Temporal Extension of the Select Statement

Michal Kvet, Karol Matiaško
University of Zilina
Slovakia
michal.kvet@fri.uniza.sk

ABSTRACT: *Effective timed data processing belongs to one of the most important task of the development of current information and database systems. It is not, however, only the changes in time management, but also the complex record of changes during the whole life cycle of the object – historical values, actual states, but also data valid in the future. Existing temporal solutions are inadequate in terms of performance - effectiveness of the whole system, which is manifested by the size of the required data and processing time. There is no temporal solution for Select statement defined and user has to manage it explicitly. This paper deals with the principles of temporal data modelling on object and attribute level. It also describes the characteristics of Select statement used in temporal system, extends it with new characteristics and defines new layer for transformation into existing syntax.*

Keywords: Temporal Select Statement, Column Level Temporal Data, Object Level Temporal System, Validity, Epsilon Definition

Received: 27 November 2014, Revised 29 December 2014, Accepted 5 January 2015

1. Introduction

Massive development of data processing requires access to extensive data using procedures and functions to provide easy and fast manipulation. The basis is the database technology [10].

Database systems are the root of any information system and are the most important parts of the information technology. They can be found in standard applications, but also in critical applications such as information systems for energetics, industry, transport or medicine. The development of data processing has brought the need for modelling and accessing large structures based on simplicity, reliability and speed of the system. However, even today, when database technology is widespread, most databases process and represent current valid data. However, conventional relational database definition can be extended for temporal data modelling, which allows changes, evolution monitoring, process optimization, prognoses and analyses creation [1] [11] [12] [13].

This paper deals with the temporal architecture and defines problems of data selection, therefore new layer for temporal data manipulation has been developed and Select statement has been extended. Extended parts are characterized with the emphasis of current relational database transformation.

2. Theory

Temporal system has been developed soon after the development of databases. In the first phase, historical data were saved using log files and archives. Thus, historical data could be obtained, but it is a complicated process, these data are in the raw form and handling them was difficult, required too much time. The main problem was data operation loss, if the backup granularity was not suitable. Thus, decisions based on historical could not be used, because large backup images had to be loaded manually, which took long time. Another problem is the impossibility to future valid data modelling and management [1] [5] [6] [7].

Later, the temporal systems have been developed defining new paradigm for selecting one or more rows based on the specified criteria, for projecting one or more columns to the output sets and for joining the tables by specifying relationship criteria. It means that individual operations must contain also time definition. This paradigm is still valid.

The first model (conventional approach) in figure 1 does not use time for definition at all, it cannot provide management for non-current data in the main structure. The primary key is defined by the attribute *ID* (can be composite). In non-timed table, each row represents specific instance identified by a primary key. The uniqueness of the primary key values without defining additional conditions ensures that the number of rows in the table is identical with the number of managed objects. Any change is directly reflected to the database and the old value is deleted.

The second model is uni-temporal system, *ID* is a unique identifier; *PK* refers to a primary key. *BD* and *ED* is a pair of columns defining the beginning and end value of the period – validity. The uni-temporal system always uses the composite primary key – object identifier and time interval defining the validity state characterizing the row [2] [3] [4] [6] [7].

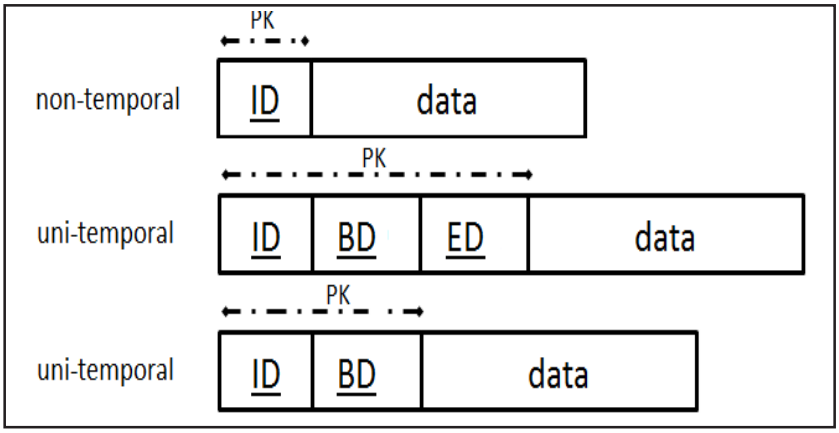

Figure 1. Structure of validity temporal table [11]

Special type of uni-temporal system consists of only one time attribute (*begin date – BD*) that is part of the primary key (third model in figure 1). This means that any change of the corresponding object determines the validity of the prior state.

