Raise The Temporal Database From Architecture

Lalit Kumar Saini1, Vishal Shrivastava2
M.Tech1 Research Scholar, Professor2
Department of Computer Science and Engineering,
Arya College of Engineering. & Information Technology,
Jaipur, India

Abstract:

A wide range of database applications are manage the time-varying data. In existing database technology gives the little support for managing such a time-varying data. The research area for temporary databases aims to change the state of affairs by characterizing the of temporal data and providing large and effective ways to model, store, and query the temporal data. This paper provides a brief introduction to temporal database research system. It concisely introduces fundamental temporal database concepts, surveys state-of-the-art solutions to challenging aspects of temporal data management, and also offers a look into the future of temporal database research.

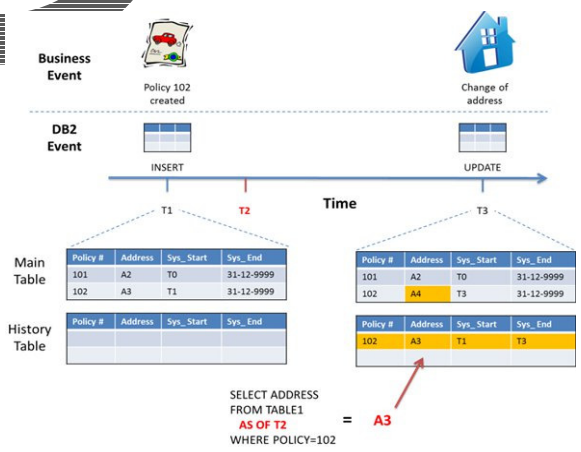
1 Introduction

Most applications of database technology are temporal in nature. Examples in- clude financial applications such as portfolio management, accounting, and bank- ing; record-keeping applications such as personnel, medical-record, and inven- tory management; scheduling applications such as airline, train, and hotel reser- vations and project management; and scientific applications such as weather moni- toring. Applications such as these rely on temporal databases, which record time-referenced data.

Temporal database management is a vibrant field of research, with an active community of several hundred researchers who have produced some 2000 papers over the last two decades.

Temporal Data Semantics:

Before we proceed to consider temporal data models and query languages, we ex- amine, in data model-independent terms, the association of times and facts, which is at the core of temporal data management. Initially, a brief description of terminology is in order. A database models and records information about a part of reality, termed either the modeled reality or the mini-world. Aspects of the mini-world are represented in the database by a variety of structures that we will simply term database entities. We will employ the term "fact" for any (logical) statement that can meaningfully be assigned a truth value either true or false. In general, times are associated with database entities.



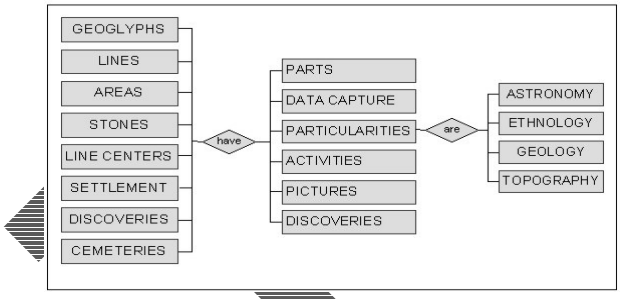
Temporal Data Models and Query Languages

Temporal data management can be very difficult using conventional(non-temporal) data models and query languages [54]. Accommodating the

time-varying nature of the enterprise is largely left to the developers of database applications, leading to ineffective and inefficient ad-hoc solutions that must be reinvented each time a new application is developed. The result is that data management is currently an excessively involved and error-prone activity.

Temporal Data Models:

The first step in providing support for temporal data management is to extend the database structures of the data models supported by a conventionalDBMS. Assum- ing a relational data model, mechanisms must be provided for capturing the valid and transaction times of the facts recorded by the relations, leading to temporal relations. Adding time to the relational model has been a daunting task, and more than two dozen extended relational data models have been proposed Most of these models support valid time only; some also support transaction time. We will con- sider three of these latter models and related design issues. As a simple example, consider a video store where customers, identified by a CustomerID attribute, rent video tapes, identified by a TapeNum attribute. We consider a few rentals during May 1997. On the 2nd of May, customer rents tape for three days. The tape is subsequently returned on the 5th. Also on the 5th customer rents tape with an open-ended return date. The tape is eventually returned on the 8th. On the 9th, customer rents tape to be returned on the 12th. On the 10th, the rental period is extended to include the 13th, but this tape is not returned until the 16th. The video store keeps a record of these rentals in a relation termed. This data modeltimestamps tuples, corresponding to facts, with values that are sets of (transaction time, valid time) pairs, captured using attribute figure. provides a graphical illustration of the three timestamp values, which are termed bitemporal elements. In the general case of infinite and continuous time domains, these are finite unions of rectangles in the two-dimensional space spanned by transaction and valid time.



Adding Time to Query Languages:

Given the prevalence of applications that currently manage time-varying data, one might ask, why is a temporal query language even needed? Is the existence of all this SQL code of ostensibly temporal applications not proof that SQL is sufficient for writing such applications? The reality is that in conventional query languages like SQL, temporal queries can be expressed, but with great difficulty.

Designing Temporal Databases:

The design of appropriate database schemas is critical to the effective use of data- base technology and the construction of effective information systems that exploit this technology. Database schemas capturing time-referenced data are often partic- ularly complex and thus difficult to design. The first of the two traditional contexts of database design is the data model of the DBMS to be used for managing the data. This data model, generally a variant of the relational model, is assumed to conform to the ANSI/X3/SPARC three-level architecture.

