DATA MODEL FOR BIGDEEPEXAMINATOR

Janusz Bobulski and Mariusz Kubanek

Department of Computer Science, Czestochowa University of Technology, Poland

ABSTRACT

Big Data is a term used for such data sets, which at the same time are characterized by high volume, diversity, real-time stream inflow, variability, complexity, as well as require the use of innovative technologies, tools and methods in order to extracting new and useful knowledge from them. Big Data is a new challenge and information possibilities. The effective acquisition and processing of data will play a key role in the global and local economy as well as social policy and large corporations. The article is a continuation of research and development works on the design of the data analysis system using artificial intelligence, in which we present a data model for this system.

KEYWORDS

Big data, intelligent systems, data processing, multi-data processing.

1. INTRODUCTION

The rapidly growing amount of information, related to the growing popularity of cloud computing and technologies such as the Internet of Things, requires increasingly powerful data processing and storage systems. The access to the collected information itself also turns out to be a big problem, when it suddenly turns out that reading or analysing everything we have on a small scale is both a feasible and estimable thing about the time needed. It will take a while, but in the end we'll get some result that we expected more or less. If we add the Big Data factor to it, it suddenly turns out that errors at the level of systematizing data or their structures are costly not only at the level of the user, but also of the entire cloud solution. If for some reason we have to read all the collected records, we may fall into the trap that we are not able to finish the process before e.g. the next iteration of daily reports, which in this case may have to operate on two data sets, further destabilizing architecture. This is not a scenario in which we would like to find ourselves. Processing large data sets presents many challenges, e.g. problems with understanding and acceptance of large data sets, diversity of Big Data technology, high system costs, scalability, data quality and transformation of large data sets into valuable information [1,2,3].

Insufficient understanding and acceptance of large data sets causes problems with the implementation of projects in many companies. Lack of knowledge about what large data is, what infrastructure is necessary, etc. causes companies to lose a lot of time and resources, which causes delays in the company's progress and development. Big data and related knowledge should be obtained and accepted by the company's management, and then transferred down the structure. To ensure understanding and acceptance of large data at all levels, IT departments must organize numerous training and workshops. In order to better accept changes in data management, all Big Data implementation processes must be monitored and controlled on an ongoing basis.

Currently, there is a large variety of technologies used on the market. In the course of their analysis, the questions arise: do I need Spark, or the speed of Hadoop MapReduce is enough, can it better store data in AWS or Cassandra? Finding the right solution is difficult, especially if you lack the knowledge you need. It is worth to hire an expert in this situation or turn to a professional company that provides Big Data solutions.

This solution will also reduce the costs of implementing new technologies in the company. Open-source solutions seem to be cheaper at the beginning, but considering all the additional costs, i.e. new equipment and personnel, it may turn out that this solution is more expensive and less productive in the long run, but it will certainly slow down the company's development.

In justified cases, one should use the offer of suppliers offering ready solutions operating in the cloud, thanks to which we will also avoid some problems with scalability. If we additionally choose the right architecture, we will guarantee the proper system performance. An additional important issue from the point of view of scalability is the adequately design algorithms of large data sets, including the possibility of expansion.

In Big Data there is a big problem with the integration and homogeneity of data, because they come from different sources and there are different formats, for example, the format of dates in the US and in Europe, whether upper and lower case letters in the name of their own. This results in the need to control their condition [3,4]. The key factor here is the correctness and quality of the data, because in the case of their lack the results of the system will be unsatisfactory, according to the principle: garbage at the input-garbage at the output. Therefore, the system should be equipped with a data control and conditioning module, analogous to signal processing systems where the input signal is filtered to eliminate interference [5,6,7].

2. BIG DATA ARCHITECTURES

2.1. Lambda Architecture

One of the two architectures used in Big Data systems is the Lambda architecture, which provides parallel processing of large data sets and the possibility of continuous access to them in real time. The idea be-hind this architecture is to create two separate flows, where one is responsible for data processing in batch mode, while the other one for accessing them in real mode [9]. A steady stream of data is directed to both layers (Fig.1). In the batch layer, calculations are performed on the entire data set. It happens at the expense of time, but the data received in return contain the full history and high quality. It is assumed that the data set in the batch layer has an undivided form, which should only be expanded rather than delete data from it. In this way, you can ensure data consistency and access to historical data.