The principles of transformation uni-temporal system defined by begin date of the validity to the standard approach and time intervals are described in [7] [16]. special approach has been developed by us, which is based on the attribute granularity, not the entire object. thus, if the value of the single attribute is changed, only that value is updated (not the whole state, which consists of various number of temporal columns) [5] [8] [14].

3. Temporal Select Statement

The Select statement in relational database approach is considered as the most important and most frequently used SQL statement based on performance. With this statement, we get desired data from the database using relational tables. The basic syntax of the Select statement in conventional database consists of these six parts - Select, From, Where, Group by, Having and Order by.

Although conditions can be defined in the Where clause, this segment does not cover the complexity and structure of the temporal system. Therefore, the following section describes ways to enhance the whole concept of management of temporal data.

```
[CREATE TABLE table_name AS ]
SELECT [ALL | DISTINCT | UNIQUE]
   { *| attribute_name| function_name[(parameters)]} [,...]
   FROM table_name [alias] [,...]
   [WHERE condition]
      [EVENT_DEFINITION]
      [EPSILON_DEFINITION]
      [MONITORED_COLUMN_LIST]
   [GROUP BY attribute_list]
   [HAVING condition]
      [TYPE_OF_GRANULARITY]
   [ORDER BY column_name [ASC | DESC] [,...]]
```

Figure 2. Temporal Select extension

Designed and implemented syntax shows the temporal extension of the Select statement using these parts:

- EVENT_DEFINITION,
- EPSILON_DEFINITION,
- MONITORED_COLUMN_LIST,
- TYPE_OF_GRANULARITY.

The importance of them is subsequently described in the next section.

4. Type_Of_Granularity

TYPE_OF_GRANULARITY expression is included after the Having clause, it is a way of formatting the final set, defines the sensitivity and details of the changes. There can be three types defined. Object - defines the granularity on object level – list of object states changes regardless the type of change. Changed attribute values themselves can be taken using Column characteristics. The difference between the options shown in the following figure. At time t_1, object O_1 state is changed from S_1 state to S_2 state. If the Object definition is used, we get information about the change of the object O_1. However, specific new values (new value of the attribute $H = H_{A12}$) are obtained through the clause Column.

The access rule COLUMN_CHANGES_MONITORING displays also comprehensive information – new (H_{A12} - shown in red in figure 3) and also previous value of the attribute (H_{A11} - shown in green in figure 3).

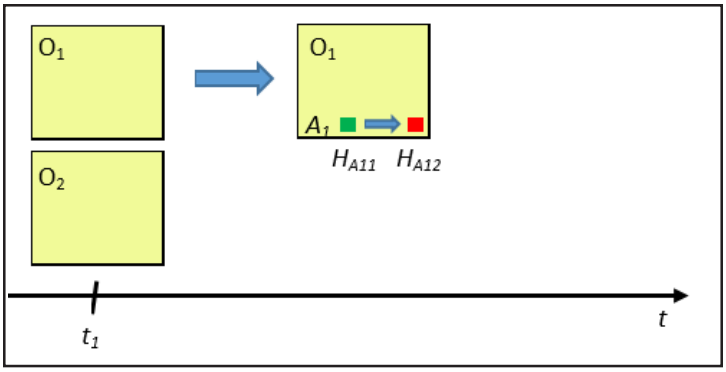

Figure 3. Update of the state of the object O_1

Next figure shows the output of the Select statements with respect to the granularity access rules. Figure 4 shows the principle of transformation to the existing syntax.

OBJECT: t_1 - state of the object O_1 has been changed.
COLUMN: t_1 - state of the object O_1 has been changed.
Attribute A_1 now has value H_{A12}.
COLUMN_CHANGES_MONITORING: t_1 - Attribute A_1 of the object O_1 has been changed. Old value = H_{A11} ; new value = H_{A12}.

