Dealing with Temporal Schema Anomalies in History Databases

N.G. Martin, S.B. Navathe and R. Ahmed

Database Center
Univ. of Florida, Gainesville, FL 32611

Abstract

Current databases do not process temporal information well. Databases containing historical information (history databases) have been proposed to provide more temporal capability by appending new data to relations instead of destructively updating them.

Because history databases do not discard data, they cannot discard outdated database schemas. Thus, in any proposal for a practical history database system, some method must be provided for accessing data from outdated, yet historically valid, database schemas. This paper investigates the problems resulting from restructuring a history database. It then explores different techniques for maintaining multiple schemas. Finally, it presents Schema Temporal Logic (STL) as a means of retrieving data from multiple historically valid database schemas using temporal logic.

1. Introduction

Current databases provide their users with only a snapshot of the universe of discourse. A snapshot is provided because current databases update destructively, replacing old information with new information. Though logs and rollback mechanisms used in transaction processing provide a primitive concept of time, they do not allow the user access to previous data. Destructive updates improve efficiency. Recent improvements in data storage and processing have, however, led researchers to reconsider the possibility of non-destructive updating to provide support for history queries.

History databases provide support for temporal processing through non-destructive updates. [1,2,3,4,5,6] Updates to history databases are achieved by appending new information marked with the time during which that information is valid. By allowing manipulation of time stamps, history databases allow users complex temporal processing. These time stamps give the users access to current information and historical information. Updates for the future can be specified by marking data with time stamps which will become valid at a time following the current time.

Non-destructive updating introduces new problems in a truly evolving database. If the database schema is allowed to change, how should one update the schema non-destructively? This is an open question. If we assume that a history of the schemas is maintained in a manner similar to the history of the data itself, it may place an additional burden on the users in that the users must now retrieve from one or more schemas applicable to their time frame before formulating a query. Moreover, if the query spans schemas, it leads to some even more difficult mapping problems.

Traditional techniques for restructuring databases do not solve these problems. Such restructuring is similar to destructive updating in that the old database schema is discarded. Destructive redesign is possible in static databases because the information stored is current. If more information is needed, it can be collected. A policy of destructive redesign is problematic for history databases. Because historical data is maintained, new information about old items may not be available. Furthermore, the amount of data may be quite large making a transfer impractical.

The approach proposed here involves non-destructive updating of database schemas coupled with a translation mechanism based on temporal logic which provides the user with a coherent virtual schema built on top of the multiple underlying schemas. This method avoids the problems of simple non-destructive changes to the database schema by providing an automatic translation procedure. It avoids the problems of destructive schema change by maintaining old database schemas.

Table 1: Example Database

EMPLOYEE

EmpNo	Sal	Position	Ts	Te
25	20K	Mechanic	10	35
47	32K	Inspect.	23	55
134	35K	Mechanic	45	Now
25	35K	Inspect.	35	Now

AIRPORT

Place	#Hangars
Atlanta	5
L.A.	2
Boston	6
N.Y.	1

MAINTENANCE

SerNo	Part	Place	Ts	Te
91	Wheel	Atlanta	10	20
105	Door	N.Y.	35	47
105	Door	L.A.	55	62
142	Wing	Boston	60	72

2. Examples of Temporal Schema Anomalies

To clarify the issues involved in restructuring history databases let us examine a simple example. This example database contains information pertaining to airplane maintenance. The initial database schema consists of three relations: MAINTENANCE which contains information about the repairs of particular airplanes; EMPLOYEE which contains information about the employees involved in the repairs; and AIRPORT which contains information about the locations at which these repairs take place. This example database is presented in the Temporal Relational Model (TRM) [1], but the issues examined are applicable to other history database models.

2.1. An Overview of the Temporal Relational Model (TRM)

TRM [1] supports a variation of attribute time-stamping within the framework of classical relational database theory to capture the independent behavior of time-varying relations (TVR). TRM is designed to deal with sychronous behavior among attributes in an effective and efficient way.

