Using R with different types of databases: An overview

A. Gužvanj*, Grigor Ćorić**, Marko Horvat and Sergej Lugović*
*Zagreb University of Applied Sciences, Department of Computer Science and Information Technology, Zagreb, Croatia
**mStart, Zagreb, Croatia
slugovic@tvz.hr

Abstract - R is a working environment and computer language used in statistical computing. It is widely used as a tool in knowledge discovery and big data analytics as it provides an environment in which different statistical methods can be applied efficiently. As the size of analyzed data grows different data sources may be added to enrich statistical models and improve analysis quality. However, with R there is no need to store those datasets into the database. At the same time, R as environment support different data structures that make design and development of the information systems even more complex. R data structures could be divided into two main categories, homogeneous data such as atomic vector, matrix or array, and heterogeneous such as list and data frame. Additionally, we are witnesses to new trends in database design such as NoSQL, New SQL, graph database, in memory data storage, columnar and different application of SQL. Aim of this paper is to give an overview of different databases types in the context of R, covering concepts from navigational databases, to rational and post relational databases and how they are suited to host data for the purpose of access, storage and manipulation by statistical procedures in R.

Keywords: R, Database, Big Data, Data Science, New SQL

I. INTRODUCTION

There is evidence of several different trends that are having impact on information system design. The first trend is evident in more and more data being collected as the ratio between stored data and analyzed data decreases [1]. Another trend is an increase of frequently used data sources. This new situation imposes a question about what data sources could be leveraged and what data infrastructure is required to support data processing [2]. The third trend is a design of information systems being built on one or more clouds [3]. Those trends are creating new requirements for future information systems design and adoption of existing ones. These requirements are (based on [3]): *i)* use of machine learning and statistical techniques to enable automatisation of data processing scalability, *ii)* as the size of data and their sources increase there is need to clean data using efficient algorithms, *iii)* as more jobs depend on data there is a need to design end user interfaces in a way that they are able to perform data combining and analysis tasks without using computer code, and *iv)* information systems should be designed in a way that new data sources can be incrementally integrated. To meet those requirements information system designers have to look for the database and data processing techniques that are beyond data warehouse concepts. Some of the ideas discussing these issues could be found in [5,6,7] just to name a few. One of the possible direction in research is to investigate the use of computer statistical languages, in particular R, as a component in the design of information systems. Overview of the potential use is presented in Figure 1.

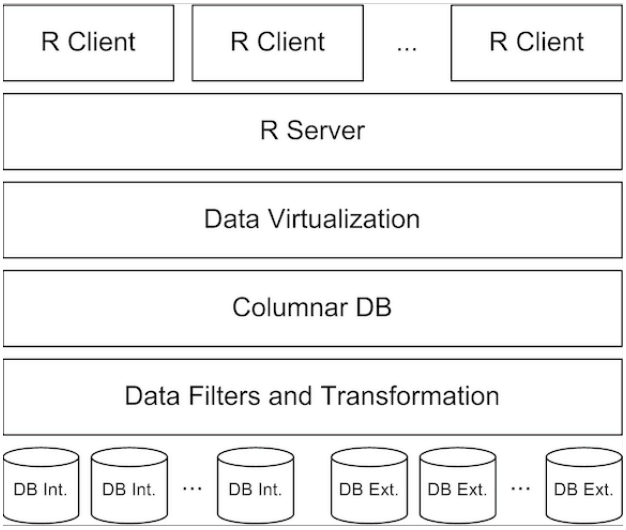

Fig. 1. Application of R in data processing procedures.

At the bottom tier of the scheme there are different types of databases used by information system. They can be internal or external. Directly above the databases there is a layer responsible for filtering the data for further analysis in terms of detecting anomaly. For example, filters could be set up to automatically analyze the number of returns (of goods sold), data collected that represent the customer satisfaction with shop assistants and the sentiments of the social media content relevant to company brand. If anomalies are detected, relevant data from different databases is "pushed" into the column-oriented database systems for temporary analysis. This system is connected with R server that manages access to the data for further analysis by data scientists with R clients based on data virtualization approach. From the R server access to data can be given to numerous clients

