IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 4, 2021, accepted September 2, 2021, date of publication September 14, 2021, date of current version September 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3112880

A Systematic Review of Data Models for the Big Data Problem

FAEZEH MOSTAJABI[1], ALI ASGHAR SAFAEI[2], AND AMIR SAHAFI[3]

[1]Department of Computer Engineering, Islamic Azad University, Qeshm Branch, Qeshm 7953163135, Iran
[2]Department of Medical Informatics, Faculty of Medical Sciences, Tarbiat Modares University, Tehran 14115-111, Iran
[3]Department of Computer Engineering, Islamic Azad University, South Tehran Branch, Tehran 1584743311, Iran

Corresponding author: Ali Asghar Safaei (aa.safaei@modares.ac.ir)

ABSTRACT Nowadays, data are generated in a continuous streaming manner as the inputs of various applications. The sources of such generated data can be wired or wireless sensor networks commonly used in various fields of geographical, traffic, Internet of Things (IoT), financial tickers, Web2 and Web3, e-commerce, social networks, and online communities. The high volume, high variety, and high velocity of data have recently posed the challenge of 3Vs to this field, also known as the Big Data Problem. The 3Vs dimensions of complexities for the big data entails high-speed storage, scalability of database systems, suitable data models, real-time responsiveness and so on. Data model, as the representation schema of data is an essential issue since many others (e.g., DBMS systems' design, DB languages, etc.) rely on. So, the study of data models is a key and fundamental aspect in structuring, organizing, storing, and manipulating big data. It is also the essence in various areas of cloud migration, web-scale, and so forth. In this paper, we have systematically reviewed different types of data models, the rationale behind them, their applications and support capabilities, and the technologies to switch from one model to another. To address the user needs in various fields, a systematic review method is adopted to classify and present different types and characteristics of data models.

INDEX TERMS Big data, database, data model, NoSQL, schema.

I. INTRODUCTION

In previous decades, relational database management system (RDBMS) [1] was considered as the optimal solution for many data consistency and management services [2]. The relational database was rapidly developed as a popular choice for most data-driven companies. However, the traditional relational data model [3] used Cartesian products. This yielded a mound of worthless results when ran on complex data, thereby reducing the efficiency of the queries. With the increase in the volume and the change in the variety of data over time, the unscalability problem arose in the relational data models. The other deficiencies of these models were the lack of open-source databases and no support for all data structures. Since then, data models have been proposed to remedy such deficiencies. The XML model was developed to manage semi-structured data [5], [6]. The object-relational [7] was another model developed, still facing the scalability problem. Additionally, the explosive growth of data traffic by social networks, e-commerce as well as multimedia data streams at the sensors [8] was an emerging challenge to IT companies and other data resources. According to [9], this growth rate doubles every two years and has increased tenfold within the years 2013 to 2020 (from 4.4 to 44 ZB). There was an urgent need for horizontal scalability and greater flexibility of databases due to the exponential growth of data volume, the change of data from structured to semi-structured [10] and non-structured, and the challenge of their storage. Hence, some practical solutions have been presented to satisfy such challenges to big data. The need to substitute new databases for old ones was felt after the emergence of big data issues, 3Vs [3], [12], IoT [13]–[15] in computer and data sciences. As a possible remedy, NoSQL came into view. It has emerged in a general term with unstable and flexible conceptual data models [16] and varied strategies for the databases. Among features of NoSQL databases are high scalability, availability without the need for ACID feature support [9], open-source possibility of the presented models [17], and capability of dynamic data modeling. Given the variety and various applications of data

The associate editor coordinating the review of this manuscript and approving it for publication was Genoveffa Tortora.

models in NoSQL [18], information on data modeling, data model types, and data retrieval from schema-less databases is a requirement for engineers, scientists, and big data analysts in making the best decisions. Also, knowledge about the rationale for using different types of the data models, switching techniques from one data model to another, migration strategies from source to target database at the conceptual level [19] is among other requirements in the field that form the topic of research in this article.

II. BACKGROUND

This section provides key concepts and terminologies associated with big data, data model, and schema.

A. BIG DATA CONCEPTS

Big data is an abstract concept with distinctive characteristics that distinguish it from other types of data [12], including massive data and very big data. Big data deals with sets of information, not capable of being received, acquired, managed, and processed at an acceptable time through information technology and traditional software and hardware tools [12], [20]. In the digital world, data is generated from different sources where the rapid growth of data results from the fast transfer of digital technologies. Big data can be classified into two general types, the information produced by humans and the information obtained from the physical world and utilities such as sensors, GPS devices, and CCTV cameras. In addition to the internet, these types of information can be grounds for the exponential growth of big data over the years. Big data is facing the problem of 3Vs, including processing large streams of high-velocity data in the shortest possible time. Meanwhile, software technologies have been making attempts to overcome this problem [11], [14]. Unfortunately, datasets with these features cannot be fully managed by the current systems, tools, and technologies. Instead of being exploited or valued, a mass of data is given up without reaching its ultimate goal. This will increase the demand for technologies, systems, tools, methods, models, structures, and concepts to receive, manage, process, and analyze big data, extract the embedded knowledge and turn it into value [21]. According to [22], big data aimes to use different techniques for data analysis. However, there is a growing gap between the possibility of data collection and the power of data processing during the life cycle of data [23]. Therefore, data representation aims at managing data heterogeneity to make it meaningful for data analysts and scientists (users or applications) [24]. Improper data representation may reduce the value of original data and thus the efficiency of data retrieval and analysis [21]. Instead, data modeling provides a representation of big data to be managed. In this paper, we examine different types of data models focusing on different representations of big data.

B. DATA MODEL CONCEPTS

The data model represents the organization and the structure of data elements. It shows how data elements relate to one another [2], [3], [9] or determine [25] how data can be stored, organized, and manipulated. In recent years, interest has been aroused in developing of various tools and technologies for exploitation and adding value to the mass of data to tackle the 3Vs (volume, velocity, and variety) of big data. In this regard, significant challenges are related to the storage, analysis, and representation of data in large volumes. In fact, data modeling with the help of data representation enables data management. In [2], the presented model is a core model for storing, analyzing, and processing data in large systems. In [26] and [27], the data model is defined as a set of conceptual tools used to model the original set of data and the relationships between data items. Based on [28], a data model is aimed to make data meaningful and data communication possible for information needs. In [4], the data model includes three concepts:

(I) Data model is a set of data structures that mainly describes data types, properties, and relationships. The data structure is the basic part on which operations and constraints are structured.
(II) Data model is a set of operators and inference rules that mainly describe types and methods of operation in a particular data structure.
(III) Data model is a set of comprehensive constraints that can be used to describe syntax, dependencies, and constraints of data to ensure its accuracy, validity, and compatibility.

The first concept of data structure is sometimes considered by the data model regardless of the two others [29]. Based on [30] and [25], a data model design consists of three phases, namely a conceptual data model, logical data model, and physical data model:

- Conceptual level deals with communicating ideas to the users about the program's domain specifying entities and relationships in the scope of the model.
- The logical level is concerned with communicating ideas to the designers about the domain of applications specifying more details than the previous level. This level is also referred to as the schema.
- Physical level refers to the physical storage and practical implementation in a database management system. Therefore, the main step in designing a database is the logical level of data model design at which the database is specified from the perspectives of both users and designers. Based on [31], the conceptual data model is composed of three layers with the corresponding architecture of Attribute, Collection, and Family.

To design a database at the abstract below the modeling level entails structuring data in an abstract environment, fulfilled by the data model. The data model is important in understanding the type of data structure and the form of data storage. There is a fixed concept of the data model in relational databases. In contrast, different conceptual data models are developed in NoSQL databases [32]–[34] with flexible and

dynamic schema and strategies for storing data structures. These databases also face the following challenges:

- Current NoSQL databases make no distinction between data models at the logical and physical levels.
- Different types of data structures are represented not in the same way, although they must have been, at the logical level.
- There is no common data model for different types of NoSQL databases at the conceptual level.

C. SCHEMA CONCEPTS

In general, databases can be categorized into two groups of schema-based and schema-less. The former utilizes a schema to describe the structure of the database, while there is no need to predefine a specific data structure for the latter, which leads to the higher flexibility of the database. The data is stored regarding a predefined structure of schema, which can be defined either in accordance with some rules or in the mind of developers. Therefore, a non-defined data structure does not necessarily ascertain a schema-less database; its existence is relatively efficient for understanding data structure and effective data management. Sometimes, schema is implicitly embedded in data or program code [35]. In NoSQL data models, a schema structure [36] is dynamically defined by the data stored and the entities of that class [37]. The major challenge of schema-less databases is lack of a definition to check for data compatibility [38]. The existence of a schema allows static data analysis and manipulation compatible with the schema itself [35]. Given the importance of schema as a pre-stage in data model design and its relation to the concept of data model, we are intended to review the related articles. In this article, attempts have been made to review the methods, solutions, and reasons for presenting different types of data models in the field.

The rest of the article is structured as follows. In section 3, methods are discussed for selecting papers to be reviewed. Then, a summary of the selected papers is given in section 4. Section 5 is devoted to the comparison of results and their classification. Finally, the main conclusion of the results is drawn in section 6.

III. RESEARCH METHODOLOGY

In this section, according to the Systematic Literature Review (SLR) method [39], [40], we classified the data models reviewed from 2014 to 2020. First, the research needs were identified in the ''Identification of Needs'' section. Next, the related articles were collected, and then, refined based on the identified needs. Last, the key questions to extract from the articles were identified in the ''Determining Research Questions'' section.

A. IDENTIFICATION OF NEEDS

This research aimed to answer the following analytical questions on the types of data models proposed in recent years:

- Why are data models of importance?
- How are data models classified, and the rationale behind this classification?
- How are different types of data models compared and the application of each type?
- Which kind of properties are required for data models, and what are future research directions to be followed?