In this model, every time-varying relation schema has two mandatory time-stamp attributes (TSA's): Time-Start (Ts) and Time-End (Te). These time stamp attributes correspond to the lower and upper bounds of the time interval.

Ts = time from which tuple is effective

Te = time to which tuple is effective.

In a TVR, an attribute value is associated with the time-stamps Ts and Te, if it is continuously valid in the interval [Ts,Te]. The domain of the time-stamps Ts and Te are the integers together with a the variable "Now" which is always equivalent to the current time.

TRM allows both time-varying and non time-varying relations. In TRM, logical time forms an integral part of the time-varying relations, in order to better represent changes in the real world. Logical time refers to the instant of time when an event actually occurred in the real world, allowing successful represention of both proactive and retroactive updates in TRM. Registration time or user-defined time can be incorporated in the model, if needed, for any particular application by defining them as additional time-stamp attributes. Incorporation of double time-stamping in TRM results in a scheme which is both algorithmically and semantically simple.

2.2. Problems with Adding Attributes

Reorganizations which increase the number of attributes in a relation cause serious problems. Because data can no longer be collected for tuples which describe a previous state of the database, the additional information required by the new attributes cannot be collected. Thus it may be impossible to enter historical data into a reorganised schema.

Suppose a new attribute is to be added to a history database. In particular, suppose it is decided to keep the EmpNo of the inspector requesting a repair and the EmpNo of the mechanic effecting that repair in the MAINTENANCE relation. The database administrator has the following options.

(1) Modify the schema of the MAINTENANCE relation into a new schema called. Under this option all the previous data from MAINTENANCE will be transferred over to the new schema. Null values will appear under newly added attributes; if any attributes are deleted, that information will be lost.

(2) Keep the old relation MAINTENANCE with its schema, along with a new relation, MAINTENANCE[2], with its schema, but do not transfer old data into the new schema, thus avoiding the problems in (1). As we shall see, the approach proposed in this paper is a variation of (2) in which a mapping is provided between the old and new schemas.

2.2. Problems with Adding Time Stamps

Because changing a non time-varying relation into a time-varying relation can be accomplished by adding the attributes Ts and Te, adding temporal information to a non time-varying relation is similar to adding an attribute to a time varying relation. Retrieving information from relations which have been expanded by adding temporal information, however, is difficult. The system must be able to retrieve information from outdated relations which contain no time stamps. This is difficult because, though the information contained in the outdated relation is historical, there are no time stamps explicitly encoding this fact. Because the earlier relation is non time-varying, the items in that relation are always current. Because a new, time-varying relation has been added, the items in the new relation are current between time start and time end. Thus there is conflict between the data in the new relation and the data in the old relation which, since it is always current, is current at the same time as the data in the new relation.

2.3. Problems with Adding Relations

Though adding new time-varying relations to a database does not increase the number of attributes in a relation, such additions may still cause problems. These problems arise due to the existence of foreign keys in the relation. When these foreign keys refer to other time-varying relations conflicts may arise.

Adding a new time-varying relation to a database is troublesome when that new relation refers explicitly to other existing relations. For example suppose a new relation stating the assignments of the employees is to be added. This new relation is to be represented by the time-varying relation ASSIGNED as in Table 2. Now the relation could be interpreted as saying that some employees working for the company (such as EmpNo 47) were assigned to no particular airport before the introduction of this relation. The meaning of the relation is unclear because its existence does not extend through the entire life of the database.

Table 2 : ASSIGNED

Empno	Airport	Ts	Te
134	N.Y.	100	Now
25	L.A.	100	Now

In the above discussion we concentrated on the "insertion anomalies" for history database--borrowing the term Codd originally used in [7]. We could similarly discuss deletion anomalies which would refer to the side effects of deleting attributes, deleting time stamps or deleting entire entities or relationships. Also, the problems of updating or changing the definition of attributes, entities and relationships, or of making the time varying relations non-time varying, and vice versa, can be thought of as "update anomalies." Our discussion in the rest of the paper addresses developing an approach to deal with these anomalies.

