Classification Rules for Temporal Databases

Kvet, Michal and Matiaško, Karol

Abstract: ***Temporal database management is currently an inevitable part of information technology. Intelligence and robustness of the systems are enhanced by the decision making reflecting the current data images, as well as historical data and future plans. Many times, existing solutions do not provide sufficient power, because they cannot be unambiguously identified and classified. In this paper, we deal with temporal architectures highlighting granularity and propose own complex classification rules to determine individual access principles, technology, architecture, and security aspects. The classification consists of 14 evaluated aspects. In this paper, discussion about the undefined states and reliability provided by the consistency is done, as well.***

Index Terms: ***data integrity, data rating, temporal classification, temporal granularity, undefined states***

1. *Introduction*

THE importance of the data in the industrial environment forms the inevitable part for the control systems, management, decision making, and problem identification. Their significance even sharpens the aspect of intelligence of the current information systems. It is, therefore, non-suitable to deal only with currently valid data, but the processing must be shifted to the temporal database system level. Current database technology allows you to manage and evaluate the data during the whole life cycle. The important role just plays the efficiency of the whole process covering the storage principles, architecture, query management, and reliability. There are many technologies, structures, and access principles surrounding and covering databases. However, there is no complex classification resulting in the necessity to describe an environment step by step in the research, as well as in the industrial manner. This paper aims to fill this negative aspect and propose complex classification rules defining the structure, access principles, and operations.

Manuscript received May 11, 2020.

Michal Kvet, Karol Matiaško, Faculty of Management Science and Informatics, Zilinska univerzita v Ziline, Slovakia. E-mail: Michal.kvet@uniza.sk, Karol.matiasko@uniza.sk

Thanks to that, temporal technology can be described just with a few characteristics. This paper chronologically describes and evaluates temporal principles from history up to current trends. It tends to summarize knowledge and technology regarding temporality. We propose classification rules for temporal databases shifting the management from the conventional principles, where just current valid data are reflected, to the temporal aspect, modeling the whole time-spectrum and object states evolution.

The evolution of the temporal paradigm started just after the first releases of the conventional relational database systems. Conventional databases are characterized by storing only currently valid data, which means that historical images cannot be obtained the the definition. Thus, if any data tuple is updated, the original value is replaced by the newer version, and original ones are thrown away. Later, developers and researchers concluded, that historical images would provide relevant added value for the processing, management, and evaluation. Industry and machine parameter values can be optimized to reach better performance resulting in reducing costs, increasing performance, and reaching a better score.

The first temporal approaches were really simple and originating from the transaction structures. Each data change is logged before the processing itself using the two-phase protocol [2]. This log consists of the reference to the encapsulating transaction, executed operation, and UNDO and REDO image of the changed tuple. From the temporal point of view, if the log files are accessible, it is possible to reconstruct the image, as it was in the past, although it is a very complicated and demanding process. In the standard environment, all log files form a cyclically linked list and are sequentially rewritten, if they do not cover relevant data of the active transaction. The Archiver background process allows you to copy the log file into a separate storage repository before its overwritten [1]. Thanks to that, historical values can be obtained. The process of reconstruction requires access to the current image, as well as loading and parsing all the log files, which were created by any transaction executed after the timepoint that we are looking for. The main disadvantage is reflected in efficiency, namely it is not possible to evaluate directly whether a particular log file consists of data modifying the required object or not. As a consequence, the process is too demanding and time and resource consuming. Technically, it is possible to start not only with the current data image, but any time point from the backup can be used, however, in that case, the whole backup must be loaded into the system, which is not possible to accept in the operative decision making.

Flashback is another technology allowing you to automate the reconstruction process based on the transaction, time, or System Change Number (SCN). It offers to mark objects or individual attributes, which are monitored over time in the historical spectrum. Thanks to that, the size of data to be parsed and loaded is lowered. Specific data structures are used to optimize the whole process, as well [1] [5] [6].

The limitation of the previously mentioned techniques is just the time spectrum and efficiency. It is too complicated and resource-consuming, as well as it requires too much time. Moreover, it can manage only currently valid data and history. No information about plans, future set parameters, and settings can be stored resulting in the necessity to develop a separate structure for the data valid in the future, forcing to manage additional rules to ensure reliability and consistency of the data and the system itself.