B. SELECTION OF RESOURCES

In the first step, articles were searched from leading scientific publishers, including IEEE, Springer, Elsevier, Taylor & Francis, Wiley, ACM, and Sage pub, corresponding to the topic of data modeling.

Table 1 shows the number of articles collected from each publisher in the first step of the search. Regarding the number of published articles, only journal and conference papers indexed in the WoS and ISI were considered to be evaluated in the second step.

TABLE 1. Number of publishers on the searched topic.

Publishers	Result
SPRINGER	35
IEEE	10
ACM	36
T &F	5
WILEY	6
SAGE	1
SCINCEDIRECT	37

The next step was to refine the selected articles. Figure 1 shows the refinement and selection process of related articles to the topic under investigation, summarized in a flowchart. The deletion index for removing articles is as follows:

- Articles not written in English.
- Articles not indexed in ISI.
- Articles with data models irrelevant to the study database design or data storage.

As articles on data modeling encompass a wide range of topics and applications, the search was done for the keywords relevant to the concept of data model and database. The results are shown in terms of publication index in Figure 2.

Table 2 represents the number of initial and selected articles for review. In the refinement stage, we excluded short articles, book chapters, low-quality papers containing no technical or scientific information about the discussion, unrelated articles to our data model classification, and those exclusively related to data model products. The selection index for the finalized studies was based on:

- Articles published between 2014 and 2020.
- Articles related to the data model in the databases.
- Articles related to the data model applications.

Figure 3 illustrates categories of finalized articles according to the publication year and Table 3 shows the classification of articles according to the journal and the year.

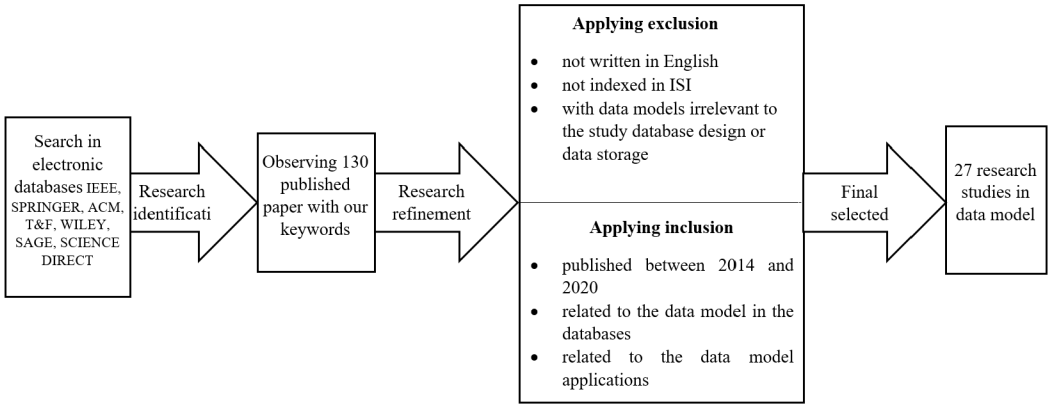

FIGURE 1. Evaluation and final index of article selection.

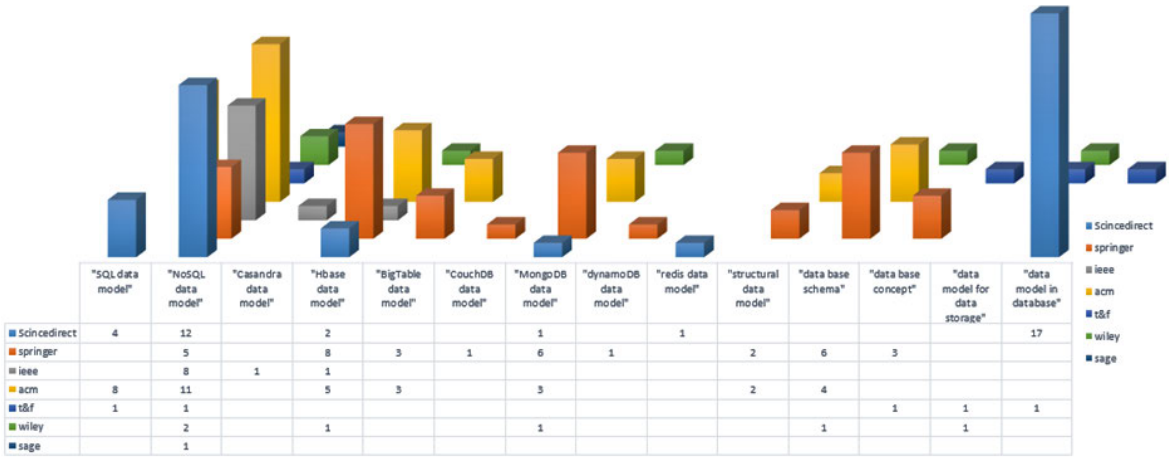

FIGURE 2. Distribution of keywords searched in each journal.

TABLE 2. Distribution of remained articles after refinement stage.

Text Searched	Related	Unrelated	Repeated	Product	Sum
SPRINGER	4	9	2	20	35
IEEE	6	3	0	1	10
ACM	6	14	2	14	36
T &F	2	3	0	0	5
WILEY	1	2	0	3	6
SAGE	0	1	0	0	1
SCINCEDIRECT	8	22	3	4	37

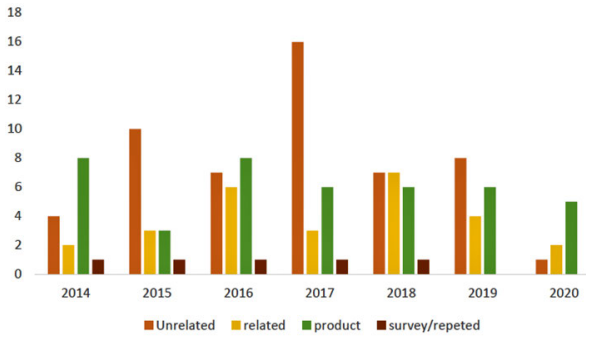

FIGURE 3. Illustration of final articles by the publication year.

C. DETERMINING RESEARCH QUESTIONS

In each article, there was an assumption based on introducing a new data model, applying an existing data model for a specific purpose, or comparing several data models. We were striving for responding to the following questions. The study aimed at making a comparison between the extracted outcome of each article and their total output to conclude a systematic review of data models, their basic features and applications. The questions posed to be answered were as follows:

- RQ1: What is the new data model proposed or the data model applied in the article? (Including the title, the used data model algorithm, the reason for the data model use, and the way it works)
- RQ2: What is the reason behind if there is any comparison drawn between the proposed data model and other data models?
- RQ3: What is the application of the presented data model?

TABLE 3. Selected papers for review distinguished by categories and publishers.

Publisher	Paper's Title	Year	Author
ACM	GraphSE2: An encrypted graph database for privacy-preserving social search	2019	S. Lai, X. Yuan, S. F. Sun, J. K. Liu, Y. Liu, and D. Liu
	SQLtoKeyNoSQL: A Layer for Relational to Key-based NoSQL Database Mapping	2015	G. A. Schreiner, D. Duarte, and R. Dos Santos Mello
	Data Storage Management in Cloud Environments: Taxonomy, Survey, and Future Directions	2017	Y. Mansouri, A. N. Toosi, and R. Buyya
	Logical Design of Graph Databases from an Entity-Relationship Conceptual Model	2018	V. M. De Sousa and L. Mariano
	GSQL: Fast query processing via graph exploration	2016	H. Ma, B. Shao, Y. Xiao, L. J. Chen, and H. Wang
	Couchbase analytics: NoETL for scalable NoSQL data analysis	2019	M. Al Hubail et al
IEEE	Traceability Architecture: Extending EPCIS to Enhance Track and Trace with NoSQL Data Model	2016	Y. Kidane and D. Kim
	Big Data-Oriented Open Scalable Relational Data Model	2014	Z. Zheng, Z. Du, L. Li, and Y. Guo
	Logical level design of NoSQL databases	2016	S. Banerjee and A. Sarkar
	A NoSQL Data Model For Scalable Big Data Workflow Execution	2016	A. Mohan, M. Ebrahimi, S. Lu, and A. Kotov
	A Model-Driven Approach to Generate Schemas for Object-Document Mappers	2019	A. H. Chillon, D. S. Ruiz, J. G. Molina, and S. F. Morales
	Comparison of MongoDB and Cassandra Databases for supporting Open-Source Platforms tailored to Spectrum Monitoring as-a-Service	2018	G. Baruffa, M. Femminella, M. Pergolesi, and G. Reali
Science Direct	A Classification of NoSQL Data Stores Based on Key Design Characteristics	2016	A. Makris, K. Tserpes, V. Andronikou, and D. Anagnostopoulos
	Mortadelo: Automatic generation of NoSQL stores from platform-independent data models	2019	A. de la Vega, D. García-Saiz, C. Blanco, M. Zorrilla, and P. Sánchez
	Towards an operational database for real-time environmental monitoring and early warning systems	2017	B. Balis, M. Bubak, D. Harezlak, P. Nowakowski, M. Pawlik, and B. Wilk
	Critical Analysis of Data Storage Approaches	2020	S. Sachdeva, P. Sharma, R. Jain, and S. Parashar
	Expanded Cloud Plumes Hiding Big Data Ecosystem	2016	S. Sharma
	Evaluating the effects of access control policies within NoSQL systems	2020	P. Colombo and E. Ferrari
	PanDA Workload Management System meta - data segmentation	2015	M. Golosova, M. Grigorieva, A. Klimentov, and E. Ryabinkin
Springer	A Clustering Algorithm for Planning the Integration Process of a Large Number of Conceptual Schemas	2015	C. Batini, P. Bonizzoni, M. Comerio, R. Dondi, Y. Pirola, and F. Salandra
	A graph-based meta-model for heterogeneous data management	2018	E. Damiani, B. Oliboni, E. Quintarelli, and L. Tanca
	Data Modeling for Big Data	2017	P. S. Engineer, A. Wang, and S. Engineering
	Automatic schema suggestion model for NoSQL document-stores databases	2018	A. A. Imam, S. Basri, R. Ahmad, J. Watada, and M. T. González-Aparicio
Taylor & Francis	Big spatial vector data management: a review	2018	X. Yao and G. Li
	Migration from an SQL to a hybrid SQL/NoSQL data model	2019	M. V. Sokolova, F. J. Gómez, and L. N. Borisoglebskaya
WILEY	Tool for materializing OWL ontologies in a column-oriented database	2017	L. Reyes-Álvarez, M. del M. Roldán-García, and J. F. Aldana-Montes
	A generic methodology for geo-related data semantic annotation	2017	K. Razzaq et al

- RQ4: What is the designed data model intended to achieve?
- RQ5: What are the critical features of the studied data model and what is the comparative result?
- RQ6: Which type of supported data (including structured, semi-structured, unstructured, or mixed) is used in the data model?
- RQ7: Which type of schema in the study is referred to?