Figure 4. Output of Select statement - granularity types

Default selection criterion is Column. If this clause was not used, we would get all temporal attribute values regardless the change of them at defined timepoint (interval).

An essential part of the state monitoring is to get direct predecessor (state) for changes monitoring. Next figure shows the algorithm for obtaining previous change, if exists. Solution is based on using With clause, which eliminates multiple same table definition (figure 6).

```
OBJECT:
  -- only identifier of the object (object_id) and timepoint of the
     change (ch_timepoint) is selected.

             select object_id, ch_timepoint ...

COLUMN:
  -- identifier of the object (object_id), timepoint of the
     change (ch_timepoint) and new value (new_val) is selected.

             select object_id, ch_timepoint, new_val
              from ...
               where new_val != old_val ...

COLUMN_CHANGES_MONITORING:
  -- identifier of the object (object_id), timepoint of the
     change (ch_timepoint), old (old_val) and new value (new_val) is
     selected.

       select object_id, ch_timepoint, new_val, old_val
       from ...
         where new_val != old_val ...
```

Figure 5. Transformation to existing syntax

5. Event_Definition

This definition extends the Where clause of a Select statement specifies a range of time and processed data obtained by way of a point in time (defined_timepoint) - just a reference to the point in time - or time interval (defined_interval), the definition of which includes two time values - the begin and end timepoint of the validity (figure 7).

In the session, user can set the type of used interval (closed-closed, closed-open representation). However, it can be redefined directly for executed Select statement. Thus, the second (optional) parameter is the interval type to be used (CC - closed-closed, CO - closed-open type)

Figure 8 shows input interval with the closed-closed characteristics, figure 9 shows the closed-open representation defined

inside the method. Blue colour represents input time interval (defined_interval (t_1, t_2)), red colour highlights the states including output set based on real representation in database.

```
With tab_with (
    Select t2.*
     from TAB1 t1, TAB2 t2
      where t1.ID = p_state.ID
         AND t1.BD= p_state.BD
          AND t2.ED<=t1.BD)
 Select *
   from tab_with
     where ED= (select max(ED)
                  from tab);
```

