Temporal Organization of Data for Solving Problems of Computer Information Systems

Ahmad Ali Al-Zubi and Turki Al-Tameem

***Abstract*—Vendors of Database Management Systems (DBMS) and Enterprise Platforms, actively compete between each other to develop new features of their products. However, many problems, such as joint data processing or management of Data Lifecycle can be solved easier and more efficient with the Data Management Technology (DTM).**

Increasing amount of information and rapid development of Information Systems (IS) –are two interrelated processes: the high growth of data requires a similar development of its processing tools, which's improvement stimulates the processing of ever larger amounts of data. Today, practically in every more or less large organization, activities are being built around the Corporate Information System (CIS) (and often multiple systems), most of which are built on Relational Database Management Systems (RDBMS) with three-tier architecture: client applications - application servers - databases servers. When designing every such system several problems related to data management have to be solved.

***Index Terms*—Temporal data organization, information systems, tracking, electronic digital signature, problem solving, database management systems**

I. Introduction

A. Changes Tracking Problem

One of the most complicated problems today still remains the problem of controlling concurrent access to data. Usually, the solution for this problem involves protection from the problem of "Dirty Read", (Non-repeatable Read) and (Lost Update). To solve the problem of "Dirty Read", which consists of reading data recorded of cancelled operations, may well be enough the facilities of database server implementing standard transactions.

The solution for the problem of "Non-repeatable Read" (composed in the difference between the reading results of initial and subsequent data by one client, without any modification), as well as the problem of "Lost Updates" (occurs when two or more clients make some modification to the value of the same element at the same time, in which the earlier modification is lost) caused by only transactions on the level of database server and / or application server, is only effective with respect to server-processing tasks that run on an application server or database server in the background of a predetermined algorithm and without the active participation of the user.

However, protection from similar problems must be also provided for manual processing of data by users using their client applications. The main difference here is that every user does not work directly with the data contained in the DBMS, but with some of its local copy loaded in the client's application. This often leads to the fact that when two users save changes to the same data element (and their modifications could be done on various properties of this element), the earlier change is lost, and the client, whose change is lost, May sometime does not even see.

It is considered, that protection from such kind of problems should be provided by the application logic through the more active use of transactions [1]. But not all problems of concurrent access can be bypassed by the software with additional checks and transactions. And the key factor here is the user. It is impossible to distinguish programmatically sequence of read and write for user data forming atomic operation. Therefore, user cannot undo such a transaction when the read data is changed – The system will simply not be able to determine how many recent actions of the user form a single entity, and which of this read data the user has taken into account when making a decision. What to do?

As in all cases, where it is impossible to prevent a problem, all efforts should be concentrated on tracking this problem. And in corporate systems, control of concurrent access - is not the only area where it is required to solve the problem of tracking changes. Another very important problem is the difficulty of identifying data, which is formed on the basis of other, usually aggregated data. As a typical example, we can present various reports that are generated based on data from the system and not stored in it. In the future, to determine the source of each composite indicator of the report is possible only if the source data is unchanged.

As one of the possible solutions, administration may prohibit data edition, based on which reports are generated, but this is not always possible, and not too convenient. On the other hand, it is a very common practice, after the primary report in case of inaccuracies detection, all these inaccuracies are corrected in subsequent reports, based on this corrected data is formed the proximate report (such as, for example, the procedure for tax reporting). In general, the purpose of changes tracking is to be able to retrieve the data that exists in the system (the local copy of data) at the time of the user's decision to change the data, generating reports, etc. In addition, often also requires information about the authorship of changes and correction. Moreover, the information needed about all modifications, not just the last ones, since errors in the data (intentional or unintentional) could be made earlier, and other users in the future can be based on them.

Particularly, this given task looks like the widespread

Manuscript received April 8,2012; revised May 24, 2012

The authors are with King Saud University (e-mail: aalzubi@ksu.edu.sa, altameem@ksu.edu.sa)

transition from paper-based information resources to electronic resources, which requires giving legal significance to the data (by EDS). Separately, it should be noted that this is not about giving the legal significance to individual documents transmitted between organizations, but a formation of a fully legally significant amount of data, each sample of which also has its own official status. Naturally, this should provide a personal responsibility by the people forming this data set. This is especially important for government systems, where every employee has the authority and the proper responsibility for that.

The task of tracking data modifications is also crucial to ensure the synchronization of joint processing of data in multiple information systems. If the systems are working consistently (ie, the output from one system is used as an input to the next system) [2], Then all relatively will be simple. It is much more difficult with the organization of parallel data processing, for example, in the management of master data. It is widely used a centralized model where all the master data is stored in a dedicated system, but it's not always convenient and possible, especially in the interaction of information systems of different organizations. Besides, this model is not based on the data management technology, and represents only an architectural solution.

If we go back to governmental information systems, then the organization of parallel data processing in electronic information resources, each of which contains legally sensitive information, can be considered as one of the key infrastructure elements of the approach to building an e-government.

