Received: February 10, 2023. Revised: March 12, 2023.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

EMEMODL: Extensible Metadata Model for Big Data Lakes

Mohamed Cherradi[1]* **Anass El Haddadi[1]**

[1]*Data Science and Competetive Intelligence Team (DSCI), ENSAH, Abdelmalek Essaadi University (UAE) Tetouan, Morocco*
* Corresponding author's Email: m.cherradi@uae.ac.ma

Abstract: The evolution of the huge amount of heterogeneous data sources introduces the emergence of several fresh new concepts. Among one of the most well-known concepts emerging as a recent and trending topic in big data is the data lake. Which represents a central repository that stores heterogeneous data sources in its native format without any predefined schema. In the absence of any enforced schema, effective metadata management based on metadata models remains an active research topic to address the data lake issues knowing by a data swamp. However, the examination of existing metadata models shows that none of them proposes a complete model. In this article, we propose a generic and extensible metadata model that supports high flexibility and better scalability in the integration of metadata. With these capabilities, EMEMODL enables comprehensive metadata management for data lakes. We show EMEMODL's feasibility through a prototypical implementation based on TPC-H datasets of different sizes to prove the scaling feature, using MySQL and Neo4J. The findings of these experiments revealed encouraging results for big data queries that give us a graph database, which shows an efficient ability to manage and process large amounts of data compared to a relational database in terms of retrieval time queries and resource consumption, including cpu and memory usage.

Keywords: Big Data, Data lake, Database management, Metadata models.

1. Introduction

Nowadays, the exponential expansion of created data has given rise to the concept of data lakes in the big data era. Data lakes have emerged as one of the trending topics for big data that has attracted an increasing amount of interest from a group of researchers. The basic idea of the data lake concept is simple: "*If you think of a warehouse as a store of bottled water, one packaged for easy consumption, the data lake is a great source of water in its natural state*". It enables the storage of heterogeneous data sources "as-is" in their native format. There is no predetermined schema necessary for data stored in the lake; this makes it possible to store vast amounts of heterogeneous data quickly and easily. However, without an effective metadata management system, the integration of data without any rigorous structure results in a massive collection of undocumented data that can quickly turn a data lake into a data swamp [1]. The term "data swamp" refers to useless, undocumented, or inexploitable data. Therefore, enhancing the value of big data is made possible in large part by the data lake metadata management system, also known as "data intelligence" or "large-scale data" [2]. Thus, the data lake concept was first introduced by Dixon, as an efficient solution to the gaps introduced by data marts [3]. It is defined as a large storage space for various data sources that allows users to explore different data stored in the lakes. It provides not only a means of storing various data sources but also an efficient system of data processing. Furthermore, the data lake quickly gained importance and significance. Indeed, many researchers have started to work in this direction to address the data lake issues [4, 5]. Thus, metadata management is one of the key topics explored in the literature to prevent the conversion of data lakes into data swamps [6-8].

The key subject of metadata management is to understand the metadata model concept [9-12]. A metadata model is defined as an abstract model to show how the metadata associated with the data

stored in the lake is organized and related to each other. However, most metadata model proposals are not sufficiently generic [7, 13, 14]. It frequently focuses on specific use cases and provides a few details on data design. This constitutes a significant limitation, as several important use cases require flexibility and scalability. Therefore, defining a metadata model for a specific use case is not realistic because the number of use cases is something that is not constant and varies with the requirements of each organization.