Several architectures for managing temporal data have been developed over the years. They are, mostly, based on the object granularity.

This paper aims to propose and describe models based on their architectures of the attribute and synchronization groups. Whereas several principles and techniques can be applied, it is necessary to define the classification rules to cover the temporality. Therefore, the main part of the paper proposes its complex techniques that participated from [6] [8] [10]. Section 2 deals with the temporal aspects from the granularity point of view. In section 3, undefined states and attribute values are highlighted defining storage and access principles. In section 4, transaction management is described. It is, namely, not suitable to use conventional principles of transaction management, whereas consistency in a temporal manner should be emphasized. Therefore, we propose our technique of collision management. Section 5 covers the classification itself.

2. Temporal Architectures – Granularity Aspect

Each object in the relational database can be clearly distinguished by a unique identifier – primary key (PK), which can consist of one or more attributes. If the time spectrum attributes are added to the identifier, the temporal model is created. In that case, each object is delimited either by the identifier itself (ID), but also by the time spectrum expressing mostly the validity interval. Thus, the object is characterized by the multiple states, but in different time points or intervals forming individual versions. The time spectrum can be modeled by the begin (BD) and endpoints (ED) of the validity or by just one attribute (begin point). In that case, each newer version of the object automatically delimits the previous one. Generally, time spectrum expresses validity, but the transaction interval expressing time of transaction approval can be used (BD2, ED2), as well. In principle, multiple time spectra can be present. The characteristics of the model reflect the number of time spectrum zones located in the system. The uni-temporal model deals with one spectrum, the bi-temporal approach is characterized by two spheres to be handled. In general, the multi-temporal model can be identified. Data model principles are shown in fig. 1. Label ID defines a set of attributes defining the object itself. It uses the same principle as in a conventional database. Validity is the most often used temporal spectrum forming a uni-temporal table. It is defined by the BD and ED reflecting validity interval. Thus, the object state temporal identification (primary key – PK) is a composite consisting of three attribute sets – ID, BD, and ED. Note, that ID is itself a set and can consist of several attributes.

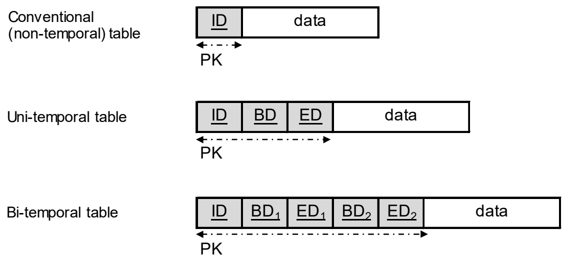


Figure 1.Object architecture [10]

If the main processed granularity is shifted to the column, attribute-oriented model is created. When using the previously described model – object-level – each change forces the system to create a completely new state. Thus, if some column values are not changed, original values are copied to newer states uncovering the storage and performance efficiency. Moreover, identification of real change requires to manipulate with two consecutive states. The attribute system encloses each attribute by its validity. Therefore, the complex data image arises as the projection of individual attributes and their validity in the defined time interval. Thanks to that, the size of the whole structure is optimized, whereas only veritable change is present and stored. On the other hand, if several attributes are changed at once, each change forces the transaction manager to add a new row to the temporal layer. The architecture of the system is shown in fig. 2. It contains three layers dividing current and inactive data into separate levels. The first layer consists of only currently valid data, which can be queried directly outside the temporal system. Temporal evaluation and change management are covered by the hearth of the system – second layer with a temporal table. It manages all the changes and interconnects them to individual layers and objects. Non-actual values (historical and future plans) are separated in the third layer. Temporal transaction manager, as an extension of the background process set, covers the whole process and ensures consistency, as the transformation from the current data layer into historical or vice versa, from the future plans into the current states. Performance definition and comparison to other architectures can be found in [9].

