Analysis of Big Data Storage Tools for Data Lakes based on Apache Hadoop Platform

Vladimir Belov, Evgeny Nikulchev
MIREA—Russian Technological University, Moscow, Russia

***Abstract*—When developing large data processing systems, the question of data storage arises. One of the modern tools for solving this problem is the so-called data lakes. Many implementations of data lakes use Apache Hadoop as a basic platform. Hadoop does not have a default data storage format, which leads to the task of choosing a data format when designing a data processing system. To solve this problem, it is necessary to proceed from the results of the assessment according to several criteria. In turn, experimental evaluation does not always give a complete understanding of the possibilities for working with a particular data storage format. In this case, it is necessary to study the features of the format, its internal structure, recommendations for use, etc. The article describes the features of both widely used data storage formats and the currently gaining popularity.**

Keywords*—Big data formats; data lakes; Apache Hadoop; data warehouses

I. Introduction

One of the most important tasks of any systems for data processing is a problem of storing the data received. In traditional approaches, the most popular tools for storing data have been the use of relational databases [1], which represent a convenient interface in the form of SQL for manipulating data.

The growth in the volume of data and the needs of consumers of data processing systems has led to the emergence of the big data concept [2, 3]. Big data concept is based on six aspects such as value, volume, velocity, variety, veracity, and variability [4]. It means that big data can be understood through not only the volume, but also their ability to be sources for generating valuable information and ideas [5].

New concepts have replaced traditional forms of data storage, among which NoSQL [6] solutions and so-called data lakes [7-9]. A data lake is a scalable system for storing and analyzing data retained in their native format and used for knowledge extraction [6]. A data lake can either be designed from scratch or developed on the basis of existing software solutions [8]. Many implementations of data lakes use Apache Hadoop as a basic platform [9].

For data lakes built based on the Apache Hadoop ecosystem, HDFS [10] is used as a basic file system. This file system is cheaper for use than commercial data bases. Using such data warehouse, choosing the right file format is critical [11]. File format determines how information would be stored in HDFS. It is required to take into account that Apache Hadoop and HDFS does not have any default file format. This determined the emergence and use of various data storage formats in HDFS.

Among the most widely known formats used in the Hadoop system are JSON [12], CSV [13], SequenceFile [14], Apache Parquet [15], ORC [16], Apache Avro [17], PBF [18]. However, this list is not exhaustive. Recently, new formats of data storage are gaining popularity, such as Apache Hudi [19], Apache Iceberg [20], Delta Lake [21].

Each of these file formats has own features in file structure. In addition, differences are observed at the level of practical application. Thus, row-oriented formats ensure high writing speed, but column-oriented formats are better for data reading.

A big problem in the performance of platforms for storing and processing data is the time to search and write information, as well as the amount of data occupied. Managing the processing and storage of large amounts of information is a complex process.

In this regard, when building big data storage systems, the problem arises of choosing one or another data storage format. To solve this problem, it is necessary to proceed from the assessment results according to several criteria.

However, testing and experimental evaluation of formats does not always provide a complete understanding of the possibilities for working with a particular data storage format. In this case, it is necessary to study the features of the format, its internal structure, recommendations for use, etc.

The aim of this paper is to analysis the formats used for data storing and processing in data lakes based on Apache Hadoop platform, their features, and possibilities in application for various tasks, such as analytics, streaming, etc. This study is useful when developing a system for processing and storing big data, as it comprehensively explores various tools for storing and processing data in data lakes. In turn, a misunderstanding of the features of the structure and recommendations for the use of tools for storing data can lead to problems at the stage of data processing systems maintenance.

The article describes both well-known and widely used formats for storing big data, as well as new formats that are gaining popularity now.

The paper is organized as follows. In the Background section, the main prerequisites for the emergence of data lakes, as well as the features of the file formats used to store data in

data lakes built on the basis of the Hadoop platform will be discussed. Challenges section explores emerging storage trends for building data lakes.

II. Big Data Storage Formats

Relational databases are the traditional way of storing data [22]. One of the obvious disadvantages of this storage method is the need for a strict data structure [23].

