# Transforming Product Catalogue Relational into Graph Database: a Performance Comparison

Josip Lorincz*, Vlatka Huljić**, Dinko Begušić*

* Faculty of electrical engineering, mechanical engineering and naval architecture (FESB)/Department of electronics and computing, University of Split, Split, Croatia, e-mail: josip.lorincz@fesb.hr; dinko.begusic@fesb.hr

** Kron d.o.o., Split, Croatia, e-mail: vlatka.huljic@kron.hr

***Abstract* - Due to the continuous expansion of telecom operators (TOs) services and subscriber's base, a number of objects in TO product catalogue (PC) traditionally developed employing some relational database (RD) rapidly increase. This can result in degraded PC database performance in terms of reduced object visibility or slow query response time, which mandates a search for new types of databases that can cope with such challenges. A possible solution to these challenges is the transformation of RD into some graph database (GD), with visualisation and organisation capabilities that can cope with the increasing complexity of PC RDs. Hence, in this paper, a novel implementation concerning the transformation of a set of PC data and functions of the TO RD into ArangoDB GD is proposed. The proposed approach allows the upgrading of PC RD into a higher presentation layer through the implementation of the GD, which enables advanced data analysis and visualization. The developed solution shows that GD with appropriately structured queries, data and links between them, improves the visibility and simplifies the entire search process. Obtained results show that GD outperforms RD in terms of search query response time for different functions executed on an equal amount of database objects.***

***Keywords* - Product Catalogue, eTOM, SID, REST API, Graph, Relational, Database, ArangoDB, NoSQL, BSS/OSS**

## I. INTRODUCTION

The telecom operator's (TOs) product catalogue (PC) containing all details of the products (services) to costumers is a very important module for their business. The PC structure defined with Enhanced Telecom Operations Map (eTOM) standards consists of entities and relationships based on which, packages and telecommunication service offers to the costumers are formed. eTOM is a part of Frameworx, developed by the TeleManagement Forum (TM Forum) [1]. The TM Forum is an industry association focused on transforming business processes, operations, and systems for managing and monetizing online information, communication, and entertainment services [2]. Hence, Frameworx is a suite of best practices and standards that provide the blueprint for effective and efficient business operations of enterprises such as TOs.

TM Forum Frameworx contains the following frames: application framework, business process framework, information framework and integration framework. The business process framework or eTOM describes different levels of enterprise (TO) processes, according to their significance and priority for the business [3]. Shared Information Data (SID) model or the information framework describes the information entities over which the processes operate, with their characteristics and relationships. SID provides definitions for all the information that passes through the enterprise among business partners and corresponding service providers. SID domain corresponds to the lowest level of the eTOM model and is central to the TOs next generation operations system support (NGOSS). This model represents a key principle for enhancing operations support systems (OSS) and business support systems (BSS) processes interoperability of TOs [4].

Therefore, the SID model offers a "common language" for software developers and integrators for describing information management of TOs processes. It enables simpler and more efficient integration of BSS/OSS software applications deployed by different vendors into the TO infrastructure. SID model takes into account the principles and concepts needed for defining shared information. This model also precisely prescribes many of the business elements (known as "*entities*") of interest to TOs and corresponding *attributes* that specify these entities (known as Aggregate Business Entities (ABSs)).

SID as model covers a vast area of TO activities and many companies in the world are working on the development and implementation of TO systems covering SID related process. Currently, the SID model is based on seven domains (market and sales, product, customer, service, resource, engaged party and common business entities domain) [5]. Product domain presented in Figure 1. contains different ABEs, which deals with product, product specification, product offering, strategic product portfolio plans, product performance, product usage, product configuration, product test and loyalty [6]. In fact, SID product domain elements are those elements which are fully covered and specified within the PC of most of the TOs.

