Concept and assumptions about the temporal graph database

Jarosław Koszela[1], and *Paulina* Szczepańczyk-Wysocka[1,*]

[1]Military University of Technology, Faculty of Cybernetics, Institute of Computer and Information Systems, Warsaw, Poland

Abstract. The article outlines existing solutions in the area of graphs and temporal databases. It provides explanation for why the temporal graph database was created. Furthermore, the article also describes the concept and assumptions about the temporal graph database, including a proposal of two methods for representing temporal data in graph databases. Full write method assumes creating a new database object for each state of being. While incremental method writes only such features and relationships that were subject to change. Regardless of the data write method used, the data may be returned in a historically unordered or ordered manner. The article outlines assumptions for both methods of representing data.

1 Graph database

The databases built on the basis of the graph (or network) data model are called graph databases [1]. It is one of the basic types of data models used in ICT systems.

The graph databases belong to the so-called schemaless databases. Such databases - similarly to schema databases - have a schema (model), which, together with all the terms defined therein, is necessary to use the database. The manner of building the schema constitutes the difference between the said databases. In the schema databases, the schema must be implemented before starting the collection of data, whereas in the schemaless databases, the schema is built dynamically, simultaneously with the data inflow [2].

Currently, Neo4j is the most popular database management system implementing the graph data model [3]. The database built on the basis of the above-mentioned system consists of two types of objects: entities (nodes) and relationships between them (arcs). Particular objects (single item) are represented by entities, wheres arcs symbolize certain correlations between them. The objects and phenomena represented by elements of the network database are distinguished on the basis of labels and types. Furthermore, they may be described by properties assigned thereto [4]. The described structure of the data model allows to process tree, hierarchical and network structures, go deeper into the correlations and search for the routes. Furthermore, constant speed of the network ensures high performance of the database - regardless of the amount of data.

Another advantage of the Neo4j system is a possibility of using the CQL language (Cypher Query Language). CQL is a declarative query language, dedicated to Neo4j, composed of a combination of SQL and SPARQL. The system not only allows operations on nodes and relationships, but also patterns. [5]

The organization of the data storage in databases created using the Neo4j system is based on files. Individual files creating the database include the information about only one type of objects (e.g. neostore.nodestore.db file stores the information on the nodes only, whereas neostore.propertystore.db - on the properties assigned thereto).

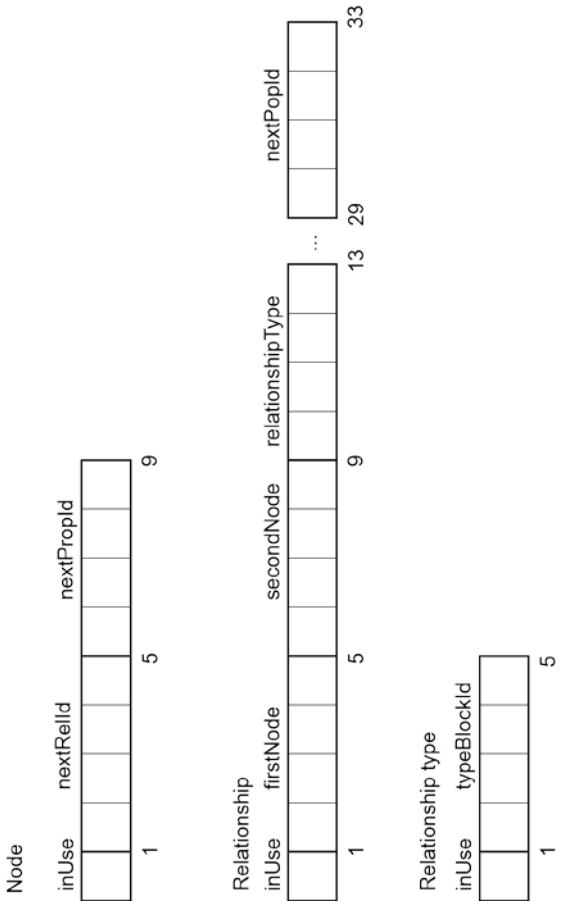

Fig. 1. Organization of data storage in DBMS Neo4j - examples

* Corresponding author: paulina.wysocka@wat.edu.pl

The structure of particular files is similar, as the files are composed of byte sequences describing individual items of the database elements. The description of model elements is shown on the figure 1.

