database is being designed (e.g. associated items are often grouped into one table). In order for these patterns to be used however, they must be known when the database is designed. Also, some patterns may come into conflict with one another, and therefore all cannot be incorporated in the initial design.

When users access a particular piece of data, or a particular type of data, they will often take certain actions either prior to or just after the access of that data. If these actions occur often, they will mark a pattern in the use of that data, even when these patterns were not known during the initial database design. In addition, these patterns are likely to shift with time. The way in which a user accesses data may change as the needs of that user change.

The system can identify patterns by looking for repeated behavior that occurs with relation to particular types of data. Since the patterns are likely to change over time, the system can continue to identify new patterns, and discard old patterns that are no longer useful.

Basic Components of the AMS

The AMS is a system that may be used in several different ways and, therefore, it is flexible in its layout and structure. To allow experimentation with different techniques during development, the system is modular in nature. The AMS itself consists of four primary internal components (Figure 1): the AMS Interface, the AMS Control Center (ACC), the Abstract Relation Rule Base (ARRB) and the User Model Knowledge Base (UMKB). Brief descriptions of these components are given below.

The **AMS Interface** is the primary link between the AMS and the User/DBMS. This window-based interface accepts queries from the user and passes them on to the DBMS as well as on to the ACC. The interface can display information to the user from both the DBMS and from the AMS.

The **AMS Control Center** controls the actions of the AMS. It is where decisions are made about what information to display to the user, when to invoke the rule base and how to use the information generated by the AMS components. Since the ACC is the main control center of the system, its operations must be efficient and its interactions with the UMKB strictly procedural in nature. The slower rule-based operations are invoked only on an as-needed basis.

The **User Model Knowledge Base** is used to store knowledge about the users of the database. A model of each individual user is stored in the UMKB and is used to represent how that user accesses the database. The model is represented as tables in a relational DBMS, and consists of four primary levels: The Syntactic level, the Semantic level, the Inter-Query level and the Data level. The first two levels are used to represent syntactic and semantic information about queries of a given user. The last two levels are used to represent more abstract relationships between queries and between data items stored in the user's database.

The **Abstract Relation Rule Base** is used to store rule-based information about the system's model of the user's behavior. It consists of tables that represent the premises and conclusions of the rules and how they should work together. The rule base is generally reserved for the control of more abstract types of modeling and is the primary mechanism for the modification of the Inter-Query and Data levels of the knowledge base.

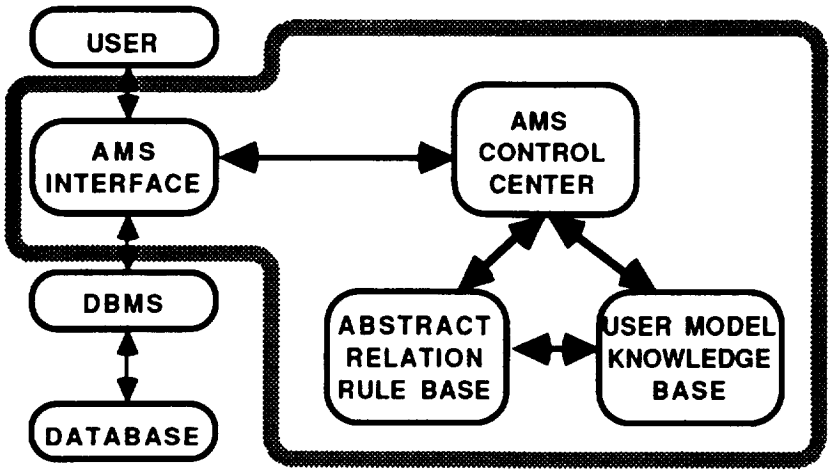

Figure 1: Layout of the AMS

The User Model Knowledge Base

The UMKB is structured as a set of relational tables with three levels of knowledge representation. The Semantic level represents knowledge about interactions within past queries. The Inter-Query level represents knowledge about relationships between separate queries. The Data level represents knowledge about relationships between database items.