The product domain as part of the SID is one of the TO system domains which has been analysed in this paper and PC as the main component of the SID product domain has been developed utilizing graph database (GD) concept. Since the number, lifecycle and contract operations related to TO products' can be huge, especially for large TOs, PCs eventually become very complex and hard to manage by means of relational databases (RDs) which are commonly used so far for PC management. Thus, in this paper, several functions of the real PC used by a prominent European TO are extracted from an implemented RD and transformed into the newly developed multifunctional GD. The capabilities and functions of a GD implemented on a real model of PC are explored and compared with that equivalent in the RD.

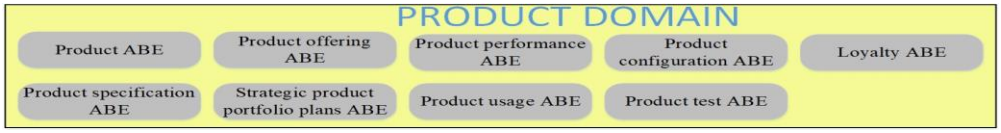


Figure 1: Product domain aggregate business entities of the SID model [6]

The comparison of the two PC database implementations shows that the proposed GD implementation outperforms the RD implementation.

The remaining structure of this paper is organised as follows: related work presenting an overview of relevant literature dedicated to the transformation of RD into GD is discussed in Section II. Section III explains research motivation and describes the analysed PC of TO. In Section IV, the transformation of the RD into GD is presented. In the next Section V, the functionality of PC as GD is described. A comparison of developed multifunctional GD and RD in terms of performance are shown in Section VI. Finally, some concluding remarks are given in Section VII.

## II. Related Work

There has been some extent of work in the area of transforming an RD into a GD. Soussi in [7] analyses the possibility that an RD could be transformed into some other model that can further be used for developing the GD. Some of the models proposed in the literature are based on eXtensible Markup Language (XML), Entity-Relationship (ER) or Resource Description Framework (RDF) [8-10]. The basic drawback of these approaches is in the lack of possibility to detect interactions among generated nodes without the appropriate ontology which is needed for defining tables and data being built beforehand. Hence, efficient algorithms for converting RD into GD by building a graph through some other in-between model are yet to be proposed.

The other approach to the transformation of RD into GD is based on tools that enable feedback from an RD in the form of a graph, through posing a query in a format of search keywords that could be found somewhere in the database tuples. Such tools proposed in literature differ in modelling the tuples as the graph nodes which are connected by edges representing foreign keys or transitive tables [11-14]. Recently proposed tools for generating a GD from an RD enable graph mining techniques on relational data, however, the drawback of these tools is that they rely on the user which must strictly define the data being converted [15-16]. Another approach in transforming RD to GD proposed in [17] and [18] are based on conceptual graphs or relation-of-relations graphs, respectively. The first approach is not suitable for frequent subgraph mining from the perspective of entities, while the second approach, although having appropriate conversion duration, lacks testing on larger databases. Approaches for converting the entire RD to a GD are proposed in [19 -21]. The approach presented in [19] is based on transforming dependency graphs for the entities in the system into a hypergraph model for the RD, which can be used to develop the domain relationship model that can be converted into a GD model. In [20], an approach to automatically migrate data and queries from RD to GD is presented. Algorithms proposed in [21] are independent of the data semantics and provide the conversion between RD and GD without data loss.

Presented overview of solutions for transferring RD into GD confirms that there is no ideal tool that can merge automatic transfer of any RD into (any) GD. Due to differences in structure and data organization of versatile RDs and GDs, the conversion tools must be adopted for specific RD and GD, in order to read the relational data of RD and load it into the GD in as few steps as possible. Hence in this paper, another approach to transferring RD characteristic for PC of TOs into contemporary GD is presented.

