Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 61 (2013) 17203-17205

ISSN: 2229-712X

Intelligent Query Manipulation and Retrieval in Temporal Database

K.Murugan[1,*] and T. Ravichandran[2]

[1]Karpagam University, Coimbatore, Tamil Nadu, India.

[2]Hindusthan Institute of Technology, Coimbatore, Tamil Nadu, India.

ARTICLE INFO

Article history:

Received: 25 May 2013;

Received in revised form:

14 August 2013;

Accepted: 16 August 2013;

Keywords

System Architecture,

User Interface,

Query Processor,

Operators.

ABSTRACT

More and more organizations each day are now storing achieves of data to help them make essential business resolutions. The Commercial Database Management Systems (DBMS) that are available in the I.T. market do not afford any considerable and practical methods of storing and manipulate such data. Time is ubiquitous in information systems. Almost every enterprise faces the trouble of its data becoming out of date. However, such data is frequently valuable, so it should be archived and some means of accessing it should be afforded. Also, some data may be naturally historical, e.g., medical, cadastral, or judicial records. Whilst there has been research into the subject of Temporal Databases, there is a lack of marketable tools. The Paper aims to design and develop a Temporal Database Management System (TDBMS) that provides means to manipulate temporal data. The TDBMS will provide a Temporal Relational Algebra (TRA) structured query language that is based on and is an extension to Relational Algebra (RA) for extracting data from the Temporal Database. The TRA is to contain most the common operators that are associated with RA such as Cartesian Product, Natural Join, Union and many more. The TRA will introduce new temporal operators, which are useful for querying Temporal Databases.

Introduction

A temporal database is formed by compiling, storing temporal data. The differentiation among temporal data and non-temporal data is that a time period is appended to data expressing when it was valid or stored in the database.[1][2] The data stored by conventional databases reflect on data to be valid at present time as in the time instance "now". When data in such a database is modified, removed or inserted, the condition of the database is overwritten to form a new state. The state prior to any changes to the database is no longer available.[3][4] Thus, by associate time with data, it is possible to store the different database states. In essence, temporal data is formed by time-stamping ordinary data (type of data we associate and store in conventional databases). In a relational data model, tuples are time-stamped and in an object-oriented data model, objects/attributes are time-stamped.[5] Each ordinary data has two time values attached to it, a start time and an end time to establish the time interval of the data. In a relational data model, relations are extended to have two additional attributes, one for start time and another for end time. Time can be interpreted as valid time i.e. data occurred or is true in reality otherwise transaction time (when data was entered into the database).[6] A historical database stores data with respect to valid time. A rollback database stores data with respect to transaction time. A bi-temporal database stores data with respect to both valid and transaction time. A database object is stored in a database at some point in time. The transaction time of an object is the time when the object is stored in the database [7], the time that it is present in the database. For example, in a banking system, the transaction time of a withdrawal would be form the time the clerk entered the payment of withdrawal into the database to the time that it was made invalid in the database.[8][9] Another example would be, in a company situation, an employee receives a pay rise but it comes into effect when the payroll clerk enters this salary rise into the database. Transaction time values cannot be after the current time. Valid time The valid time of a database object is the time when the object is effective or holds (is true) in reality [10][11]. The time when the event occurred, took place in reality. For example, in a banking system, the payments and withdrawals made by a customer have a valid time associated with the time the customer performs the transaction at the bank.

System Architecture

The overall architecture of the Temporal Query Processing and Access power structure is depicted in Fig. 1. It consists of five basic elements as explained below:

User Interface: This part provides a Visual Query Builder for SQL. The input for this is the query and the output is the result set from the database. The User can interacts with the TDBMS by User friendly graphical interface that permits the user to compose requests to the TDBMS and present the results. This user interface be supposed to trouble-free to use and also easy to learn. The user interface is the only way the user can communicate with the temporal database system.

Middleware: The central part that contains the functionality of the system, which processes the request from the user through the interface and get back the information from the primary conventional relational database

Analyzer:

The input for this is the query. The given query is split into tokens and assigned the respective types with the help of the dictionary. The output of this analyzer is Lexical Tokens. Tokens are defined for all operators that are used in form TRA expressions, e.g. UNION, PRODUCT, JOIN, SINCE, UNTIL, PAST, FUTURE etc. Tokens are defined for arguments or variables given in TRA expressions, e.g. relation names,

Tele: +91-9443284308

E-mail addresses: dhuvash_2004@rediffmail.com

attribute names, integers, real numbers, string values etc. If a token cannot be found for a particular input then the system returns a user friendly error message.

Parser: The parser validates every or a collection of tokens as they are produced, by matching them against the predefined grammar integrated into the system. While parsing tokens, if the parser is unable to recognize any part of the input, an error is returned back to the user by means of interface highlighting the problem and the reason why the input has been rejected. For example, if the user has mistyped an operator name, then the analyzer stops translating and validating any further input and returns an error message with this mistyped name.