In all of the levels, the system uses a unique identification (ID) number associated with each of the user's queries. By tagging information about each item in a query with the query's unique ID before storage, the entire query can potentially be recreated later. The IDs are also used in the precedence of relationships between queries since the IDs are assigned in sequential order. This ordering is exploited when looking for relationships between queries. For example, with the unique IDs, the knowledge base can be searched for such things as "What are the attributes of any relation that was created within the last five queries?" or "Has relation A been modified within the last several queries?" Also the tuples of some tables have weights associated with them. These weights represent how frequently and how recently the particular tuple was used. This information is accessed and updated during the traversal of the tables by the ACC and the ARRB. It is also used to flush old information from the system.

Tuples in the UMKB are subject to being flushed after a period of dormancy. If a tuple is not accessed by the system after an appropriate period of time, it is removed from the system in order to keep the size of the tables manageable. If a tuple is used frequently (i.e. the user takes the same path frequently) the query ID associated with that tuple remains current, and where appropriate, the weights remain high. This information is updated each time the tuple is accessed. If a tuple is not used frequently, the ID becomes old and any weights are decreased. A bookkeeping mechanism of the ACC will periodically eliminate tuples with lower weights.

The Semantic level is a representation of past queries made by the user. It consists of a set of tables that represent the possible SQL queries (CREATE TABLE, CREATE VIEW, CREATE INDEX, SELECT, etc.). Each command is structured as a set of tables that represent the important components of that command. For example, the SELECT command is represented by a set of tables that contain the table names that the SELECT command is operating on, the attributes selected, the ORDER BY clause information and others. Each tuple of a table contains information about one particular query and is tagged with that query's unique ID. By using the ID of a query to select information from the appropriate tables, the query can be recreated in its entirety. This level is used by both the ACC for straightforward traversal as well as by the ARRB for more complex analysis. Since represented queries are broken out into their component parts, the AMS is able to look for patterns that are close but do not necessarily match.

The Inter-Query level is used primarily by the ARRB for making abstract connections between queries. The structure is fairly simple and consists of only one

table. Each tuple of the table represents one relation between queries. Each tuple consists of a pair of query IDs, each with a code that marks the type of query (e.g. SELECT, NESTED SELECT, UPDATE, etc.) and a textual string that designates the relation between the ID pair. The rule system is able to use this table to store and retrieve information about relations between query pairs by either looking for the type of relationship necessary, or by looking for queries related to some specific query (via the query's ID). The primary key of the table includes both of the query IDs as well as the relation type. This allows the AMS to keep track of more than one relation between the same two queries.

The Data Level is also used primarily by the rule base and works in a similar way to the Inter-Query Level. It too consists of only one table. Each tuple consists of a pair of textual attributes that represent some type of data in the relational system (e.g. a table name, column name, etc.), a code that marks the data type of data and another textual string that designates the relation between the pair of data items. The rule system is able to use this table to store and retrieve information about relations between any pair of database entities by either looking for the type of entity or by looking for specific entities (e.g. all entries for TABLE-A).

The ACC and AMS Interface

The ACC is responsible for the control of the other three components of the AMS. Whenever the user performs any sort of action such as entering a word or pressing a function key, the AMS Interface passes this information on to the ACC which performs two primary actions: 1) It invokes the ARRB and gathers the results and 2) It makes any necessary changes to the Semantic level of the UMKB. Once these actions have been taken, the results are examined by the ACC to determine what set of information should be displayed to the user via the AMS Interface.

This information is presented to the user via a series of windows and menus. The user is first presented with a menu containing a list of the potential SQL commands he may choose (SELECT, INSERT, etc). After choosing one of these commands the Interface brings up a set of windows that represent that command. Each SQL command has three types of windows: the Initial window, the Current Choice window, and the Potential Choice window. Figure 2 shows the initial window for the SELECT command. The boxes represent fields that the user is allowed to move the cursor into. Each box represents one component of the command and map directly into the tables that represent the Semantic level of the UMKB. In other words, these boxes represent the major components of a complete SQL command.