There are a number of GDs currently available, such as Sparksee (formerly known as DEX), AllegroGraph, Neo4j, HyperGraphDB, Infinite Graph, ArangoDB, OrientDB, etc. and comparison of these and some other GDs are given in [22-23]. According to [23], ArangoDB and Neo4j GD obtain the highest scores in terms of comparison based on features such as flexibility, query language, sharing, backups, multimodality, multi-architecture capabilities, scalability and cloud readiness. This motivates the selection of ArangoDB as one of the best candidates for analyses presented in this paper. Although ArangoDB and Neo4j outperform other currently available GDs in terms of mentioned features, results of performance comparison among RDs and Neo4j GD have been already reported in [24-25]. However, to the best of our knowledge, a performance comparison between the RD and ArangoDB GD on the concrete application has not been reported in the literature. To fill this gap, this paper presents performance comparison among ArangoDB GD and an RD for PC as the specific TO use case.

## III. Research Motivation for Product Catalogue Transformation

To achieve the functionality of the PC, modules that contain product information such as the basic entities (the general names for any service), attributes, prices, product offerings, etc. are required [26]. The existence of the relationship between these entities enables the creation of complex products grouped in packages offered to the users by the TO. The structure and number of elements of each TO PC can differ in complexity and size.

The data model of the PC module analysed in this work contains different basic entities (attributes, products, product relations, attribute mapping, products business rules, product offerings, prices, price lists and sales context) which are presented in Figure 2. Links between entities of the PC model emphasize the complex structure of the PC.

In the presented analyses, part of the real PC model used by one European middle-size TO was originally

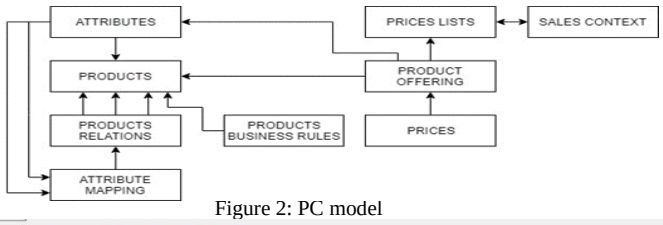



Figure 2: PC model

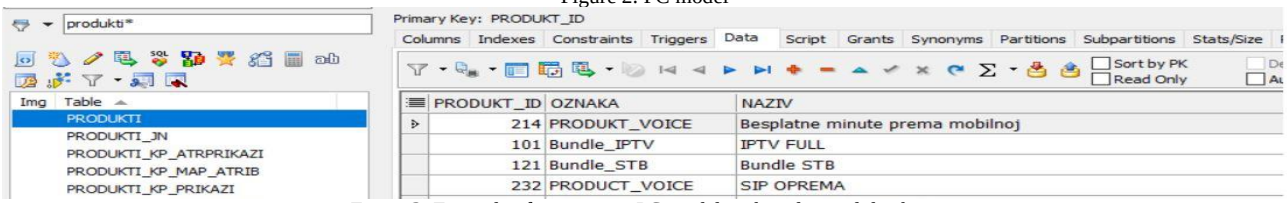



Figure 3: Example of an existing PC model in the relational database

implemented and used in the form of the RD. An example presenting a listing of the PC elements in the form of the table, with main products structured in the RD are shown in Figure 3. A *Toad* software tool was used to work with RDs (Figure 3). This tool allows developers, database administrators and database analysts to manage RDs and non-relational databases by Structured Query Language (SQL) queries.

Since it is challenging to cope with the multitude of data and have a good overview of all database elements, administrators of such RDs start to have the problem of managing the PC. The question that arises was can some other PC data structure offer faster PC database search and better visualisation of PC elements, especially in light of necessity for the continuous expenditure of TOs PC with new products and corresponding elements. The transformation of the existing RD into new datasets with an upgrade to the GD was a valuable option to the TO. Hence, the research presented in this paper was motivated with the goal related to the transformation of data from an RD into the GD format. This can gain a new visualisation dimension of PC and additional information that will serve for better analysis of the PC data. The idea was to use the graph of the database to attribute a PC in order to get information about: where the services are used and under what conditions, what are the geographic locations of services used by users and how used services are represented. All this information cannot be easily visualized in existing TO RD. However, such information is important for TO marketing, which must optimize offering of the service in the geographical area covered by TO and also for the technical implementation and maintenance of the TO equipment including fault recovery.