IV. REVIEW OF THE SELECTED ARTICLES

The remained articles after the refinement stage were individually studied and summarized in this section. In the process of reviewing, it was attempted to find answers to the questions listed in section III considering the nature and the content of research.

In [4], Open Scalable Relational Data Model (OSRDM) was introduced, which supports data variety with full horizontal scalability compared to the relational data model. This data model exhibits similar properties of performance and scalability to the NoSQL data model. In this model, a public key was used to identify specific data items similar to the key-value data models. In other words, this model developed on a key-value model with split keys and public hashes, providing full horizontal scalability. Feature keys are also used for flexible storage and data aggregation. The data structure and the communication of operators can be defined by the users themselves. Compared to the relational, column-based, and document-based data models, various data structures are supported by the model and various data types are definable by the users. Representing data relationships as entities and separating entities in the data structure provide this model with full horizontal scalability.

In the design of NoSQL database [19], a model-based process, called Mortadelo, was planned to automate database implementation when switching between NoSQL models. A conceptual data model is required for the database to determine the entities and the relationships between them. The way to retrieve and update entities at runtime is also determined at this level which forms the first step of data model design. From a technical point of view, processing separate but linked metamodels can cause random system complexity. A metamodel was developed to model a NoSQL database independent of the platform, to cope with such complexity in Mortadelo, called Generic Data Management (GDM). It can be used as input for different instances of NoSQL in which structured data and data access patterns are aggregated in a metadata scheme. Two logical metadata instances (column family data model and document data model) were also used as input. The proposed model was evaluated in terms of expressiveness, efficiency, required resources, and adaptability to change. This model claimed to be high-efficient in terms of system performance. In this approach, the query integration mechanism was used and compared with the approach in [41]. Except the two-query integration, the answers to all queries were exactly the same with the shorter response time for the model in [41]. In comparison with MONGODB and Cassandra as the two top NoSQL databases, a case study was carried out to test the performance of this model.

In [16], a data model was designed for MongoDB and Cassandra to compare their performance on storing radio spectrum data obtained from sensors and Binary Large Objects. The purpose was to select the optimal text database of the game for the time-series data as having timestamps. Each Avro record from Kafka was converted into a document in MongoDB. The bucketing pattern was used for the minimum number of documents. Avro fields were mapped from Kafka record in Cassandra data model to columns in a database table, and bucketing was used for size limitation of partitions.

Another mechanism was presented for converting the NoSQL conceptual data model to the schema JSON logical data model [30]. The steps of converting data and their relationships were described from the conceptual to the logical level. Similar to the NoSQL databases, the proposed model was claimed to be simpler for converting data at the physical level. This model aimed to make schema more flexible, represent both structured and unstructured data, and evolve heterogeneity of NoSQL databases. Its early formation at the conceptual level occurred on the base of Collection, Family, and Attribute layers with their structural types. Its performance was then evaluated through a case study.

In a designed architecture for scalable and efficient Electronic Product Code Information System (EPCIS) tracking [42], a column-based data model was adopted to retrieve information in NoSQL. It was shown that the column-based data models were the most suited to data track and trace applications. The data from the EPCIS system were stored in Cassandra clusters using distributed hashes on the column-based data model. The syntax comparison between SQL and Cassandra showed that SQL required more than one index to access the recorded events available for a particular tag, analogous with the document-based data model. This is while only one index was used for Cassandra database. In the column-based model, there was no need for encryption and decryption of information based on the storage path of stored data in the linked columns. The scalability of the column-based data model was finally investigated from an operational point of view. It was found that the column-based model had better performance than other data models on tracking specific data for path-based queries.

In [23], a NoSQL data model was introduced to support the high-performance of MapReduce, providing two algorithms for automatic partitioning and parallelization of data. The purpose was to present a model of NoSQL data collection not only scalable but also suitable for workflows of big data-driven applications. The first algorithm splits the workflow into multiple clusters, each runs in a virtual machine. The second algorithm executes data partitioning, virtual machine preparation, and deprivation reduction automatically. It also develops scalability of workflows in the MapReduce style and a new DataView for workflows of big data systems. The suggested model was used to analyze collected data from vehicles and provide insight into associated risks with the driving style.

In [43], the NoSQL data models were compared with the relational data model and categorized according to their key features. These features include horizontal scalability, partitioning ability of large datasets in distributed sources, data replicability for fault tolerance, and mechanism facility for data item compatibility. Other properties of data placement, partition, replication, consistency, fault tolerance, as well

as lack of global query language are fully explored for all NoSQL data models.

NoSQL databases, in particular the column-store Cassandra database were used for storing linked data and OWL ontology [44]. Given that linked data storage necessitate scalable databases and distributed parallel algorithms, a schema was designed to store the ontology classification in the Cassandra database and help OWL scalability. Combining MapReduce algorithms on Cassandra, an optimal model was implemented for the storage of OWL ontology classification. The RFD data model based on directional graphs was deployed to distribute and link data structures on the Web. This model facilitated the link between Cassandra database and the OWL file.

In [45], an encrypted graph database model was introduced for online social networking services, called GraphSE, to provide privacy in social networks based on queries. The needs for storage and computation of big data in social networks, control over data confidentiality in the cloud environment, and prevention of data leakage by cryptography were justifications for this model. The graph data models have practical application in social networks concerning their rich and complex search demand. This poses the challenges of scalability which limit query performance by using the algorithms such as neighbor search or k-nearest neighbor. The distributed and encrypted graph was introduced to overcome this limitation. The data model efficiency, operational power, scalability, memory consumption, and search time latency were studied on YouTube data.

In [46], a graph-based metamodel, GSMM, was proposed on a straight line and tagged graph for structured and unstructured data accompanied by a general semi-structured language, GSL. The goal was to achieve flexible feature entries within a single framework. It was asserted that data models of relational, OEM, DOEM, XML, TGM, XML, and RDF (for a triple-based database) can be extracted from this metamodel. This metamodel could also be used to represent Neo4j and BigTable data models. The cardinality of each data model was then expressed as a rooted GSMM graph. By comparing the constraints and limitations of the two data models, this paper achieved a qualitative, quantitative, and flexible evaluation of GSL to represent the limitations of the data model.

Providing a focal data model called SQLtoKeyNoSql [47], a relational layer was developed on NoSQL key-value databases such as key-value, column, and document. In fact, a mapping of the relational data model was defined in NoSQL in such a way that the user could manage data making use of SQL DDL and DML orders. The purpose was to facilitate migration from the relational to NoSQL data models while avoiding the risk of poor performance. In this model, a relational schema was mapped into a three-level focal model. This focal schema of simple hierarchical structure and key-value representation was next mapped into NoSQL data models. In this architecture, there were defined modules for query execution and presentation. The model data scalability and performance were investigated in this research. Although experiments showed satisfactory results, this model was not cost-effective for key-value databases in terms of time and overhead costs. The results of this model were compared with those of MongoDB, Cassandra, Redis, HBase, and Key-Oriented data models, excluding semi-structured XML and DOM. Neither specific data nor special data model applications were observed.

First, data models of four databases, MongoDB, Redis, PostgreSQL, and InfluxDB, were modified to represent time series data in [48]. Simulating reading and writing operators, a comparison was then drawn on their writing performance, defined query performance, memory usage, and operational power. Finally, Redis and InfluxDB data models showed the best operational power. The InfluxDB data model showed the best results in terms of memory usage of the tested data volume, query performance, and average execution time.

To transform a logical model into a conceptual model, a process was designed for converting relation-entity data model to a graph-based data model in NoSQL [38]. The process was intended to increase flexibility of the conceptual models. Another NoSQL abstract model (NoAM) was presented based on the common features to NoSQL database models. Based on the mapping rules from the conceptual to the logical level, a simplified ER model known as Extended Bity Entity-Relationship (EB-ER) was used at the conceptual level. The XML model was used at the logical level in addition to a graph-based data model. Also, several characteristics have been introduced for flexibility in the conceptual data schema. Ultimately, an algorithm was developed for the conceptual model transformation on a set of integrated constraints by which EB-ER that was automatically mapped into a graph-based data model at the logical level. To implement the algorithm, Neo4j database and Cypher query language were utilized.