In recent years, the direction of development of the so-called data lakes has gained popularity [7–9]. A data lake is a scalable system for storing and analyzing data retained in their native format and used for knowledge extraction [7]. The prerequisites for this were the following factors:

- Growth in the volume of unstructured data, such as the content of web pages, service logs, etc. For these purposes, it is not assumed that there is a common format.
- The need for storing and analyzing large amounts of semi-structured data, such as events from the data bus, unloading from operational databases, etc.
- Development of OLAP technologies and analytics-oriented data storage facilities.
- Development of streaming processing and data transmission.

A data lake can either be designed from scratch or developed on the basis of existing software solutions [8]. Many implementations of data lakes use Apache Hadoop as a basic platform [9]. For such systems, HDFS [10] is used as a basic file system. The traditional way of storing data in HDFS is to create files of various formats.

According to the internal structure of the file, the formats used for working with big data can be divided into the following groups:

- Textual formats: CSV, JSON.
- Hadoop-specific formats: Sequence files.
- Column-oriented formats: Parquet, ORC.
- Row-oriented formats: Avro, PBF.

Each of the groups is focused on solving specific problems. Thus, row-oriented formats ensure high writing speed, but column-oriented formats are better for data reading.

Each data storage format will be discussed below.

A. Textual Formats

JSON (JavaScript object notation) is a textual file format, represented as an object consisting of key-value pairs. The format is commonly used in network communication, especially with the rise of REST-based web services [12]. In recent years, JSON have been becoming popular in documented NoSQL databases [24] such as MongoDB, Couchbase, etc.

In addition, JSON is popular in systems that require data transfer because many programming languages support serialization and deserialization using this format by default. This also applies to streaming data processing systems.

- JSON supports following data types:
- primitive: null, boolean, number, string;
- complex: array, object.

CSV (comma-separated values) is a textual file format presented in the form of a table, the columns of which are separated by a special character (usually a comma). The file may also contain a header containing the names of the columns. Despite its limitations, CSV is a popular choice for data exchange because it supports a wide range of business, consumer and scientific applications [13]. In addition, many batch and streaming systems (e.g. Apache Spark [25]) support this format by default.

B. Hadoop-specific Formats

SequenceFile [14] is a binary format for storing data. The file structure is represented as serialized key-value pairs. The peculiarity of this file is that it was specially developed for the Apache Hadoop ecosystem. The structure allows you to split the file into sections during compression, which provides parallelism in data processing.

SequenceFile is a row-oriented format. The file structure consists of a header followed by one or more entries. The header provides technical fields such as the version number, information about whether the file is compressed, and the file's metadata.

There are three different SequenceFile formats depending on the type of compression.

- no compression;
- record compression – each entry is compressed as it is added to the file;
- block compression – compression is performed when data reaches block size.

C. Column-oriented Formats

Apache Parquet [15] is a binary column-oriented data storage format. The format architecture is based on "definition levels" and "repetition levels". An important part of this format is the presence of metadata that stores basic information about the data in a file, which contributes to faster filtering and data aggregation in analysis tasks.

The file structure is represented by several levels of division:

- row group - row-by-row data splitting into rows for faster reading when working in parallel using the MapReduce algorithm.
- column chunk - data block for a column in a row group. This partition is intended to speed up work with a hard disk - in this case, data is written not by rows, but by columns;
- page - is a conceptually indivisible unit containing meta information and encoded data.

Apache Parquet supports the following data types:

- primitive (int32, int64, int96, float, double);
- complex (byte array, time, maps, lists, etc.);
- logical (boolean).

ORC (Optimized Row Columnar) [16] is a column-oriented format for data storing in Apache Hadoop system. This format is optimized for reading big data streams.

Architecturally, this format is similar to the Apache Parquet format. The format structure is divided into metadata and data itself. Metadata stores statistical and descriptive information, indexes, data partitioning information. The data itself is divided into so-called stripes. Each lane is an atomic unit for distributed data manipulation.

ORC supports a full set of types, including complex types (structures, lists, maps, and unions).

D. Row-oriented Formats

Apache Avro [15] is row-oriented format for data storing widely used for data serializing. Apache Avro stores the schema in an implementation independent JSON format making it easier to read and interpret by programs. The Avro file consists of a header and data blocks. The header contains file metadata containing a schema and a 16-byte random number marking the file. For data blocks, Avro can use a compact binary encoding or JSON format, convenient for debugging.