## IV. Implementing Product Catalogue by Means of Graph Database

According to presented in the previous section, PC database management can be a difficult task when it comes to large amounts of data, and the problem can be compounded when multiple data formats are involved. For the development of GD, in this work, the open-source ArangoDB as not only SQL (NoSQL) database type having GD capabilities were selected for administration of PC data transformed from Oracle v12c RD. This database type is specifically designed to manage multiple data formats in a single instance. The ArangoDB provides GD capabilities, database document management and key-value storage [27].

Unlike RDs that have a strictly defined data schema, ArangoDB has a dynamic data schema [28]. Data in ArangoDB are stored as JavaScript Object Notation (JSON) documents. The documents are grouped into "collections" that correspond to a table in RDs. The documents in the collection have a unique *primary document key* that automatically stores and encodes the identity of the document, thus storing the document in the key/value container. The created collection consists of documents stored as vertices and edges in the form: *_from* the source point value *_to* the value of the target vertex [29]. Combining vertices and edge collections produce a graph. The proposed transformation of data from the RD into GD is performed through the following phases:

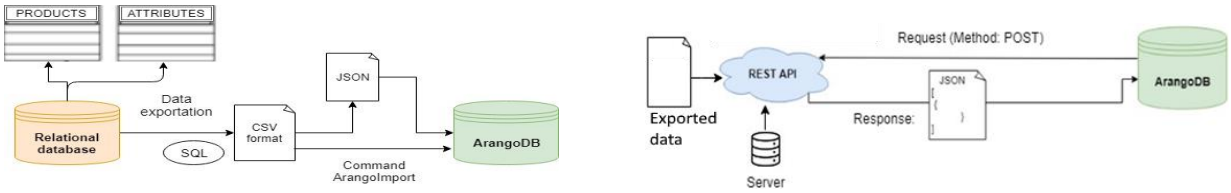
### A. *The first phase: export data from a relational database*

The first phase in the process of transforming PC from the RD into GD structure is related to the export of selected data from an RD into a GD, more precisely into the ArangoDB database (Figure 4). These data operations are performed using SQL queries (Figure 4a).

The first two queries extract the data that will be entered into the vertices collections of GD, and the third query extract data which will be extrapolated into the edges collection of GD. The first phase ends when data from the RD are extracted in a .xls file format that is stored in Comma Separated Values (CSV) data format (Figure 4a).

### B. *The second phase: import data into ArangoDB*

The second phase of transferring RD into GD is dedicated to the import of exported data form RD into the GD (Figure 4). The ArangoDB contains a command-line client tool called *Arangoimport* (Figure 4a). This client command tool can be used to import JSON, CSV and tab-separated values (TSV) format data into ArangoDB. In our analyses, JSON and CSV as common data formats are used (Figure 4a). *Arangoimport* is called from the command line every time when each file (JSON or CSV) wants to be imported into the GD (Figure 4a). Hence, the




a) b)

Figure 4. The proposed concept for transforming relational into GD based on relational data export and: a) direct SQL queries, b) REST API methods

CSV data format can be imported directly (Figure 4a), or it can be formatted into JSON format using format converter and imported in GD (Figure 4a). The data importing results with the transfer of PC data objects consisted of attribute-value pairs and array data types in GD. When data import is done, it is possible to continue working with the same data model previously used.

Another way of importing data from an RD is to expose the ArangoDB application programming interface (API) over the hypertext transfer protocol (HTTP) server, making the server easily accessible for different clients and tools (Figure 4b). Data exported in the first phase are reachable through exposed API and converted into JSON format for purpose of transmission into ArangoDB. According to Figure 4b, the JSON data is sent and received over a well-known HTTP server protocol which provides a RESTful API (REST API).

