UDC 004.652.4
© Safonov V.S.

Theoretical aspects of application of temporal relational DB to store the social and economic information

This article deals with the theoretical aspects of temporal databases. There is one of the definitions of temporal databases in the first part of the article which reflects the essence of temporal databases exactly. The existing approaches to temporal databases are also described in the article as well as three approaches to create such databases are highlighted. The article proposes a way to implement temporal extension for existing relational databases. Therefore new data types and new operations with them are proposed. All descriptions are given through the abstract data types. The final part of the article deals with the application of temporal database to store the social and economic information and creating of DB to keep the results of monitoring studies.

Relation database, temporal database, ADT, PostgreSQL, operation.

Vadim S.
SAFONOV
Junior scientific researcher of Institute of Socio-Economic Development of Territories RAS
VadimSafonov@gmail.com

All information which we have to work with has a temporal component in a varying degree. The only difference is the following: a temporal component can be stored in the information system or it can't be stored.

Several years ago the temporal component was stored only in those systems where its absence led to information distortion or inability to restore proper sequence of events. Such systems included financial databases, data storage system for scientific experiments, i.e. all the systems where time was an integral part of information, such as the status of the researching system or object at a certain time and the time of financial transactions.

Such systems that accumulate historical information are not widespread because the tasks can be solved successfully without saving any time in the majority of cases. In addition, these systems are demanding to hardware resources such as information carriers' capacity primarily which were actual recently. Therefore, even the systems storing information with temporal attributes had only those objects marked as temporal which lost their properties without the temporal qualities. For example, as for accounting systems, information about financial transactions would have a temporary marking while information about the person who committed the operation would be stored only in its current state.

Interest in systems that store time reference information has increased recently. Such systems are called temporal. Temporal databases are the databases which store data related to the time and have controlling means of such information.

The main difference between temporal database management systems (DBMS) and the conventional relational database systems is the following: any object that is created at time t1 and is removed at time t2 retains all of its state in this time interval [t1, t2], whereas there is only current state of an object at the particular time in the conventional DBMS.

Thus, the temporal database stores the history of changes of object states and the user can obtain information about the status of records in the database at any time of the specified period.

There are several approaches to implement the temporal properties of DBMS at present. The first approach is the creation of temporal DBMS from scratch. In this case all temporal properties will be founded in the core of database management system.

Although such developments exist, but we can't talk about their mass using because non-temporal relational DBMS dominate in the up-to-date market and existing temporal databases are inferior to them in functionality. There are also works that invite you to query temporal databases in natural language, but such approach is not promising because lexical analyzers are not perfect and they can recognize only a limited number of lexical structures. Whereas a natural language query can be formulated in different ways.

The third approach is the most rational. It is an extension of the existing DBMS functionality through the creation of some additional functional block which is responsible for transformation of temporal queries into the relational form and transformation of data from the storage format into the structure that is convenient to the user. The location of this functional block is the main problem in such approach.

There are three main options: the DBMS's core, the user's application and the form of an intermediate module.

The first way is bad because it is available only to developers of DBMS, but it provides maximum data transparency in applications and great potential for optimizing of queries.

The second way is more available to developers of applications. Temporal properties are available for your application without reference to the chosen DBMS. At the same time such properties can be laid only in the newer versions of programs. Currently this method is used more often.

The third approach creates the transparent extension both for front end and back-end. But you can easily see that the intensity of data exchange between the module and the application is less than between the module and DBMS. That's why it is reasonable to place this intermediate module as close to the DBMS's core as possible.

In our opinion it is more rational to combine the first and the third approaches. It is reasonable to introduce a part of the module into the DBMS's core and use the middleware. In this case you can get support for temporary data integrity at the level of DBMS, and it is possible to transform the queries in the independent module.

