8-25-1995

Using Expert Systems to Ensure Temporally Consistent Schema Evolution

Susan J. Chinn
Kent State University, schinn@kentvm.kent.edu

Gregory R. Madey
Kent State University, gmadey@synapse.kent.edu

Using Expert Systems to Ensure Temporally Consistent Schema Evolution

Susan J. Chinn
(216) 672-2750 x306
schinn@kentvm.kent.edu

Gregory R. Madey
(216) 672-2750 x348
gmadey@synapse.kent.edu

College of Business Administration
Kent State University
Kent, OH 44242

Problem Statement

As the capacity for storing large amounts of data has grown, managers have been able to consider using temporal databases for problemsolving. Temporal database management systems allow users to access both current and historical versions of data. If a company chooses to use historical data, it may need to consider changes that have occurred in the composition of the conceptual schema throughout the history of the enterprise. Queries to data spanning multiple intervals where the schema has changed should be able to return consistent, nonredundant results. In addition, the timestamps associated with data fields should be interpreted properly so that inconsistent or redundant results are avoided.

An expert system can help enforce consistency of temporal attributes by tracking schema changes and the constraints associated with those schemas. It should also be able to recommend which constraints should be added, changed or removed in response to a change in the schema. Most important, the resulting series of current and past schemas must result in no loss of information. Currently, maintaining constraints to the schema has been the responsibility of the applications programmer. As the size of an historical databases grows, however, it becomes more difficult to track all of the changes and constraints without considerable expertise.

This research in progress has the following objectives:

1. to develop an expert system prototype to detect temporally inconsistent schema changes, and
2. to use the expert system to recommend alternate schema changes if a proposed change results in conflicting or redundant modifications.

Problem Background

As databases change, not only do the information values change, but the schema changes to reflect changing business needs. Navathe and Fry (1976) developed a set of operations that occur in a traditional nontemporal database environment: renaming, compression, expansion, merging, partitioning, instance merging, instance splitting, and inversion. One can think of these operations as occurring in an entity-relationship environment. Renaming refers to changes in attribute and entity names without accompanying changes in basic relationships. Compression refers to the shortening of a hierarchy of relationships between attributes and entities. Expansion, the inverse of compression, occurs when the hierarchical relationship between attributes and entities grows with the addition of new intermediate levels. Merging refers to the compression of relationships at the same hierarchical level. Partitioning, the dual of merging, creates new relations at the same level. In instance merging, two or more groups of instances with the same parent are combined. Instance splitting creates new instances by dividing an old relation into several relations to handle more detailed information. Inversion reverses the hierarchy of relationships so that parents become children and viceversa. These operations can be used in temporal database settings to classify transitions from one schema to the next.

Changes to the schema are temporal in nature, because they reflect the changes to the enterprise's view of the world. Schema changes should therefore result in a truthful state of the enterprise (Dean, 1989). As databases evolve, temporal consistency is measured by temporal completeness, temporal density, and temporal isomorphism (Ariav, 1986). A database that is temporally complete is one in which events in the database completely describe the changing enterprise over the history of the database. If the criterion of temporal density is satisfied, then the state of the database or any object in it can be derived for any time based on the events already recorded therein. Temporal isomorphism mandates a close correspondence between the evolving database and the enterprise that it reflects.

Research using expert systems in schema design and evolution has focused on consistency enforcement and schema refinement in a nontemporal setting. In one research project to develop knowledgebase management systems, the expert system contains rule entities for consistency checking and error handling when a new (empty) schema is generated (Eick and Werstein, 1993). These rule entities model pairs of conditions and actions. The activation patterns associated with each rule entity define the calls to the knowledgebased management system that activate the rule. The expert system captures inconsistencies at the activation level rather than by using the forward chaining strategy of examining the data itself. In another research project, an intelligent database component examines data that is added to the database to detect whether the schema itself should change (Borgida and Williamson, 1986). Features common to a set of objects are used to generate a class description against which exceptions can be measured. In explaining why the exception falls outside of the description or matches the constraints, the expert system can then generalize the information from the exception to aid the data base administrator in adjusting the schema.