The REST API methods (GET, POST, PUT, DELETE, PATCH, etc.) represent the architectural style used for creating a web service for the transfer of resource state views (Figure 4b). Usage of these methods is popular in practice due to their simplicity and the fact that they rely on existing HTTP protocol and corresponding features, while HTTP defines the format and method of messaging [30].

### C. *The third phase: creating collections in ArangoDB and graph formations*

In ArangoDB, documents are stored in collections. Collections have names, which describe what kind of information the documents contain. Each document requires a key that uniquely identifies it within a collection. It can be assigned by the user upon creation, or ArangoDB will generate one. Document keys and document identifiers (IDs) are always indexed. The index on the *_key attribute* is called a primary index. It exists for each collection and can't be removed. There are two types of collections in ArangoDB: vertex and edge collections (Figure 5). Documents in an edge collection have two additional attributes, *_from* and *_to*. Both must have document IDs to link documents together. The document that links them is called an edge and the linked documents are called vertices in the graph model (Figure 5). When creating a graph model based on an RD model, it is important to be aware that the row in the RD is equivalent to a node in the GD, and that the table name is equivalent to the node label in the GD model.

The ArangoDB uses ArangoDB query language (AQL) for executing procedures over graphs, as well as over individual collections. Although some keywords overlap, AQL syntax is different from RD SQL syntax. The most notable difference is in the concept of loops in AQL, which makes AQL look like programming language [31]. AQL has a language construct for looping through data and the most common form is the FOR loop used with a collection of documents (similar to SELECT in SQL). Complex object and array manipulations can be done using AQL queries (INSERT, UPDATE, REPLACE, DELETE, etc.) and operators (FILTER, SORT, LIMIT, etc.), what makes AQL very powerful query language that remains easy to write and read.

## V. THE FUNCTIONALITY OF PRODUCT CATALOGUE AS GRAPH DATABASE

As previously indicated, the PC presents description and characteristics of TO services offered to customers and contains all necessary information related to the specific product. An example of graphical representation obtained through developed GD for specific TO product is presented in Figure 5. The graphical representation obtained through implementation of GD shows the relationship between templates (e.g. user mail addresses ("MAIL"), contracts ("OTT ugovor"), etc. in Figure 5) of PC and their attributes which can be characteristic for the specific template (e.g. additional service ("Dodatna usluga"), service and service packet ("Usluga i paket usluge"), etc. in Figure 5) or shared among different templates (e.g. service name ("Naziv usluge"), name and surname of the owner ("Ime i prezime vlasnika"), etc. in Figure 5).

The template is prepared textual document used when generating a new document instance and its attributes are used to describe an entity or to extend the meaning of an entity. This means that if there is a link between templates and their attributes, internal logic generates a document with its attributes representing dynamic fields in it. The view is filtered by the name contained in the template or by attribute description, what generates query results as those presented in Figure 5. This approach gives a credible view of all related and unrelated elements in the graph of GD. In this way, an error can easily be detected if there must be a connection between PC elements (templates and attributes), but no link exists in the graph model. Unlike RDs where data search is a long process, GD improves the entire search process by simple visualization of the data and the links between them (Figure 5).