Thus, it is necessary to use two additional fields in the temporal tables as opposed to the conventional relational tables to store the beginning-of-period and end-of-period timestamps. In this case it is necessary to use composite primary keys, i.e. you have to unite the tables in three or more fields. In our opinion, it is more rational to define a new composite data type that contains a unique identifier and two timestamps. A new temporal data type for working with the urgency time can be described as an abstract data type (ADT) [2] in the following way:

```
    DATA_TYPE ValidTimeKey IS
    OPERATIONS:
    CONS: → ValidTimeKey(Key, Time_Begin, Time_End=NULL)
    CLOSE: ValidTimeKey(Key,Time_Begin,NULL) →
ValidTimeKey(Key,Time_Begin, Time_End)
    IS_OPEN: ValidTimeKey → bool
    IS_LINKED: ValidTimeKey → bool
    IS_INTERSECT: ValidTimeKey → bool
    LESS_THAN: ValidTimeKey < ValidTimeKey → bool
    LESS_THAN_EQ: ValidTimeKey <= ValidTimeKey → bool
    LESS_EQ: ValidTimeKey == ValidTimeKey → bool
    GREATER_THEN_EQ: ValidTimeKey >= ValidTimeKey → bool
    GREATER_THEN: ValidTimeKey > ValidTimeKey → bool
    GET_KEY: ValidTimeKey (Key,Time_Begin, Time_End) → Key
    END ValidTimeKey
```

There are the following operations for this type. IS_OPEN – it allows to determine whether the period is open, IS_LINKED – determines the presence of pointers to the column from other tables, CONS – a constructor of the type, CLOSE – it closes the open period, IS_INTERSECT – it checks whether the actual periods of the same object intersect, GET_KEY – it gets the numeric part of the type.

It should be noted that the possibility of defining of the new types is laid in the current version of SQL standard, and if you use ADT to describe a new type it allows you to implement a new data type to any modern relational database management system that supports the addition of the new types.

We used PostgreSQL database version 8.2 [3] as an experiment in the current work. This system was chosen for several reasons.

First of all, it has free and open source code. Secondly, functional capabilities of the system are wide.

Thirdly, there are detailed documents here. The implementation is regarded in this article later in respect to this DBMS.

New data type for PostgreSQL database was defined as the following:

```
CREATE TYPE validtime (
INPUT = valid_time_in,
OUTPUT = valid_time_out,
internallength = 24,
alignment = double
);
```

In this example, the function valid_time_in() converts the data which are sent to its entrance into the internal structure of the following form:

```
valid_time_in ()
typedef struct ValidTime {
int ValidBegin;
int ValidEnd;
int key;
}
```

This structure stores the information about beginning-of-period timestamp and end-of-period timestamp, as well as the unique identifier; a combination of these three values provides the uniqueness of each tuple. A function valid_time_out () performs the reverse action. These functions have been implemented in the language C.

The type is represented for the user of database as a string of the following form:

```
(key, validbegin, validend)
```

type is:
(Key, validbegin, validend)

where the key – a unique numeric identifier, validbegin and validend – beginning-of-period and end-of-period timestamps.

Element key must be an integer. A timestamp in the format Unix Timestamp date in YYYY-MM-DD or YYYY-MM-DD HH: MM: SS can be specified as the timestamps. The function valid_time_in () converts automatically the indicated dates into the timestamp. All dates are stored as the timestamps regardless of format specified by the user.

The developing type should be able to be used as a key because one of the objectives pursued by the creation of a new type is flight from composite primary keys. Thereto it was necessary to determine the class of operators in respect to PostgreSQL database which were responsible for the construction of the index tree. B-tree was used as the index.

Five operators were identified to create the B-tree: less, less or equal, equal, more or equal and more. Each operator was implemented as a function of the C language that compared two values of the new type and returned value of the type of Boolean. Here is an example of implementation of the operator "less" into the database.

```
    CREATE OPERATOR < (
    leftarg = validtime, rightarg = validtime,
procedure = valid_time_less_than,
    commutator = > , negator = >= ,
    restrict = scalarltsel, join = scalarltjoinsel
    );
```