3. Approaches Dealing with Temporal Schema Anomalies

Any approach to dealing with the problem of accessing historical data from history databases with dynamically changing schemas must provide two things. First, it must provide a coherent view of the data. That is, it must be able to represent old data and new data with equal facility. Second, it must provide a comprehensible view of the data. That is, it must not require the use of a schema which may be beyond the capability of the user to understand.

3.1. The Traditional Approach to Schema Reorganization

Reorganizing history databases is essentially being able to update history database schemas. This update problem is solved in non-history databases through destructive updating. There is only one current schema and updates to this schema entail simply replacing it with a new schema. As shown above, however, destructive schema updating is inadequate for history databases.

Snodgrass and Ahn [2,3] suggest an extension to the traditional approach which in part corrects this deficiency. They propose providing a "rollback" mechanism which allows the user to specify the database state against which a query is to be evaluated. A rollback database records a sequence of database states indexed by transaction time, the time at which the information was entered. By selecting a particular temporal index, a standard relational query can be made against the resulting static database. Because database schemas are in effect for a particular time frame, queries against the static database which result from the effect of the included temporal index are possible. Valid time stamps, as opposed to the temporal index provided by transaction time stamps, record the temporal information associated with a particular record. By including valid time stamps in a rollback database, the result of a query against a particular transaction time will give a full history database. Thus the combination of transaction time and valid time provides a means of dealing with old and new data uniformly.

The difficulty with this approach is its comprehensibility. Users must be familiar with all database schemas in effect for the scope of their queries. For example the simple query "List the serial number of the planes repaired at Atlanta" requires the user to explicitly access both MAINTENANCE and MAINTENANCE[2] and get a union of the information retrieved. Such a query, though possible, could be quite cumbersome. Writing queries will become more difficult as the database schema is restructured because queries must be written to embrace all schemas ever in effect. Our proposed approach reduces the burden on the user by providing a mapping to the appropriate schema for a given user query.

3.2. The Logic Approach

All relational databases are founded on first order logic. Recently, however, there has been a resurgence of interest in the logical foundations of databases [8]. Two views of databases as first order logic theories have been developed: the model theoretic and the proof theoretic. The model theoretic view of databases sees the data as a structure against which the logical formulae representing the integrity constraints and queries are evaluated. The proof theoretic view sees the data as ground atomic formulae, the integrity constraints as axioms in a first order theory, and queries as statements to be proved against this theory [9]. One advantage claimed for such logical views of databases is that they provide support for more real world knowledge including the specification of temporal information. Can such support be extended to the problem of maintaining multiple historically important schemas? We feel that it can.

3.2.1. First Order Temporal Logic

Several first order theories of time have been proposed, a few of these are the situational calculus [10] and the temporal logics of Allen [11] and Hayes [12]. More recently, Kowalski [13] proposes a first order temporal language called the event calculus to deal with the problem of updates in deductive databases. McDermott [14] follows a strategy similar to the one proposed here in the development of his first order logic temporal language. He starts with a modal temporal language and proposes a first order language to express the intention of a modal temporal language. He, however, is interested in a general purpose temporal language; a language applicable in all temporal situations. We propose a special purpose temporal language tailored to the requirements of schema updates.

3.2.2 Modal Temporal Logic

Temporal modal logic grew out of modal logic. Propositions in modal logic can be thought of as pertaining to different worlds and relations between these propositions can be thought of as defining the relationships between these worlds. In different worlds the same proposition may be both true and false. Complete and sound inference strategies for modal logics generate new sentences about a modal structure from a theory which describes that modal structure. [15] Temporal logics are modal logics in which the worlds are generated by the passage of time. Some recent work has concentrated on the use of modal temporal logic as a language for reasoning about the execution of sequences of programs, especially concurrent programs [16,17].