The real-time layer processes incoming data in real mode. The short time of access to data in this layer translates into the possibility of faster information retrieval. Unfortunately, the lack of access to historical data means that not all calculations are possible. Often the quality and faith-dignity of data from the real time layer is not as high as from the batch layer. The latter should be considered more reliable, but due to the longer time needed to load them, the real-time layer proves to be an invaluable help in order to guarantee the possibility of processing data in real mode.

The access layer is responsible for creating views based on the batch and real-time layers. The data is aggregated in such a way that the end-user sees it as a single, coherent whole. Views should be prepared in such a way as to ensure the possibility of performing all kinds of ad-hoc analyses, while enjoying fast access to data.

The Lambda architecture concept provides many advantages, primarily a perfect compromise between batch and real-time processing. The biggest and most often mentioned disadvantage is the need to maintain two independent applications - one for powering the batch layer, and the other for the real-time layer. The tools used in each layer are different, so it's hard to choose one that can be used for two purposes. Unfortunately, this architecture is more complicated and more expensive to maintain, so for our system we chose the second of the popular architectures used in Big Data - Kappa.

2.2. Kappa Architecture

Kappa's architecture appeared as a response to criticism related to implantation and maintenance of systems based on Lambda architecture [9]. The new architecture was based on four main assumptions:

1. Everything is a stream - the stream is an infinite number of completed data packages (batches), so that each data source can be a data stream generator.
2. Data is immutable - raw data is persisted in the original form and does not change, so that you can use them again at any time.
3. The KISS rule - Keep is short and simple. In this case, using only one engine for data analysis instead of several, as in the case of Lambda architecture.
4. The ability to restore data state - calculations and their results can be refreshed by retrieving historical and current data directly from the same data stream at any time.

It is critical for the above rules to ensure that the data in the stream remains unchanged and original. Without this condition being met, it is not possible to obtain consistent (deterministic) calculation results.

The idea of Kappa architecture is shown in Fig.1, and we may see a simplified structure in relation to Lambda architecture. Just like in the Lambda architecture, we have a Real-Time layer and an Access layer that performs the same functions here. In contrast, there is no layer of the Batch, which has become redundant because history can be reconstructed at any time from the data stream at the Access layer using identical data processing engine.

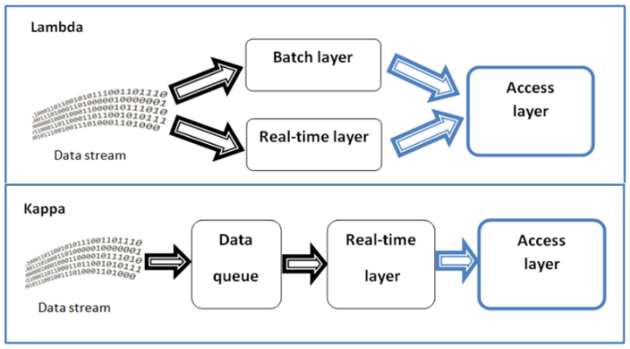

Figure 1. Comparison of Lambda and Kappa architectures

3. BIGDEEPEXAMINATOR SYSTEM PROJECT

The name of BigDeepExaminator system is an abbreviation of the names of the component system technologies that is: Big Data, Deep Learning and Examinator of data.

3.1. Elements of the System

The general scheme of the system is shown in the Figure 2. Main parts of the systems are:

1. System: A system for processing any type of data that is able to handle the system, having its own language and grammar, can learn, is intelligent and able to find solutions to new problems; universal for tasks requiring reasoning in a general sense, using the knowledge base; a system capable of learning and gathering knowledge, as well as data mining and extracting new knowledge from the knowledge possessed; the goal of the system is to gather knowledge, not data.