Services offered by the TO are tracked through the concept of the customer's assets. They represent the state of services that the customers use at any given time. For TOs, it is important to have a correlation between used services and corresponding products and locations where they are installed. As an example, for some selected dataset, Figure 6 presents a relationship between products

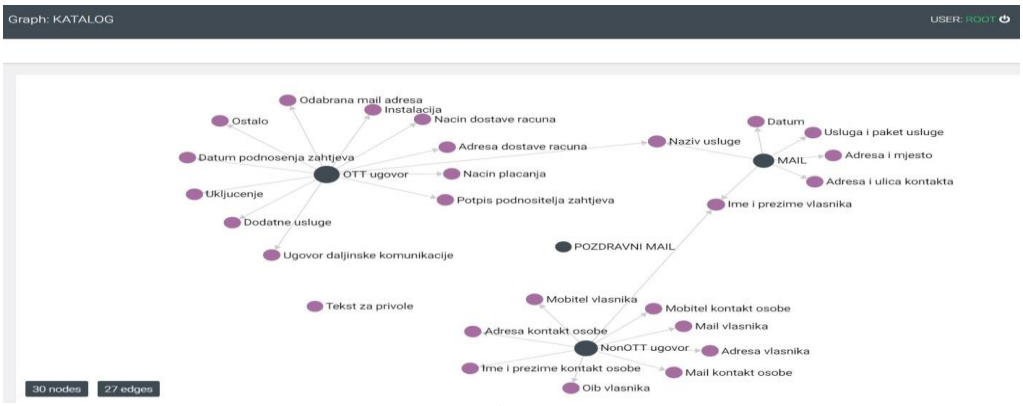



Figure 5. The relationship of templates and attributes in GD

and locations of product usage, where templates (e.g. “SESVETE”, “ZAGREB”, etc. in Figure 6) represent locations of used products, while attributes (e.g. S service packet no. 19 (“S paket 19”), L service packet no. 19 (“L paket 19”), etc. in Figure 6) identify a specific product. The selected dataset realistically shows the relationship of products sold to a customer in a particular place of residence (Figure 6). Displaying the density of services and corresponding product distribution by location (place), gives the TO an insight into the actual state of use of the specific service and corresponding product. This information is useful for making decisions about the need to change the current state, i.e. create new services and corresponding products or use existing ones in the areas not covered with specific TO service.

Hence, developed ArangoDB GD ensures one of the main advantages of GDs, which is performing simple queries on a diverse set of data, what results with a comprehensive and detailed view of the requested information.

## VI. Performance Analyses

Performance analysis is based on a comparison of developed multifunctional GD (ArangoDB) and existing RD (Oracle). Analyses are based on the measurement of SQL and AQL query execution response time. For realistic comparison, the same amount of data related to the PC of TO was used for comparing query response time in RD and GD. This measurement is performed for the three queries (Q1, Q2, and Q3) executed in the RD and GD (Figure 7). Queries differ in terms of scope and complexity of the search, which is directly related to the number of analyzed database elements.

The Q1 query in the RD search for a custom set of data (e.g. places, products, link type, etc.) and 90 database elements are the result of the search. The Q1 query execution time in the RD is 109 ms, while in the ArangoDB GD is significantly faster and equals to 0.695 ms (Figure 7).

The Q2 query provides information about which attributes are attached to the selected template and gives 9 elements as a result. The speed of query execution differs significantly in this example, because the GD has a great advantage in terms of execution time which is equal to 0.960 ms and the 43 ms for the RD (Figure 7).

To have reliable analyses, a larger amount of PC data was taken in the next example with the Q3 query, where the query results in 1000 database elements. The Q3 query search for all the attributes contained in the products table (collection) of the PC (e.g. group (“GRUPA”), type (“VRSTA”), model (“TIP”), line (“LINIJA”)). In this case, query execution time in the RD is 226 ms (Figure 7), while in the GD equals to 5.355 ms.

Based on the measured execution time of individual queries, it can be concluded that query execution in the GD is significantly faster compared to the RD (Figure 7). For some query types, GD outperforms RD with more than 42 times faster response time. Also, in Figure 7 can be seen that better performance is evident for GD where the query execution time does not increase significantly with the increase in the number of search data objects, which is not the case with RD. Hence in developed GD, queries are executed faster and outcomes of queries give more comprehensive results in terms of their visualization. The faster execution time of GD is a direct consequence of the better structural organization of ArangoDB GD, what confirms that this type of GD is suitable for transforming the PC of TO from a conventional RD.