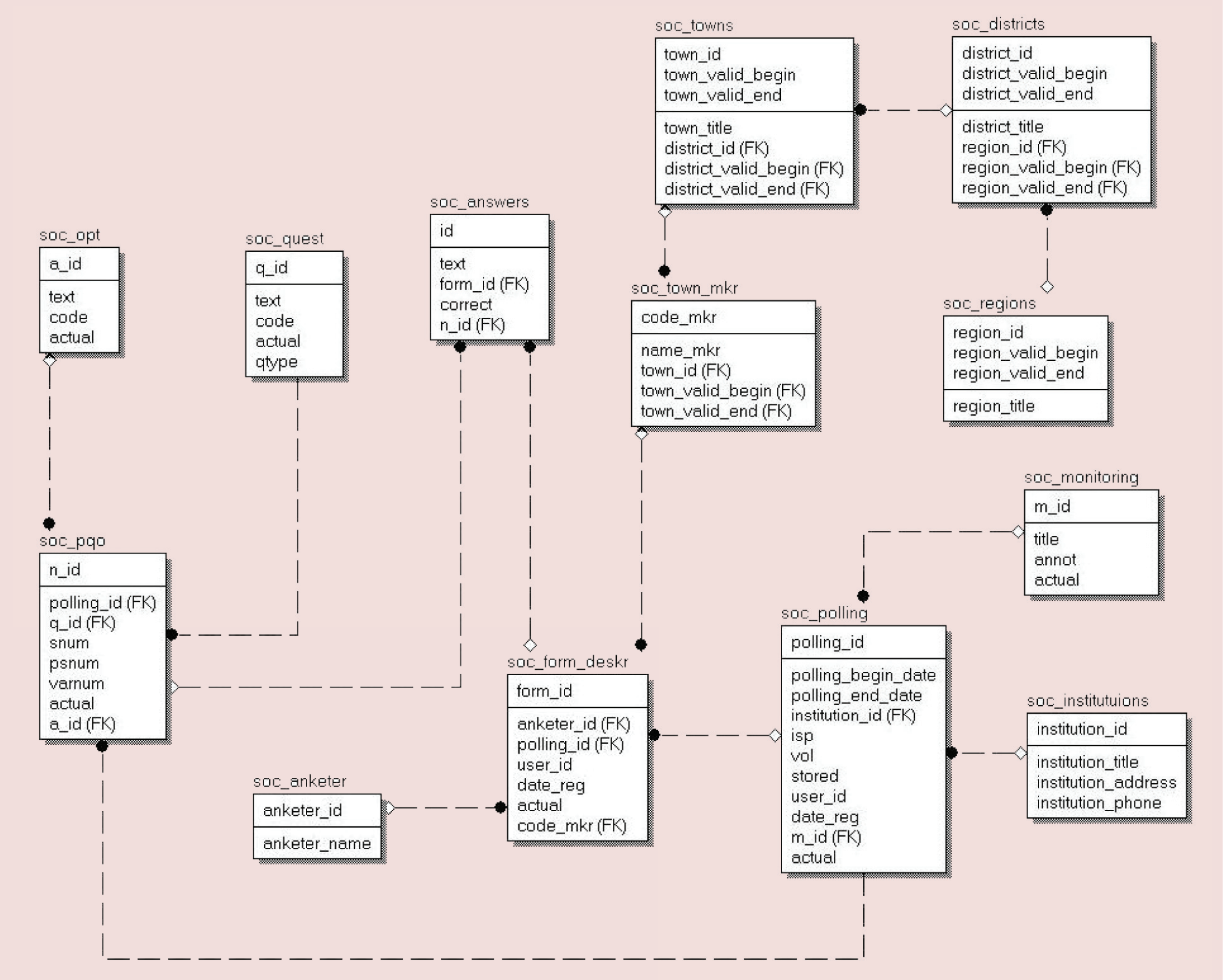
Then we defined an operational class using those operators. It allowed using a new type in the primary key.

```
    CREATE OPERATOR CLASS valid_
time_abs_ops
    DEFAULT FOR TYPE validtime
USING btree AS
    OPERATOR   1 < ,
    OPERATOR   2 <= ,
    OPERATOR   3 = ,
    OPERATOR   4 >= ,
    OPERATOR   5 > ,
    FUNCTION   1 valid_time_abs_
cmp(validtime, validtime);
```

The steps described above allowed to use not a group of three fields but one field of a new type as the primary keys and simplify writing queries. However, creation of a new type does not convert the relational database into the temporal database; it is only the first step towards such a transformation. In addition, it is necessary to transform queries from the temporal form into the format specified SQL standard. It is reasonable to perform all these transformations in some intermediate module. And queries to the temporal database that uses the new types of data should not differ from queries in the classical language for the users. Simple queries to the temporal databases should not vary from queries to the conventional relational database. In other words, the query SELECT * FROM table1; in both cases must return the same result (as applied to the temporal databases – all current at the time of writing).