2. User: a person, device or system having the ability to communicate with the system, having the ability to handle system input / output streams. This applies to formulating questions, characterizing the analysed problem, and interpreting the results of the system's operation.

3. Data: a set of information in the form of a data stream as well as a file. The system should process data in both known formats and new types.

4. External commands: queries from the user in natural language, possibly as simple as possible, constituting correct sentences of a subset of natural language defined in accordance with syntax - grammar, i.e. alphabet, and construction rules.

5. Methods library: a set of functions that will work on a set of external data provided to the system, based on algorithms of artificial intelligence, deep learning, application algorithms, data processing and analysis; this library will be used by the processing unit.

6. Pre-processing: recognition and interpretation of input streams or streams in order to select optimal methods of information processing; recognition of data forms and their processing to the necessary internal form with the analysis of correctness and error signalling.

7. Language interpreter: program that verifies the correctness of external commands, intelligent, interactive parser and generator of executable code or an internal language suitable for the system to perform in the processing; the interpreter can be in the form of an agent or bot using preliminary analysis and algorithms based on fuzzy sets and artificial neural networks, speech recognition [10].

8. List of internal commands: a set of internal system instructions that are not available to an external user; using a library of methods, its own internal data format and its own language and grammar.

9. Knowledge database: Knowledge of the system stored in accordance with specific rules; objects and relations between them introduced from outside or deduced by the system and stored in the system in a specific way (human and system-readable) enabling quick access to them and their use in taking partial actions in solving problems and in requesting; knowledge base will use elements 3-9.

10. Analysis module: transformation of input streams into output streams based on elements 3-9; it is a part of the system that enables the data to be processed using all of its isolated elements; execution of the code or internal language generated by the language interpreter and features extraction for recognition module.

11. Recognition module: artificial intelligence sub-systems based on Deep Learning neural networks; recognizes patterns and interprets responses.

12. Results: streams of output data that meet the expectations formulated by the user and presented in a way that is understandable to him; they also supply the knowledge base; this is every effect / result of the system operation.

13. System input: there are data on the system input - see point 3.

14. Additional data and information entered into the system: the system has the option of downloading additional external data in the form of knowledge or new methods and information about new data formats.

15. System output: a data stream obtained in the results of data processing by the system.

16. The data will not be collected. They will be used to extract knowledge. The "raw" data being the basis of the knowledge discovery process is very often characterized by the following features: measurement errors, missing values in data, disruption during sampling, human mistakes. The system should be resistant to such errors. Therefore, replacing such data with empty values is more justified than with null values, because "zero" is also a value and can lead to incorrect conclusions being drawn [11].

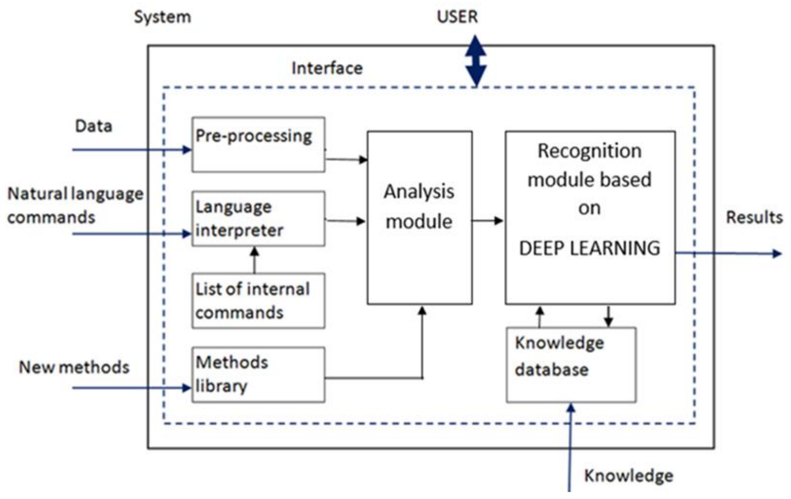

Figure 2. BigDeepExaminator system architecture

4. Data Model

4.1. Nosql Database Models

Database model is understood as a set of rules that describe the structure of data in a given database. Allowed operations or data structure are defined by specifying representations of entities and relationships allowed in a given model. In the context of Big Data processing, NoSQL (Not only Structured Query Language) databases that have no SQL database (Structured Query Language) restrictions apply.