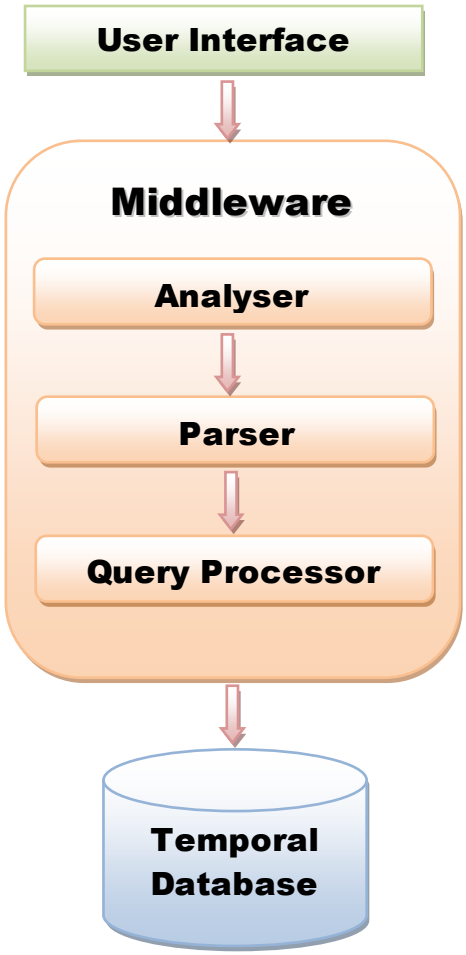

Fig 1.

Query Processor: The Query Processor takes in the key words as input and separates the fields that have to be retrieved. The main purpose is to obtain the meaning of the query. Initially, the given input is scanned for domain specific phrases. For example, customer identification number is identified by cust id in the database. So, the transformation becomes important. Similarly, the temporal operators and relational operators equal to, greater than etc., are replaced by appropriate symbols. From the transformed list of words, the relational operators and other keywords such as 'between', 'in', 'like' etc., are located as these specifies the required condition. The conditions are separated out and have to be transformed to fit into the SQL query directly. In this system, the keywords are the ones that represent the field name in the database and also the table name if it is explicitly specified. The keywords are of type noun. The words that are noun alone are taken out. This will contain the field names and the table name. A separate table is maintained where in we have the list of fields and the respective table name and the kind of data that it represents whether it is a string, number or date is maintained. The entire field names should point to one table. A key identifier helps to identify every table. Even if the table name is not directlyspecified, the table name can be obtained from the key identifier. For example, SB account refers to sb master table. The table from which the data has to be obtained is identified from the fields. All the fields have to point to the same table. The SQL query is formed with the field name, table name and the condition that is separated out.

Operators

The semantics of the usual relational operators have been extended to temporal tables, while new ones have been designed to cope with specific problems of temporal data [11]. This model includes four extraction operators, namely selection, projection, join, aggregation, and normalization transformations.

Temporal projection. This operator returns, for any correct source entity table S and for any subset A of its columns, a temporal table in which only the values of the columns in A are kept. If the set of the projection columns includes the entity identifier, the result is a normalized entity table. Conceptually, the temporal projection can be perceived as a standard projection followed by the merging (or *coalescing*) of the rows that have the same values of the non-temporal columns, and that either overlap or are consecutive.

Temporal selection: This operator returns the rows that meet some selection predicate in a correct source table. The selection can involve the temporal columns, the other columns, or both.

Temporal join: Considering two correct temporal source tables S1 and S2, and a predicate P, this operator returns, for each couple of rows (s1, s2) from S1xS2 such that P is true and the temporal interval i1 of s1 and i2 of s2 overlap, a row made up of the column values of both source rows and whose temporal interval is the intersection of i1 and i2. The result is a correct temporal table.

Temporal aggregation: Due to the great variety of aggregation queries, the process has been decomposed into four steps that are easy to encapsulate. Let us consider a correct entity table T with one time dimension. The query class coped with has the general form : select A, f(B) from T group by A, where f is any aggregation function. First, a normalized state table minT is derived by collecting, for each value of A, the smallest intervals during which this value appears in T10. This table is joined with T to augment it with the value s of B, giving the correct table minTval. Then, the aggregation is computed through the query select select A, f(B) from T group by A. Finally the result is coalesce d. In particular, this technique provides an easy way to compute temporal series (in this case, minT is a mere calendar).

Temporal Normal Form

A particular and well-organized technique for makeing temporal relational databases is by way of use of time intervals that are correlated with the tuples. In short, it is planned to facilitate tuples will have the subsequent structure: (a1,….,an) [start, end] in a relation R where start and end are the temporal time attributes and describe the time period for when the tuple exists. This temporal formation is applied and stored in a relational database as R(a1,….,an,start,end), a set of attributes . All intervals that are stored in the relational database are and should be maximal . That is to state, for any tuple R(a1,….,an,start,end) there must be no other matching tuple R(a1,….,an,start',end') where the non-temporal attributes a1,….,an are identical and the temporal attributes, the intervals overlap by confirming to start ≤ end'+1 and end ≥ start-1. Thus, adding the constraint that all such intervals are maximal gives the temporal normal form (TNF).