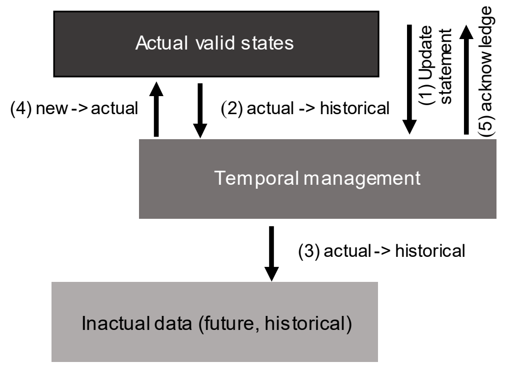
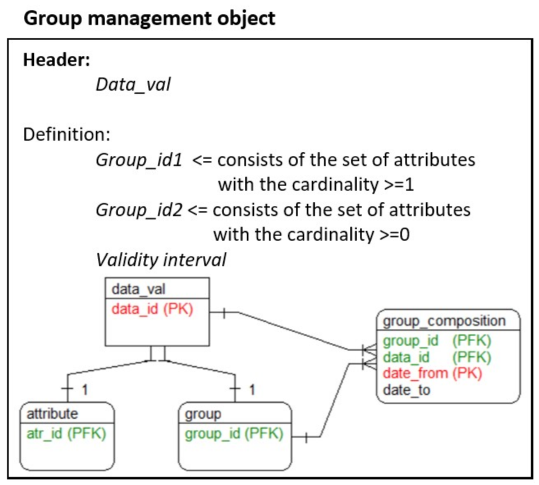


Figure 2.Column architecture [9]

The limitation of the attribute granularity is just the case of multiple attributes are updated synchronously. In that case, each attribute requires one insert operation in the temporal layer. Therefore, we introduce our system with synchronization groups. The data model is shown in fig. 3. Attribute reference is replaced by the expression data_val, which can express the attribute itself but can be linked to the temporal group covering multiple attributes. Thus, the attribute is not referenced directly, but in the temporal layer, an object reference to the synchronization group is located. Thanks to that, performance is ensured. Physical implementation and architecture can be found in [8] [14]. In this paper, we introduce an improved version of the management and data model.


Group management object

Header:
Data_val

Definition:
Group_id1 <= consists of the set of attributes with the cardinality >=1
Group_id2 <= consists of the set of attributes with the cardinality >=0
Validity interval

Figure 3.Group architecture

Another processing level is formed by the spatial aspect of the processing. In that case, however, it is not necessary to change the identification model of the data tuple, just new attributes defining spatial position are added forming Spatio-temporal architecture [15]. The database itself can be located either in the on-premise, but the cloud technology is widespread now. Current temporal trends are defined in [13]. Big data architecture is described in [16].

Our proposed architectures delimit the data by the temporal validity intervals. Many times, however, large objects (LOBs) are not covered by the temporal change monitoring. Temporal data archive characteristics and architecture dealing with the multimedia can be found in [17].

3. *UNDEFINED STATES*

As already mentioned in the previous sections, there can be several models and multiple approaches for dealing with the states themselves. If each object must be defined at any time point, an undefined state must be stated explicitly. The point is, how to model and express the undefined state. Generally, the undefined state does not automatically mean, that all of the attributes are unknown, respectively do not hold relevant data. Each object definition consists of the list of attributes, which can be undefined, but the object from the outer view can be considered as (partially) valid. There can be also a list of attribute sets, which decompose the object state resulting in a complete invalid object state.

Data to be stored in the database can be produced by various sources, data streams, or sensors. A typical problem is just the uncertainty of the communication channel, like wireless connection, etc. For these purposes, the input database connector is extended with a reliability module to cover the data correctness. If some data portion, attribute, or a whole object state is not defined, it must be clearly stated. The first representation is NULL expression, that the values are not present. In our solution, we do not use such notation for several reasons. First of all, such value does not hold sufficient information value. What does it mean in practice, that the object, group, or just one attribute is NULL? Does it mean, that the system cannot obtain the value? Or that the input value cannot be evaluated as reliable? Or the value came to the system with latency? Moreover, NULL values are not part of the index structures, forcing the system to scan all data blocks sequentially [11] [12]. Shifting to the temporal spectrum, the problem is even sharper. From the NULL value in the time notation, we cannot even distinguish that the event has not occurred yet. Therefore, in many literatures and papers, MaxValueTime representation is used, physically modeled by the maximal date to be processed (31/12/9999). Once again, when managing data, a particular representation must be extracted from the real timepoint in the future. Based on the previous analysis, we concluded that it is necessary to develop our own solution for dealing with the undefined values. The principle is based on the memory object for each data type or the object definition. If the undefined value is present, a pointer to the memory object is used instead. Based on the definition, such a pointer can be part of the sorting and index itself. Undefined object definition is always present in the instance memory, it is created during the mounting process of the instance, managed by the background processes and released during the instance drop (closing operation). It is supervised by the System Monitor background process and it is part of the core. Thus, if corrupted or there is an attempt to drop it, the whole instance is automatically and immediately closed. Object state can be routed to the object part of the memory object (Undef_obj), the attribute is pointed to a particular position based on the data type. If the synchronization group is present, it is operated on the same principles as the undefined object itself. The architecture is composed of the attribute definition and object definition separately.