The obvious change introduced by NoSQL databases is the departure from the relational data model. Each type of NoSQL database uses a different data representation model. These types are very diverse, so they can be divided into two basic groups, presenting the aggregation data model and graph. There are usually four types of NoSQL databases: key-value, document, column and graph. The first three are similar enough to include the term aggregation oriented databases [12].

The aggregation approach is definitely different from the relational option, which stores data broken down into tuples, which are a limited data structure that cannot contain any nests. Such simplicity is the basis of the relational model, so you can identify with database operations the modification and return of tuples.

The aggregation data model is willing to operate on data in more complex units than a set of tuples. We are talking here about the category of the extended record model, which allows storing other records or lists. Key-value, document and column databases allow the creation of such records, so they have been classified into this model group.

The term "aggregations" comes from Domain-Driven Design modelling. In this approach, aggregation is a collection of objects that we treat as an entity. It is usually a unit designed for manipulation and consistency management. Usually, we want to update aggregations using atomic operations and communicate with the data warehouse by passing aggregations. This definition fits very well with how key-value databases, documents and column families operate.

Operating on aggregates will make it easier for databases to work in clusters, because aggregation is a natural unit for replication and sharing [12].

The document database stores and retrieves documents on the basis of an electronic document cabinet. Documents often contain maps and lists, which allows hierarchies such as JSON and XML. In the simplest terms, documents can be stored and obtained by using ID, they are equivalent to rows / records [12].

Document databases may behave like key-value databases, but in general they are based on indexes in order to facilitate access to documents based on their attributes. You can refer to fields that are not a primary key. An entry in the database consisting of key-value fields is considered to be a document.

Key-value warehouses are the second family of NoSQL databases blurring the document model of data storage. They store key-value pairs. Access to data is only possible via a key. This type of database was modelled on the Amazon Dynamo database. Key-value databases behave like large, distributed hash table arrays that store and retrieve data of an unspecified type (opaque data) using a key. Key-value databases, like document databases, are strongly based on aggregations. Both types of databases have a significant number of aggregations, each containing a key or identifier enabling access to it. The difference between these models is that in key-value databases, the data is transparent to the database (a set of usually insignificant bits). However, in the document option, the database has the ability to view the aggregation structure. As an advantage of key-value databases, it can be assumed that any items can be stored in the database. The database has the option of introducing size restrictions, but beyond that, the user has full freedom of use. The document database has restrictions on stored structures included in the definition of acceptable structures and types. In return, however, there is greater flexibility in accessing data, because in key-value databases you can access aggregation only through its key, where in the document database you can send queries to the database based on aggregation fields (you can download some aggregation, and the database has the ability to create indexes based on aggregation content).

Another type of NoSQL database is column family database that contains columns of related data. It was one of the first influential NoSQL databases, created by Goggle under the name BigTable. The name refers to the tabular structure, implemented with the help of loose columns and no diagram. The basic unit of data storage is the column itself, containing the name-value pair. Any number of columns can create a super column structure that provides a name to the sorted column set. The columns are stored in rows. In the event that a row contains only columns, it is called a column family. When a row contains supercolumns, it is called the supercolumn family [12].

A common feature of the NoSQL databases discussed is indexing aggregation using a key that the user can use to download it. Aggregation is the basis for cluster operation, because the database is designed to store all data contained in aggregation on one server. Aggregation is also defined as the basic unit during data update, which results in obtaining useful, but limited, transactional control.

The difference between aggregation-oriented database types lies in the structure of internal aggregation notation. For the key-value database, aggregation is a transparent whole, which translates into the ability to retrieve data only using a key.

In document databases, the aggregation structure is familiar to the database, which makes it possible to perform queries and retrieve any part of the data. The document has no strictly defined structure. The database cannot optimize the storage and retrieval of aggregation fragments based on the structure.\\

The databases from the column family introduce the aggregation division into column families. This option allows the database to aggregate aggregates as a unit of data within a row. This solution adds a structure that the database can use to increase the flexibility of data access [13,14].