## VII. Conclusion

The PC structure based on eTOM standards contains entities and relationships according to which packages of telecommunication services are formed and offered to users. PCs are traditionally developed in the form of an RD. Due to the constant increase in a number of services and subscriber base, the complexity of relational PC database management became challenging for TO database administrators. This requires new concepts in the development and management of the relational PC databases of TOs. This paper gives a contribution to devising such new concepts through the presentation of a new approach for transformation of TO relational PC database into GD. The process phases of transforming RD (Oracle) into GD (ArangoDB) are described, with the presentation of developed GD structure and performance comparison among analyzed databases.

It is shown that GD with advanced queries supported through AQL, graph data structure and links between GD elements, simplify the entire search process and it is applicable for use when a larger amount of different data structures must be managed. A comparison of analysed

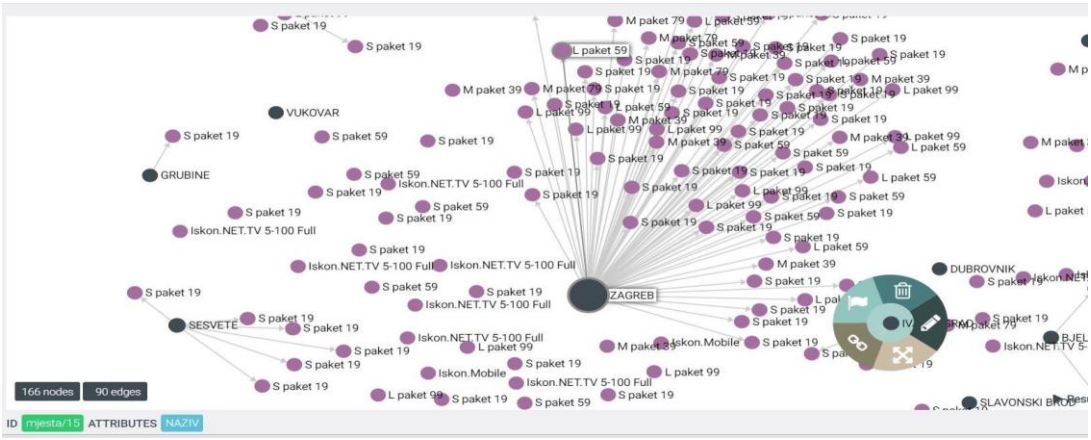



Figure 6. Relationship between product and location of product usage presented in GD


Figure 7. Query execution response time

databases show that GD offers a higher presentation level which enables more comprehensive data analysis and visualization. The greatest advantage of developed GD over the RD is reflected in the speed of query execution, for which GD significantly outperforms RD. Due to a very important feature of managing and storing large amounts of versatile data structures, the ArangoDB is expected to be widely used in practice, and it is particularly suitable for the development of TO PCs. Our future research activities will be focused on the automatization of data transfer form RDs into GDs.

## REFERENCES


[1] Internet, August 2019, https://www.tmforum.org/

[2] Internet, August 2019, https://www.cisco.com/c/en/us/products/collateral/services/high-availability/white_paper_c11-541448.html

[3] Internet, August 2019, http://www.ilsa.kz/etom/main/tamdomain5.htm

[4] Internet, August 2019, http://www.sebconsulting.ie/understanding_ngoss_sid.html

[5] Internet, August 2019, https://www.tmforum.org/Browsable_HTML_SID_R18.5/content/3E3F0EC000E9_root.html

[6] Internet, August 2019, https://www.tmforum.org/tm-forum-frameworx-2/

[7] R. Soussi, "Querying and extracting heterogeneous graphs from structured data and unstrutured content", PhD thesis, Ecole Centrale Paris, 2012.

[8] M. Hert, G. Reif, H. C. Gall, "A comparison of rdb-to-rdf mapping languages", In Proc. of the I-SEMANTICS '11, 2011. pp: 25 – 32.