When the user moves the cursor into one of these boxes, the system shows the user two additional windows (Figure 3). This first window (the Current Choices window) shows the user the choices he has already made for this component. The second window (the Potential Choices window) show the user the predicted choices that the AMS has chosen. The user may then pick choices from this window, or may type in a component on his own. In either case, the new component is added to the

current choices list and the user's choice is logged into the various locations of the UMKB for future reference. The user is allowed to move to and from any box in the Initial window. This means that the user is allowed to build the SQL command components in any order he wishes. For example he may choose Table names before he chooses Attributes, which is of course not valid in standard SQL.

When the user has built a SQL command, he can then send the command on to the database management system for execution. Since the AMS still retains the information about the last command, the user can then go back and make any changes to this command without having to reenter the entire query. Since the user can enter the components of a SQL command in any order, the AMS can use the ordering to its advantage. For example, if the user enters the Table names before he enters the Attribute names, the system will have more knowledge about what Attributes to predict. Once the user has filled in any appropriate components (enough for a compete SQL command) he may then have the command sent on to the RDBMS for execution.

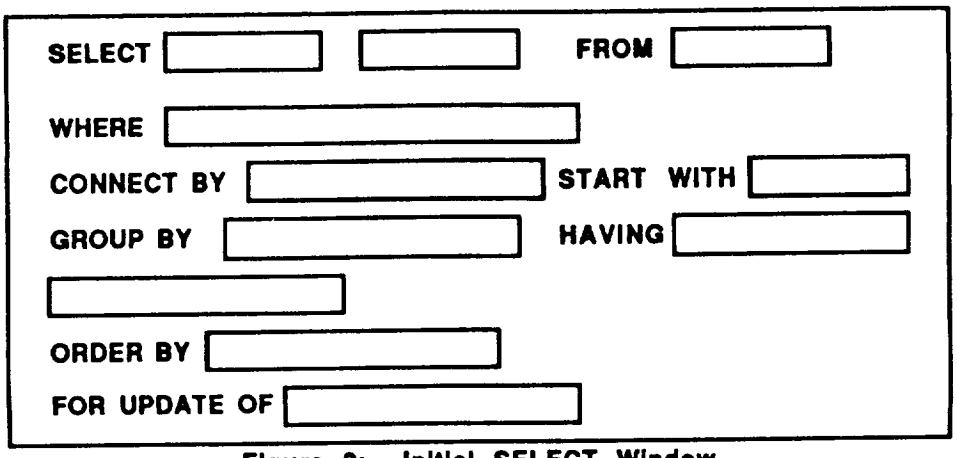

Figure 2: Initial SELECT Window

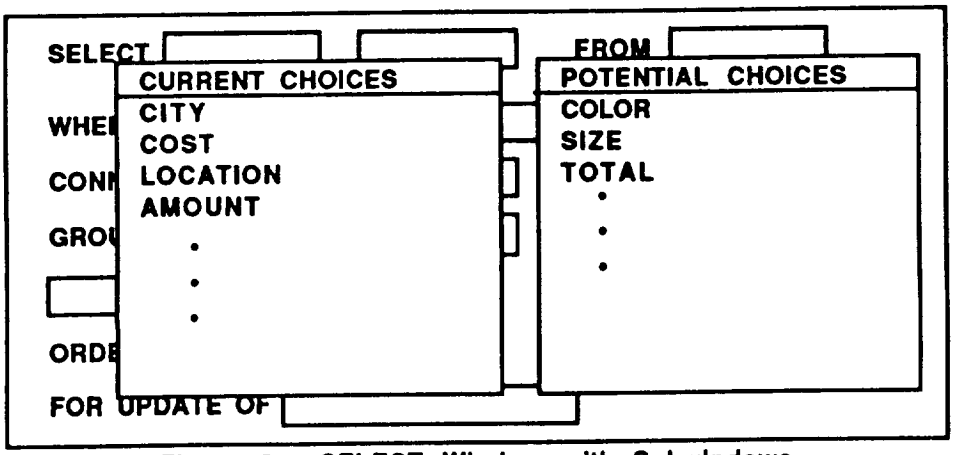

Figure 3: SELECT Window with Subwindows

The Abstract Relation Rule Base

The ARRB is the rule base that does most of the abstract reasoning with the user model stored in the knowledge base. It is based on a set of tables that represent information about the rules and some routines in the ACC that represent when and how the rules are invoked. The ARRB tables contain information about the rules, their premises and their resultant actions. Each rule is represented in three tables: the PREMISE Table, the ACTION Table and the RULE_SET table. In addition, each premise and action of every rule may have a resultant table associated with it.