and they may use different statistical techniques to simultaneously analyze detected anomalies. Data processing tasks can also be divided into smaller tasks and run asynchronously on a single client. By using this combined approach analysis time can be drastically reduced and different insights into same datasets can be concurrently generated. Datasets can be accessed by data scientist that are the part of the organization and others that are only temporarily employed. This could be useful if analysis is domain specific and there is no inside expertise in the organization. Except the above described "push" process, "pull" process could be used so that data scientists can access the data stored in different databases and "pull" them through different layers (filter, columnar, virtualization, R server) to their R clients' statistical environments. But to implement and apply the proposed process there is a need to first understand R data structures and how well they fit with different data models and databases. This paper first elaborates R, its data types and structures, following with the brief overview of different types of databases. In the discussion synthesis of those two in terms will be presented together with outlined benefits and limitations in using R.

II. R

R is a working environment and computer language used in statistical computing. It is a GNU package and is available freely under GNU General Public Licence. It was created at the University of Auckland (in New Zealand). Its creators are Ross Ihaka and Robert Gentleman. The initial work of the beforementioned two authors appeared at Statlib as a binary copy of code in August of 1993. Receiving a good feedback from peer community (notably Martin Machler of ETH Zurich) the initial source code of R has been made available under Free Software Foundation's GNU general license in June of 1995. From the early beginnings many members of science community have been active in proposing enhancement to the core source code, and bug fixes through the established mailing list. All these efforts led to establishment of larger „core group" in 1997 whose responsibility was to make changes to the source code of R, in response to suggestions from community [8]. R has been created as a successor to S programming language (S was created by Jon Chambers) and was inspired by functional programming language Scheme. Development of R began by creating an interpreter for a subset of Scheme. The interpreter supported initial mechanism for symbol-table management and counted approximately 1000 lines of code written in C programming language [8]. R, referred to as a system, is written primarily in C, Fortran and R programming language. Even though there are similarities between R and S, there have been made some design time choices specific to R. The way memory is allocated is that it is set to some constant size. It is controlled with a built-in garbage collector resulting with the heap not getting too spaced up. There are substantially less paging issues in R design [8].

Another major difference in R implementation are scoping rules. Functions can potentially reach variables which were present in time when the functions were being defined [8]. The syntax of R has some resemblance with the syntax of the C programming language, even though it has functional programming language semantics [9] R is an interpreted language and is used widely by statisticians for the purpose of data analysis.

A. R Data types and Data Structures

Everything in R is an object. To elaborate on that statement, it is important to present main mechanisms in R which are used for efficient data representation as well as performing operations on supported data types. There are Object types which are internal types implemented in C. Another step was introducing vector structures which are closely related to vectors. S3 classes present another approach, that is objects which have no class definition, but can have attributes. With S3 classes there is a one-argument method dispatch. Formal (S4) classes have a class definition and support generic functions. S3 approach lacks some flexibility and S4 approach to programming is recommended [10].

Internally in C implementation of R language, all objects are pointers to a structure of type SEXPREC. In R looking at its low level implementation in the C programming language objects are pointers. They point to a structure of type SEXPREC. Also in C, SEXPTYPE is used to represent various data types from R language [9]. Various data structures will be presented summarizing all major types of objects in R, and later the main data structures will be presented more thoroughly. Vectors are one of the basic data types in R and can be thought as contiguous structures holding data. R has six basic (also referred to as 'atomic') vector types. These vector types are: integer, real, complex, logical, string (or character) and raw. Single numbers are vectors, and they have vector length of one. Lists are another basic data type and they can be thought of as 'generic vectors' meaning they can contain different elements which and each of them can be of a different type. In essence lists are vectors, but it is important to differentiate between atomic vectors and lists. Language objects are objects that are integral part of R language. They can be divided into: symbol objects, expression objects and function objects. There are also special objects (NULL type), built-in objects, promise objects (they are tightly coupled to lazy evaluation) [9]. These are not the only objects that exist but represent the most crucial types in R. There are also special compound objects: Factors and Data Frame objects. Factors are implemented using an integer array and are used to represent items that have a finite number of values. A data frame is a list of factors, matrices, vectors. What is common for them is that their length is identical [9]. General classification of R's data structures can also be made based on dimensionality and homogeneity. Homogenous data structures are atomic vector (one dimensional), matrix (2 dimensional) and array (many dimensional). Lists and data frames are heterogeneous data structures with list being one dimensional, and data frame being two dimensional [9]. Looking at the low-level implementation of R most elements of the language are internally represented by a special data structure. This

data structure is sometimes referred to as BLE (basic language element) and is composed of five machine words. The first element is a tag that contains information about the type. The second word contains a pointer and serves as a mean of associating attributes with an object. According to the specific type of object the content of the last three words can vary [8].