To address the genericity issue with metadata models that offer flexibility and scalability, which reflect the essential components that complete a metadata model's extensibility. Thus, analyzing the issue of different data formats and their effectiveness in storing and processing data using DBMS scalability proved to be one of the best solutions for dealing with this type of schema-less data [15]. Furthermore, our purpose aims to build a generic model by enhancing existing metadata models with new capabilities like flexibility and scalability, which will help us address the issues that massive data lakes have brought about, such as volume, variety, value, veracity, etc. Therefore, our objective is to create a generic model that can manage and adapt to a collection of use cases in order to overcome the limitations provided by a group of recent metadata models that only concentrate on particular use cases [16]. In this article, we propose an efficient approach to building metadata models based on a typology of metadata, which covers the ideal set of metadata model functionalities. The construction of our model passes through modeling at three levels: 1) conceptual, 2) logical, and 3) physical. For each level, we associate with it the list of identical concepts with their mathematical formalization. Additionally, we have introduced the passage of transformation from one level to another. To assess the effectiveness of our model, we compared it with recent metadata models. The benchmark study reflects the highest level of abstraction for our model, which is explained by its genericity. The experimental evaluation focuses on three sample TPC-H datasets of different sizes. The experiments were carried out with MySQL and Neo4j.

The remainder of this article is organized as follows: section 2, examines related literature on data lake metadata management. In section 3, we investigate the methodology followed by our metadata models. In section 4, we report the results and discussion of our study. Finally, section 5, concludes the paper and suggests future research perspectives.

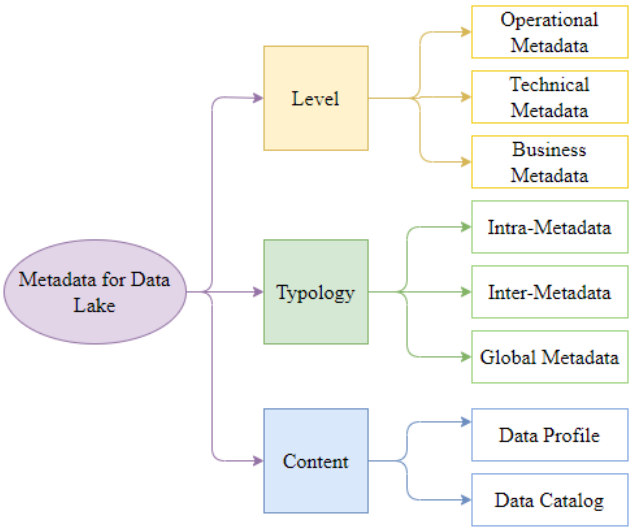


Figure. 1 Metadata categorization in data lakes

2. Related works

A literature review was carried out to provide us with an overview of metadata management for data lake information systems. In fact, metadata management plays a significant role, contributing to maximizing the value of the data stored in the lake. Within this section, we examine recent literature reviews about metadata categorization and metadata models for data lakes.

2.1 Metadata categorization

Metadata management plays a significant role in allowing users to efficiently analyze and explore various data sources collected in the lake. Indeed, as shown in Table 1, a variety of approaches aid in differentiating metadata categorisation. Other works exist that are not listed in Table 1. For example, J. Riley [17] categorizes metadata according to three types: structural, descriptive, and administrative metadata. Structural metadata describes the types, versions, and relationships between subsets of data and how these belong together. Descriptive metadata describes data sources with the aim of discovery and identification. It can contain items such as the author, title, keywords, etc. Lastly, the administrative metadata provides information that aids in resource management, such as when and how data is created and who can access it.

In this article, we address metadata categorization by adapting the existing classification of [18], which we have extended by adding a data catalog at the content metadata category level, as shown in Fig. 1.

Concern level metadata category represents the metadata typology adopted by Diamantini [13]. It encompasses three types of metadata: business,

Table 1. The two main approaches for creating metadata models for data lakes.