If the time validity interval is defined explicitly, a virtual calendar must be created for the defined data precision in time, by which the undefined time frames can be identified.

4. *Consistency Management*

Transaction in the temporal system is defined in the same manner as in the conventional approach – atomicity, consistency, isolation, and durability. When dealing with the consistency, the time spectrum must be handled, as well. It means, that each referenced object must cover the time interval of the child record in the hierarchy. Naturally, such an object can be defined by multiple states. There cannot be, however, time period during which the parent object is undefined or non-reliable. In terms of transactions, data collisions must be taken care of, too. As mentioned, each object is defined by no more than one valid state anytime, with emphasis on versioning and state corrections, as well. It requires a temporal manager to be extended to cover such functionality. In our proposed solution, we define four collision rules to ensure the reliability and consistency of the system.

1. Complete-reject automates the collision management by refusing the new state, which is in collision with the existing state.
2. Complete-approve is characterized by accepting the new plan, thus the existing state validity is shortened.
3. The partial-approve method is similar, in that case, the validity of the new state is shortened to limit the collision. This rule can be extended by the parameter shifting, which can hold value True or False. If True is selected, the end point of the new state validity is shifted to the right, so the original validity duration remains unchanged. Vice versa, if the False option is used, the original validity end point is applied.

The whole management is secured by our three-phase Commit protocol shown in fig. 5. BOT denotes Begin of Transaction, EOT expresses End of Transaction.

The following rules apply to this method:

- The transaction cannot change the data in the database until it is confirmed (before reaching RC point).
- The transaction cannot change the data in the database unless all conflicts between the time intervals of individual object images - states in time have been resolved.
- A transaction cannot be committed if transaction rules have also been applied that does not allow the modification of the validity period of existing states.

The advantage of this method is that in the event of an accident before reaching the RC point, none of the changes have been recorded in the database and therefore no changes are required. Another advantage is that if a collision situation occurs that does not allow the transaction to continue (e.g. Restricted rule is used), there is no need to modify the database as no changes have been applied yet.

If an accident occurs after sending an RC signal, then the solution is based on rolling back the transaction.

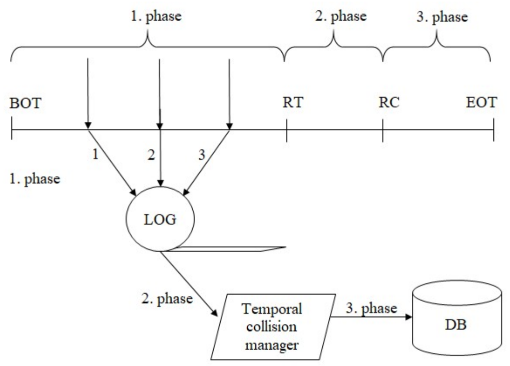


Figure 5.Three-phase Commit

5. CLASSIFICATION

Gradual extension of temporal approaches and time-oriented database systems brought the need for formalization and categorization of individual types. There was no organized form for managing types in terms of modeled granularity or system architecture, so in 2015 we decided to create normative classification rules [9] originating from [7], which consist of three types - α / β / γ /, where:

- **α** represents the type of database system used.
- **β** characterizes the type of temporal structure.
- **γ** denotes the type of online transaction processing (OLTP).