III. DATABASES

A database can be referred to as a collection of dynamic data which provides persistency and integration. It also provides operations to manipulate and access its data. A database management system(DBMS)presents languages and services which enable administrators and users to efficiently work with the underlying database [11]. When categorizing databases they are often distinguished by the underlying data model. A data model gives definition of the objects and operators enabling interaction for end users. It presents an abstract, self-contained definition. When talking about its implementation we are referring to its implementation on a physical machine. Knowing about implementation details of the model is not mandatory from a user's perspective, whereas knowing the data model is mandatory [11].

Database model consists of three components: a set of data structure types, a set of operators or inference rules and a set of integrity rules [12]. It is important to mention a well-known DBMS three-layer architecture consisted of the conceptual layer, the physical layer and the external layer. The physical layer is used for describing how data is organized and stored on a physical storage media. External layer presents data to the users, and the conceptual layer is used for describing the total data model [11] DBMS systems are being coordinated through usage of the data definition language (DDL), data manipulation language (DML) and data query. By describing schemas in DDL language data model of the DBMS is described [11]. When developing different data models there is a set of criteria which have to be taken into consideration. These are: the characteristics or structure of the domain to be modeled, the type of theoretical tools that are of interest to the expected users and the hardware and software constraints [12]. Before the introduction of relational data model two data models existed. They were the hierarchical model and network model. In the hierarchical data model, information is organized as a collection of inverted trees of records. The records contains fields and a field may contain simple data value, or a pointer. It is interesting that the pointer graph is not permitted to contain cycles [11]. Hierarchical databases could be explained, based on the hierarchical data model. In terms of storing, accessing and manipulating data represented in formal (S4) R classes if there is need to express complex models, hierarchical databases are impoverished in doing so, and it is not always suitable to form a parent-child relationship required to navigate in the database. Navigation in hierarchical databases starts from the root and continues in the opposite direction (from parents to children) until the desired record is found. Searching down the hierarchical tree is fast, but all other queries require sequential search techniques. It is also important to note that storing NULL values was implemented in only few concrete database implementations [11] In the network data model there are graphs of records. They can be accessed via pointers. Many individual fields constitute a record, and each field contains a simple value [11]. Some network DBMS doesn't allow NULL values which again limits the flexibility in storing certain objects from the R language data model. For the network model navigation in the database is implemented close to the physical storage structures (through pointers and indexes), implying that the network data model supports only limited data independence. Inevitably as the data model becomes more complex over time it is more difficult to maintain a network database. While providing more flexibility than hierarchical databases, network databases are not fit for efficiently persisting varying and changing data created by statistical computations. The relational database model was proposed by Codd in 1970's and proposed separation between physical and logical levels of the relational database, introducing them as different abstraction levels in design [12]. The relational data model finds its roots in relational algebra. Relational data model organizes data in two dimensional tables (relations). Every relation has a set of records (tuples). Additionally, tuples contain fields. A relation can be viewed as a truth predicate [11]. It defines attributes which are part of the predicate. An attribute has a name and a domain. Domains can be considered data types [11]. Most relational database management systems allow NULL values and that is accommodating the storage of R's NULL objects. Relational databases introduce a more logical view on data but it is also important to stress out that from the perspective of persisting R data structures it is often not desirable to adhere to relational constraints imposed by relational data model. In terms of efficient querying of the stored data it is also important to differentiate between traditional row-oriented database systems („row-stores“) and recent column-oriented database systems („column-stores“). As stated in [13]: “Column stores have been shown to perform more than an order of magnitude better than traditional row-oriented database systems on analytical workloads”. An object-oriented data model is based on concepts from object-oriented orientation. Objects are closely related to classes which serve a blueprint for object creation. Object orientated paradigm appeared as a result of joined effort in mitigating advanced systems [12]. It provides flexibility where each object has a number of attributes which can be simple values, complex values (a reference to another object due to composition) or methods. According to object-oriented database system manifesto, an OODBMS (object-oriented database management system) must satisfy two criteria, which are stating that an OODBMS must satisfy all the features of a DBMS system and it should be an object-oriented system. Implementing the second