Taking the model theoretic view of databases, modal logics seem admirably suited to the problem of reasoning across schemas. Each schema is represented by a first order theory which is modeled by the data it contains. A language which is to express the relationships between schemas must therefore be able to express relationships between logical theories. Modal logics were designed precisely for this situation. Each theory or schema represents a world and therefore the modal propositions represent relations between theories or schemas.

We have developed a model temporal logic, called Schema Temporal Logic (STL), to deal with temporal schema anomalies. It is based on a temporal logic developed by Manna and Wolper [16] and is, in turn, a variant of one appearing in [18]. The details of this approach are provided in the next section.

4. Details of Schema Temporal Logic

To translate queries made against the current state of the database schema to comparable queries against previous states of the database schema, the temporal logic need only translate queries against a current database into queries against a previous database. Other types of temporal statements are not required in this problem domain. For example, one need not deal with alternative possible past database schemas because they contain no data. In this particular domain the sequence of database schemas forms a total order terminated by the present and the origin. Schema Temporal Logic (STL) is an attempt to design a temporal logic which fits these narrow requirements.

4.1. Temporal Operators

Informally, this temporal modal logic is a logic oriented to reasoning about sequences. It is the classical propositional calculus extended by three temporal operators. These temporal operators are, Prev, All, and Some. Intuitively these operators generate true statements in the following cases:

Prev f iff in the previous world f is true
All f iff in all previous worlds f is true
Some f iff in some previous world f is true.

4.2 Schemas as Modal Logic Worlds

In STL, the states are the schemas which have been generated by changes to the history database. If the history database is viewed as a model theoretic deductive database, it represents a first order theory which reasons about the data contained in the database (the universe of discourse). This universe of discourse, because it is temporal, is a half open interval on a total order generated by the passage of time. The origin of this half open interval is the earliest time in the database.

The schemas included in a history database consist of a set of closed intervals on the total order. The origins and terminations of these intervals correspond to the creation and reorganization times of these schemas. The termination of the last interval is arbitrarily chosen as infinity. Because the schemas represent totally ordered closed intervals, STL need only reason about a discrete situation. Each of these intervals, however, represents a theory about the database; thus the choice of a temporal modal logic for reasoning about this system.

If the database is viewed as a model theoretic deductive database, the queries are proved by retrieving data from the database schema. Such queries can fail incorrectly in a situation which contains more than one schema if the data is present in the database, but is not present in the schema queried. To insure that data is retrieved correctly from all schemas, the query must be translated from its initial form into a form appropriate to the schema containing the data.

4.3. Associating Schema Temporal Logic Formulae with Schemas

In STL, each schema is associated with a set of STL formulae which model the current collection of database schemas. The previous schemas correspond to the worlds in the modal structure about which STL reasons. Queries generated by an STL theory of the current schema are interpreted in the context of the STL theories of the previous schema. The ground clauses indicate the relations contained in the schema; the sentences indicate the relations contained in other schemas.

Queries are made against the set of STL formulae. The relations mentioned in the query are first matched against the ground atomic formulae. If all relations can be found, the query is answered immediately. If some ground atomic formula cannot be found, STL translates a query into one against the previous schema using the sentences. If such a sentence cannot be found, the query fails and STL responds that the query cannot be answered and the names of the relations which cannot be found are returned. This process is repeated for the queries against the previous schemas until the query is answered or no further translation is possible.

4.4. Inference Strategy

STL retrieves relations from previous schemas through a natural deduction principle. The rule used here is that if A must be found, and A -> B, then B must be found. For example, suppose the query requires relation A. If A is a ground clause, the query is answered against the current schema. If, on the other hand, A is not found among the ground clauses, STL searches for a sentences of the form A -> B. If such a sentence is found, the query is translated so that it is a query against B rather than against A. If no such sentence can be found, the query fails and STL responds that B cannot be found.