Temporal Relational Algebra – TRA

TRA is a set of operators that consists of the entire RA operators and with the addition of new temporal explicit operators. The evaluation of a query in TRA using the information of presently a snapshot of the historic database would indicate that the traditional relational operators have their regular significance, are reliable. To project data during manipulation and connecting of tuples from other databases Dt' with the tuples from the present database Dt (where t' $\bar{t}$), we can initiate new temporal operators. There is the beginning of two new temporal operators that can only be applied to historical temporal databases, which are known as *Until* and *Since*. These operators and their logical interpretation, US logic (Until and Since logic) are first order complete for a distinct historic database, they are a simple expansion of classical logic. Research into temporal logic and their properties has resulted in the formation of US logic that has been shown to be mathematically sound . They have verified semantics, which makes them popular temporal operators to be used for querying temporal relational databases.

Results

Temporal table stores both valid and transaction time. Consider a table Employee(EmpNo, EName Dept,Salary) with validtime and transaction attributes.

Emp. No.	Ename	Dept	Salary	From	To	Time Stamp
CS11	Rama	CS	9000	01/06/13	01/06/15	15/06/13 10:10
CS12	Seetha	CS	8000	01/06/13	01/06/15	15/06/13 10:10
EC21	Aswin	EC	8500	01/06/13	01/07/16	15/06/13 10:20
EC22	Joes	EC	9000	01/06/13	01/07/16	15/06/13 10:23
EE31	Dhuvash	EE	8600	01/04/13	01/04/14	15/06/13 10:30
EE32	Gopi	EE	9000	01/06/13	01/04/14	15/06/13 12:10
IT41	Edwin	IT	9000	01/06/13	12/03/16	15/06/13 12:20
IT42	Abdul	IT	8800	12/03/13	12/03/16	15/06/13 03:25
M51	Ashok	MC	8300	18/02/13	18/02/15	15/06/13 05:45
M52	Hamsi	MC	9000	20/02/13	20/02/15	15/06/13 05:53

SQL Query:

SELECT * from Employee during salary Change (5000,9000,01/06/2010)

Procedure::

```
salaryChange(int v1,int v2,day d)
{
setTableAs(employee);
setEventDomain(true);
moveCursorToPreviousDay(d);
match(salary,v1);
moveCursorTo(d);
match(salary,v2);
}
```

Results:

Empno	Ename
CS101	**Rama**
EC202	**Joes**
EE303	**Gopi**
IT401	**Edwin**
M502	**Hamsi**

Conclusion

This paper mainly focuses on and concerns the extraction and manipulation of temporal data. The system developed has revised and implemented the RA operators common to most commercial relational database systems that now evaluate temporal data. The system has introduced new and specific operators that can only perform over temporal data. The overall system is an independent software tool, more specific, a temporal relational database management system that allows itself to be attached to a temporal database, which has been modeled on a standard relational database system. This tool is appealing to organizations that store archives of data but find it difficult to extract required information.

References

[1].John F. Roddick "Schema Vacuuming in Temporal Databases" IEEE Transactions On Knowledge And Data Engineering, Vol. 21, No. 5, May 2009

[2].G. Antoniol, V. F. Rollo, and G. Venturi, "Linear predictive coding and Cepstrum coefficients for mining time variant information from software repositories," ACM SIGSOFT Software Engineering Notes, vol. 30, no. 4, pp. 1–5, 2005.

[3].Vijayalakshmi Atluri and Avigdor Gal. An authorization model for temporal and derived data: Securing information portals. ACM Transactions on Information and System Security, 5(1):62–94, 2002.

[4].Advanced Databases 2001 (Imperial Course Notes), by P.J.McBrien and J.McCann.

[5]. Rios Viqueira J. and Lorentzos N.A. Spatio-temporal SQL Extension. roceedings of the 8th Panhellenic Conference on Informatics, Nicosia, Cyprus, Vol. 1, 264-273 (2001).

[6]. Piero Andrea Bonatti Elisa Bertino and Elena Ferrari. Trbac: A temporal role-basedaccess control model. ACM Transactions on Information and System Security, 4(3):191–223, 2001.

[7].Snodgrass R. Developing Time-Oriented Database Applications in SQL. Morgan Kaufmann Publishers (2000).

[8].Garani G. Temporal Database Models: A Critical Approach. *University of London Publications*, ISBN: 0718716396 (2000).

[9].Jensen C.S. *Temporal Database Management*. Dr. Techn. Thesis, Aalborg University, Denmark (2000).

[10].Gadia S.K. and Nair S.S. Algebraic Identities and Query Optimisation in a Parametric Model for Relational Temporal Databases. *IEEE ransactions on Knowledge and Data Engineering*, **10** (5), 793-807 (1998).

[11].Tansel, Cli_ord, Shashi Gadia, and Richard Snodgrass. Temporal Databases: Theory, Design and Implementation. Database Systems and Applications Series. Benjamin/ Cummings, Redwood City, CA, second edition, 1993.