criterion in OODBMS provides means for encapsulation, complex objects, inheritance, polymorphism and extensibility [14]. When evaluating benefits of object-oriented databases in storing R data it is important to point out that one of the key benefits in OODBMS is the possibility of modelling complex data models and extensibility via creation of new custom data types. These kinds of databases are suitable for persisting complex data structures from R's type system and since there is no impedance mismatch (due to object nature of both the object-oriented data model and R data structures). When listing the downsides of using an object-oriented data model, there are traditional setbacks in terms of no standard mathematical foundation via relational algebra. More importantly absence of a standard query language and nested queries may present a problem in efficiently accessing and manipulating persisted data in a standardized way despite the ease of manipulation via means of custom, more complex querying. There are other specialized forms of DBMS data models which emerged as a response to finding optimal ways of storing and organizing semi-structured data, or data that mainly depicts relationship between entities, or any other data for which these data models facilitate better means of database representation. The notion of graph database models is conceptualized with respect to three basic components. These components are basic data structures, integrity constraints and a transformation language. A graph database data model is a model where data is modelled as a directed, labeled graph. Graph oriented operations enable graph data manipulation. Type constructors and integrity constraints are defined over graph structures [12]. This text is referring to a full-fledged property graph model with all the benefits it introduces. A property graph is a directed multi graph. It consists of set of vertices (nodes) and edges (arcs) They are finite and mutable. [15]. The properties of vertices and edges (attributes) are simple name value pairs. If there is a need to uniquely identify a vertice, or an edge, dedicated property could be used. The graph database model gained a significant momentum in the early nineties, but its significance decreased to other database models being developed at that time (namely spatial and semi-structured data models). Taking into consideration flexibility and natural expressiveness of graph related structures, graph databases have been used lately in traditional business applications (where graph data representation proves to be more efficient) and in newer applications such as Social Media Analysis, Context aware search and social asset management [15]. R data structures which depict complex interconnections between objects are suited for storing in graph databases as they are suited for representing component interconnectivity. Retrieving complex data can be also significantly faster. Not all applications of statistical analysis in R produce complex interconnected objects and therefore suitability of using a graph database is limited to storing a subset of complex data types in R.

Further there are NoSQL (also referred to as not only SQL) databases which are supported by various data models such as key/value or documents. It is important to say that there is a distinction between traditional graph data model (and graph databases) from NoSQL underlying graph model. This text refers to graph databases as a separate entity separating it from the NoSQL class of databases. NoSQL databases as type of distributed database systems gained momentum in 2000's with the arrival of [16] applications that presented more resource requirements. Initial shortcomings of scaling the existing hardware infrastructure vertically and the impracticality of sharding middleware led to design of new distributed databases. This class of NoSQL databases focuses on availability and performance presenting eventual consistency of data [16]. This class of databases is suited for scaling out, and lacks strong transactional guarantees. As presented before they introduce alternative data models such as key/value and document [16] NoSQL databases try to solve three major problems which are not adequately addressed in relational databases [16]. These are growing data set size, growth of connectivity (as information is getting more connected) and semi-structurality of information which can be presented with few mandatory, but many optional attributes in the data model [17]. Considering the benefits of NoSQL databases when storing R data structures, R objects having few mandatory attributes which can present a basic data type or reference another object, and a handful of optional attributes which can be NULL, are suitable for persisting in document-oriented types of NoSQL class of databases. A major setback of using document databases with R data structures arises in need to write an additional code for handling inconsistent data when large quantity of data is being retrieved. There are also many applications that cannot give up strong transactional and consistency requirements making.