5. Example of an STL Solution to Temporal Schema Anomalies

To see how STL deals with the history database schema update problem let us examine the example history database. We will show the database represented in table 1 after two different reorganisations and the modal formlae associated with each schema in the reorganised databases.

5.1 Example Database after First Reorganization.

Suppose the database in table 1 has been reorganized at time $t = 100$ to produce the database in table 3. Three reorganizations have taken place. Two new attribute has been added to MAINTENANCE producing a new relation also called MAINTENANCE; a new relation, AIRPLANE, has been introduced; and a new relationship, ASSIGNED, has been added. The new database schema (Table 3B) holds current information; the previous schema (Table 3A) holds the old data which cannot be included in the new schema. For example, the data in EMPLOYEE gets carried over to the next schema, whereas data from the old MAINTENANCE relation does not. All of the time stamps in the old schema which read "Now" are changed to "100" at the time of reorganisation. In general, to "close" a database schema all "Now's" in the old schema will be computed and actual values stored. New data is inserted in the current schema (Table 4).

Associated with the new database schemas are a sets of modal formulae. These formulae tell how the information in this schema relates to the previous schemas. The set of modal formulae associated with this database is as shown in table 5. The modal formula associated with the previous schema is shown in table 5A; the formulae associated with the current schema are shown in table 5B.

Table 4: DB Some Time After Reorganization

A. Relation According to the Previous Schema

MAINTENANCE

SerNo	Part	Place	Ts	Te
91	Wheel	Atlanta	10	20
105	Door	N.Y.	35	47
105	Door	L.A.	55	62
142	Wing	Boston	60	72

B. Relations According to the Current Schema

EMPLOYEE

EmpNo	Sal	Position	Ts	Te
25	20K	Mechanic	10	35
47	32K	Inspect.	23	55
134	35K	Mechanic	45	Now
25	35K	Inspect.	36	Now
156	40K	Inspect.	100	120
156	45K	Inspect.	121	Now

AIRPORT

Place	#Hangars
Atlanta	5
L.A.	2
Boston	6
N.Y.	1

MAINTENANCE

SerNo	Part	Place	Inspect.	Mechanic	Ts	Te
105	Door	N.Y	25	134	111	113
97	Wheel	L.A.	156	134	124	132
105	Door	N.Y.	25	134	152	162

AIRPLANE

SerNo	Type	Ts	Te
97	747	100	143
105	727	100	Now
127	747	100	Now

ASSIGNED

EmpNo	Place	Ts	Te
134	N.Y.	100	Now
25	N.Y.	100	Now
156	N.Y.	100	120
156	L.A.	120	Now

Table 3: DB Immediately After Reorganization

A. Relation According to the Previous Schema

MAINTENANCE

SerNo	Part	Place	Ts	Te
91	Wheel	Atlanta	10	20
105	Door	N.Y.	35	47
105	Door	L.A.	55	62
142	Wing	Boston	60	72

B. Relations According to the Current Schema

EMPLOYEE

EmpNo	Sal	Position	Ts	Te
25	20K	Mechanic	10	35
147	32K	Inspect.	23	55
134	35K	Mechanic	45	Now
25	35K	Inspect.	36	Now

AIRPORT

Place	#Hangars
Atlanta	5
L.A.	2
Boston	6
N.Y.	1

MAINTENANCE

SerNo	Part	Place	Inspect.	Mechanic	Ts	Te

AIRPLANE

SerNo	Type	Ts	Te

ASSIGNED

EmpNo	Place	Ts	Te

Table 5: Modal Formulae Associated Example DB

A. Modal Formula Associated with Prev. Schema

(1) MAINTENANCE (SerNo, Part, Place, Ts, Te)

B. Modal Formulae Associated with Curr. Schema

(1) EMPLOYEE (EmpNo, Sal, Position, Ts, Te)

(2) AIRPORT (Place, #Hangars)

(3) MAINTENANCE (SerNo, Part, Inspector, Mechanic, Ts >= 100, Te)