Metadata (MD) Models	**1st approach** MD Categorization	**2nd approach** List of MD Management Features
GEMMS[26]	Semantic MD Structure MD	-
Ground [10]	Behaviour MD Application MD Versioning MD	-
Diamantini [13]	Technical MD Business MD Operational MD	-
Ravat and Zhao [11]	Inter-MD Intra-MD	-
MEDAL [12]	Inter-MD Intra-MD	Data Indexing, Semantics, enrichment, lineage, linkage, versioning, polymorphisme
HANDLE [27]	-	Multiple granularity levels, Zone MD, MD properties
goldMEDAL [28]	Inter-MD Intra-MD Global-MD	Data Linkage, Semantic enrichment, Usage tracking

technical and operational metadata. Business metadata provides meaning and contexts to technical metadata [18], for example, book title, publisher, authors, etc. Technical metadata provides information on the technical aspects of data, for example, size, type, table name, column name, etc. Finally, operational metadata describes details of the accessing and processing of data, for example, access rights, provenance, quality requirements, etc.

For the typology metadata category, this is the metadata typology adopted by MEDAL [12]. It divides metadata into three types: intra-metadata, inter-metadata, and global metadata. Intra-metadata points to a specific data source (or object in the jargon of MEDAL). It includes sex functionalities, such as semantic enrichment (SE), data indexing (DI), link generation and conservation (LG), data polymorphism (DP), data versioning (DV), and usage tracking (UT). Inter-metadata types represent the links or relationships between different objects. It contains the following functionalities: Object groupings, similarity links, and parenthood relationships. Finally, global metadata is defined by its name, i.e., neither intra-metadata nor inter-metadata. It concerns the entire resource stored in the lake.

Lastly, for the content metadata category, this category groups together two types, namely: data profile and data catalog. Following [19-21] through big data profile, we can investigate a posteriori data analysis and data discovery. Hence, the main characteristic of the data lake is "schema-on-read", i.e., the analysis performed when needed. We cannot talk about the metadata content category without citing a data catalog. A data catalog is defined as a collection of metadata combined with data management that helps different users to find data that they need. Furthermore, data catalogs have increasingly become the standard for metadata management in the big data era and self-service analytics.

2.2 Metadata models

Several approaches have been proposed to handle metadata in data lakes. Nonetheless, the majority of them provide a few details on how data is conceptually designed [22, 23]. Furthermore, a review of all recently proposed metadata models in the literature is absolutely essential. Yet, metadata models serve as a broad framework for organizing metadata [24, 25]. Thus, each metadata model is unique from other models in terms of the concepts it has presented, the functionalities it offers, or a variety of other variables. GEMMS [26] is an illustration of a generic metadata model that can be utilized with data lakes, despite the fact that this was not its intended usage. When considering the concepts of the metadata models, GEMMS provides two concepts: the data file and the data unit. Nevertheless, GEMMS needs data structure information, which is incompatible with unstructured data. Ground [10] is suggested as a way to manage unstructured data without being aware of its structure. Despite its greater significance in depicting the relationship between the various data sources, Ground does not support data linkage. In order to address this issue, Diamantini [13] added similarity linkages despite the fact that they lacked data versioning and tracking. According to the Diamantini model, Ravat and Zhao [11] suggest that each step is connected to a particular zone in the metadata zone. Yet, data granularity aspects are not under the control of Ravat and Zhao. Therefore, MEDAL [12] is suggested to evaluate granularity features but does not handle various data granularity levels. Then, HANDLE [27] is proposed as a solution to overcome this gap. However, HANDLE is incomplete due to the fact that its architecture does not permit data lineage. As a result,

Table 2. Ideal features supported by data lakes metadata models

Features / Models	SE	DP & MZ	DV	MGL	UT	DC	SL	MP	DS	Total
GEMMS [26**]**	√			√		√		√		**4/9**
Ground [10**]**	√		√		√	√		√		**5/9**
Diamantini [13**]**	√	√		√			√			**4/9**
Ravat and Zhao [11**]**	√	√	√		√	√	√	√		**7/9**
MEDAL [12**]**	√	√	√		√	√	√	√		**7/9**
HANDLE [27**]**	√	√		√	√	√	√	√		**7/9**
goldMEDAL [28**]**	√	√	√	√	√	√	√	√		**8/9**
EMEMODL's	√	√	√	√	√	√	√	√	√	**9/9**

goldMEDAL [28] is proposed to cover the gaps left by MEDAL and HANDLE. Subsequently, goldMEDAL is not considered generic in the sense that it doesn't provide data scalability. This feature will be the major asset of our article.