There is also emergence of different kinds of applications that are based on data stream processing. Mostly they could be found in the area of sensor generated data. Such data streams could be processed using R language in two ways, inbound and outbound. With inbound processing models of execution, possible streams of data could be pushed and processed as they fly by in memory, making reads and writes to traditional storage optional [18]. The main difference between inbound and outbound data processing is that former is explaining situations when data is processed before transaction and in the later model transaction appears first and then data stream is processed. At the heart of outbound processing is committing the transaction and only then processing data. This approach presents latency issues [18]. This „inbound processing“ model is employed by a stream processing engine such as StreamBas. There is also evidence in increase of the popularity of a new breed of DBMS systems collectively referred to as NewSQL DBMS that are also suitable for storing and retrieving R data structures. New SQL databases are modern databases that maintain ACID guarantees for transactions, while offering scalability of NoSQL databases for OLTP reads and writes [16]. Those systems could be categorized according to their nature of

development and for the purpose of actual use. Main categories proposed in are; new systems built from scratch, middleware reimplementing the sharding infrastructure developed from 2000s and database as a service offerings based on new architectures [16]. Analyzing systems with a completely new architecture there is a trend of storing the complete database in main memory. Thus, these systems can get better performance because traditional database components are simply left out (such as a buffer pool) [16].

Another difference with NewSQL Systems is its ability to move data from main memory to hard disc if there is an increased demand for random access memory [19]. These new types of databases are well suitable for working with R data, as in main memory data processing introduce large performance benefits. Being ACID compliant, and supporting storage of vast quantities of data due to increase in main memory price, they are suitable for data-intensive, fast workloads. In some cases there is a need to consolidate data from disperse data sources in order to do statistical analysis in R. There are technologies addressing this need enabling virtual connections to remote database objects where only metadata about remote sources is stored in virtual tables. Unification of disconnected sources presents a new approach in statistical analysis of data being present in different external databases [19]. Cell stores is also one among of new approaches in structuring a database model. The cell store data model organizes data at the cell level. Cells can be seen as atoms of data. Cell stores are associated with dimension data and a big collection of cell stores can be easily clustered, replicated or retrieved efficiently. Cells can also be assembled into data cubes or into spreadsheet views [20]. Structuring R data types into the cell store data model allows storing highly dimensional data.

IV. DISCUSSION

In Table 1 different types of databases (or streaming engines) and their suitability to store R data structures (in terms of key benefits and setbacks) are discussed. Streaming data processing is not directly related to a particular database but as more and more data from sensors is collected, processing it «on the fly» before it is stored in a database (or after transactions are executed) deployment of R language in those procedures is of a high importance.

TABLE I.
Properties of different database types in relation to R data structures storage ability

Database types and streaming engines	R usage key benefits and deficiencies
Hierarchical	Searching down the hierarchical tree is fast but there is no possibility to store NULL values
Network	Network databases are not fit for efficiently persisting varying and changing data created by statistical computations.
Relational	Imposed data relations and constraints are not very suitable for statistical computing because to be able to perform different analysis, data models should be easy to change. Column store database types could help to overcome such limitations, as data could be stored in one or few tables, making them more suitable for statistical computing. They also support storage of NULL values.
Object	Object-oriented databases are beneficial for processing R data as they give the possibility to analyze and process complex data models and are able to support data extensibility via provisioning of new custom data types. The limitations of using this type of database are that there is no standard mathematical foundation via relational algebra and there is an absence of standard query language.
NoSQL	Analyzing data sets with few mandatory fields but with a high velocity of additional attributes are the main benefits of using NoSQL for the purpose of storing and processing R data. The main setback is manifested in the need to write a lot of additional code for handling inconsistent data when a large quantity of data is being retrieved.
New SQL	Use memory to store data, utilize the opportunities of cloud computing based on data virtualization techniques and use different data sources stored in different databases.
Cell	Store and analyze highly dimensional data.
Streaming	Reduce the amount of data before transaction (inbound) and/or analyze data streams that are a result of a high number of transactions. In both cases, out of the large volume of data streams, systems use only those parts that are relevant to triggering a particular function (by using R scripts) and that data is stored in a database system chosen by a designer.

V. CONCLUSION

We have described new trends influencing the design of data processing information systems. The trends are: 1) decreasing ratio between continuously collected and analyzed data; 2) increasing number of available data sources; 3) accelerated introduction of cloud and multi-cloud information systems. These trends will have a significant impact on the development of current and future data mining systems. Knowledge discovery research will also have to adapt to the new situation and requirements from the IT industry. Incremental system integration and intelligent user interfaces regulated by information need will become a necessity [21] [22] [23]. The standard data warehouse paradigm will become inadequate and new concepts for acquisition, integration and analysis of heterogeneous data sources will have to emerge. The most immediate and efficient solution is the use of a standardized data model built on computer statistical language R. We believe that the importance of

R in the design of information processing systems will continue to grow.