Figure 6. Getting previous state

- **defined_timepoint(t)**
- **defined_interval(t_1, t_2, [CC | CO])**

Figure 7. Event_definition

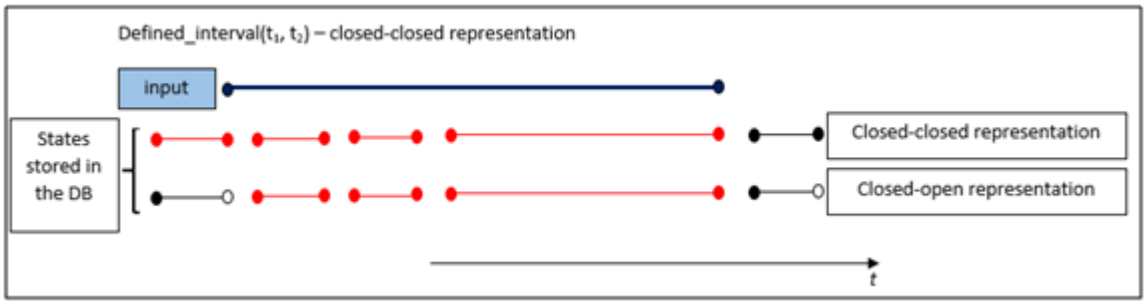

Figure 8. Closed-closed input representation

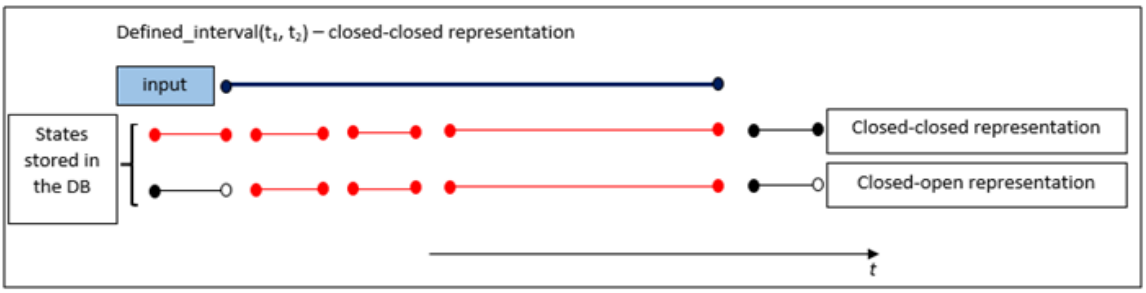

Figure 9. Closed-open input representation

The Following Block Shows The Transformation To An Existing Select Statement Syntax. We Assume That The Object State Is Bordered By The Beginning Of The Period (Bd) And End Date (Ed). Terms T_1 And T_2 Represent Boundary Of The Monitored IntervaL (Defined_Interval):

6. Epsilon_Definition

For the purposes of changes and progress monitoring over the time, it is convenient to define rules that affect the size of the

- **defined_timepoint(t):**

  --for closed-closed time validity representation <BD, ED>.
  ... *where BD <= t AND t<=ED* ...

  --for closed-open time validity representation <BD, ED).
  ... *where BD <= t AND t<ED* ...

- **defined_interval(t_1, t_2, CC | CO):**
  - **closed-closed input interval representation <t_1, t_2>:**
    - Validity modelled using **closed-closed** representation:
      ... where BD <= t2 AND ED >= t1 ...
    - Validity modelled using **closed-open** representation:
      ... where BD <= t2 AND ED > t1 ...
  - **closed-open input inteval validity <t_1, t_2):**
    - Validity modelled using **closed-closed** representation:
      ... where BD < t2 AND ED >= t1 ...
    - Validity modelled using **closed-open** representation:
      ... where BD < t2 AND ED > t1 ...

Figure 10. Transformation to existing syntax – Event_definition

output processed sets. EPSILON_DEFINITION is the determination of the method by which it is possible to filter out irrelevant changes, especially in sensor data. Each referenced temporal attribute may have defined the precision – relevance – minimal value of the significant change - Epsilon (ε) value. If the difference between two consecutive values of the attribute is less than the value of the Epsilon (ε) parameter defined for the corresponding temporal column, this change will not appear in the result set returned by the Select statement. If this clause is not used, then the default value of the minimum change (ε = 0) is used. Thus, any change will be processed regardless the relevance.

To use this functionality, it is necessary to define function that will map values of the non-numerical data types to the types in order to quantify the change. For each data type, we need to define a Map function (or use implicit conversion function). Input parameter of this function is the value of the primary data type, the result returned is a numeric value (integer, float, longint, ...):

```
Create or replace map function f_date_type(val data_type)
return longint
is
begin
  return transformed_value(val); -- into longint;
end;
/
```

Figure 11. Map function

The diagram in figure 12 characterizes the use of the Epsilon (ε) principle for the definition of medically processed data – brain tumour detection. The input values are first filtered based on the position (only areas, where anomaly, respectively tumour can be located, are monitored). Then the marker values are compared during the time evolution (tumor markers means a substance, usually a protein, the occurrence of which indicates the presence of cancer in the patient´s body [15]).

If the new marker value (m_{new}) expresses significant change:

$$|m_{new} — m_{old}| > = \varepsilon$$

specific algorithms checks incorrect positivity. If not detected, the value is stored in the database. This process reduces the amount of stored data.