Unlike many other Big Data formats, Avro supports schema evolution by handling schema changes by skipping, adding, or modifying individual fields. Avro is not a strongly typed format: the type of each field is stored in the metadata section along with the schema. This means that no prior knowledge of the schema is required to read the serialized information.

Apache Avro supports following data types:

- primitive (null, Boolean, int, long, float, double, string, bytes, fixed);
- complex (union, record, enum, array, map);
- logical (decimal, date, time, timestamp, uuid).

PBF (Protocolbuffer Binary Format) [18] is row-oriented format. A format contains a header followed by a sequence of data blocks. The structure of the format is intended to allow random-access to the file content skipping unwanted data.

The format contains a repeating sequence of the following parts:

- the number presenting the length of the BlobHeader message in network byte order;
- serialized BlobHeader message;
- serialized Blob message.

One of the features of the format is that when serializing integers, it defaults to variable length format, which takes up less space for small positive numbers. However, the format adds the field number and its type to the binary stream, which increases the total size.

PBF supports following data types:

- primitive (bool, int32, int64, uint32, uint64, float, double, string, bytes, etc.);
- complex (oneof, message, enum, array, map);
- logical (date, time, timestamp).

E. Analysis of Data Storage Formats

Within the framework of this study, an analysis of the main characteristics of the previously described formats was carried out. Comparative characteristics of the formats are presented in the Table I.

This section may be divided by subheadings. It should provide a concise and precise description of what data is contained, which format, how to read and interpret the data. E.g., for tabular data a note about what's contained in each column of the data table.

TABLE I. COMPARATIVE CHARACTERISTICS OF THE FORMATS

	avro	**csv**	**json**	**orc**	**parquet**	**pbf**	**sequence**
Platform independence	+	+	+	-	-	+	-
Changeability	-	+	+	-	-	-	-
Complex structures support	+	-	+	+	+	+	-
Compliance with ACID	-	-	-	+	-	-	-
Format type	row-oriented	text	text	column-oriented	column-oriented	row-oriented	row-oriented
Compression support	+	-	-	+	+	+	+
Metadata presence	-	-	-	+	+	-	-
Schema integration	+	-	+	-	+	-	-
Readability	-	+	+	-	-	-	-
Schema evolution	+	-	-	-	+	+	
Usability for streaming systems	+	+	+	-	-	+	-

Table II summarizes the main advantages and disadvantages of each studied data storage format.

In addition, other studies have been explored aimed at choosing a format for various purposes.

In [26], the Apache Parquet and Apache Avro formats are compared in terms of performance, but in this study, there is no justification for choosing this particular alternative. The study proceeds from an experimental assessment of two formats in the absence of a specific task of choosing alternatives. The authors of [27] pursue the goal of finding an alternative for the WARC format when developing web services. Apache Parquet and Apache Avro are also alternatives in this study. The author in [28] offers extensive research on various data storage formats for the analytical task in bioinformatics. This article provides an assessment of all the formats described here. Apache Parquet and ORC were chosen as the most suitable format. The authors also give recommendations on the use of a particular format. Specifically, when running multiple queries, it is recommended to use Apache Parquet, while ORC should not be used [28]. Research [29] is aimed at evaluating the Avro and Parquet formats when performing data queries. The research results are recommendations on the use of each format for specific tasks. [30] is a comprehensive study of the Apache Parquet and ORC formats. Both formats are column-oriented and share similar characteristics and properties. The study carried out a number of experiments focused on the applied properties of each format.

The study [31] developed a methodology for analyzing data storage formats based on comparative analysis, experimental evaluation and a mathematical model for choosing an alternative. For the experimental evaluation, Apache Spark [24] framework was used, which is one of the most popular tools for analyzing data in the Apache Hadoop system.