VI. REFERENCES

[1] J.Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," IDC iView: IDC Analyze the future, pp. 1–16, 2012.

[2] A. Abbasi, S. Sarker, and R. H. Chiang, "Big data research in information systems: Toward an inclusive research agenda," Journal of the Association for Information Systems, vol. 17(2), 2016.

[3] I. A. T. Hashem, I.Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: Review and open research issues," Information Systems, vol. 47, pp. 98–115, 2015.

[4] M. Stonebraker, D. Bruckner, I. F. Ilyas, G. Beskales, M. Cherniack, S. Zdonik, A. Pagan, and S. Xu, "Data Curation at Scale: The Data Tamer System," In CIDR, 2013.

[5] M. Golfarelli, S. Rizziand, and I. Cella, "Beyond data warehousing: what's next in business intelligence?," In Proceedings of the 7th ACM international workshop on Data warehousing and OLAP , pp. 1–6, 2004.

[6] S. Rizzi, A. Abelló, J. Lechtenbörger, and J. Trujillo, "Research in data warehouse modeling and design: dead or alive?," In Proceedings of the 9th ACM international workshop on Data warehousing and OLAP, pp. 3–10, 2006.

[7] L. H. R. M. B. Niswonger, P. M. Roth, Schwarz and E. L. Wimmers, "Transforming heterogeneous data with database middleware: Beyond integration," Data Engineering, vol. 31, 1999.

[8] R. Ihaka, "R: Past and future history," In Dimension Reduction, Computational Complexity and Information: Proceedings of the 30th Symposium on the Interface, Minneapolis, Minnesota, pp. 13–16, 1998; Computing Science and Statistics, vol. 30; Interface Foundation of North America, pp 392– 396, 1998.

[9] R. C. Team, "R language definition," Vienna, Austria: R foundation for statistical computing, 2000.

[10] J. Chambers, "Software for data analysis: programming with R," Springer Science & Business Media, 2008.

[11] L Bergholt, J. S. Due, T. H. Daimi, J. L. Knudsen, K. H.Nielsen, T. S. Olesen, and E. H. Pedersen, "Database management systems: relational, object-relational, and object-oriented data models," 1998.

[12] R.Angles and C. Gutierrez, "Survey of graph database models," ACM Computing Surveys 40, 1, pp. 1–39, 2008.

[13] D. J. Abadi, S. R. Madden and N. Hachem, "Column-stores vs. row-stores: how different are they really?," In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp. 967–980, 2008.

[14] H.Alzahrani, "Evolution of Object-Oriented Database Systems," Global Journal of Computer Science and Technology, 2016.

[15] M. Rudolf, M. Paradies, C. Bornhövd, and W. Lehner, "The Graph Story of the SAP HANA Database," In BTW, vol. 13, pp. 403–420, 2013.

[16] A. Pavlo and M. Aslett, "What's really new with NewSQL?," ACM Sigmod Record, vol. 45(2), pp. 45–55, 2016.

[17] C. J.Tauro, S.Aravindh and A.B. Shreeharsha, "Comparative study of the new generation, agile, scalable, high performance NOSQL databases," International Journal of Computer Applications, vol. 48(20), pp. 1–4, 2012.

[18] M. Stonebraker and U. Cetintemel, ""One size fits all": an idea whose time has come and gone," In Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on, pp. 2–11, 2005.

[19] A. Pattanayak, "Data Virtualization with SAP HANA Smart Data Access," Journal of Computer and Communications, 5(08), 62, 2017.

[20] G. Fourny, "Cell Stores," arXiv preprint arXiv:1410.0600, 2014

[21] S.Lugović, I. Dunđer and M. Horvat, "Primary and secondary experience in developing adaptive information systems supporting knowledge transfer," In Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2017 40th International Convention on, pp. 1207–1210, 2017.

[22] S., Lugović, I., Dunđer, and M. Horvat, "Secondary Experience of an Information System Enabling Scientific Communication," In Proceedigs of the 1st International Communication Management Forum (CMF2015), pp. 562–587, 2015.

[23] S., Lugović, I., Dunđer, and M. Horvat, "Patterns-based information systems organization," In Proceedings of the 5th International Conference: The Future of Information Sciences (INFuture), pp. 163–174, 2015.