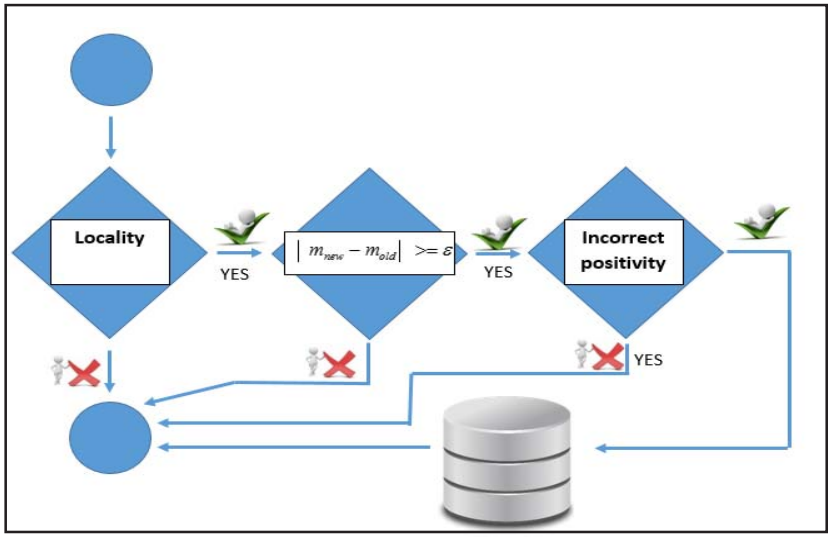

Figure 12. Incorrect positivity (brain tumour detection)

Figure 13 shows the marker values without using Epsilon (ε) definition, whereas figure 14 uses this approach - (ε = 0.5%). Note that if you do not use Epsilon (ε) definition, complete image is stored. However, by using Epsilon (ε) principle, not processed values are replaced by evaluated areas from the past. Green color borders region of interest (which should be monitored over the time).

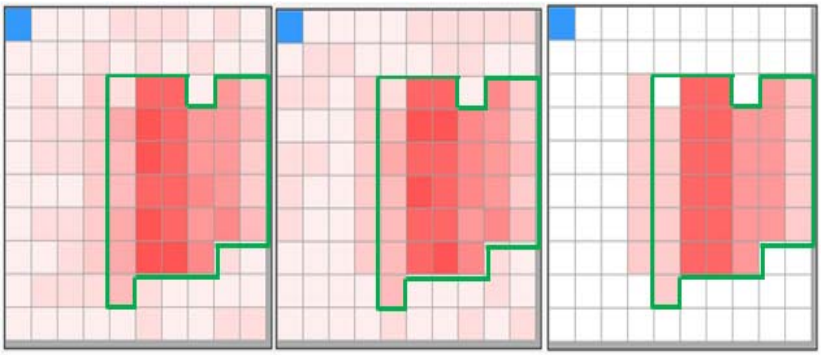

Figure 13. Solution without using Epsilon (ε) approach

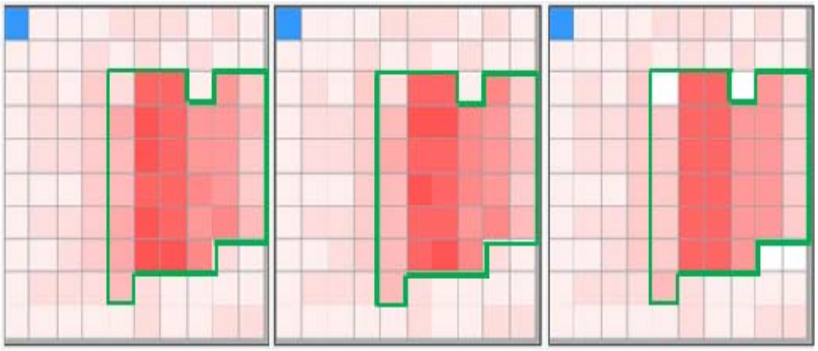

Figure 14. Solution using Epsilon (ε) principle

The following expressions illustrate the transformation to the existing *Select* statement syntax

```
--implicit conversion:
where ABS(new_val - old_val) >= Epsilon

--explicit conversion:
where ABS(f_data_type(new_val) - f_data_type(old_val)) >= Epsilon
```

Figure 15. Epsilon (ε) transformation

In this approach, however, specific situation can occur – attribute value change rate is high, but the difference between actual and previous value is lower than Epsilon (ε) parameter. Thus, in global view, the progress reflected to time t_1 can be significant. The problem also expresses the following figure.

Suppose attribute value $A = 5$ (represented at time t_1) and Epsilon temporal change parameter value $\varepsilon = 1$. Next figure shows the progress of changes - observed value is still growing, but not so much, thus the neighboring changes difference is not greater than defined parameter ε (figure 16).