Making use of clouds, NoSQL big data models were investigated in [49]. The purpose of this paper was to reveal the role of clouds in hosting big data for meeting big data challenges to the NoSQL data models. Given the concept and objectives of DBaas, the cloud environment's ability to manage big data systems and the simplified access to complex databases were of concern. The XCLOUDX data model has been expanded to represent large data models. With the aim of cloud utilization, the study investigated different types of XCLOUDX data model classification. JSON storage was employed by the built Cloudant from CouchDB and the used CloudKit as a schema-less technology. The Cloud Datastore as a schema-less and non-relational NoSQL database in the cloud environment, non-cloudy LightCloud, as well as Cloudera were studied. Numerous big data models in the X…X cloud class, including the cloud-based columnar database of 1010data System, Azure DocumentDB, Amazon DynamoDB, and Datameer were also investigated. In addition to basic features of the cloud itself, key features of the cloud environment, including scalability, performance, and configuration were finally introduced for investigation on big data storage.

In [9], a data model was examined as one of the key aspects of cloud-based data storage. The hierarchical data model called NewSQL was suggested combining goals of RDB and NoSQL deliver. It aimed to cover horizontal scalability similar to NoSQL and maintain ACID properties similar to the relational model. NoSQL was proposed as a schema-less data model with respect to the stored data volume in the cloud environment, data growth rate, and big data challenges. Among NoSQL data models, BigTable, Dynamo PNUTS, and Cassandra were suggested for OLTP and OLAP of big data owing to their velocity coverage. The low latency and the availability of reading and writing are requirements for both systems. One of their limitations in the cloud environment is the poor processing of queries. As a solution, changing their data models is suggested. The hierarchical data model is efficient in the cloud environment for the analysis of program entities and identification of relations among them as well as placement of related data close to the servers/DC.

How Big Spatial Vector Data (BSVD) is stored with their associated data models was discussed in part of [50]. The column-based and key-value data models in NoSQL were considered as tools that able to store and integrate spatial data from multiple resources effectively. The relational data models of Oracle and PostgreSQL were also discussed. Moreover, the data model and the vector data model within the database were considered necessary for a new spatial data model. Having reviewed index and query, this paper found the features of data partition and scalability required for the data storage.

In [51], a hybrid data model was developed from SQL and NoSQL utilizing ontology. Combining relational and non-relational databases to create an ontology, this model provides migration from SQL to NoSQL. The system's overall structure is composed of multiple nodes linking together in a single semantic space. Three logical levels constitute the architecture of this system. First, node client, which provides clients with interfaces depending on their permission. The second level, graphical user interface, connects the node client to the server. Third is the node server dedicated to data storage management. This model aimed at improving access to data, integrating heterogeneous data, presenting a hybrid version of DBMS by mixing local paths of prior DBMS versions. MySQL and RDF Store were used for implementation of the model. The flexibility, mobility, and efficiency were the considered features in this model.

In [8], Another model was designed for transferring and mapping spatial data from RDB schema to RDF and XML schemas. This model thoroughly represents data and metadata structures and data constraints in detail. Indirect mapping to RDF was done without the use of XML, and schema from RDB was transformed to RDF with the help of XML. Employing big data techniques, several models were also proposed for direct mapping while making comparisons among the existing data models.

To test access control policies in the schema-less data model, an integrated data model was presented to evaluate the accessibility of schema-less resources [52]. This approach was presented in three stages of extracting the integrated resource model, defining access control policies, and mapping to the main data model in a two-way process. MongoDB was applied to the model evaluation due to the support of NoSQL systems for MapReduce. Local NoSQL data model and unifying model data resource were used in the presented two-way mapping method. To show data sources in various NoSQL data models, an integrated data model is provided and then utilized to test accessibility as a key feature in data management systems. The flexibility of this model makes it applicable to complex systems with several heterogeneous NoSQL databases. This model can be applied to key-value, document-based, and column-based models in NoSQL. It can also be applied to different data models since integration of data sources was independent of data model. The model was tested on six different datasets, increasing the average runtime by 2.5 In [53], the Couchbase Server document-based data management system was analyzed. Its data model was described and its architecture was investigated in contrast to other traditional NoSQL and HTAP models. Furthermore, a comparison was drawn between NoSQL and SQL. The data service of this approach uses JSON format to display data structures in a flexible and self-describing manner. Similar to the relational model, this model applied the main key to access documents. On the contrary, the documents in this model are stored in the server based on the defined logical structure in the program code. The distinguishing feature of this model is its agile management in adding new objects and properties to new JSON data with a new program code and no schema change. The distinction between the relational model and this model is horizontal scalability, independence scaling, and clustering without shared storage.

In [35], the schema of the MongoDB data model was studied for schema inference from the document-oriented data models in NoSQL, and schemas were inferred for the ODM, Morphia, and Mongoose. The purpose of this article was to use a mapper rather than API and to facilitate reading data during migration. In this paper, a metamodel was developed as an object-oriented conceptual model to describe relationships and concepts. In addition, a strategy was adopted to automate object-document mapping (ODM) when the database is already available. To apply the semi-structured data, JSON text format was used. In the end, the studied schema is suggested for the Ecore models (a language for changing the model) to be used in NoSQL database applications.

In [54] refers to the step before data model design, conceptual schema, which is reviewed here as a bright idea in data model design. This paper focused on a unique global schema aggregated from the database including many schemas whose complex management is done using the clustering algorithm. This procedure is done by repeating two options: i) automatically clustering schemas through discovering cluster matches, and ii) manually aggregating schemas cluster by cluster. The ER data model was used to represent schema,

generalized to the generalization relationship model in which clusters were represented as a graph. The analysis time, development time, rest time, and total aggregation time are among the studied properties, whereas flexibility, scalability, and efficiency are not among key properties considered in the design of data model in this study.

In [55] A schema was suggested for the document-based database in three areas of NoSQL, namely architecture, algorithm, and association rules. The purpose of this model was to maintain a balance of compatibility, availability, and scalability features in the data model. The architecture consisted of four phases, the representation layer on the client side, the selection of selectable and non-selectable parameters, the semantic mapping of entities by the model itself, and the production of schema as the final phase. The proposed model was mathematically formulated and run in Java to be evaluated in MongoDB and Couchbase databases. This model was claimed to be superior to the other two models in terms of writing time and security.

In [56], schema-based Postgres database was compared with schema-less MongoDB and Cassandra databases. Referring to the features of a schema-less database, it is concluded that schema is important even in NoSQL databases. The three databases were examined in terms of schema. The results showed that as a schema-based database, Postgres stores data as a single JSON object. This was, however, different from JSON storage in MongoDB as a document-based which stores dynamic data in datasets. In the column-based Cassandra, each attribute was stored as a column family in the database schema which can be readily changed by adding a new column family. Therefore, non-schema databases perform faster on schema evolution. It was finally shown that in spite of being more flexible, schema-less databases increase sparseness and query execution time, having been checked by changing schema and datasets.

In [57], different studies were compared on combining graph database and RDF models to find out graph problems using relational database capabilities. The article stressed the query language called G-SQL, obtained from the combination of a relational database with a graph processing engine for graph navigation. Regarding local cache problems to navigate large graphs, one layer of the graph was simulated on the relational database using the power of graph navigation in combination with mature technology of relational database. This yielded accelerated processing of combined queries and a schema for synchronized SQL database and graph updates.

In [25], a schema was developed for document-based data models such as MongoDB to support and store temporal data from sensors that continuously transmit data. With the aim of scalability of data integration in addition to flexibility evolution, this schema examined the challenges of temporary data modeling. Also, an algorithm was developed for integrating JSON data as hierarchical documents. In the proposed schema, the SchemaMinute (for minutes, including nested documents (for second), was used to store per-second data from sensors. The second sub-document schema introduced SchemaSensor (for hours) and nested documents (for minute). The schema design was then compared with the document per event approach. The two proposals resulted in lower latency in data reading and more efficient data retrieval. Instead of complete document writes for updates, the query was performed on a smaller part of nested document.

In [58], a Cassandra-based model was suggested for metadata storage. The metadata was divided into the operational and the archive categories, part of which was sent to SQL and part needed for data analysis was sent to this new data model in NoSQL. The purpose of this data model was to improve performance and scalability in analysis and distributed systems.

V. RESULTS AND COMPARISON

A. DATABASE CATEGORIES

In general, data models can be divided into different classifications based on the type of database. According to the concepts and challenges of big data, there are two main categories of data models before and data models after the emergence of NoSQL databases. With the emergence of big data issues, the challenge of 3Vs, IoT, current affairs in computer science and data science, required types of NoSQL databases received priority over previous types of data models including, relational, object-relational, TimeSeries, and etc. However, some types of databases before the emergence of NoSQL are still being used for some applications. Regarding the applications and frequencies of reviewed data models, databases were classified under three general categories of time series, relational, and NoSQL. They were further classified into five main classes of NoSQL database, including key-value, document-based, column-based, graph-based, and Native XML. NoSQL systems are a very heterogeneous group of database systems. It can be said that there are a total of sixteen different data models to date [64], of which eight are subsets of NoSQL [64], although any attempt to classify one or more systems fails. However, according to the data model examined in this paper, we have considered this category. These classifications have their own products according to their characteristics and applications, separately shown in Figure 4.