Let's consider the task of forming a common database of public opinion polls. In spite of the fact that it is more correct to refer such data to the information contained in the time series rather than to temporal information, temporal attributes would be useful for some aspects of these databases because questions and answers may change with the lapse of time. It should be taken notice of the fact that some parts of the forms are identical although the polls may pursue different goals or they can be focused on different population groups (for example, groups which differ in their places of residence or social status, etc.). Such data com-

The proposed structure of the temporal DBMS



bined into a single database will expand greatly the background information for analysis.

Thus, the creation of a single normalized relational database is an important task, and the addition of temporal attributes in those tables, where it is necessary, will expand its capabilities.

The results of all periods of measurements are generated in the form of files of SPSS program. They represent a table where each row stores the data of one survey. It is difficult to combine the data of polls carrying out in different time, although this type of representation is useful for the analysis of individual results.

Although the polls are not temporary data, but logic circuit shows that some tables may be added by the time aspect: for example, a group of tables that describe the place of filling in of the questionnaire. In spite of the fact that administrative and territorial division is constant, it is necessary to take into account the recent trend to merger of administrative and territorial areas when we develop the database. It would be difficult to maintain the integrity of the data without introducing a temporary component to them.

In addition, the temporal aspect is required in the tables which contain questions and their answers because they can change in the course of time. The keys for all tables which are proposed to be converted into the temporal form are simple in the scheme of the database and they represent a unique identifier that satisfies the normal forms and ensures the tuple's unrepeatability. It is impossible to use the simple primary keys for these tables when the database is converted into a temporal form because it can't ensure uniqueness of records. So it is necessary to use compound primary keys of the following type: [record identifier, the beginning of the urgency period and the end of it]. And then the database schema will have the following form (figure).

This scheme can easily be represented in the non-temporal form; it is just enough to remove the fields which are responsible for the time component, therefore it isn't given in this article. Although the transformation into the temporal form expands the database's opportunities, but it can complicate the queries. But in this case the information for the temporal database will be more detailed and reliable. So, if we change the answers to any closed question (for example, add a new answer) in the non-temporal database, the changes will affect all existing data. If we turn to the temporal database, the temporal fields which are added can solve this problem: they point out the period when one or another variant of the answer exists although the query itself does not change.

Thus, we can draw some conclusions. Firstly, the definition of relevant periods for all elements of the questionnaire can remove ambiguity in question formulations, questionnaire venues, etc. Secondly, addition of the extra information to the database can extend the analysis allowing to retrieve and analyze data in dynamics. Thirdly, it is easier to link the temporal database to events which are not directly reflected in it, but which are connected with specific time intervals.

In general we can say that nowadays the temporal databases are the promising area of researches. And, in our opinion, the creation of special extensions of temporal data management for existing DBMS is the most optimal.

References

1. Snodgrass, R.T. Developing time-oriented database application in SQL / R.T. Snodgrass. – San Francisco: Morgan Kaufmann Publishers, 2000. – 528 p.

2. Futi, K. Programming languages and VLSI circuit design: trans. with Japanese. / K. Futi, N. Suzuki. – Moscow: World, 1988. – 224 p.

3. PostgreSQL 8.2.0 Documentation [Electronic resource] // PostgreSQL. – Available at: http://www.postgresql.org/files/ documentation/pdf/8.2/ postgresql-8.2-A4.pdf

4. Education. Science. Business: Features of regional development and integration: collected papers of All-Russian scientific and practical conference. – Cherepovets: IMIT SPSTU, 2005. –344 p.

5. Poray, D.S. Implementation of the temporal database concept through the relational database means [Electronic resource] / D.S. Poray, A. V. Solovyov, G.V. Korolkov. – Available at: ftp://ftp.dol.ru/pub/users/cgntv/download/sbornic/sbornic5/Doc8.doc