[9] W. Hu, Y. Qu, "Discovering simple mappings between relational database schemas and ontologies", In Proc. of the ISWC/ASWC '07, 2007., pp. 225-238.

[10] J. Sequeda, M. Arenas, D. P. Miranker, "On directly mapping relational databases to rdf and owl", In Proc. of the WWW '12, 2012., pp. 649-658.

[11] V. Hristidis, Y. Papakonstantinou, "Discover: Keyword search in relational databases," in Proc. of the 28th VLDB '02, 2002. pp. 670–681.

[12] S. Agrawal, S. Chaudhuri, G. Das, "Dbxplorer: enabling keyword search over relational databases," in Proc. of the ACM SIGMOD '02, 2002, pp. 627–627.

[13] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, H. Karambelkar, "Bidirectional expansion for keyword search on graph databases", in Proc. of the VLDB '05, 2005, pp. 505–516.

[14] H. He, H. Wang, J. Yang, P. S. Yu, "Blinks: ranked keyword searches on graphs," in Proc. of the ACM SIGMOD '07, 2007., pp. 305–316.

[15] K. Xirogiannopoulos, U. Khurana, A. Deshpande, "Graphgen: Exploring interesting graphs in relational data", Proceedings of the VLDB Endowment, vol. 8, no. 12, 2015., pp. 1-4.

[16] D. Simmen, K. Schnaitter, J. Davis, Y. He, S. Lohariwala, A. Mysore, V. Shenoi, M. Tan, Y. Xiao, "Large-scale graph analytics in aster 6: bringing context to big dana discovery," Proceedings of the VLDB Endowment, vol. 7, no. 13, 2014., pp. 1405–1416.

[17] S. Palod, "Transformation of relational database domain into graphs based domain for graph based data mining," Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, SAD, 2004.

[18] S. Pradhan, S. Chakravarthy, A. Telang, "Modeling relational data as graphs for mining" in Proc. of the COMAD '09, 2009., pp. 1-6.

[19] S. Bordoloi, B. Kalita, "Designing Graph Database Models from Existing Relational Databases", International Journal of Computer Applications, Volume 74, No.1, 2013, p.p.: 25-31.

[20] R. De Virgilio, A. Maccioni, R. Torlone, "Converting relational to graph databases," in Proc. of the GRADES '13, 2013, pp. 1-6.

[21] O. Orel, S. Zakošek, M. Baranović, Property Oriented Relational-To-Graph Database Conversion, Automatika, Vol. 57(2016), 3, 2016, p.p.: 836–845.

[22] R. Angles, "A comparison of current graph database models", in Proc. of the ICDEW '12, 2012, pp. 171–177.

[23] D. Fernandes1, J. Bernardino, "Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB", In Proc. of the DATA 2018,2018., pp. 373-380.

[24] O. Est, "Comparative analysis of relational and graph database behavior on the concrete web application", Master thesis, Tallin University of Technology, 2016, p.p.: 1-68.

[25] Yaowen Chen, "Comparison of Graph Databases and Relational Databases When Handling Large-Scale Social Data", Master thesis, University of Saskatchewan, pp: 1-91

[26] Kron Company, "Kron - Product Catalog - Assets – Order", Internal Document, 2019.

[27] Internet, August 2019, https://www.arangodb.com/arangodb-graph-course/

[28] Internet, August 2019, https://repozitorij.pmf.unizg.hr/islandora/object/pmf%3A3234

[29] J. Stücke: „From Zero to Advanced Graph Query Knowledge with ArangoDB" Internet, August 2019, https://www.arangodb.com/2017/05/zero-advanced-graph-query-knowledge-arangodb/

[30] Internet, August 2019, http://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer

[31] Internet, August 2019, https://www.arangodb.com/docs/3.4/getting-started-coming-from