(4) AIRPLANE (SerNo, Type, Ts >= 100, Te)

(5) ASSIGNED (EmpNo, Type, Ts >= 100, Te)

(6) MAINTENANCE(SerNo, Part, Place, Ts, Te < 100)
->
Prev MAINTENANCE(SerNo, Part, Place, Ts, Te)

5.2. Mapping Queries of the Example Database after First Reorganization

To process a query involving a relation with added attributes, the modal formulae translate the query against the current database into queries against the current database and previous databases. For example the query:

```
SELECT SerNo FROM MAINTENANCE
        WHERE Place = "Atlanta"
```

would be translated into two queries: one against the first example database; one against the second. (see [19] for a discussion of the TSQL language.) The new translated query would be:

```
SELECT SerNo FROM Prev MAINTENANCE
        WHERE Place = "Atlanta"
UNION
SELECT SerNo FROM MAINTENANCE
        WHERE Place = "Atlanta".
```

To perform this transformation STL first looks in the ground atomic formulae to see if there is a relation called MAINTENANCE from which the attribute SerNo" can be retrieved. Formula 3 provides such a relation, but it is restricted to queries which specify a time start of greater or equal to 100. Because no time start is specified in the query being processed, a time start of 0 is assumed. Thus STL looks for a formula of the form A -> B where A is a relation named MAINTENANCE from which a attribute called "SerNo" can be retrieved. Formula 6 is such a formula, so STL queries from the MAINTENANCE relation found in the previous schema. The query against the previous schema can be answered because it matches the ground atomic formula in the set of modal formulae for that schema.

Queries which retrieve information only from the current schema or only from previous schemas can also be handled. For example,

```
SELECT SerNo FROM MAINTENANCE
        WHEN Te < 100
```

retrieves information only from the previous MAINTENANCE relation. On the other hand,

```
SELECT SerNo FROM MAINTENANCE
        WHEN Ts > 100
```

retrieves information only from the current MAINTENANCE relation.

To retrieve information from relations which have been reorganized by adding temporal information, the schema from which they are retrieved must be returned with the tuple. In that way data from old schemas without time stamps will not be confused with data from the current schema for which the time stamps were not requested. Some measure of ambiguity will result from such retrievals due to the lack of information in the earlier schemas. The user will know that the earlier information was current during a certain time frame, but will not know the precise boundaries of that currency.

Queries which generate negative propositions will return "unknown" with the negation of the negative proposition as a parameter. For example if presented with the query

```
SELECT SerNo FROM AIRPLANE
        WHEN Ts = 50,
```

the system can find the required relation in neither the ground clauses, nor the sentences which translate the query to one against another relation. Therefore the system will return "unknown(AIRPLANE(Ts >= 0,Te < 100))".

Similarly, queries involving added relationships may return "unknown". As in the above example, the query

```
SELECT Sal FROM EMPLOYEE
        WHERE EmpNo =
        SELECT EmpNo FROM ASSIGNED
                WHEN Ts = 50
```

will also return "unknown". Using the information encoded in the modal formulae, however, the system can give an indication as to why the information is unknown. Here, the system will return "unknown(ASSIGNED(Ts >= 0,Te < 100)).

5.3 Example Database after a Second Reorganization

Suppose now that at time t = 200 the database represented in table 4 has again been reorganized. The only changes to the database are the deletion of the attribute, Place, from the MAINTENANCE relation, and the removal of the relation, ASSIGNED. Because none of the other relations are changed, they are all moved to the new current schema.
he new database will appear as in table 8C. The two previous databases will appear as in tables 6A and 6B. The database in 6A is not changed; the database in 6B is the current database from table 4.

Because the database represented in table 6B has been changed, the modal formulae associated with this database also need to be changed. The formulae are altered by moving forward all formulae associated with the relations in the current schema. The formulae associated with relations which are not moved forward (i.e., those designated as being in the previous schema) are not changed. The modal formulae associated with the schemas created by the second reorganization of the example database are shown in table 7.