As part of the development of new temporal techniques and the definition of robust tools, to which we also contributed significantly through our scientific and experimental activities, it was necessary to extend the categorization to include additional attributes. Besides, the above definition has also been slightly modified - in the new system, we distinguish between non-timed (conventional) models and static attributes that were originally considered identical in terms of time management.

In this paper, we propose a complex extension and new classification rules. The definition consists of the following parts **A / B / C / D / E / F / G / H / I / K / M / O / P / R:**

A – database system type

- **N** - no database system support (e.g. file system).
- **R** - relational DBS.
- **H** - object-relational DBS.
- **O** - object-oriented DBS.
- **X** - DBS storing data in the form of XML documents.
- **N** - non-relational DBS.
- **x** - the unspecified type of DBS used.

B – Temporal dimensions

- **0 - S** - static data (codebooks, configuration parameters, constants, etc.)
- **0 - C** - conventional (non-timed) data - objects in which changes are not tracked over time.
- **1 - U** - uni-temporal data (records are limited by the validity).
- **1 - T** - uni-temporal data (records are delimited by the timestamp of insertion or status update - transactional validity without recording of the state validity limit itself).
- **2 - B** - bi-temporal data bounded by the validity and reliability expressed by the transactional validity.
- **2 - O** - bi-temporal data bounded by the validity and synchronization timestamp. This modeling method is used in systems with asynchronous processing in offline mode followed by synchronization with the central system (fig. 6). There are different times for the local system and the reflection in the central database node (synchronization timepoint).

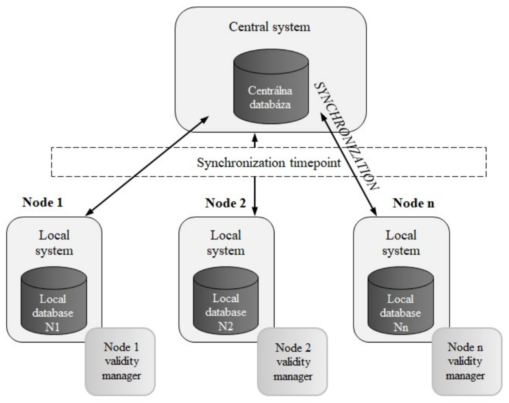


Figure 6. The architecture of the synchronization tool

- **3 - F** - multi-temporal data with three dimensions (validity, transactional validity, and timestamp of the transformation of the states valid in the future to the current ones).
- **3 - O** - multi-temporal data with three dimensions (validity, transactional validity, and synchronization timestamp).
- **4 - K** - multi-temporal data with four dimensions (validity, transactional validity, timestamp of the transformation of states valid in the future to the current and timestamp of synchronization).
- **x - M (x)** - multi-temporal access to data, where "x" represents the number of dimensions.

C – the type of granularity

- **O** - object granularity (any change in the temporal attribute causes a completely new state to be created; effective, if all individual changes are synchronized, they occur at

uniform times).
- **C** - column granularity (each change is made at the attribute level; the complete state of the object is created by composing individual images of the attributes).
- **G** - granularity at the sync group level.

D - representation of time on the axis - time spectra that are processed and evaluated in the temporal system (several options can be used):
- H - historical states.
- C - currently valid states.
- F - states of objects which validity begins in the future (the solution must include an automated projection of these plans into a set of current states).
- E - use the Epsilon (Ɛ) principle. Only significant changes in the values of individual attributes, respectively attribute groups. The value difference must be greater than Epsilon, otherwise, the validity of the original state will be prolonged (or retained).

E - Online transaction processing (OLTP)
- N - no transaction processing support.
- L - OLTP with logs.
- O - OLTP at the level of temporal objects.
- A - OLTP at the level of temporal attributes of the object.

F - definition of the way of the collision of states:
- R - restriction.
- P - partial validity.
- C - complete validity.
- W - notification (the solution is left to the user). Ignoring collisions can cause inconsistent states and database integrity violations.

G - definition of indices (structure, type, and characteristics of indices) [3] [4] [11]

H - index localization [8] [11]:
- **T** - data is stored in the same tablespace as the data itself.
- **S** - data is stored in another tablespace, optionally with a different data block size:
    - by the same size (8kB by default) inherited property of the main database structure,
    - 2, 4, 16, 32, 64kB data block.
- **D** - indexes are distributed to individual DDBS nodes.