TABLE II. ADVANTAGES AND DISADVANTAGES OF BIG DATA STORAGE FORMATS

Format	Advantages	Disadvantages
csv	Readable and manually editable; Provides a simple table layout; Can be handled by almost all existing applications; Compact.	Doesn't support complex data structures; Allows to work with flat data; There is no support for column types; There is no standard way to represent binary data; Problems with CSV import (for example, there is no difference between NULL and empty string); Poor support for special characters; Lack of a universal standard.
json	A readable format that allows to work with it without the use of special software; Support for a hierarchical structure, which allows reading a complete set of data; Supported by many programming languages and default data tools. Support for complex types such as arrays and objects. Data schema support.	Format consumes large amount of memory due to repeatable field names; Poor support for special characters; Less compact compared to more binary formats.
avro	High speed of information recording; Fast reading of all fields of the record; JSON data schema provides support for many programming languages and facilitates debugging during development. The availability of extensive capabilities for describing objects and events, including creating your own data schemas, Compatibility with previous versions as data evolves over time.	Reduced speed of information reading, since it is required to read all fields of the record; Lower performance when performing selective queries; Higher consumption of disk space for data storage.
pbf	Compressed data storage format; Self-described data storage format;	Small community, which makes it difficult to develop in case of problems with the format; Storing data type information for each stored value.
parquet	Column-oriented format allows to allows you to significantly speed up the work of the analyst. Efficient storage in terms of space occupied. It provides fast reading experience.	Doesn't support changing data; Does not support schema evolution; Transactions are not supported; No possibility of using in streaming systems; Loss of information due to loss of metadata.
orc	Indexing that speeds up I/O operations; The presence of metadata to facilitate the optimal execution of queries; Transactional support.	Doesn't support schema evolution; Loss of information due to loss of metadata; Transactionality occurs by adding new files.
sequence	Compact format; There are 2 types of file compression - at the record level and at the block level; The ability to parallelize tasks by independently unpacking and using different portions of the same file; Can act as a container for many small files	Lack of multilingual support - this format is specific to the Apache Hadoop ecosystem, which determines the use of only the Java API. Doesn't support complex structures; Doesn't support column types.

A number of disadvantages of the described formats for storing big data have been identified. The main ones are the following:

- Failure to comply with the requirements of the "General Data Protection Regulation" [32]. This regulation defines the human right to "oblivion". In this case, the storage tools must be able to delete the record. In the formats described earlier, only text formats have the ability to delete one record. Other formats require deleting the entire file and writing a new one.
- The need for a transactional data record. Of the above formats, only the ORC format has this property, which requires the addition of delta files to update records.
- Building data storages on dimensional model;
- Requirements for schema enforcement.

The conventionally described data storage formats can be divided into groups containing alternative formats, depending on the tasks assigned to these formats when they are used in big data processing systems.

Accessibility to the data described formats can be divided into the following groups:

- changeable (JSON, CSV);
- unchangeable (Parquet, Avro, ORC, SequenceFile, PBF).

The following groups are distinguished by the internal structure of the file:

- textual (JSON, CSV);
- column-oriented (Parquet, ORC);
- row-oriented (Avro, PBF, SequenceFile).

According to their application in tasks of processing and storing data in big data systems, the formats can be divided into the following groups:

- formats for data streaming (JSON, CSV, Avro, PBF);
- formats for data storing (Parquet, ORC, SequenceFile).

III. Challenges

The limitations of the previously described formats have determined further research and development in the storage of information in data lakes. Among the most well-known emerging big data storage facilities are the following projects: Apache Hudi [19], Delta Lake [20], Apache Iceberg [21].

A. Apache Hudi

Apache Hudi (Hadoop Upserts Deletes Incrementals) [19] is a framework developed for managing big data storage in distributed file systems such as cloud storage, HDFS and other storage combined with Hadoop FileSystem. A distinctive feature of this system is a support of transactional operations (ACID).

Changes to data tables are achieved in two ways: copy on write and merge on read.

- Copy on write: Data is stored in the Parquet file format and each update creates a new version of the file at write time. This storage type is most suitable for read-intensive batch downloads.
- Merge on Read: The data is stored as a combination of the Parquet and Avro file formats. Updates are logged in delta files. This type of storage is better suited for streaming write-intensive workloads.

Data queries are divided into following types:

- Snapshot: The last snapshot of the table as of this commit action. For "Merge on Read" tables, the snapshot query will merge base files and delta files on the fly, resulting in latency.
- Incremental: changes in the table since commit.
- Read-Optimized: The last snapshot of the table at the time of this commit action. For "Merge On Read" tables, read-optimized queries return a view that contains only the data in the underlying files, without merging delta files.

Recent privacy regulations such as the GDPR [32] require companies to be able to perform record-level updates and deletions in order to satisfy the human right to be forgotten. With support for deletes in Hudi datasets, the process of updating or deleting information for a specific user or over a period of time is greatly simplified.