Focus of the Research

An expert system component of a knowledge base management system that can manage both current and historical schemas and their related constraints can provide businesses with more consistent data. It also can ensure that queries that span both the current version and past versions of the database will reflect some "reasonable" subset of constraints for the returned result. Reasonableness means that the result of the query yields temporally consistent results. For example, attributes that are present in one schema but are merged in a subsequent schema must be managed so that the user is unaware of the changing state of the enterprise. Reasonableness also applies to the problem domain, which is applicationdependent. The objectives of the research-in-progress are:

1. to develop a prototype of an expert system that checks temporal constraints after a schema change,
2. to provide the user of the expert system with recommendations for changing a schema based on an examination of attribute time-stamps, and
3. to allow the user of the expert system to capture a history of schema changes that fit into a time-line.

The schema operations chosen for the initial prototype are the expansion/compression and the merging/partitioning actions. The knowledge base of the expert system contains the original schema, with time-stamped attributes, and the rules to check for inconsistent time relationships that may result from the operation. For example, suppose an expansion operation occurs which splits a relation, with part of the relation becoming the "child" of the other part of the relation. The time-stamps must be "migrated" to both relations. If, subsequently, a relation is merged with the new parent relation, will time-stamps associated with attributes in the merged relation conflict with each other? Will they conflict with the dependent relation created by the expansion?

The expert system should also be able to examine time-stamps associated with attributes to recommend potential changes to the schema. For example, suppose time-stamps are linked to an attribute in relation A, and the attribute is a foreign key in relation B. If other attributes in relation B have time-stamps that are inconsistent with those in relation A (the time-stamps do not "meet," overlap, or have some other connection), then the relationship may signal a change to the schema.

Finally, capturing the history of schema changes involves capturing the changes in relations and the time-stamps applied to the attributes. Linking the changes to a time-line allows the user to refer to events in the history of the database (Kahn and Gorry, 1977). Thus, the schemas themselves should be time-stamped to allow references to them as entities in themselves. This information would provide a form of documentation to the data base administrator.

Proposed Solution

For each schema change, the expert system will update the time-value associated with the change by modifying the construct that describes the schema to maintain a time-line. Then, it will look at the time-stamps for the attributes affected by each operation, alerting the user to temporal inconsistencies. Upon correction by the user, the expert system can be rerun until 1) no more inconsistencies are detected, or 2) inconsistencies continue to be found, suggesting a need for further changes to the schema.

We are using the C Language Integrated Production System (CLIPS) as the expert system tool. CLIPS is an expert system shell developed at NASA. It has been used for military, government, research, medical, and industry applications (NASA, 1993). Its forward chaining inference engine is suitable for this data-driven problem. After queries are parsed and passed to the knowledge base, the expert system will check the underlying schemas for constraints and coordinate the data retrieval so that inconsistent or redundant results are not returned. As the schema evolves and is replaced with a new definition, the expert system will also check the new constraints for consistency, and if conflicts are apparent, it will recommend an alternate logical design to the data base administrator.

Future research will involve adding and testing the additional schema operations. The expert system tool will then need to be linked to a temporal DBMS. Research efforts are currently underway using CLIPS to induce knowledge from a set of relations so that a reduced set of data can fit into working memory (Guiffrida, 1994). Using a similar approach, we envision our system extracting the set of schema changes from changes in the database relations themselves. We also plan to automate the corrections to temporally inconsistent data in the database.

References

Ariav, G. "A TemporallyOriented Data Model," ACM Transactions on Database Systems (11:4), 1986, pp. 499-527.

Borgida, A., Mitchell, T., and Williamson, K."Learning Improved Integrity Constraints and Schemas from Exceptions in Data and Knowledge Bases," In On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies, M. Brodie and J. Mylopoulos (eds.), New York, SpringerVerlag, 1986, pp 259-286.

Dean, T. "Using Temporal Hierarchies to Efficiently Maintain Large Temporal Databases," Journal of the Association for Computing Machinery (36:4), 1989, pp. 687-718.

Eick, C., and Werstein, P. "Rule-Based Consistency Enforcement for Knowledge-Based Systems," IEEE Transactions on Knowledge and Data Engineering (5:1), 1993, pp. 52-64.

Guiffrida, G. "Expert System Shell to Reason on Large Amounts of Data. Third CLIPS Conference Proceedings, Houston, Johnson Space Center, 1994, pp. 34-42.

Kahn, K. and Gorry, G. "Mechanizing Temporal Knowledge," Artificial Intelligence (9), 1977, pp. 87-108.

NASA. CLIPS Version 6.0 Basic Programming Guide, Houston, Software Technology Branch, Johnson Space Center, 1993.

Navathe, S., and & Fry, J. "Restructuring for Large Databases: Three Levels of Abstraction," ACM Transactions on Database Systems (1:2), 1976, pp. 138-158.