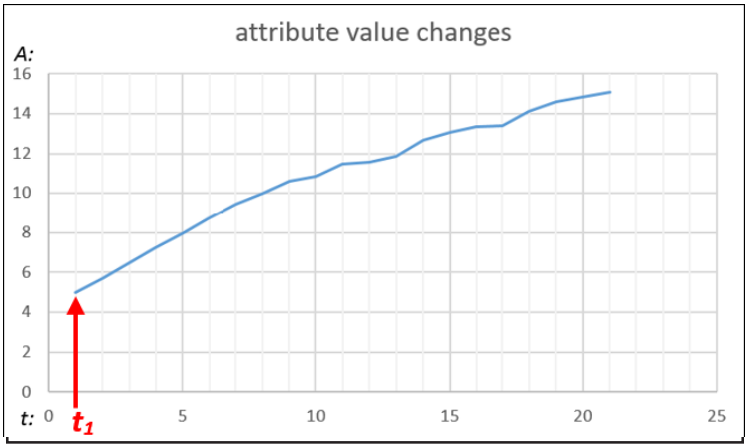

Figure 16. Time changes evolution

Therefore, EPSILON_DEFINITION clause allows you to define not only management for neighboring changes. Another optional parameter is the size of a controlled time frame (time_frame). Each column in the temporal clause EPSILON_DEFINITION is characterized by the parameter Epsilon ([) and also time frame. When defining the time frame, you need to specify the method of processing - the smallest granule – time point or attribute change.

The last option of this method is the fixed (global) referential value, thus processed frame is still growing during the time.

```
EPSILON_DEFINITION_attribute A1(ε1, [time_frame1], [frame_type1],
                                [referential_value1]),
EPSILON_DEFINITION_attribute A2(ε2, [time_frame2], [frame_type2],
                                [referential_value2])
...
```

Figure 17. Epsilon (ε) definition

7. Monitored_Column_List

The clause MONITORED_COLUMN_LIST as the extension of the Select for temporal approach allows list of columns definition, which are relevant for processing and should be monitored. This list does not need to be identical to the first part of the Select statement, however, it can consists only of the temporal attributes (not conventional or functions).

```
MONITORED_COLUMN_LIST(attribute1, attribute2, attribute3, ...)
```

Figure 18. Monitored_column_list

If there is a change of at least one column defined in clause MONITORED_COLUMN_LIST, resulting command will reflect this change. If the clause is not specified, it will automatically be replaced by monitoring all temporal attributes:

```
MONITORED_COLUMN_LIST(*)  --all temporal columns monitoring
```

Figure 19. All temporal columns monitoring

8. Experiments

Our experiments and evaluations were performed using medical information system – measured and processed parameters were performed using volunteers – long term magnetic resonance imaging (MRI) results monitoring [9] [15].

All experiments were provided using the Oracle 11g database system. In this part, total number of records in main structure was 10 000, each record contains 10 MRI results.

Time to get required data and size of the structure – experiments in this section are based on time comparison of the implemented temporal solution on column level with the standard uni-temporal system and also extension by the transaction management of the developed system compared to bi-temporal structure (uni-temporal approach on object level extended by the definition of the transaction time validity).

Although in general we are talking about transaction management, processing in this meaning is based on measurement error reduction. If it is possible to reduce measurement error, new transaction inserts the corrected marker value into the database obtained by the approximation and monitoring the progress and dependencies of marker values over time.

The first model deals with the standard uni-temporal approach (reference 100%) based on object level. In comparison with the temporal structure on column level (model 2 and 3), there is significant acceleration of the system:

without using Epsilon (•) principle (model 2):

- Size: 41,60%.
- Time to get current image (T_1): 73,88%.
- Time to get all life-cycle MRI data result: 59,98%.