Since every rule is made up of a set of premises and actions, these groupings are represented in the RULE_SET table. It contains four fields: RULE_NUMBER, PREMISE_ACTION_NUMBER, PREMISE_ACTION_FLAG and LOGICAL_CONNECTION. The primary key is the RULE_NUMBER, PREMISE_ACTION_NUMBER and the PREMISE_ACTION_FLAG. Each rule has a unique rule number and is made up of a set of premises and actions, represented by the PREMISE_ACTION_NUMBER. Premises are distinguished from ACTIONS by the PREMISE_FLAG. In addition, each premise has a logical connection between it and the other premises represented by the LOGICAL_CONNECTION attribute.

In the case of premises, the PREMISE_ACTION_NUMBER is used to reference the PREMISE table. This table is made up of several attributes which represent how the premise should be tested and where the results of that test will be located. The table has the following fields: PREMISE_NUMBER, PREMISE_TRIGGER, PREMISE_RESULT_LOCATION and PREMISE_RESULT_FLAG. The PREMISE_NUMBER is the primary key in the PREMISE table and (if it were supported) is a foreign key on the PREMISE_ACTION_NUMBER in the RULE_SET table. The PREMISE_TRIGGER contains the name of a user routine that should be invoked to do the processing that is required to see if the premise is valid. This routine will be invoked whenever the premise needs to be tested. In other words, whenever the rule system needs to test a given rule, the PREMISE_TRIGGER for each premise contained in that rule will be invoked. Also, the routine will be responsible for flagging either a success or failure in the PREMISE_RESULT_FLAG. If there is an associated table containing further results of the premise test, the routine is also responsible for placing any results in the table. The PREMISE_RESULT_LOCATION attribute contains the name of this table so that other functions may reference the results.

When an event causes the rule set to be invoked, the ACC clears the PREMISE_RESULT_LOCATION and begins firing the premises associated with the rules. It combines the results of the premises in the logical manner defined for each rule. If one of the sets of premises results in a true result, then the actions for that rule are invoked. These actions are executed in the same manner as the premise. The ACTION table contains the following fields: ACTION_NUMBER, ACTION_TRIGGER, ACTION_RESULT_LOCATION and ACTION_RESULT_FLAG. The ACTION_NUMBER is the primary key. Each action associated with a firing rule (the association is defined in the RULE_SET table) is invoked by having the routine name

stored in the ACTION_TRIGGER attribute executed. The routine is responsible for taking any action necessary, and returning a result into the ACTION_RESULT_FLAG and, if appropriate, results into the table defined in the ACTION_RESULT_LOCATION attribute.

The rule structure of the rule base system allows the AMS a great deal of flexibility, although this comes at the expense of speed. The premises and actions of rules are not limited by some specific rule syntax or even a specific language or set of possible events. In addition, if a defined action is used as a premise in more than one rule, it will only be invoked once per rule firing. Furthermore, defined actions can be used as both premises or actions. In the future, this could be the basis for a truly forward or backward chaining system.

The rule base makes extensive use of the UMKB and is allowed to modify both the Inter-Query and the Data levels of the KB structure, but it is only invoked if the ACC determines that it is necessary. In this way, the ACC can retain some level of control over the knowledge base. The rule base is expected to generate any appropriate query component predictions based on heuristic assumptions that are contained in the rules. These predictions are passed to the ACC along with weights that represent the likelihood of the correctness of each prediction. The ACC will compare these predictions with those of its own and choose the most likely candidates to display to the user.

The Use of the AMS Interface and Semantic Level

Before there are any queries made by the user to the database, the AMS Interface is able to represent the "bare bones" information necessary to support the query language syntax, but no semantic information. The three levels of the knowledge base are empty at this point. The tabular structure is there, but the tables contain no tuples.