II. Transactional and Non Transactional Data

To solve the problem of tracking all modifications made on data, it is reasonable to use the technology of temporal databases. But to date, the complete industrial implementation of temporal databases, in fact, still absent. Curiously, they could appear five to ten years ago, but because of fashion trends in data management to shift the focus towards the support of XML and solving problems of integration, temporal technology remained aside. Some modern databases contain specialized mechanisms that allow the use of background versioning attribute values. However, it is not always convenient to use.

But more importantly - the Relational Data Model (RDM) provides a very large fund, which may well be used to solve the problems listed above. Maybe we should not artificially extend this model with additional time dimension?

Data processed in each information system can be divided into transactional and non-transactional. Transactional data – is data, each record of which is related to a fixed point of time and contains information that is fixed at a given time, and no change in the future. Accordingly, all the rest are non-transactional data.

Transactional data are typically repeated examples of events, phenomena, occurrences of the same type. This includes all requests, bills, invoices - and indeed all documents (as entities), as they are all locked (already processed documents) and tied to a certain point of time (the time of their compilation or registration).

Non-transactional data that is often referred to as reference or baseline data, are lists of similar objects: entities, objects and abstract categories. Typically, these references and classifications are of diverse kinds, in other words - master-data or, as a special case of normative-reference information.

It is easy to see, that the relationship between these two data categories, is usually one-sided: when describing transactional schemas can be used both transactional and non-transactional data, but when describing non-transactional data, only non-transactional data can be used. For example, when preparing a bill different references can be used (contractors, nomenclature) [3], and the basis for the bill may be, for example, an account.

On the other hand, if you do not take into account the specific informational representation of data and procedures for its archiving and utilizing, it can be argued that the number of information objects of transactional data is growing as the registration of each new event creates a new record. At the same time, the number of information objects of non-transactional data is relatively constant.

The value of each transaction data record, as a rule, remains unchanged since it was fixed. Exceptions are cases of some records correction due to inaccuracies or errors, which, incidentally, in the information system are generally considered as not quite correct action - correction must be done in a separate operation [3,4]. Non-transactional data is not tied to a particular point in time, but during its life cycle, it can be determined, and the values of attributes of each element of such data are subject to change.

Thus, the problem of tracking modifications, mainly relevant to non-transactional data. For transactional data, it only makes sense in terms of tracking corrections.

One of the main causes of the problems associated with the representation of non-transactional data is that non-transactional data is often perceived as a quasi-permanent, and as a consequence of having a static representation in databases. To record data in SQL statements is used (insert) for input, (update) for change and (delete) for remove. Typically, these operators are used quite straightforward: for a new object - insert a new record, if there is some changes on its characteristics - update them, object disappeared - delete record. In fairness, it should be noted that to improve the referential integrity instead of the delete operation is now increasingly used update additional record attributes (status) [5].

However, the update operation is also not harmless. Termination of an object – is new information, and the result should not be an overall reduction of information in the database, but on the contrary, it is an increase. Similarly, if you change the characteristics of some object, means a new information will be added to the database, so the total amount of information will also increase. The same thing can be said about correction. In other words, the information elements should store information about the life cycle of real objects, rather than repeating it.

Using SQL instructions, we can say that it’s impossible to drop objects or change their characteristics using “delete” or “update” commands. These commands are used for data manipulation (deleting, changing the data itself). Thus, we can conclude that all data circulating in the database should be transactional; non-transactional data should be provided

in the form of a chain of transactional data.

III. Temporality in Relational DBMS

Transactional data is usually associated with only one value of time. To ensure the same concurrency control and implementation of separate reflections in a state database operations patch requires the use of bitemporal model that includes real and transaction time. Real time – is the time indicating the relevance time of the attributes values existence of real objects. Transactional time – is the time of introducing a new information about the object in the information system.

Double temporality not only allows more accurately describe the system model, but also to determine correct operation. To do this, in addition to the existing record with an incorrect value, should be entered correcting value with the same real time, the current transactional time and corrected values.

In addition to extending opportunities to present data in an information system, the use of bitemporal model allows the concurrency control for protection from the problems of non-repeating readings. This is achieved due to the fact that in a single operation (transaction) during the data processing is used a restriction for data selection: Used only the records added in the information system prior to the transaction.

The double temporality of data in conjunction with the data management in the "Insertion Only" mode, can provide storage of information together with user personal, who made the changes, and his digital signature as well, calculated based on entered data. This allows you to organize a legally significant data warehouse with the repository of personal responsibility for its content.

In addition, this technology provides the possibility of tracking the data that existed in the system before making any changes by the user. Similarly, the same way you can keep track of primary data that present in the system while formatting secondary data, provided storing time stamp together with the secondary data, as of which data was formed.