According to the above-mentioned schemas, ID numbers constitute an important elements of the data storage in Neo4j. However, unlike the identifiers used as value-keys by the administrators in the relational databases, they are not attributes given by users, but unique identifiers automatically assigned by the system to all objects in the database, used to facilitate data management.

The organization of the data storage in the Neo4j system does not provide for a possibility of writing temporal data. [4]

2 Temporal databases

The temporal database is the database storing at least one type of the time stamp connected with its elements (e.g. data, relationship, metamodel). The most common type of the time stamp used in temporal databases is the data valid time. It defines the time, during which the fact represented by such data is true in terms of the modeled reality. The data valid time may be represented by the point in time or time period. In the event when the temporal database also includes time stamps for actions performed in the database (transaction time), such database is called the bitemporal database. [6] Such databases are currently the most often used type of the temporal databases. It is also possible to introduce other time stamps to the database - i.e. the multitemporal approach. [7]

The first solution related to temporal databases was proposed by Richard Snodgrass in 1992. The solution was introduced into the SQL language as an extension of TSQL2 and constituted grounds for further work. In 2011, as part of the SQL:2011 standard, extended support for temporal databases was presented. The standard includes, among other things, definitions of time periods, valid time tables, temporal primary keys, temporal referential integrity constraints and temporal predicates. The above-described standard allows to represent data concerning the present time and the past - without a possibility of using the data, whose validity is situated in the future. Unlike the currently used standard, the concept outlined in the article covers a full spectrum of time - including a possibility of storing and processing the data concerning the future.

Time stamps allow to analyze the changes in the status of objects represented thereby in time. The changes usually refer to the changed values of attributes and properties of the objects. However, it is possible to time stamp the schema of the database. Metamodel temporality allows to analyze the changes in relationships and attributes of the objects. It is especially important in case of schemaless databases.

Temporal databases support temporal integrity constraints and supplement relational algebra with time-based operations. Furthermore, to use temporal databases, it is necessary to be able to use the temporal query language based on temporal logic. [8] This type of logic deals with time structure. Two types of temporal logic may be distinguished: linear temporal logic (LTL) and computation tree logic (CTL).

LTL presents time in a linear manner, using the set of ordered points, and allows to determine the status of objects at particular points in time and their changes. On the other hand, CTL presents time in a linear manner up to a certain point, and then branches off to several paths, which represent alternative versions of events (parallel time). Branching logic is usually used in versioning file systems. [9]

3 Structure of the storage of temporal data in graph database

Currently, temporal data and metadata [10] constitute an important element of data analysis and processing. [8-9] Commonly used systems for managing temporal databases are based on relational data models. [11-13] However, the characteristics of phenomena not always allow to accurately represent them in the form of relational models. The processes of data collection and analysis are more and more often executed using the theory of graphs and networks. [1,14]

To construct temporality models in the database that does not implement them, it is necessary to develop a method for storing the "Date/Time" and "Time period" data, modify the manner of storing such data so that they would include the created data types, and to develop other methods for data processing, including temporal logic [12-13,15]. In the subsequent part of the article, two methods of storage of temporal data in the network databases were outlined, together with the assumptions for each of them. The studies were conducted on the basis of DBMS Neo4j. The above-described methods concern the storage of information with the data valid time.

3.1 Full write method

The valid time is the information feature, which shows in which period of time the information was valid. Each object stored in the graph database may have a set of attributes, out of which every attribute may change in time. Therefore, it seems justified to consider the data valid time as an additional attribute of each object. When the value of at least one object feature changes, its status changes as well. The proposed method consists in closing the data valid time describing the status of the being in case of its change and presenting the new (current) status by way of the new object added to the database. The presented method requires identification of particular items and their versioning.

DBMS Neo4j identifies individual objects in the database by automatically assigning them unique ID numbers. The identifiers may be used to identify particular versions of each object. To determine which object version concerns which item stored in the database, it is suggested to introduce an additional ID number.