(1) Relational database

The relational model is based on a traditional database that originated from Mr. Codd's dynamic thinking, which has been incorporated into the software industry in recent decades. In this model, all entities are expressed in the form of one and only one structure called relations. This model can be displayed in a two-dimensional table understandable to everyone. The simplicity of a relational model is the first determining factor in its success. Other key factors are capability and functionality of software, general and uniform understanding of concepts, and simplicity of comprehensiveness in this model. The complex data processing and big data challenges have

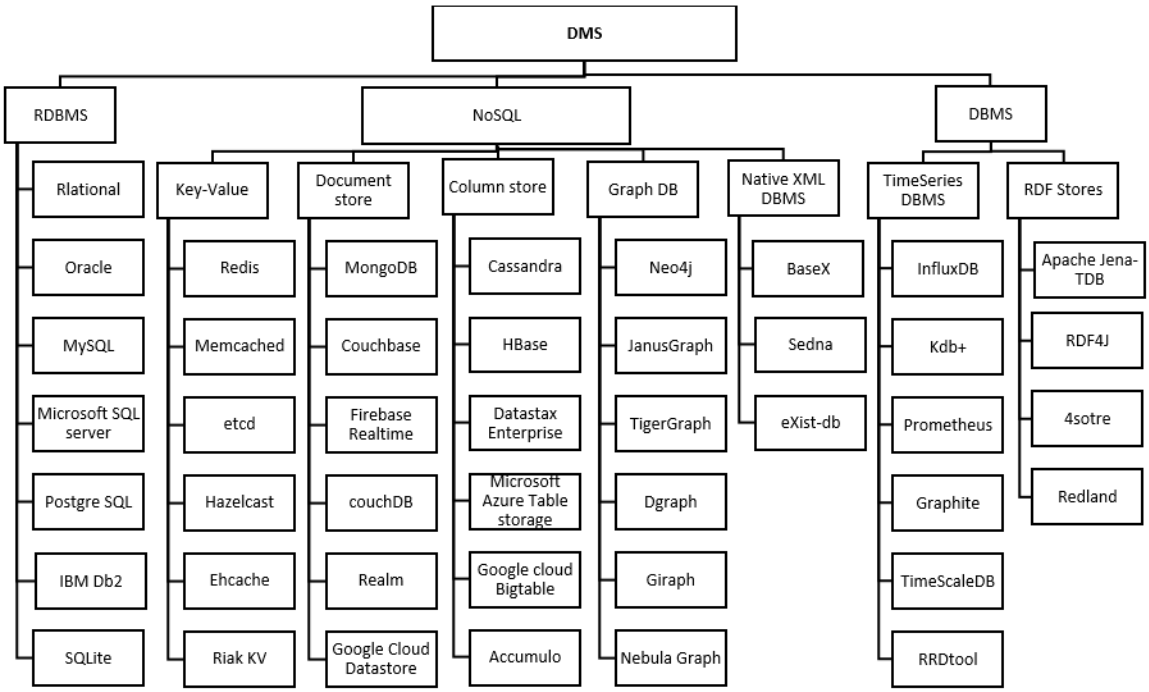

FIGURE 4. Distribution of product types according to database classification.

recently caused deficiencies to this model. Among such deficiencies are having static schema and ACID transaction format, separating data and processing, focusing on structured data and attitude toward tables, and holding no support for nested relationships. Although this model has been satisfying the needs of previous decades, it is now under the influence of changing needs of data processing.

(2) NoSQL

This system has been developed to overcome the challenges of big data and provide a strong adaptation of traditional and predefined schemas to traditional databases. These databases were a new generation of more flexible and higher scalable (horizontally) data storage systems since traditional databases had limited capabilities with large-scale, unstructured, or semi-structured data [23]. NoSQL systems were expanded into different scenarios with the focus on further applications and classified into four main groups based on different data models [61]:

(a) Key-value

This model consists of a value and an index or key to determine the value. The key-value model lists all data in ascending order, and the values are accessible only by the key [43], [62]. In modern types of this category, scalability has priority over compatibility. This system is useful when there is only one type of object for which query is done based on a type of property.

(b) Document store

The storage structure of this type is based on documents. These documents are indexed and a simple query method is presented for them. This system is more advanced and complex than the key-value model. The attribute-names are dynamically specified at runtime. In contrast to the key-value database, this category benefits from the secondary index capability [43]. It stores data in a structured and hierarchical manner and with accessibility to different IDs [62].

(c) Column store

This model, also known as Wide Column Store, structures the data into columns with the number of key-value pairs [43], and thus provides high scalability [62]. Extended records can be vertically or horizontally partitioned across nodes. In this category, the data model is in the form of rows and columns, and the scalability model is obtained from splitting both rows and columns over multiple nodes. The table columns are distributed as a group of columns in other nodes. Partitioning can be done on a table both horizontally and vertically at the same time [43].

(d) Graph database

It uses graph structures with nodes and edges to store data. Graphs and edges respectively play the role of objects and the relationships between them. This type of database employs the index-free adjacency technique in which each node directly relates to its neighbor node. Millions of records can be navigated in this way [43], [62]. Data can be easily transferred from one model to another.

(e) Native XML

It is a database in which data are represented as XML elements. The semantic data must be defined by an

TABLE 4. The data model in each article reviewed.

Paper	XML	TGM	NoSQL	Column store					Key-value				Document store						Graph database					RDF		Relational Model						Time series	
				PNUTS	BigTable	HBase	Cassandra	Azure Table	Redis	Key-Oriented	MemCached	Dynamo	Dynamo	Couchbase	CouchDB	Mongoose	Morphia	Mongo DB	Neo4j	RDF Graph	graph	HyperGraphDB	OEM	RDF(triplestore)	RDF store	sql	sql server	MYSQL	PostgreSQL	Relational	Oracle	influx DB.	Riak Triplestore
42							✓											✓								✓		✓					
4					✓		✓		✓			✓	✓		✓							✓					✓				✓		
30			✓																													
43				✓			✓		✓		✓	✓	✓		✓			✓	✓			✓											
23	✓						✓																									
35																✓	✓	✓															
19							✓											✓															
45									✓																								
54	✓																												✓			
44							✓														✓				✓							
46	✓	✓			✓														✓		✓	✓	✓	✓					✓			
47						✓	✓		✓	✓								✓								✓							
48									✓									✓											✓			✓	
50			✓			✓																							✓			
9				✓	✓		✓					✓	✓																			
16							✓											✓													✓	✓
51																									✓							
38	✓																		✓													
56							✓											✓										✓				
49					✓	✓	✓	✓	✓		✓	✓	✓												✓					✓		
57																			✓	✓	✓								✓			
53			✓								✓			✓	✓														✓			
25																✓		✓														
52			✓															✓														
55													✓	✓	✓			✓														
58							✓																						✓	✓		
8	✓																								✓				✓			

XML element as a combination of delimited start and end tags for description of data elements. The XML document is developed for semi-structured data with arbitrary elements in terms of presence or position. The elements have an underlying tree structure in an XML document. The XML emerged as a response to the weaknesses of relational and object-relational data models. These models have structures with poor scalability in which system performance is reduced with increasing size. Nonetheless, there was no response to the unstructured data.

(3) Time-series database

In the past, attempts were made to use relational or object-relational data models to store timestamp data or data streams or adjust data models to this type of data. These databases were not responsive to the needs of big data. It was impossible to load and process data either continuously or with low latency. Accordingly, time-series databases flourished which were data-driven with data streams as their input [63]. Their models have real-time applications for data streams and timestamp data as well as data-based computations [25]. These databases have features of error tolerance, high accessibility, continuity, or low latency in real-time processing and responding to queries. In these data stream management systems, the main trade-off is between the amount of memory and the approximate accuracy used to store stream [63]. The architecture of Data Stream Management System and its comparison with DBMS are described in [63] and [25], respectively.

Table 4 shows the type and the change of products, relational, RDF, NoSQL, and Time-series, on which data models were tested in each article reviewed. Figure 5 shows the frequency and efficiency of each data model compared to each other.

Based on the reviews and the results, among all the products in database classifications, MongoDB and Cassandra were the most applied in recent years, followed by Relational and Redis databases. As a result, document-based and key-value are the most widely used classifications. Today, data streams from sensors, networks, IoT, and so forth account for the majority of data volume. As time-series data models are required for real-time or low-latency processing of data streams, MongoDB in [25] is discussed to be suitable databas.

Therefore, the popularity of data stream research can be considered one of the reasons for giving emphasis to this type of data model. Based on [8], the lack of solutions to prevent data loss and the problem of compatibility are drawbacks of relational databases in converting RDB data sets to Semantic Web (SW) for storing spatial data. We compared the results of this study with the latest ranking of databases (2021) in terms of popularity. The obtained ranks were based on 378 products (Figure 7) in [64] and 107 products (Figure 6) in our study.

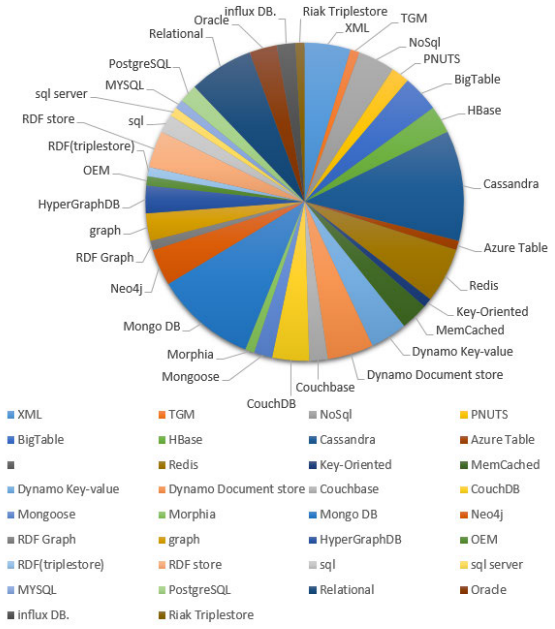

FIGURE 5. Schematic of database products in reviewed articles between 2014 and 2020.