Table 6: Database Immediately After Second Reorganization

A. Relation According to the First Schema

MAINTENANCE

SerNo	Part	Place	Ts	Te
91	Wheel	Atlanta	10	20
105	Door	N.Y.	35	47
105	Door	L.A.	55	62
142	Wing	Boston	60	72

B. Relations According to the Previous Schema

MAINTENANCE

SerNo	Part	Place	Inspector	Mechanic	Ts	Te
105	Door	N.Y	25	134	111	113
97	Wheel	L.A.	156	134	124	132
105	Door	N.Y.	25	134	152	162

ASSIGNED

EmpNo	Place	Ts	Te
134	N.Y.	100	Now
25	N.Y.	100	Now
156	N.Y.	100	120
156	L.A.	121	Now

C. Relations According to the Current Schema

EMPLOYEE

EmpNo	Sal	Position	Ts	Te
25	20K	Mechanic	10	35
47	32K	Inspector	23	55
134	35K	Mechanic	45	Now
25	35K	Inspector	36	Now
156	40K	Inspector	100	120
156	45K	Inspector	120	Now

AIRPORT

Place	#Hangars
Atlanta	5
L.A.	2
Boston	6
N.Y.	1

MAINTENANCE

SerNo	Part	Inspector	Mechanic	Ts	Te

AIRPLANE

SerNo	Type	Ts	Te
97	747	100	143
105	727	100	Now
127	747	100	Now

Table 7: Modal Formulae Associated Example Database

A. Modal Formula Associated with First Schema

(1) MAINTENANCE (SerNo, Part, Place, Ts, Te)

B. Modal Formulae Associated with Previous Schema

(1) MAINTENANCE (SerNo, Part, Place, Inspector, Mechanic, Ts >= 100, Te)

(2) ASSIGNED (EmpNo, Type, Ts >= 100, Te)

(3) MAINTENANCE(SerNo, Part, Place, Ts, Te < 100)
->
Prev MAINTENANCE(SerNo, Part, Place, Ts, Te)

C. Modal Formulae Associated with Current Schema

(1) EMPLOYEE (EmpNo, Sal, Position, Ts, Te)

(2) AIRPORT (Place, #Hangars)

(3) MAINTENANCE (SerNo, Part, Inspector, Mechanic, Ts >= 200, Te)

(4) AIRPLANE (SerNo, Type, Ts >= 100, Te)

(5) ASSIGNED (EmpNo, Type, Ts, Te < 200)
->
Prev ASSIGNED (EmpNo, Type, Ts, Te)

(6) MAINTENANCE(SerNo, Part, Place, Inspector, Mechanic, Ts, Te < 200)
->
Prev MAINTENANCE(SerNo, Part, Place, Inspector, Mechanic, Ts, Te)

5.4. Mapping Queries on the Example Database after Second Reorganization

Now the query:

```
SELECT SerNo FROM MAINTENANCE
        WHERE Place = "Atlanta"
```

must be translated into a union of three queries: one against the first database, one against the second, and one against the current database. This query will be:

```
SELECT SerNo FROM Prev Prev MAINTENANCE
            WHERE Place = "Atlanta",
    UNION
    SELECT SerNo FROM Prev MAINTENANCE
            WHERE Place = "Atlanta"
    UNION
    SELECT SerNo FROM MAINTENANCE
            WHERE Place = "Atlanta".
```

STL performs this translation by first looking at the ground atomic formulae of the current schema. The query can be partially answered from the current schema, so the current schema is queried. The rest of the information is retrieved by querying against the previous schema. This new query is again translated by the modal formulae associated with the previous schema into two queries: one against its previous schema (Prev) and one against the first schema (Prev Prev). Because the first schema can completely satisfy the query it receives, no further translation is necessary.