I - represents the use of an own Flower Index Approach (FIA) to eliminate the impact of using the Table Access Full (TAF) method.

If no suitable index for the query is present, the system must use the TAF method to scan all blocks associated with the table regardless of the data, individual block holds. As a result, inappropriate, fragmented, and even empty blocks are loaded to the memory for the evaluation. To remove such an impact, we have proposed our own FIA approach. It uses the main index technique, that the leaf layer of the B+tree index (most often used database index) consists of the direct pointers to the data in the database (ROWID). ROWID is a unique row identifier and contains the address to the database file, block, and references the position of the row inside it. Our approach, however, uses only block level for the processing. Thus, only relevant blocks are loaded. If no ROWID points to the block, it is clear, that it does not hold data of the table. If the industrial environment, individual parameters, and object values are changed dynamically, the size of the row can vary resulting in creating fragmentations inside the block and the file itself. Our proposed solution, therefore, limits the impact of the fragmentation, no structure rebuilding is necessary to be executed. Thanks to that, resource extensive method can be omitted and performance does not degrade.

Individual records are located in the appropriate database blocks using pointers, so it is not necessary to sequentially check all the blocks associated with the table. The index so defined is permanently available in the server instance memory (as long as the STATUS instance is OPEN):
- **Y** - yes - FIA access can be used.
- **N** - no - use of FIA access is prohibited.

The principle of the proposed method is shown in fig. 7. Notice, that several ROWID values point to the same data block, which is then marked as already processed. After the loading process of the block into the memory, it is completely scanned to locate data.

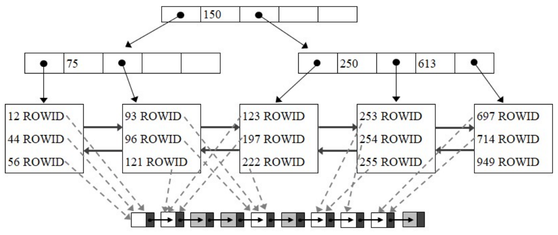


Figure 7.FIA index architecture

K - Expression of security rules with emphasis on audits:
- **A** - audit at the level of whole database objects (e.g. whole tables) = database audit.
- **T** - Trigger audit.
- **F** - processing of attributes = level audits =

fine-grained audit.

M - the architecture of the database system itself:

- **S** - Single instance.
- **R** - Real Application Cluster.
- **S** - Streams.
- **D** - Data Guard.
- **C** - Cloud.

O - selected integrity rule applied in time to reference the integrity and composition of objects (ISA hierarchy):

- **S** - the strict condition of coverage - the object in the role of "child" must be completely covered by the valid status of its "parent", respectively ancestor in the hierarchy. Thus, foreign keys are not only checked at the existing level of the object, to which the record is referenced. Individual time intervals of validity are monitored and evaluated, as well.
- **R** - relaxed coverage condition - reference integrity ignores validity intervals completely.

P - state classification

- **Exact** - each object must be defined by exactly one state at any time. In this case, undefined object states are also managed explicitly.
- **Max** - this approach is defined by a relaxed rule, just correctly defined states are recorded, undefined and incorrect states are calculated automatically, however, they are not stored in the database.

6. *Conclusions*

The conventional database approach does not bring sufficient power for intelligent information systems or management systems in the industry, whereas only current valid data are modeled, monitored, and evaluated. The temporal paradigm has brought the wide possibilities to manage data over the whole life spectrum of each object with emphasis on the information value. Thanks to that, it is easily and effectively possible to deal with decision making, creating prognoses, and complex analytics. In this paper, we deal with temporal architectures, we bring our own 3 level architecture of the column temporal granularity, in which each attribute is bordered by the time frame expressing modeled and processed time interval, mostly validity. Such architecture aims to limit the efficiency of object architecture and remove the necessity for storing the same data values many times. Another proposed temporal system interconnects object and column granularity by defining the temporal group. In this case, if multiple attribute values are changed during the same event, the synchronization group can be created and managed as the one composition, not individual attributes separately.