B. Apache Iceberg

Apache Iceberg [20] - is a tabular format for storing tables larger than a petabyte. Iceberg was designed from the ground up for use in the cloud, and the key was to address the various data consistency and performance issues that Hive [33] suffers from when used with data residing in S3 [34]. Iceberg defines how to manage large analytic spreadsheets using immutable file formats such as Parquet, Avro, and ORC.

All information is stored in several different files:

1) Snapshot metadata file contains metadata about the table, such as the table schema, section specification, and the path to the list of manifests.

2) Manifest List contains an entry for each manifest file associated with the snapshot.

3) Manifest file contains a list of paths to related data files.

4) Data file is a physical data file written in formats such as Parquet, ORC, and others.

Apache Iceberg has the following benefits:

1) Lack of "dirty reading" [35]. The use of a snapshot guarantees isolated reading and writing. Readers will always see a consistent version of the data without having to lock the table. Writers work in isolation without affecting the live table.

2) Performance benefits. Instead of listing O(n) partitions in a table during scheduling, Iceberg performs O(1).

3) Data schema evolution. Iceberg ensures that schema changes are independent and have no side effects.

4) Evolution of partitions. Through the implementation of hidden partitioning, Iceberg is also able to propose an evolution in the partitioning specification. This means that the project provides the ability to change the granularity or the column that is split without breaking the table.

5) Support for the query engine. Iceberg is supported by the Apache Spark project, that is, data can be read and written using Spark DataFrames [25], and also read using SparkSQL [25].

C. Delta Lake

Delta Lake is a storage layer for improving the reliability of data lakes [36–38]. Delta Lake can operate on the basis of implemented data lakes using Apache Hadoop [11], Amazon S3 [32] or Azure Data Lake Storage [39].

Delta Lake is characterized by the following properties:

- Support for ACID transactions. Delta Lake Brings ACID Transactions to Data Lakes for Serializability and Highest Isolation.
- Scalable metadata processing. Delta Lake processes metadata using the distributed computing power of Apache Spark.
- Data versioning. Delta Lake provides snapshots of data, allowing you to access and downgrade to earlier versions.
- Open format - all data in Delta Lake is stored in Apache Parquet columnar format, which allows you to efficiently compress and encode data.
- Unified batch and streaming source and consumer in one.
- Evolution of the scheme. Delta Lake allows for table schema changes that can be applied automatically.

IV. Conclusion

Today, data lakes are the most advanced area of big data processing and analysis. In recent years, many platforms have emerged that provide the ability to build data lakes. This study explored the storage tools provided for building data lakes based on the Apache Hadoop platform.

As part of the study, we reviewed the main formats for storing big data in data lakes. Three groups of big data storage formats have been studied: textual, row-oriented, column-oriented. Each group describes the alternatives among the formats. The distinctive characteristics of each format are presented, including features of the internal file structure, supported data types, recommendations for use; highlighted the advantages and disadvantages of each format.

A comparative analysis of the most popular formats for storing big data has been carried out. The studies aimed at identifying the effectiveness of a particular data storage format in relation to the task have been analyzed. The main advantages and disadvantages of the most popular big data storage formats are highlighted.

During the study, the main prerequisites for further research and development of tools for storing big data in the construction of data lakes were studied and determined. One of the main reasons for further research was the requirement for the confidentiality of personal data. This requirement determines the ability to delete a record from the data store. In addition, one of the obvious disadvantages of data storage formats is the lack of transactional operations.

New trends in the field of building data warehouses in the context of data lake architectures are considered; highlighted new requirements for the development of data warehouses. A review of modern tools that meet new requirements is carried out. Their distinctive characteristics and advantages of use have been highlighted.

The analysis of the main properties of data storage formats, their structure and application features, as well as the study of modern trends in the storage and processing of big data in data lakes are necessary for further experimental evaluation of these tools, as well as the development of a methodology for choosing a format that meets system requirements when developing a system for processing big data. These tasks are further objectives of the authors' research.

References

[1] A.F. Alasta, and M.A. Enaba, “Data warehouse on Manpower Employment for Decision Support System,” Int'l Journal of Computing, Communications & Instrumentation Engg., vol. 1, pp. 48-53, 2014.