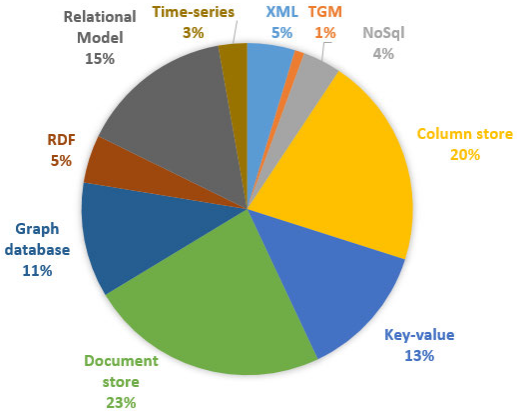

FIGURE 6. General classification of databases in the reviewed articles.

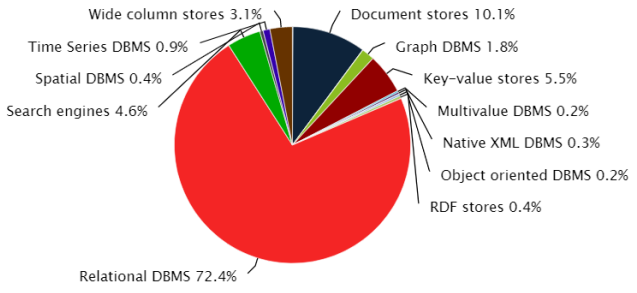

FIGURE 7. The latest ranking of databases in terms of popularity in 2021.

To establish a comparable set of products, both outputs were first transformed into percentages and then compared to each other. The emphasis given by numerous studies to data models of document stores databases is justifiable regarding their popularity. Obviously, this focus on a product can lead to further development of models for more mature products.

The comparison results of this systematic study showed that NoSQL document stores databases were the most reviewed in the field, followed by column stores databases. In spite of their challenges and shortcomings, relational database technologies still are popular, followed by key-value and graph-based databases. Overall, NoSQL databases hold the first rank in the world of database technology with respect to the cumulative frequency of their several groups. On the one hand, features such as high scalability (horizontal), availability, and schema flexibility provide these databases with the possibility to overcome big data challenges and the suitability for data streams. On the other hand, deficiencies such as static schema and limited types of predefined data to define relationships face relational models with major challenges. Hence, the popularity of NoSQL models is completely justifiable. Only those models can be adaptable to the generation of new data types that are able to dynamically define all data types. The results of this systematic review on both databases suggest that a logical format is undeniably required for storing data with covering ACID features, even with the flexible NoSQL schemas and even at the program level. Therefore, researchers are presenting new data models for day-to-day problem solving based on emerging needs, while this issue has remained a challenge in the field of data storage. By comparing our results with the latest available rankings and observing the popularity of relational databases despite the lack of support for big data challenges, it can be concluded that data models if the maturity of relational data models to cover weaknesses and shortcomings use themselves. While retaining their characteristics, they can certainly meet more needs in the future. Just as document store data models like Mango DB have recently tried to incorporate ACID features into their data model. In the next section, we have categorized considered features in each data model reviewed to check the accuracy of basic features for data models. Comparing the results will help us to better understand the concept of the data model and the reasons for emphasis given to a specific model based on its required characteristics.

B. DATA MODEL FEATURES

Apart from application and needs, a data model contains some key features to survive into the age of new information and to meet big data challenges. It is obvious that all of these features cannot be simultaneously gathered in a single model. For instance, agility and flexibility were obtained at the expense of low compatibility of data storage and retrieval with the database in the data model [35]. The scalability was increased with the decrease of ACID properties in NoSQL databases [19]. However, the more coverage of these features, the more capable is the data model when facing the challenge of 3Vs. These basic features of the data model can be summarized as follows:

- Scalability: The data model efficient performance on multiplied size or scale.
- Performance: The data model efficiency in terms of data production volume and velocity and big data processing.
- Representation: The data model management of heterogeneous data making them meaningful for analysts and commentators.
- Consistency: The new model compatibility with previous data models for data integration.
- Flexibility: The data model ability to provide support for various data structures.
- Schema-based or schema-less: The data model ability to have limited types of data with static and predefined schema or unlimited types of data with dynamic schema according to the definition of the relation.
- Data partition: The data model capability of horizontal and vertical partitioning at any time to ensure data availability and concurrent access performance.
- Support for data types: The data model ability to store structured, semi-structured, and unstructured data based on the variety of big data.
- Hash: a technique to directly search the location of desired data on the disk.
- Efficiency: The data model has the necessary efficiency in many processes on data and the database uses a small amount of computational and storage resources.
- Availability: Data should be available whenever necessary.

The determining factors in data model design are considered consistency, availability, and scalability [55]. Having reviewed various articles, we observed that scalability, performance, and schema-less with frequencies of 10% or more are important features in designing the data model (Figure 8). In this diagram, frequencies of different features are close to each other, suggestive of the necessity for the existence of each feature. In other words, a data model is more practical when includes simultaneous existence of these features. The NoSQL databases have been introduced as a complement to the previous ones, especially relational databases to remedy deficiencies of scalability, flexibility at the schema level, ability to store large volumes of data, and efficiency in widely-used applications. The frequency of each feature in the reviewed databases is shown in Figure 9. Table 5 shows

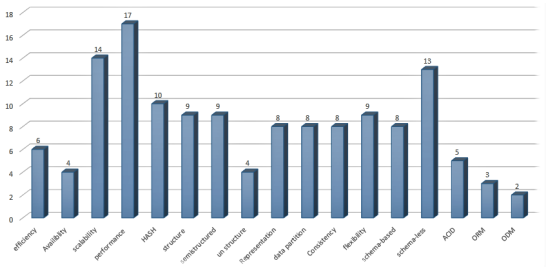

FIGURE 8. Cumulative frequency of key features reviewed in the articles.

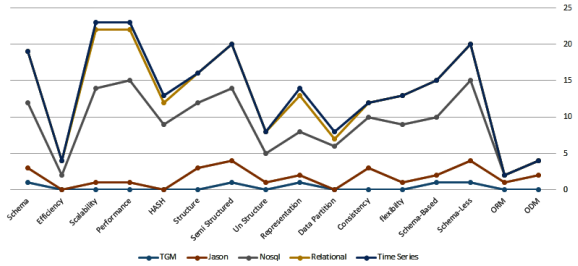

FIGURE 9. The frequency of each feature in the reviewed databases.

TABLE 5. Each data model features separately studied in each article.

Paper	efficiency	Availability	scalability	performance	HASH	structure	semi structured	un structure	Representation	data partition	Consistency	flexibility	schema-based	schema-less	ACID	ORM	ODM
4	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	✗
8	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
9	✗	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✓	✓	✗	✗
16	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
19	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗
23	✓	✗	✓	✓	✓	✓	✓	✗	✗	✓	✗	✓	✗	✓	✗	✗	✗
25	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗
30	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗
35	✓	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗	✓	✓
38	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✓	✓	✓	✗	✗	✗
42	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
43	✗	✗	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✗	✗
44	✗	✗	✓	✓	✗	✗	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗
45	✓	✗	✓	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
46	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✓	✗	✗	✗
47	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
48	✗	✗	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗
49	✗	✗	✓	✓	✗	✓	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗
50	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
51	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
52	✓	✗	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗
53	✗	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗
54	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
55	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗
56	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗
57	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
58	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗

in detail the features that each data model covers. Comparing the results of Table 4 and Table 5, it can be inferred that NoSQL databases constitute the dominant group concerning the coverage of key features and the variety of products in different applications. The relational model takes second place, and both models are merely adjacent in the Efficiency feature.

Table 6 shows responses to the questions in Section 3-C, briefly describing which question response can be achieved in which article content. Figure 10 illustrates the frequency of responses from the papers reviewed. The XML, RDF, and JSON format of initial data representation and, in some cases, data exchanges were studied in databases and data models, shown in Table 7, with frequencies shown in Figure 11. The JSON in column store models, as well as MongoDB in [16],

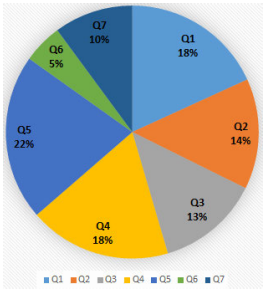

FIGURE 10. The frequency of responses from the articles reviewed.

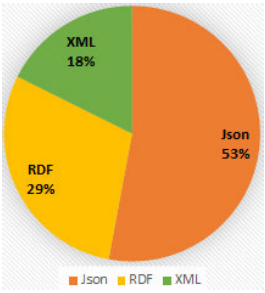

FIGURE 11. The frequency of the initial format of data representation in the data models reviewed.

TABLE 6. Responses to each question in Section 3-3 from each article.

Paper	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Paper	Q1	Q2	Q3	Q4	Q5	Q6	Q7
4	✓	✗	✗	✗	✓	✓	✗	46	✗	✗	✓	✗	✓	✗	✗
8	✗	✗	✓	✓	✗	✗	✓	47	✗	✓	✗	✓	✓	✓	✓
9	✓	✓	✓	✓	✓	✗	✗	48	✓	✓	✗	✗	✓	✗	✗
16	✓	✓	✗	✓	✓	✓	✗	49	✓	✗	✓	✓	✓	✗	✗
19	✓	✓	✗	✓	✓	✗	✗	50	✓	✓	✓	✗	✓	✗	✗
23	✓	✗	✓	✓	✓	✗	✗	51	✓	✗	✓	✓	✓	✓	✗
25	✗	✗	✓	✓	✓	✗	✓	52	✗	✓	✓	✗	✓	✗	✗
30	✓	✗	✗	✓	✓	✗	✓	53	✗	✗	✓	✗	✓	✗	✗
35	✓	✓	✗	✓	✓	✗	✓	54	✗	✗	✗	✓	✓	✗	✗
38	✓	✓	✗	✓	✗	✓	✓	55	✓	✓	✗	✗	✓	✓	✗
42	✓	✓	✗	✓	✗	✗	✓	56	✗	✗	✗	✓	✗	✗	✓
43	✓	✓	✓	✗	✓	✗	✗	57	✓	✗	✗	✓	✓	✗	✗
44	✓	✓	✗	✗	✓	✓	✗	58	✓	✓	✗	✗	✓	✗	✓
45	✓	✗	✓	✓	✗	✗	✗								

is the most widely used formats of initial data representation, providing standards for describing and testing documents. This format is suitable for unstructured, semi-structured, and structured data presentation from both schema-based and schema-less points of view using a set of key-value pairs. JSON technology is integrated with a number of distinctive NoSQL databases to represent raw data exchange formats. As a data format, JSON is also suited to NoSQL databases for serialization. It has been used in the design of NoSQL databases [30]. JSON is a simple data format that allows developers to store a set of values, lists, and key-value pairs that communicate with each other [59], [60]. Table 7 shows the representation formats used in each papers.