As a user makes queries to the system, new tuples are added to the tables of the Semantic level, and the weights of the currently existing tuples are modified. As the information in the Semantic level accumulates, the ACC is able to make more accurate predictions. When the user begins a query, the ACC examines the information in the Semantic structure to help fill in the specific information necessary. The information is filled by taking the set of tables that represent the given command (e.g. SELECT, DROP, etc) and looking for patterns that match the current command's structure on a superficial level. For example, it can find what WHERE clauses were used in the past whenever the user selected information from a given column in a given table.

The system first determines what the syntactic layout of the command is likely to be and uses that layout to extract patterns from the Semantic level. Since all tuples have an associated weight, the system can assign a level of confidence in the patterns generated. As mentioned before, these ACC generated components predictions are compared with those generated by the ARRB. The predictions that have the highest weights are then shown to the user.

Conclusions

A prototype AMS system is currently being developed and evaluated. It is being implemented on an IBM compatible PC using the C language and a commercially available relational DBMS [Tanner 1989]. Early results of the user modeling look promising, although speed can be enhanced. The rule set stored in the ARRB is being examined and modified to find the best set for consistent prediction success. In the current system, both the AMS and the DBMS use the same processor. A migration of the system to a dual processor environment in which the AMS has a dedicated processor is planned for the future.

References

[Blaha 1988] Blaha, Michael R., William J. Premerlani, James E. Rumbauch, "Relational Database Design Using An Object-Oriented Methodology", Communications of the ACM, 31, 4, April 1988.

[Brunner 1988] Brunner, K.P., R.R. Korfhage, "An Automatic Improvement Processor for an Information Retrieval System", Proceedings from the Second International Conference on Expert Database Systems, George Mason University, 1988.

[Chakravarthy 1987] Chakravarthy, Upen S., Jack Minker, and J. Grant, "Semantic Query Optimization: Additional Constraints and Control Strategies", Proceedings from the First International Conference on Expert Database Systems, George Mason University, 1987.

[Chen 1976] Chen, P.P. The entity-relationship model: Toward a unified view of data. ACM Transactions on Database Systems 1,1. March 1976.

[Codd 1972] Codd, E.F. "Relational Completeness of Data Base Sublanguages." In Data Base Systems, Courant Computer Science Symposia Series, Vol. 6. Englewood Cliffs, N.J.: Prentice-Hall 1972.

[Date 1986] Date, C.J. *An Introduction to Database Systems*, Volume I, Addison-Wesley, Reading, Massachusetts. 1986.

[Malley 1987] Malley, Christopher V. and Stanley B. Zdonik, "A Knowledge-Based Approach to Query Optimization", Proceedings from the first International Conference on Expert Database Systems, George Mason University, 1987.

[Michalski 1983] Mickalski, R.S., J.G. Carbonell and T.M. Mitchell, *Machine Learning, An Artificial Intelligence Approach*, Morgan Kaufmann, Los Altos, California, 1983.

[Michalski 1986] Mickalski, R.S., J.G. Carbonell and T.M. Mitchell, *Machine Learning, An Artificial Intelligence Approach*, Volume II, Morgan Kaufmann, Los Altos, California, 1986.

[Minsky 1975] Minsky, Marvin, "A Framework for Representing Knowledge," in *The Psychology of Computer Vision*, edited by Patrick H. Winston, McGraw Hill Book Company, New York, 1975.

[Tanner 1989] Tanner, Steve and Sara J. Graves, "*Modeling User's Access to the Oracle Relational Database Management System*", Proceedings of the Oracle International User Week, Dallas, Texas, 1989.

[Visetti 1987] Visetti, Y.M., "Plan inference and student modeling in ICAI", Proceedings of the Sixth National Conference on Artificial Intelligence, Seattle, Washington, 1987.

[Winston 1984] Winston, Patrick Henry, *Artificial Intelligence*, Addison-Wesley, 1984.

[Woolf 1987] Woolf, Beverly, "Intelligent Tutoring Systems: A Survey", Invited talk given at the Sixth National Conference on Artificial Intelligence, Seattle, Washington, 1987.