Conclusion

The clear conclusion to draw from this work is that including temporal information in a database through a policy of non-destructive updates will introduce difficulties in an evolving situation requiring database schema updates. This paper has attempted to explore these difficulties and has suggested a possible solution to them.

Modal temporal logic seems to be a promising candidate for solving the problems that occur when temporal schemas are updated. Because modal logics describe collections of worlds with well defined interactions, they appear to offer semantics tailored to the problem of retrieving information from multiple schemas. Each schema represents a single world within the modal structure about which the modal logic reasons. Thus, the reasoning required to determine whether information need be retrieved from another schema can be divorced from the logic required to retrieve information within the schema.

Acknowledgments

The authors would like to thank Alex Papachristidis for many helpful discussions.

References

[1] S.B. Navathe, and Rafi Ahmed. "A Temporal Relational Modal and a Query Language", UF-CIS Technical Report TR-86-16, April 1986.

[2] Richard Snodgrass and Ilsoo Ahn. "Temporal Databases", Computer, 19:2, September 1986, pp. 35-42.

[3] Richard Snodgrass and Ilsoo Ahn. "A Taxonomy of time in Databases," Proc. Int'l conf. Management of Data, ACM SIGMOD, Austin, TX, May 1985, pp. 136-246.

[4] Ariav, Gad, Aaron Beller, and Howard L. Morgan. "A Temporal Data Model" New York Univerisity, December 1984.

[5] Clifford, James and Abdullah Uz Tansel. "On an Algebra for Historical Relational Databases: Two Views" ACM SIGMOD Record 1985, pp. 247-265.

[6] Gadia, Shashi K. "Toward a Multihomogeneous Model for a Temporal Database" IEEE Conf. on Data Engineering 1986, pp. 390-397.

[7] E. F. Codd. "A Relational Model of Data for Large Shared Data Banks", CACM, 13:6, June 1970.

[8] Herve Gallaire, Jack Minker, and Jean-Marie Nicolas. "Logic and Databases: A Deductive Approach", Computing Surveys, 16:2, June 1984.

[9] Raymond Reiter. "Towards a Logical Reconstructions of Relational Database Theory", in On Conceptual Modeling, M. Brodie, J. Mylopoalos, and J. Schmidt (eds.), (Springer-Verlag, 1985).

[10] J. McCarthy and P. J. Hayes. "Some Philosophical Problems from the Standpoint of Artificial Intelligence", in Machine Intelligence 4, (Edinburg: Edinburg University Press, 1969).

[11] James Allen. "Maintaining Knowledge about Temporal Intervals", CACM, 26:11, November 1983, pp. 832-843.

[12] P. J. Hayes "The Second Naive Physics Manifesto", in Formal Theories of the Common Sense World, (Norwood: Ablex Publishing, 1985) pp. 1-36.

[13] Robert Kowalski. "Database Updates in the Event Calculus", Imperial College, July 1986.

[14] Drew McDermott. "A Temporal Logic for Reasoning About Processes and Plans", Cognitive Science 6, 1982, pp. 101-155.

[15] J. Jay Zeman Modal Logic: The Lewis-Modal Systems (London: Oxford University Press), 1973.

[16] Zohar Manna and Pierre Wolper. "Synthesis of Communicating Processes from Temporal Logic Specifications", ACM Transactions on Programming Languages and Systems, 6:1, January 1984, pp. 68-93.

[17] E. M. Clarke, E. A. Emerson, and A. P. Sistla. "Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications", ACM Transactions on Programming Languages and Systems, 8:2, April 1986, pp. 244-263.

[18] D. Gabbay, A. Pneuli, S. Shelah, and J. Stavi. "On the Temporal Analysis of Fairness", Proc. 7th ACM symposium on Principles of Programming Languages, Las Vegas, January 1980, pp. 163-173.

[19] S.B. Navathe, and R. Ahmed. "TSQL--A Language Interface for Temporal Databases", Proc. of Temporal Aspects of Information Systems, North-Holland, May 1987.