It should be noted that in the three-tier architecture conflicts may happen because of the fact that between the time of data reading and the time of data change, some updates can be done by another user. To avoid this, it makes sense to use two transaction times: the main transactional time – update transactional time, and data reading transactional time - to keep track of main transactional time of data on the basis of which is processed.

In this model, the disappearance (removal) of the object cannot be displayed in a pure form. However, in addition to the appearance and disappearance of data elements and the change of their attributes, may also occur some events as the merger, accession, division, separation, reorganization. To realize required operations an additional table must be used, and the corresponding event to appear in this table as a transition (vector) with an indication of the preceding identifier (null for element appearance) and the new identifier (null for element disappearance) [5,6]. Using this approach, any of the above operations can be reflected in the state of database with several inter-related entries.

The significant problem is to provide a soft upgrading for information systems, together with the relevant relationships between them. First, the complexity is the need for modernization and maintenance of metadata with the data of prior periods in the corresponding data schema. Also important is the preservation of connectivity of the systems when the data structures are changed without further development, but only by a corresponding connections reconfiguration.

In fact, besides life cycle of data element and life cycle of its attributes, it makes sense to consider the two life cycle of metadata: the life cycle of classes (tables) and life cycle of class attributes (table columns). To describe them, additional tables are needed, in which, transitions are used to reflect operations on data elements in the database status [7].

Fig. 1 shows the structure for presenting data using above described technology. To link entries an additional transaction table is allocated, which stores transactional read time, transactional record time, information about the author and his digital signature. This further allows for manual transactions rollback, and to some extent reduces the amount of memory needed to store information about users and their Electronic Digital Signature (EDS). All elements tables are combined into a single table, which contains a field that indicates the class of the element. And also all tables of values are combined together in one table, and this table contains the field, which references the table of elements and the table of attributes.

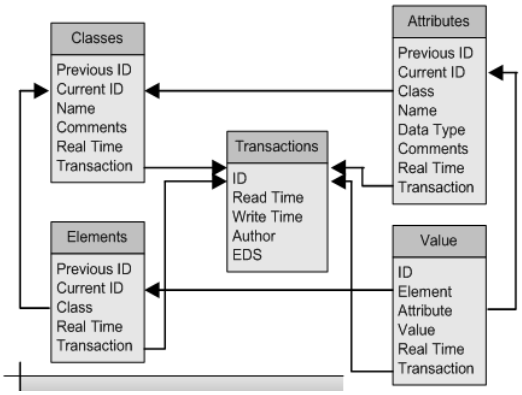

Fig. 1. Data representation using suggested technology.

IV. Conclusion.

The advantages of this model include a significant increase in the level of referential integrity, a separate reflection of the status of the database operations, data changes and corrections, the possibility of organizing a partial protection from the problems of concurrent access to data, solving the tracking problem of secondary data sources, providing authorship tracking and the possibility of forming a legally significant data set.

Temporal organization of data allows to significantly simplifying their co-authoring in several information systems and the temporal organization of metadata - to provide the Opportunity for soft upgrade of the data structures and providing working with the data of previous periods in the corresponding data schema.

The described approach can be applied in developing systems for different purposes, primarily, management systems (subsystems) of master data or reference data. In

particular, the use of this technology can solve many problems in the development of a major component of e-government systems -System of public services registers.

REFERENCES

[1] C. J. Date, H. Darwen, and N. A. Lorentzos, "Temporal Data and the Relational Model: A Detailed Investigation into the Application of Interval and, Relation Theory to the Problem of Temporal Database Management," *Morgan Kaufmann*, December 2002.

[2] J. opitovs, V. Demidovs, and N. Petoukhova, "Method of Temporal Databases design Using Relational Environment," In*: Scientific proceedings of Riga Technical University – Computer Science: Applied Computer Systems*, pp. 236-246, 2002.

[3] N. Petoukhova, "Development of a Complex Security System in Relational Databases for Railway Transport," In: *Scientific proceedings of Sixth International Baltic Conference on Computer Science and Information Technologies – Databases and Information Systems: DB, IS 2004. Scientific Papers University of Latvia*, June 6-9, vol. 673, pp. 139–150, 2004.

[4] Preliminary information for International Symposium on Secure Software Engineering ISSSE 06 – IEEE, [Online]. Available: http://www.ieee-security.org/Calendar/cfps/cfp-ISSSE06.html, 005, November 22.

[5] Microsoft Corporation with Andy Ruth and Kurt Hudson. Security+Certification Training Kit, Microsoft Corporation, 01/29/2003, 512 pages, CIA triad, pp. 5–6, 11.

[6] C. Jensen, Temporal Database Management. Dr. techn. Thesis, defended on 14.04.2000, 1328 pages, [Online]. Available: http://www.cs.auc.dk/~csj/Thesis/, 2005, November 22.

[7] N. Petukhova, "The method of providing access to relational data at the row level relationships," *Transport and Telecommunication*, vol. 4, no. 1, 2003.