TABLE 7. The data representation formats regarding resources used in each article.

Paper	Json	RDF	XML	Paper	Json	RDF	XML
4	✗	✗	✗	46	✗	✓	✓
8	✗	✓	✓	47	✗	✗	✗
9	✗	✗	✗	48	✓	✗	✗
16	✓	✗	✗	49	✗	✗	✗
19	✗	✗	✗	50	✓	✗	✗
23	✗	✗	✗	51	✗	✓	✗
25	✓	✗	✗	52	✓	✗	✗
30	✓	✗	✗	53	✓	✗	✗
35	✓	✗	✗	54	✗	✗	✓
38	✗	✗	✗	55	✗	✗	✗
42	✗	✗	✗	56	✓	✗	✗
43	✗	✗	✗	57	✗	✓	✗
44	✗	✓	✗	58	✗	✗	✗
45	✗	✗	✗				

VI. CONCLUSION

The exponential growth of data generated from different sources, high volume and variety of data, and their rapid transfer from digital technologies brought about the growth of big data, the challenge of 3Vs, and the development of software technologies to overcome this challenge. In the field of data storage and management at the abstract level of database design, a data model is required to define the data structure and storage as a way to meet the challenges of big data. In this article, we have systematically reviewed data model types proposed from 2014 to 2020. The results suggest that the research focus has mainly been on the classification of data models, storage of various data types and variety, support capabilities and basic features of data models, and the popularity and basic technology in recent years. The results of this review will be helpful for users not only in the domain of database and data storage but also in various domains of big data, cloud, data migration, and so on. The elegance of this article lies in its allowing for the possibility of comparing data models based on their critical features and corresponding products. This article depicts the significance of data models even in NoSQL databases which lack static data models and flexible schemas. Although there are a variety of data models with various purposes, there must be a logical structure or format for data storage even at the program level. This is evidence of the need for more focus and research on this issue.

REFERENCES

[1] E. J. Yannakoudakis, "Foundations of databases," in *The Architectural Logic of Database Systems*. 1988, pp. 1–16, doi: 10.1007/978-1-4471-1616-5-1.
[2] A. Y. Zomaya and S. Sakr, *Handbook of Big Data Technologies*. Cham, Switzerland: Springer, Dec. 2017, pp. 1–895.
[3] N. Kusuma, *Database Systems*, vol. 53, no. 9. Basingstoke, U.K.: Palgrave Macmillan, 2013.
[4] Z. Zheng, Z. Du, L. Li, and Y. Guo, "BigData oriented open scalable relational data model," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2014, pp. 398–405, doi: 10.1109/BigData.Congress.2014.65.
[5] G. Pavloviác and P.-L. Lazetiác, "Native XML databases vs. relational databases in dealing with XML documents," *Kragujev. J. Math.*, vol. 30, pp. 181–199, Oct. 2007.
[6] *Data Models for Semistructured Data*, The Jerusalem Talmud, Zeraim, Tractate Berakhot, 2013, doi: 10.1515/9783110800487.156.