This paper deals also with the management of undefined states, the principle of modeling, and identifying such states. We prefer our object located in the instance memory, to which object states can point. Thanks to that, such values can be indexed and effectively-identified.

Individual states must be covered by the transaction rules focusing on integrity and consistency. Several rules are defined to limit the collisions of the states. The relational paradigm of the temporal system requires no more than one valid state for each object any time, therefore collisions must be dynamically identified and removed. Principles are based on removing new or existing state or by changing their validity intervals.

The core part of this paper is delimited by the classification rules of the temporal databases. In the past, there was no complex classification of the system and was very complicated to identify real principles, data flow, and management. Therefore, we proposed basic classification is 2015. In this paper, we extend the principles to cover temporality complexly. Transaction management, architecture, volatility, dimensions, collisions, index management and location, security, or integrity. We propose 14 layers defining each temporal system.

In the past, our emphasis will be on the data distribution in the temporal systems, data loading regarding block size, indices, and fragmentations. After the system and architecture development, new classifiers will be proposed to cover a distributed environment.

Acknowledgment

This article was created in the framework of the National project IT Academy – Education for the 21st Century, which is supported by the European Social Fund and the European Regional Development Fund in the framework of the Operational Programme Human Resources.

The work is also supported by the project VEGA 1/0089/19 Data analysis methods and decisions support tools for service systems supporting electric vehicles and the Grant system of the University of Zilina.

References

[1] Ahsan, K., Vijay, P., "Temporal Databases: Information Systems", Booktango. 2014.

[2] Ashdown, L., Kyte T., "Oracle database concepts", Oracle Press, 2015.

[3] Avilés, G., et al., "Spatio-temporal modeling of financial maps from a joint multidimensional scaling-geostatistical perspective". In Expert Systems with Applications. 60, 280-293, 2016.

[4] Doroudian, M., et al., “Multilayered database intrusion detection system for detecting malicious behaviours in big data transaction”, IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2016.

[5] Erlandsson, M., et al., “Spatial and temporal variations of base cation release from chemical weathering a hisscope scale”. In Chemical Geology. 441, 1-13, 2016.

[6] Johnston, T., “Bi-temporal data – Theory and Practice”, Morgan Kaufmann, 2014.

[7] Johnston, T., Weis, R., “Managing Time in Relational Databases”, Morgan Kaufmann, 2010.

[8] Kvet, M., Matiaško, K., “Temporal Data Group Management”, IEEE conference IDT, 5.7. – 7.7.2017, 218-226, 2017.

[9] Kvet, M., Matiaško, K., “Transaction Management in Temporal System”, IEEE conference CISTI 2014, 18.6. – 21.6.2014., 868-873, 2014.

[10] Kvet, M., Matiaško, K., “Uni-temporal modelling extension at the object vs. attribute level”, IEEE conference UKSim 2014, 20.11 – 22. 11.2014, , 6-11, 2014.

[11] Kuhn, D., Alapati, S., Padfield, B., “Expert Oracle Indexing Access Paths”, Apress, 2016.

[12] Kumar, N., “Efficient data deduplication for big data storage systems”, In Advances in Intelligent Systems and Computing,714, 351-371, 2019.

[13] Lan, L., Shi, R., Wang, B., Zhang, L., Shi, J.: “A Lightweight Time Series Main-Memory Database for IoT Real-Time Services”, Lecture Notes in Computer Science, Volume 11894 LNCS, 2020, pp. 220 – 236.

[14] Li, S., Qin, Z., Song, H., “A Temporal-Spatial Method for Group Detection”, Locating and Tracking, In IEEE Access, 4. 2016.

[15] Moreira, J., Duarte, J., Dias, P.: “Modeling and representing real-world spatio-temporal data in databases”, Leibniz International Proceedings in Informatics, LIPIcs, Volume 142, 2019

[16] Ochs, A. R., et al.: “Databases to Efficiently Manage Medium Sized, Low Velocity, Multidimensional Data in Tissue Engineering”, Journal of visualized experiments JoVE, Issue 153, 2019.

[17] Saany, S. I. A., Rahman, M. N. A., Nasir, A. F.: Temporal based multimedia data archive, International Journal of Recent Technology and Engineering, Volume 7, Issue 5, 2019.