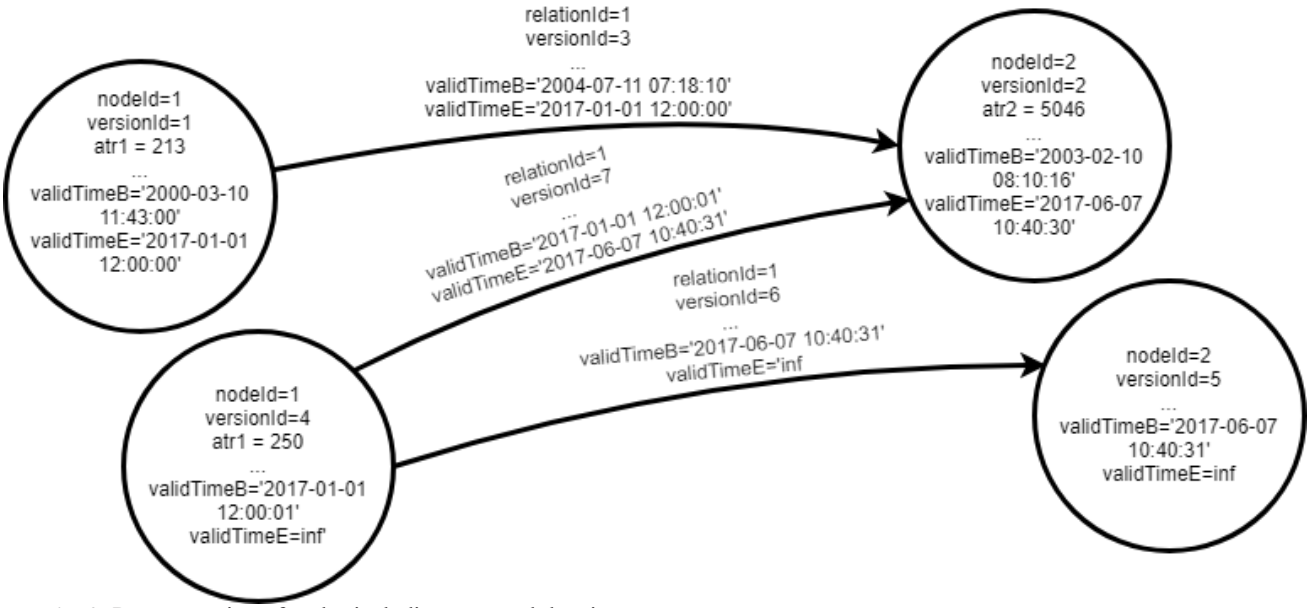

Fig. 2. Representation of nodes including temporal data in accordance with the full write method

Figure 2 shows the suggested schema of storing the information on nodes and arcs. The particular items were distinguished on the basis of "nodeId" and "relationshipId", whereas their versions were designated with "versionId". The "validTimeB" and "validTimeE" mean the beginning and end of the data valid times, respectively. The "versionId" attribute uses a unique identifier automatically assigned to each object by the system. It is possible to add historic versions of the objects to the database by designating the data valid times as points in the past. It means that the number of the object versions does not reflect the order in which the object statuses occur.

The presented method ensures high performance of data processing by using computer memory.

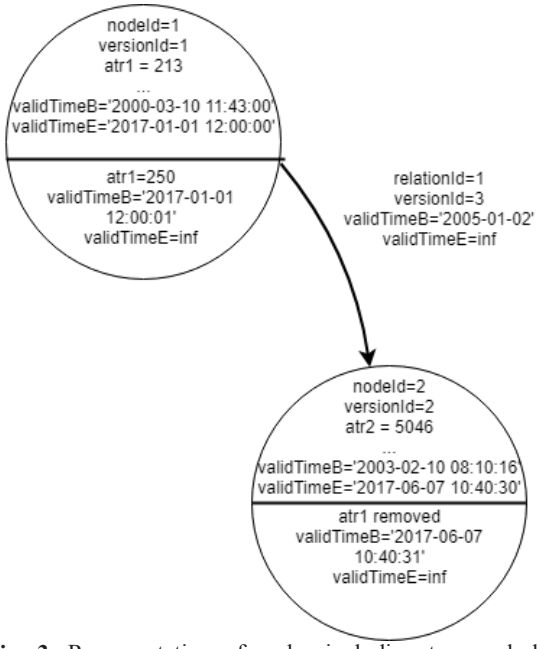

Fig. 3. Representation of nodes including temporal data in accordance with the incremental write method

3.2 Incremental write method

A method opposite to the above-mentioned method consists in recording the changing data incrementally.

According to this method, every change of the object status contributes to the writing of only such features and relationships that were subject to change. Other aspects are presented analogically to the full write version.

The incremental write method allows to decrease the amount of computer memory used by the database. The system does not have to store replications of the whole objects, only their individual changing features. However, the solution requires analysis of the historic and current information while processing the data in the database, which decreases the performance of the system.