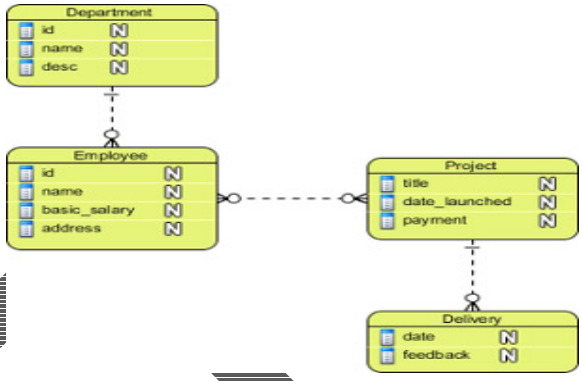
In the second context, a database is modeled using a high-level, conceptual design model, typically the Entity-Relationship model. This model is independent of the particular implemen- tation data model that is eventually to be used for managing the database, and it is designed specifically with data modeling as its purpose, rather than implementation or data manipulation, making it more attractive for data modeling than the variants of the relational model. Mappings are assumed available that bring a conceptual design into a schema that conforms to the specific implementation data model of the DBMS to be used. We proceed to consider in turn logical and conceptual design of temporal databases.

Logical Design:

A central goal of conventional relational database design is to produce a database schema consisting of a set of relation schemas. In normalization theory, normal forms constitute attempts at characterizing "good" relation schemas, and a wide variety of normal forms has been proposed, the most prominent being third normal form and Boyce-Codd normal form. An extensive theory has been developed to provide a solid formal footing for relational database design, and most database textbooks expose their readers to the core of this theory. In temporal databases, there is an even greater need for database design guide- lines. However, the conventional normalization concepts are not applicable to tem- poral relational data models because these models employ relational structures dif- ferent from conventional relations. New temporal normal forms and underlying concepts that may serve as guidelines during temporal database design are needed.

Conceptual Design:

By far, most research on the conceptual design of temporal databases has been in the context of the Entity-Relationship (ER) model. This model, in its varying forms, is enjoying a remarkable, and increasing, popularityin industry. Conventional ER diagram rentals.



Conceptual Design daigram

The research on temporal ER modeling is well motivated. It is widely known that the temporal aspects of the mini-world are very important in a broad range of applications, but are also difficult to capture using the ER model. Put simply, dia- grams that would be intuitive and easy to comprehend without the temporal aspects become obscure and cluttered when an attempt is made to capture the temporal aspects. As a result, some industrial users simply choose to ignore all temporal aspects in their ER diagrams and supplement the diagrams with textual phrases to indicatethat a temporal dimension to data exists, e.g., "full temporal support." The result is that the mapping of ER diagrams to relations must be performed by hand; and the ER diagrams do not document fully the temporally extended relational database schemas used by the application programmers. The research community's response to this predicament has been to develop temporally enhanced ER models. Indeed, about a dozen such models have been reported in the research literature

Temporal DBMS Implementation:

There has been a vast amount of work in storage structures and access methods for temporal data, and a dozen-odd temporal DBMS prototypes have been reported [7]. Two basic approaches may be discerned. Traditionally, an integrated approach has been assumed, in which the internal modules of a DBMS are modified or extended to support time-varying data. More recently, a layered approach has also received attention [59]. Here, a software layer interposed between the user-applications and a conventional DBMS effectively serves as an advanced application that converts temporal query language statements into conventional statements that are subse- quently executed by the underlying DBMS, which is itself not altered.

Query Processing:

A query formulated in some high-level, user-oriented query language is typically translated into an equivalentquery formulated in a DBMS-internal, algebraic query language. The DBMS then optimizes this algebraic expression by transforming it into an equivalent expression that is expected to be more efficient to process, the result being better query processing performance. Optimization of temporal queries offers new challenges over optimization of conventional queries. At the core of the matter, temporal database queries are often large and complex.

Implementing Algebraic Operators:

As explained earlier, a user-specified query is translated into an internal, algebraic form, which is then optimized using equivalence-preserving transformations. The DBMS has available a library of algorithms that implement the operations that oc- cur in the resulting algebraic formulation of the query. As the next step, algorithms are chosen from the library for each operation, upon which the query is ready for execution. Good performance is dependent on the availability of good implementa- tions of the operations. Focus has been on a number of temporal algebraic operators, including se-lection, joins, aggregates, and duplicate elimination. Conventional approaches to computing these operators typically have poor performance, and new opportunities exist for efficiently implementing these operators

Indexing Temporal Data:

A variety of conventionalindexes have long been used to reduce the need to scan an entire relation to access a subset of its tuples, to support the conventional selection algebraic operator and temporal joins. Similarly, a number of temporal indexing strategies are available. Many of the indexes are based on B+-trees, which index on values of a single key; most of the remainder are based on R-trees, which index on ranges (intervals) of multiple keys. The worst-case performance for most proposals has been evaluated in terms of total space required, updates per change, and several important types of queries. Most of this work is in the context of the selection operator**.**

Summary:

This chapter has briefly introduced the reader to temporal data management ,empha- sizing central concepts, surveying important results, and describing the challenges faced. This section briefly summarizes the current state-of-the-art, and Section 7 discusses challenges that remain. A great amount of research has been conducted on temporal data models and query languages, which has shown itself to be an extraordinarily complex challenge with subtle issues.

References:

S. Abiteboul, R. Hull, and V. Vianu. Foundations of Databases. Addison- Wesley 1995.

T. Abraham and J. F. Roddick. Survey of Spatio-Temporal Databases. GeoInformatica, 3(1):61–99, March 1999.

J. F. Allen. Maintaining Knowledge about Temporal Intervals. Communications of the ACM, 26(11):832–843, November 1983.

J. Bair, M. H. Böhlen, C. S. Jensen, and R. T. Snodgrass. Notions of Up- ward Compatibility of Temporal Query Languages,39(1):25–34, February 1997