[7] M. Vazirgiannis, "Data modeling: Object-oriented data model," in *Encyclopedia of Information Systems*. Amsterdam, The Netherlands: Elsevier, 2004, doi: 10.1016/b0-12-227240-4/00035-6.
[8] K. R. Malik, M. A. Habib, S. Khalid, and M. Ahmad, "A generic methodology for geo-related data semantic annotation," *Concurrency Comput., Pract. Exper.*, vol. 30, p. e4495, Aug. 2018, doi: 10.1002/cpe.4495.
[9] Y. Mansouri, A. N. Toosi, and R. Buyya, "Data storage management in cloud environments: Taxonomy, survey, and future directions," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 1–51, Jan. 2018, doi: 10.1145/3136623.
[10] C. Lv, C.-Y. Wei, and Y. Hao, "Schema discovery of semi-structured hierarchical data based on OEM model and hierarchical transactional database," in *Proc. Int. Conf. Manage. E-Commerce E-Government*, 2009, pp. 172–175, doi: 10.1109/ICMeCG.2009.82.
[11] A. A. Tole, "Big data challenges," *Database Syst. J.*, vol. 4, no. 3, pp. 31–40, 2013.
[12] K. Ahmed, "A survey on big data analytics: Challenges, open research issues and tools," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 2, pp. 511–518, 2016, doi: 10.14569/ijacsa.2016.070267.
[13] H. Cai, B. Xu, L. Jiang, and A. V. Vasilakos, "IoT-based big data storage systems in cloud computing: Perspectives and challenges," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 75–87, Jan. 2017, doi: 10.1109/JIOT.2016.2619369.
[14] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 171–209, Apr. 2014, doi: 10.1007/s11036-013-0489-0.
[15] A. Cravero, "Big data architectures and the Internet of Things: A systematic mapping study," *IEEE Latin Amer. Trans.*, vol. 16, no. 4, pp. 1219–1226, Apr. 2018, doi: 10.1109/TLA.2018.8362160.
[16] G. Baruffa, M. Femminella, M. Pergolesi, and G. Reali, "Comparison of MongoDB and Cassandra databases for spectrum monitoring as-a-service," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 1, pp. 346–360, Mar. 2020, doi: 10.1109/TNSM.2019.2942475.
[17] R. Cattell, "Scalable SQL and NoSQL data stores," *ACM SIGMOD Rec.*, vol. 39, no. 4, pp. 12–27, May 2011, doi: 10.1145/1978915.1978919.
[18] S. LaValle, E. Lesser, R. Shockley, M. S. Hopkins, N. Kruschwitz, and S. LaValle, "Big data, analytics and the path from insights to value," *MIT Sloan Manag. Rev.*, vol. 52, no. 2, pp. 21–31, 2011. [Online]. Available: http://tuping.gsm.pku.edu.cn/Teaching/Mktrch/Readings/Big Data
[19] A. de la Vega, D. García-Saiz, C. Blanco, M. Zorrilla, and P. Sánchez, "Mortadelo: Automatic generation of NoSQL stores from platform-independent data models," *Future Gener. Comput. Syst.*, vol. 105, pp. 455–474, Apr. 2020, doi: 10.1016/j.future.2019.11.032.
[20] M. Chen, S. Mao, Y. Zhang, and V. C. M. Leung, *Big Data—Related Technologies, Challenges and Future Prospects*. Springer, 2014.
[21] A. A. Safaei, "Hyper nested graph: Data model for big data," *Mod. J. Elect. Eng.*, vol. 14, no. 3, pp. 1–15, 2016.
[22] Y. Alshboul, Y. Wang, and R. K. Nepali, "Big data lifecycle: Threats and security model," in *Proc. Amer. Conf. Inf. Syst.*, 2015, pp. 1–7.
[23] A. Mohan, M. Ebrahimi, S. Lu, and A. Kotov, "A NoSQL data model for scalable big data workflow execution," in *Proc. IEEE Int. Congr. Big Data, BigData Congr.*, Oct. 2016, pp. 52–59, 2016, doi: 10.1109/BigDataCongress.2016.15.
[24] P. S. Engineer, A. Wang, and S. Engineering. *Data Modeling for Big Data*. [Online]. Available: https://docplayer.net/17683662-Datamodeling-for-big-data.html
[25] N. Q. Mehmood, R. Culmone, and L. Mostarda, "Modeling temporal aspects of sensor data for MongoDB NoSQL database," *J. Big Data*, vol. 4, no. 1, pp. 1–35, Dec. 2017, doi: 10.1186/s40537-017-0068-5.
[26] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, vol. 4, 6th ed. New York, NY, USA: McGraw-Hill, 2011.
[27] E. F. Codd, "Data models in database management," *ACM SIGMOD Rec.*, vol. 11, no. 2, pp. 112–114, doi: 10.1145/960126.
[28] T. Connolly and C. Begg, *Database Solutions: A Step-by-Step Guide to Building Databases*. London, U.K.: Pearson, 2004.
[29] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Comput. Surv.*, vol. 40, no. 1, pp. 1–39, 2008, doi: 10.1145/1322432.1322433.
[30] S. Banerjee and A. Sarkar, "Logical level design of NoSQL databases," in *Proc. IEEE Region Conf. (TENCON)*, Nov. 2016, pp. 2360–2365, doi: 10.1109/TENCON.2016.7848452.
[31] S. Banerjee and A. Sarkar, "Modeling NoSQL databases: From conceptual to logical level design," in *Proc. 3rd Int. Conf. Appl. Innov. Mob. Comput.*, Kolkata, India, Feb. 2016, pp. 10–12.
[32] X. Li, Z. Ma, and H. Chen, "QODM: A query-oriented data modeling approach for NoSQL databases," in *Proc. IEEE Work. Adv. Res. Technol. Ind. Appl.*, May 2014, pp. 338–345, doi: 10.1109/WARTIA.2014.6976265.
[33] P. Atzeni, F. Bugiotti, and L. Rossi, "Uniform access to NoSQL systems," *Inf. Syst.*, vol. 43, pp. 117–133, Jul. 2013, doi: 10.1016/j.is.2013.05.002.
[34] P. Atzeni, *Data Modeling in the NoSQL World to Cite Version*. Bengaluru, India: HAL, 2017.
[35] A. H. Chillon, D. S. Ruiz, J. G. Molina, and S. F. Morales, "A model-driven approach to generate schemas for object-document mappers," *IEEE Access*, vol. 7, pp. 59126–59144, 2019, doi: 10.1109/ACCESS.2019.2915201.
[36] M. J. Mior, "Automated schema design for NoSQL databases," in *Proc. SIGMOD Symp.*, 2014, pp. 41–45, doi: 10.1145/2602622.2602624.
[37] G. Zhao, Q. Lin, L. Li, and Z. Li, "Schema conversion model of SQL database to NoSQL," in *Proc. 9th Int. Conf. Parallel, Grid, Cloud Internet Comput.*, Nov. 2014, pp. 355–362, doi: 10.1109/3PGCIC.2014.137.
[38] V. M. de Sousa and L. M. del V. Cura, "Logical design of graph databases from an entity-relationship conceptual model," p. 7, doi: 10.1145/3282373.
[39] C. Jatoth, G. R. Gangadharan, and R. Buyya, "Computational intelligence based QoS-aware web service composition: A systematic literature review," *IEEE Trans. Services Comput.*, vol. 10, no. 3, pp. 475–492, May/Jun. 2017, doi: 10.1109/TSC.2015.2473840.
[40] D. Moher, P.-P. Group, L. Shamseer, M. Clarke, D. Ghersi, A. Liberati, M. Petticrew, P. Shekelle, and L. A. Stewart, "Preferred reporting items for systematic review and meta-analysis protocols (PRISMA-P) 2015 statement," *Syst. Rev.*, vol. 4, no. 1, pp. 1177–1185, Dec. 2015.
[41] A. Chebotko, A. Kashlev, and S. Lu, "A big data modeling methodology for apache Cassandra," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2015, pp. 238–245, doi: 10.1109/BigDataCongress.2015.41.
[42] Y. Kidane and D. Kim, "Traceability architecture: Extending EPCIS to enhance track and trace with NoSQL data model," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Jun. 2016, pp. 880–883, doi: 10.1109/SCC.2016.129.
[43] A. Makris, K. Tserpes, V. Andronikou, and D. Anagnostopoulos, "A classification of NoSQL data stores based on key design characteristics," *Procedia Comput. Sci.*, vol. 97, pp. 94–103, Oct. 2016, doi: 10.1016/j.procs.2016.08.284.
[44] L. Reyes-Álvarez, M. D. M. Roldán-García, and J. F. Aldana-Montes, "Tool for materializing OWL ontologies in a column-oriented database," *Softw., Pract. Exper.*, vol. 49, no. 1, pp. 100–119, Jan. 2019, doi: 10.1002/spe.2645.
[45] S. Lai, X. Yuan, S.-F. Sun, J. K. Liu, Y. Liu, and D. Liu, "GraphSE: An encrypted graph database for privacy-preserving social search," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Jul. 2019, pp. 41–54, doi: 10.1145/3321705.3329803.
[46] E. Damiani, B. Oliboni, E. Quintarelli, and L. Tanca, "A graph-based meta-model for heterogeneous data management," *Knowl. Inf. Syst.*, vol. 61, no. 1, pp. 107–136, Oct. 2019, doi: 10.1007/s10115-018-1305-8.
[47] G. A. Schreiner, D. Duarte, and R. Mello, "SQLtoKeyNoSQL: A layer for relational to key-based NoSQL database mapping," in *Proc. 17th Int. Conf. Inf. Integr. Web-Based Appl. Services*, Dec. 2015, pp. 1–9, doi: 10.1145/2837185.2837224.
[48] B. Balis, M. Bubak, D. Harezlak, P. Nowakowski, M. Pawlik, and B. Wilk, "Towards an operational database for real-time environmental monitoring and early warning systems," *Procedia Comput. Sci.*, vol. 108, pp. 2250–2259, Oct. 2017, doi: 10.1016/j.procs.2017.05.193.
[49] S. Sharma, "Expanded cloud plumes hiding big data ecosystem," *Future Gener. Comput. Syst.*, vol. 59, pp. 63–92, Jun. 2016, doi: 10.1016/j.future.2016.01.003.
[50] X. Yao and G. Li, "Big spatial vector data management: A review," *Big Earth Data*, vol. 2, no. 1, pp. 108–129, Feb. 2018, doi: 10.1080/20964471.2018.1432115.
[51] M. V. Sokolova, F. J. Gómez, and L. N. Borisoglebskaya, "Migration from an SQL to a hybrid SQL/NoSQL data model," *J. Manage. Anal.*, vol. 7, no. 1, pp. 1–11, Jan. 2020, doi: 10.1080/23270012.2019.1700401.
[52] P. Colombo and E. Ferrari, "Evaluating the effects of access control policies within NoSQL systems," *Future Gener. Comput. Syst.*, vol. 114, pp. 491–505, Jan. 2021, doi: 10.1016/j.future.2020.08.026.
[53] M. A. Hubail, A. Alsuliman, M. Blow, M. Carey, D. Lychagin, I. Maxon, and T. Westmann, "Couchbase analytics: NoETL for scalable NoSQL data analysis," *Proc. VLDB Endowment*, vol. 12, no. 12, pp. 2275–2286, Aug. 2019, doi: 10.14778/3352063.3352143.
[54] C. Batini, P. Bonizzoni, M. Comerio, R. Dondi, Y. Pirola, and F. Salandra, "A clustering algorithm for planning the integration process of a large number of conceptual schemas," *J. Comput. Sci. Technol.*, vol. 30, no. 1, pp. 214–224, Jan. 2015, doi: 10.1007/s11390-015-1514-5.

[55] A. A. Imam, S. Basri, R. Ahmad, J. Watada, and M. T. González-Aparicio, "Automatic schema suggestion model for NoSQL document-stores databases," *J. Big Data*, vol. 5, no. 1, pp. 1–17, Dec. 2018, doi: 10.1186/s40537-018-0156-1.
[56] S. Sachdeva, P. Sharma, R. Jain, and S. Parashar, "Critical analysis of data storage approaches," *Procedia Comput. Sci.*, vol. 173, pp. 264–271, 2020, doi: 10.1016/j.procs.2020.06.031.
[57] H. Ma, B. Shao, Y. Xiao, L. J. Chen, and H. Wang, "G-SQL: Fast query processing via graph exploration," *Proc. VLDB Endowment*, vol. 9, no. 12, pp. 900–911, Aug. 2016, doi: 10.14778/2994509.2994510.
[58] M. Golosova, M. Grigorieva, A. Klimentov, and E. Ryabinkin, "PanDA workload management system meta-data segmentation," *Procedia Comput. Sci.*, vol. 66, pp. 448–457, 2015, doi: 10.1016/j.procs.2015.11.051.
[59] P. Bourhis, J. L. Reutter, and D. Vrgoä, "JSON: Data model and query languages," *Inf. Syst.*, vol. 89, Mar. 2020, Art. no. 101478, doi: 10.1016/j.is.2019.101478.
[60] D. Peng, L. Cao, and W. Xu, "Using JSON for data exchanging in web service applications," *J. Comput. Inf. Syst.*, vol. 16, pp. 5883–5890, Dec. 2011.
[61] D. G. Chandra, "BASE analysis of NoSQL database," *Future Gener. Comput. Syst.*, vol. 52, pp. 13–21, Nov. 2015, doi: 10.1016/j.future.2015.05.003.
[62] P. Gölzer, L. Simon, P. Cato, and M. Amberg, "Designing global manufacturing networks using big data," *Procedia CIRP*, vol. 33, pp. 191–196, Oct. 2015, doi: 10.1016/j.procir.2015.06.035.
[63] S. Geisler, "Data stream management systems," *Dagstuhl Follow*, vol. 5, pp. 275–304, Oct. 2013. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2013/4297/
[64] (Dec. 2020). *DBMS Popularity Broken Down by Database Model Number of Systems per Category*. [Online]. Available: https://db-engines.com/en/ranking_categories

FAEZEH MOSTAJABI received the B.Sc. degree in computer software engineering from the University of Applied Science and Technology, Shiraz, Iran, in 2009, and the M.Sc. degree in information technology engineering from Amirkabir University of Technology, Tehran, Iran, in 2014. She is currently pursuing the Ph.D. degree with the Department of Computer Engineering, Islamic Azad University, Qeshm Branch, Qeshm, Iran. Her research interests include data modeling, user interest modeling, green computing, and programing.

ALI ASGHAR SAFAEI received the B.Sc. degree in computer software engineering from Shahid Sattari Air University, Tehran, Iran, in 2001, the M.Sc. degree in computer software engineering from Ferdowsi University, Mashhad, Iran, in 2004, and the Ph.D. degree in computer software engineering from Iran University of Science and Technology, Tehran, in 2011. He is currently an Assistant Professor with the Department of Computer Engineering, Tarbiat Modares University, Tehran. His research interests include medical software design, nursing informatics, free/open-source software (FOSS) development, advanced databases, advanced software engineering, and distributed systems.

AMIR SAHAFI received the B.Sc. degree from Shahed University of Tehran, Iran, in 2005, and the M.Sc. and Ph.D. degrees from Islamic Azad University, Science and Research Branch, Tehran, in 2007 and 2012, respectively, all in computer engineering. He is currently an Assistant Professor with the Department of Computer Engineering, Islamic Azad University, South Tehran Branch, Tehran. His current research interests include distributed and cloud computing.