3 Operations on temporal data in graph databases

The entry of temporal data into the network databases requires modification of the processing algorithms so that the operations performed in the database included time stamps, in particular the valid data time. Basic operations performed in the database are defined as CRUD (Create-Read-Update-Delete).

The algorithms for adding data to the database, upon entry of temporal data into the system, should enable a user to introduce the data valid time (also with respect to historic data) and automatically record the transaction execution time.

In case of full write method, at the time of update, the system should automatically write the end time of the validity period and create new object in the database describing new (current) status of the represented being. If the incremental write method is used, the update should trigger the recording of changes of attributes, including their valid times, in the database.

Data are not physically removed from the temporal databases (except for incorrect data). Therefore, the data

removal operation consists in indicating the end of the data valid time.

To read data in the graph databases, it is usually necessary to set the subgraph or path. It is the most important and complex operation to be performed in such databases. The main issue that needs to be considered in terms of data reading in the temporal graph databases is the manner of returning results. Regardless of the data write method used, the data may be returned in a historically unordered or ordered manner.

Unorderly presentation of query results would not much differ from the method of data representation currently applied in the system. However, it would be necessary to supplement the instructions of the CQL language with a possibility of searching for current data or data with valid time between two points in time. Upon verification of both the conditions described by the user in the query and the conditions related to time stamps, the algorithms would produce the results in a comparable manner as in the MATCH-RETURN command (unordered data). The solution ensures high performance, but requires independent (manual) data analysis and modification by the user.

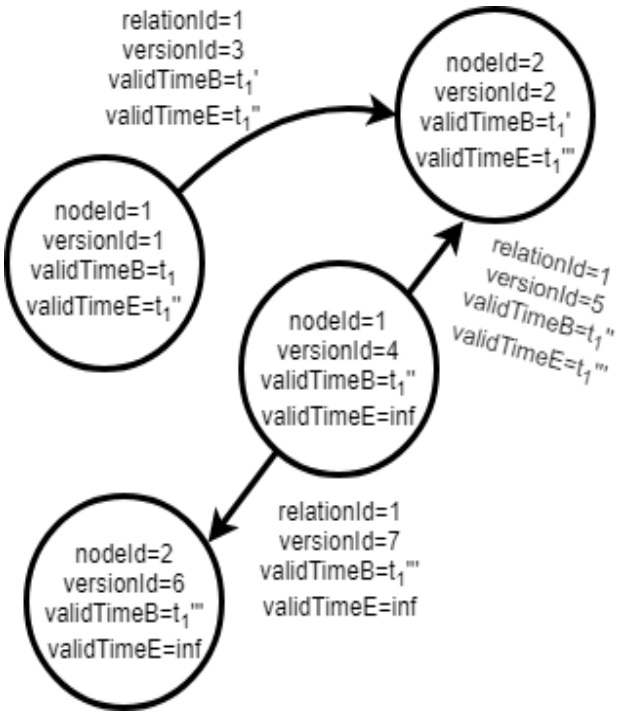

Fig. 4. Unorderly representation of the results

Representation of the results in chronological order, according to the order of changes in the object statuses, would significantly facilitate analytical work. To that end, it would be necessary to modify the manner of representation of the query results processed in Neo4j. According to the proposed method for representing the results, in case of changes in the object statuses for the n-th time as stipulated in the query, the system shall return n graphs showing individual statuses.