To sum up, aggregation is a set of data that functions as a unit in a database. Thanks to aggregations, work in clusters is definitely easier, which puts NoSQL databases in a good light. These three models described above are used to store large amounts of data, but in their nature do not offer semantic inference. Therefore, they are not suitable for use in our system.

4.2. Graph Databases

Most no relational databases were created for work in clusters, which is why the model focused on large aggregation records, which are combined using simple relationships, has become more popular. The rise of graph databases, however, was caused by other problems that relational databases could not cope with at a satisfactory level.

The graph model of data representation is a model with labelled and directed multi-graphs, which contains attributes. Labelled, because it has labels for all edges. Directed, because it has edges with certain direction (from the source to the end node). The presence of attributes in the graph causes the attribute list variable to be assigned to each node and edge. The attribute is the value associated with the name.

A multigraph can have multiple edges between two nodes. Therefore, different edges can connect two nodes many times, without paying attention to whether these edges have the same source, end node and label.

In GBD, graph is a native form of information storage. It is often stated that GBD provides index-free neighbourhoods. In GBD, information about vertex neighbours is stored locally. For comparison, in RBD there would be a reference to an index that would show the vicinity of the vertices. Therefore, in the graph option, the neighbourhood search time depends on the number of vertices neighbours. In the relational option, however, on the number of all edges [15].

The development of the Internet, the need for flexibility and large data volumes are the main needs for the introduction of graph databases. They are designed to work in situations where relational databases are not doing well. This is mainly about poor performance of graph structure storage in RBD. This is difficult, unnatural and inefficient for RBD [15].

The problem with query performance, which includes many joins, low RBD scalability, semistructural data, and problems with normalizing graph structures stored in RBD are still not all reasons why GBD should be used. Semi-structured data is often modeled as large tables, with lots of columns, blank for most rows, which significantly reduces performance. The alternative in the form of modeling this data, with the participation of many table connections, is also not an efficient solution, considering the costs of joins during queries [16].

In addition, normalization of graph structures in RBD degrades performance during queries. Such a decrease in performance is related to the recursive nature of e.g. file trees or social networks, as well as to the form of recording data as relationships. Each operation performed on the edge of the graph gives the effect of joining between tables in a relational database. This is a very slow operation, which in addition is not scalable [16].

GBD applications can be divided into two categories: complex networks and classic applications. Complex networks are characteristic for areas with large amounts of data and network structure. Among the basic applications in which GBD works much better are: knowledge representation systems, semantic networks and social networks [17].

GBD support a graph model that allows direct storage of specific objects and relationships between them in a database. GDB should allow access to query methods that cope not only with stored objects, but with the graph structure itself. The best known example is traverse, which in its simplest form can be used to obtain adjacent vertices in the graph. The use of the graph database in the BigDeepExaminator System has the additional advantage of being able to search graphs in depth and breadth.

The use of GDB technology will result in the creation of a semantic data network. The semantic web is a web in which individual elements have their meaning, accessible not only to people, but also to machines. A semantic web is not a specific product, specification or standard. It's more of an idea or vision than technology. The goal of the semantic web project is to make data available for processing: people and machines so that they can be used not only for display purposes, but also for automation, integration and reuse in many different applications, such as intelligent agents. They will use distributed databases in the form of semantic networks. This in turn will allow the creation of an automatic infrastructure that, if properly designed, will make a significant contribution to the evolution of human knowledge [18]. In addition, the semantic web enriches the current web with annotations written in a machine-process able language, which can also be associated with each other [19].

5. CONCLUSION

The article is a continuation of research and development works on the design of the data analysis system using artificial intelligence, in which we present a data model for this system. Big Data is a one of the most important challenges of the modern digital world. Big Data as a complex of IT issues requires the introduction of new data analysis techniques and technological solutions that will allow to extract valuable and useful knowledge from them. New technologies of data collection and processing force interdisciplinary research and the need to combine existing solutions. Future large IT systems will be based on techniques that use Big Data, so new technologies should be developed that can process large amounts of data and extract useful knowledge from them, which will require artificial intelligence and Deep Learning techniques.