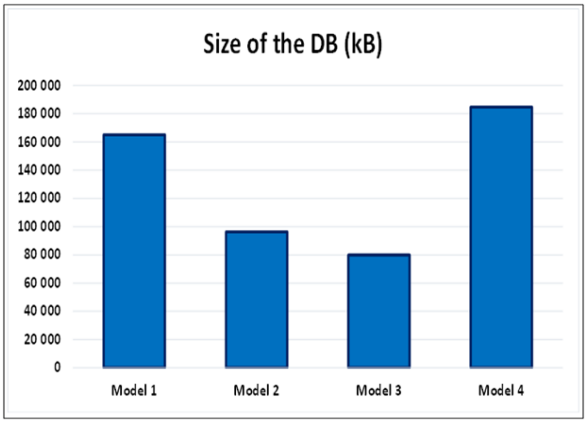

Figure 20. Size of the DB

using Epsilon (•) principle (model 3):

- Size: 51,55%.
- Time to get current image (T_1): 73,88%.
- Time to get all life-cycle MRI data result: 65,97%.

The last (4th) model represents transaction processing modelled by the extension of the primary key using transaction time definition, it provides slowest performance.

Epsilon (ε) parameter has been used, which separates the objects of interest (anomalies, tumours) from other brain tissues. However, it is very important to set the appropriate value of the parameter. Too high value can cause the reduction of the potential anomalies from the output image [15]. Figure 14 shows the size – standard approach (black color) and epsilon value dependency. If the system for incorrect positivity detection is used, the size requirement is lower (red color).

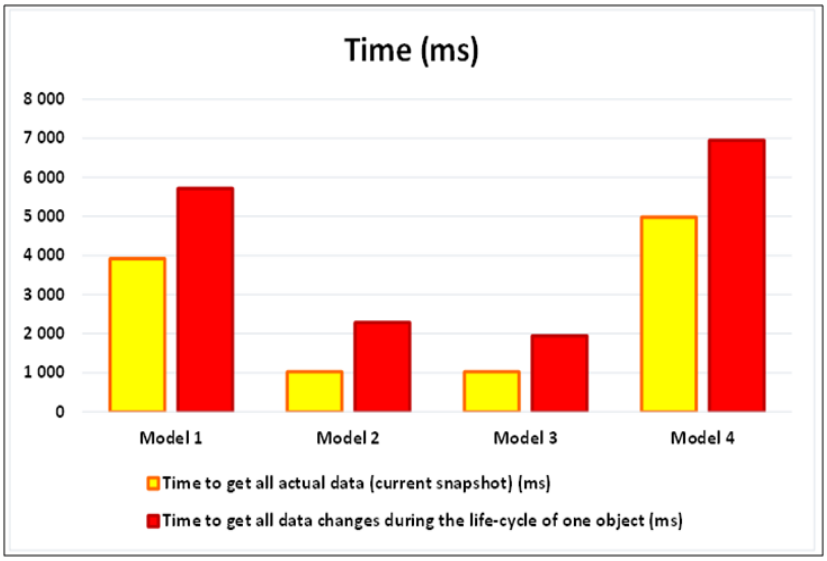

Figure 21. Results - time

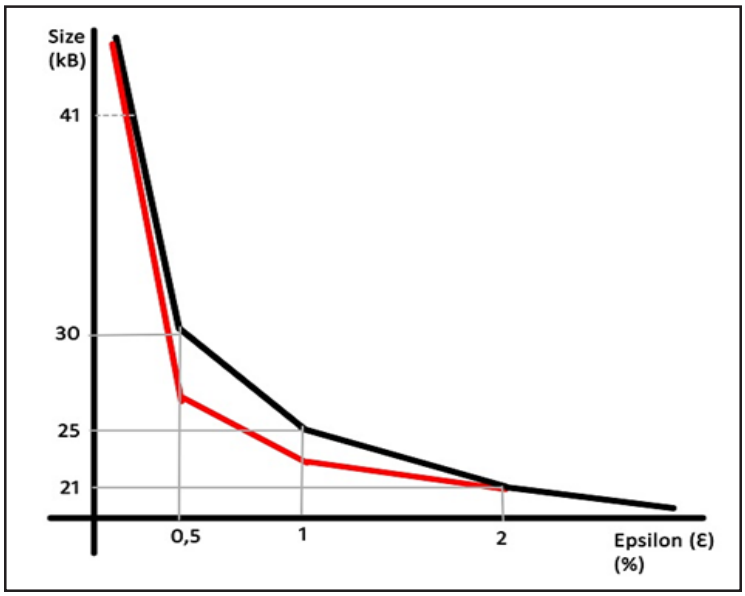

Figure 22. Epsilon value dependency

9. Conclusion

Conventional database object is represented by one row – current state of the object, whereas temporal database system offers processing object valid data and their changes and progress in time. Data processing in temporal environment requires access to the whole information about the evolution of the states during the life-cycle. Effective managing temporal data is the core of the development and can be used for decision making, analyses, process optimization, which is very significant factor in industrial environment.