[2] D. Chong and H. Shi, “Big data analytics: A literature review,” J. Manag. Anal., vol. 2, 175–201, 2015.

[3] V.S. Tomashevskaya, and D.A. Yakovlev, “Research of unstructured data interpretation problems,” Russian Technological Journal, vol. 9, no. 1, pp. 7-17, 2021. https://doi.org/10.32362/2500-316X-2021-9-1-7-17.

[4] F. Cappa, R. Oriani, E. Peruffo, and I. McCarthy, “Big data for creating and capturing value in the digitalized environment: unpacking the effects of volume, variety, and veracity on firm performance,” Journal of Product Innovation Management, vol. 38, no. 1, pp. 49-67, 2021.

[5] C. Yang, Q. Huang, Z. Li, K. Liu, and F. Hu, “Big Data and cloud computing: Innovation opportunities and challenges,” Int. J. Digit. Earth, vol. 10, pp. 13–53, 2017.

[6] D. Ilin, and E. Nikulchev, “Performance Analysis of Software with a Variant NoSQL Data Schemes,” In 2020 13th International Conference" Management of large-scale system development"(MLSD); IEEE, pp. 1-5, 2020. https://doi.org/10.1109/MLSD49919.2020.9247656.

[7] J. Darmont, C. Favre, S. Loudcher, and C. Nous, “Data Lakes for Digital Humanities,” In 2nd International Digital Tools & Uses Congress (DTUC 2020); Hammamet, Tunisia, pp. 38-41, 15-17 October 2020.

[8] P.-N. Sawadogo, E. Scholly, C. Favre, E. Ferey, S. Loudcher, and J. Darmont, “Metadata Systems for Data Lakes: Models and Features,” In 1st International Workshop on BI and Big Data Applications (BBIGAP@ADBIS 2019); Bled, Slovenia, pp. 440–451, 8 September 2019.

[9] P.P. Khine, and Z.S. Wang, “Data Lake: a new ideology in big data era,” ITM Web of Conferences, vol. 17, p. 03025, 2018. https://doi.org/10.1051/itmconf/20181703025.

[10] HDFS. 2020 HDFS Architecture Guide. Available online: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html (accessed on 24 July 2021).

[11] V. Belov, A. Tatarintsev, and E. Nikulchev, “Comparative Characteristics of Big Data Storage Formats,” Journal of Physics: Conference Series, vol. 1727, p. 012005, 2021. https://doi.org/10.1088/1742-6596/1727/1/012005.

[12] A. Agocs, and J.-M. Le Goff, “A web service based on RESTful API and JSON Schema/JSON Meta Schema to construct knowledge graphs,” In 2018 International Conference on Computer, Information and

Telecommunication Systems (CITS); Alsace, Colmar, France, pp. 1-5, 11-13 July 2018. doi: https://doi.org/10.1109/CITS.2018.8440193.

[13] J. Mitlöhner, S. Neumaier, J. Umbrich, and A. Polleres, "Characteristics of Open Data CSV Files," In 2nd International Conference on Open and Big Data (OBD), Vienna, Austria, pp. 72-79, 22-24 August 2016, https://doi.org/10.1109/OBD.2016.18.

[14] Apache. SequenceFile. Available online: https://cwiki.apache.org/confluence/display/HADOOP2/SequenceFile (accessed on 24 July 2021).

[15] Apache. Parquet official documentation 2018. Available online: https://parquet.apache.org/documen-tation/latest/ (accessed on 24 July 2021).

[16] ORC. ORC specification 2020. Available online: https://orc.apache.org/specification/ORCv1/ (accessed on 24 July 2021).

[17] Apache. Avro specification 2012. Available online: http://avro.apache.org/docs/current/spec.html (accessed on 24 July 2021).

[18] Protocol Buffers. Language Guide. Available online: https://developers.google.com/protocol-buffers/docs/overview (accessed on 24 July 2021).

[19] Apache Hudi. Quick-Start Guide. Available online: https://hudi.apache.org/docs/spark_quick-start-guide.html (accessed on 24 July 2021).

[20] Apache. Apache Iceberg. Available online: https://iceberg.apache.org/ (accessed on 24 July 2021).

[21] Databricks. Delta Lake documentation 2020. Available online: https://docs.delta.io/latest/index.html (accessed on 24 July 2021).