ACKNOWLEDGEMENTS

Supported by the project financed under the program of the Polish Minister of Science and Higher Education under the name "Regional Initiative of Excellence" in the years 2019 – 2022. Project number 020/RID/2018/19 the amount of financing 12,000,000 PLN.

REFERENCES

[1] Buhl, H., Röglinger, M., Moser F., Heidemann J. (2013) "Big Data", Business & Information Systems Engineering, April 2013, Volume 5, Issue 2, pp 65–69.

[2] Zikopoulos, P., Eaton, C., deRoos, D., Deutsch, T., Lapis, G.: Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, McGraw Hill, USA, (2012)

[3] Bekker A., Last accessed 4 April 2019 "7 major big data challenges and their solutions", ScienceSoft, https://www.scnsoft.com/blog/big-data-challenges-and-their-solutions,

[4] Jinchuan, C., Yueguo, C., Xiaoyong, D., Cuiping, L., Jiaheng, L., Suyun, Z., Xuan, Z.(2013) "Big data challenge: a data management perspective", Frontiers of Computer Science, SP Higher Education Press, vol. 7, issue 2, pp. 157-164.

[5] Katal, A., Wazid, M., Goudar, R. H. (2013) "Big Data: Issues, Challenges", Tools and Good Practices, Sixth Interna-tional Conference on Contemporary Computing (IC3), IEEE, Noida 2013, pp. 404-409.

[6] Doug, L.(2011) Data Management: Controlling Data Volume, Velocity, and Variety, Application Delivery Strategies, META Group, Gartner.

[7] Maslankowski, J.(2015) Analysis of the quality of data obtained from websites using Big Data solutions, Annals the Collegium of Economic Analysis.

[8] Bobulski J., Kubanek M. (2019) "Design of the BLINDS System for Processing and Analysis of Big Data - A Pre-processing Data Analysis Module", Advances in Intelligent Systems and Computing, Volume 889, pp. 132-139.

[9] Marz N., Warren J. (2015) Big Data Principles and Best Practices, Manning Publications Co.

[10] Zadeh, L.A.(2012) "Computing with Words: Principal Concepts and Ideas", Springer Publishing Company, Incorporated.

[11] Schank, R.C.(1975) Conceptual Information Processing, Yale University, New Haven, Connecticut.

[12] Sadalage J. P., Fowler M.,.Hubisz J. (2013) NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Pearson Education.

[13] Pokorny J.(2013) "NoSQL databases: A step to database scalability in web environment", International Journal of Web Information Systems, Vol. 9, Iss 1, pp. 69 – 82.

[14] Barbierato E., Gribaudo M., Iacono M.(2014) "Performance evaluation of NoSQL big-data applications using multi-formalism models", Future Generation Computer Science,Vol.37, pp. 345-353.

[15] Kolomičenko V.(2013) Analysis and Experimental Comparison of Graph Databases, Masters`s Thesis, Charles University in Prague.

[16] Robinson I., Webber J., Eifrem E. (2013) Graph Databases. O`Reilly Media.

[17] Slotwinski D.(2010) Graph databases - technology review, AGH.

[18] Berners-Lee T., Hendler J., Lassila O.(2001) "The semanticweb", Scientific american, 284(5), pp.34–43.

[19] Euzenat J., Shvaiko P.(2007) Ontology Matching, Springer Science & Business Media.

Authors

Janusz Bobulski is professor in Department of Computer Science, Czestochowa University of Technology, He received PhD in 2004, and Habilitation in 2018. Areas of his research: image processing, big data, artificial intelligence, multimedia and biometrics.

Mariusz Kubanek is professor in Department of Computer Science, Czestochowa University of Technology, He received PhD in 2005, and Habilitation in 2015. Areas of his research: image processing, artificial intelligence, speech recognition, multimedia and biometrics.