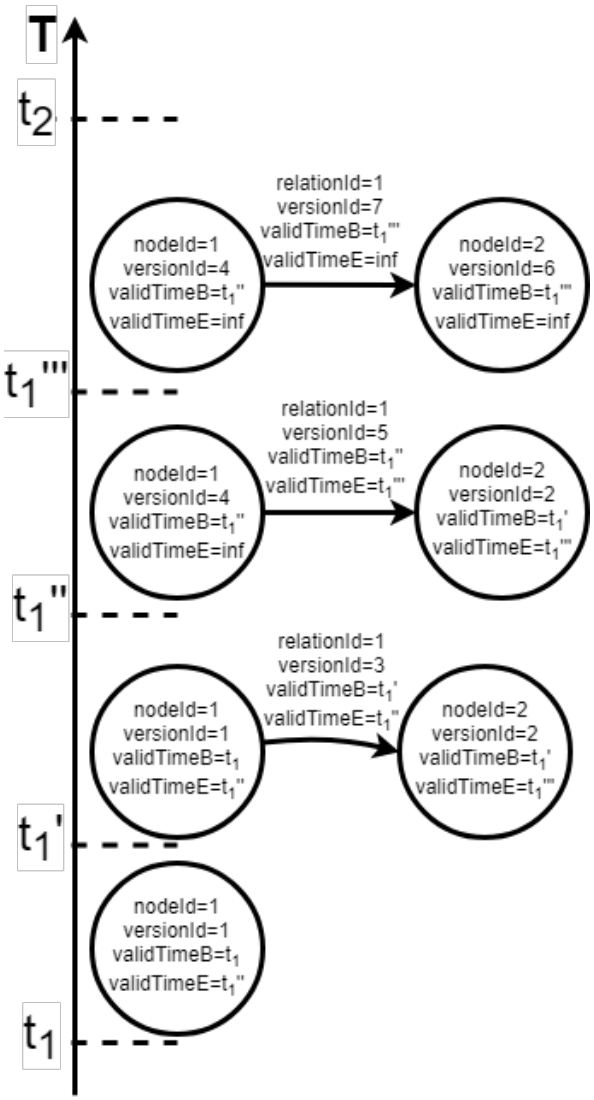

Fig. 5. Orderly representation of the results

4 Conclusions

The assumptions for the temporal database outlined in this article constitute grounds for further research. Further studies shall concentrate around the issue of implementation of the above-described methods as well as their analysis and comparison in terms of efficiency. In order to create temporal graph database methods of storing and representing data shall be implemented. As well as temporal primary keys, temporal referential integrity constraints and temporal predicates. Finally, data processing language should be extended to include temporal clauses.

The solutions may be used to analyze data, among other things, in the Systems for Detection of Cyber Incidents, systems with documents with RFI tags [16], with high reliability [17]. Possibility of representing temporal data in such systems allow to analyze incidents in chronological order and find interdependence between them. Temporal graph database is a useful mean for pattern recognition, especially in cybersecurity problems.

References

1. I. Robinson, J. Webber, E. Eifrem, Graph Databases (O'Reilly Media, Sebastopol, 2013)
2. D. Sullivan, NoSQL for Mere Mortals (Addison-Weasley, Ann Arbor, 2015
3. DB-Engines Ranking of Graph DBMS, https://db-engines.com/en/ranking/graph+dbms (access: 10.03.2018)
4. Neo4j web portal, https://neo4j.com (access: 11.03.2018)
5. Opencypher web portal, http://www.opencypher.org/, (access: 11.03.2018)
6. L. Liu, M.T. Ozsu, Encyclopedia of Database Systems (Springer US, Boston, 2009)
7. C. De Castroa, F. Grandib, M. R. Scalasa, Information Systems, **22**, 5 (1997)
8. C. S. Jensen, R. T. Snodgrass, IEEE Transactions on knowledge and data engineering, **11**, 1 (1999)
9. S. Konur, Frontiers of Computer Science, **7**, 3 (2013)
10. N. Simons, J. Richardson, New Content in Digital Repositories (Chandos Publishing, Oxford, 2013)
11. D. Petkovic, New Advances in Information Systems and Technologies. Advances in Intelligent Systems and Computing, 444 (2016)
12. A. Campos, A. A. Vaisman, CoRR, arXiv:1604.08568 (2016)
13. H. Huang, J. Song, X. Lin, S. Ma, J. Huai, TGraph: A Temporal Graph Data Management System (ACM International Conference on Information and Knowledge Management, 2016)
14. DB-Engines Ranking of DBMS, https://db-engines.com/en/ranking_categories (access: 10.03.2018)
15. C. Cattuto, M. Quaggiotto, A. Panisson, Al.Averbuch, Time-varying social networks in a graph database: a Neo4j use case (GRADES'13 First International Workshop on Graph Data Management Experiences and Systems, 2013)
16. M. Kiedrowicz, T. Nowicki, R. Waszkowski, Z. Wesołowski, K. Worwa, Software simulator for property investigation of document management system with RFID tags, MATEC Web of Conferences, vol. 76, DOI: 10.1051/matecconf/20167604012, Greece, (2016)
17. R. Waszkowski, M. Kiedrowicz, T. Nowicki, Z. Wesołowski, K. Worwa, Method for assessing software reliability of the document management system using the RFID technology, MATEC Web of Conferences, vol. 76, DOI: 10.1051/matecconf/20167604009 Greece, (2016)