	Uni-temporal system	Uni-temporal system	Uni-temporal system	Bi-temporal system
	Object level	Column level	Column level	Object level
	---	---	Epsilon approach	---
	Model 1	Model 2	Model 3	Model 4
Size of the DB (kB)	164 940	96 312	79 912	184 552
Time to get all actual data (current snapshot) (ms)	3 921	1 024	1 024	4 978
Time to get all data changes during the life-cycle of one object (ms)	5 712	2 286	1 944	6 950

Table 1. Experiment Results

Problem of current temporal paradigm is weak support for data management. Simply, temporal systems proposed in the recent past do not offer sufficient power to manage large volumes of data with emphasis to reliability and effectiveness of the statements. Most significant is the Select statement, which can be considered as the main part of the DML statements, also the critical performance factor of the whole system. New definition of the Select statement extends conventional principle by adding clauses for time management. In sensor data processing, problem is more visible due to the precision of the measured data. Therefore, layer with Epsilon (e) principle has been defined. Moreover, all new clauses can be directly transformed into existing syntax.

Temporal data processed over the time are usually large. The processing requires sophisticated access methods. In the future, we will focus on various index structure creation, index distribution, which can improve the performance of the system, too. *Select* statements to be executed will be transformed based on index structures to improve performance of the system.

Acknowledgment

This publication is the result of the project implementation:

Centre of excellence for systems and services of intelligent transport II., ITMS 26220120050 supported by the Research & Development Operational Programme funded by the ERDF.

PODPORUJEME VÝSKUMNÉ AKTIVITY NA SLOVENSKU
Projekt je spolufinancovaný zo zdrojov EÚ"

.

References

[1] Date, C. J. (2006). Date on Database. Apress.

[2] Date, C. J. (2007). Logic and Databases – The Roots of Relational Theory, Trafford Publishing.

[3] Date, C. J., Darwen. H., Lorentzos, N. A. (2003). Temporal data and the relational model, Morgan Kaufmann.

[4] Hubler, P.N., Edelweiss, N.(2000). Implementing a Temporal Database on Top of a Conventional Database,. *Conference SCCC '00*, p. 58 – 67.

[5] Jensen, Ch. S. (2015). Introduction to Temporal Database Research, Web: http://infolab.usc.edu/csci599/Fall2001/paper/chapter1.pdf - Online January.

[6] Jensen, Ch. S., Snodgrass, R. T. (2000). Temporally Enhanced Database Design, MIT Press.

[7] Johnston,T. Weis,R. (2010). Managing Time in Relational Database, Morgan Kaufmann.

[8] Kimball, R. (1996). The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley & Sons.

[9] Kvet, M., Matiaško, K. (2014). Transaction Management,. *CISTI, Barcelona*, p.868-873.

[10] Kvet, M.,Vajsová, M. (2014). Transaction Management in Fully Temporal System, *UkSim, Pisa*, p. 147-152.

[11] Kvet, M., Matiaško, K., Kvet, M., (2014)Complex time management in databases, *In Central European Journal of Computer Science*, 4 (4), p. 269-284.

[12] Lewis, P., Bernstein, A., Kifer, M. (2000). Databases and Transaction Processing (An Application Oriented Approach), Addison-Wesley.

[13] Maté, J., (2011). Transformation of Relational Databases to Transaction-Time Temporal Databases, *In ECBS-EERC*, p. 27-34.

[14] Oszu, M. T., Valduriez, P. (1991). Principles of Distributed Database Systems.

[15] Pianykh, O. (2008) Digital Imaging and Communications in Medicine, Springer.

[16] Snodgrass, R. (2000). Developing Time-Oriented Database Applications in SQL. Morgan Kaufmann Publishers, San Francisco.