[22] W. Ali, M.U. Shafique, M.A. Majeed, and A. Raza, "Comparison between SQL and NoSQL Databases and Their Relationship with Big Data Analytics," Asian Journal of Research in Computer Science, vol. 4, no. 2, pp. 1-10, 2019. http://dx.doi.org/10.9734/AJRCOS/2019/v4i230108.

[23] H.R. Vyawahare, P.P. Karde, and V.M. Thakare, "Brief Review on SQL and NoSQL," International Journal of Trend in Scientific Research and Developmen, vol. 2, no. 1, pp. 968-971, 2017. https://doi.org/10.31142/ijtsrd7105.

[24] A.B.M. Moniruzzaman, and S.A. Hossain, "NoSQL Database: New Era of Databases for Big data Analytics-Classification, Characteristics and Comparison," Int. J. Database Theory Appl., vol. 6, pp. 1–14, 2013.

[25] S. Salloum, R. Dautov, X. Chen, P.X. Peng, and J.Z. Huang, "Big data analytics on Apache Spark," Int. J. Data Sci. Anal., vol. 1, pp. 145–164, 2016.

[26] R.F. Munir, A. Abelló, O. Romero, M. Thiele, and W. Lehner, "A cost-based storage format selector for materialized results in big data frameworks," Distrib Parallel Databases, vol. 38, pp. 335–364, 2020. https://doi.org/10.1007/s10619-019-07271-0.

[27] X. Wang, and Z. Xie, "The Case for Alternative Web Archival Formats to Expedite The Data-To-Insight Cycle," In JCDL '20: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020. E China, pp. 177-186, 1-5 August 2020.

[28] S. Ahmed, M.U. Ali, J. Ferzund, M.A. Sarwar, A. Rehman, and A. Mehmood, "Modern Data Formats for Big Bioinformatics Data Analytics," International Journal of Advanced Computer Science and Applications, vol. 8, no. 4, pp. 366-377, 2017. https://dx.doi.org/10.14569/IJACSA.2017.080450.

[29] D. Plase, L. Niedrite, and R. A. Taranovs, "Comparison of HDFS Compact Data Formats: Avro Versus Parquet," Lietuvos ateitis, vol. 9, no. 3, pp. 267-276, 2017. https://doi.org/10.3846/mla.2017.1033.

[30] T. Ivanov, and M. Pergolesi, "The impact of columnar file formats on SQL-on-hadoop engine performance: A study on ORC and Parquet," Concurrency and Computation: Practice and Experience, vol. 32, no. 5, p. e5523, 2020.

[31] V. Belov, A. Tatarintsev, and E. Nikulchev, "Choosing a Data Storage Format in the Apache Hadoop System Based on Experimental Evaluation Using Apache Spark," Symmetry, vol. 13, p. 195, 2021. https://doi.org/10.1002/cpe.5523.

[32] Regulation (EU). EU General Data Protection Regulation (GDPR). Available online: https://gdpr-info.eu/ (accessed on 24 July 2021).

[33] Hive. 2020 Apache Hive Specification. Available online: https://cwiki.apache.org/confluence/display/HIVE (accessed on 24 July 2021).

[34] Amazon. Amazon Simple Storage Service Documentation. Available online: https://docs.aws.amazon.com/s3/index.html (accessed on 24 July 2021).

[35] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil, "A Critique of ANSI SQL Isolation Levels," In Proc. ACM SIGMOD 95, San Jose CA, USA, pp. 1-10, 22-25 May 1995.

[36] P. Sawadogo, and J. Darmont, "On data lake architectures and metadata management," Journal of Intelligent Information Systems, vol. 56, no. 1, pp. 97-120, 2021.

[37] S. Vats, and B. B. Sagar, "Data Lake: A plausible Big Data science for business intelligence," In Communication and Computing Systems; CRC Press, pp. 442-448, 2019.

[38] C. Diamantini, P. Lo Giudice, D. Potena, E. Storti, and D. Ursino, "An approach to extracting topic-guided views from the sources of a data lake," Information Systems Frontiers, vol. 23, pp. 243-262, 2021.

[39] B. Shiyal, "Introduction to Azure Synapse Analytics," In Beginning Azure Synapse Analytics; Apress, Berkeley, CA., pp. 49-68, 2021.