However, to keep things straightforward and so that everyone can understand how metadata models are created, HANDLE [27] describes the metadata model building according to the most common and widely used ways in the literature. The first approach is based on the metadata classification, while the second is based on the functionalities supported by each metadata model. Table 1, summaries these two approaches.

3. Methodology

In this section, we illustrate our proposed metadata model called "EMEMODL". To carry out this study, we have presented a methodology that will describe the main stages in the realization of our contribution. Such proposition of a metadata model for data lakes needed some requirements, like the set of ideal requirements for a generic and extensible metadata model defined in section 3.1. Further, the design of our metadata model, and for more sake of precision, the list of concepts which will be designed based on the conceptual and logical models described in section 3.2.

3.1 Ideal requirements for a generic and extensible metadata model

After examining the metadata models in the related work section, it appears none of the existing models propose a generic and extensible model. Thus, a generic metadata model should be flexible and scalable to support different data lake use cases. In this article, we address the genericity aspect gap by exploiting existing models and their shortcomings. Indeed, we consider the most abundant features that are not taken into consideration by a metadata model to be the generic aspect of such a model. Therefore, features supported by different models are a suitable way to compare them [12].

The generic degree of the existing models yields results that are not sufficiently generic and cannot handle different use cases. Thus, the need for a generic and extensible metadata model remains a real challenge in data management. For this reason, we offer generic requirements supported by recent existing metadata models and ones that are not supported. This is a significant add-on makes it very generic and extensible. It can support and handle any data lake use case. To the best of our knowledge, there are eight features proposed by the recent generic metadata models in the literature. These characteristics are used to compare various models. They are as follows: data categorization (DC), semantic enrichment (SE), data polymorphism/multiple zones (DP&MZ), data versioning (DV), similarity links (SL), metadata properties (MP), usage tracking (UT) and multiple granularity levels (MGL).

However, existing generic models don't take into consideration Data Scalability (DS), even though this feature has been identified as relevant and adds to the model more extensibility and genericity. Nonetheless, data scalability accommodates rapid changes in the growth of data. Therefore, for a large-scale metadata system to perform well, distributed metadata management must be flexible and scalable [22]. In addition to that, according to [25], metadata management systems need a scalable feature as a

Table 3. Symbols used for the annotations of conceptual models

Notations	Meaning
DE	Data Entity: represents the raw dataset stored in the lake.
G	Group (or Category): represents a set of entities that are grouped according to a similarity measure (intra-group relationship).
Gr	Grouping: represents a relationship between two groups (inter-group relationship)
DP	Data Processing: represents the process applicable to the data to make them interoperable.
Rel	Relationship: represents intra-group and inter-group relationship.

relevant requirement to provide consistent storage scalability and performance. Furthermore, by adding data scalability to the eight relevant features identified in the paragraph above, it appears none of all the models reviewed support all of them. Table 2 highlights the exemplary features shared by the models examined in section 2.2.

3.2 Proposed metadata model

Within this subsection, we describe the design of our metadata model to manage heterogeneous data sources stored in the lake. Section 3.2.1 describes the conceptual model, which proposes the list of concepts to model the data lake. Section 3.2.2 describes the logical model, transformation rules, and a comparison with recent metadata models in terms of the proposed concepts. Thus, in order to evaluate the performance of our model, a list of evaluation measures is proposed in Section 3.2.3.

3.2.1. Conceptual model

In this part, we illustrate our conceptual model, which provides the entire process of data modeling. It aims to define the list of concepts that will be used by our model to establish the relationship between them. Thus, the type of this model describes what the system contains. In EMEMODL, we propose a list of concepts and their formalization that will be used for data modeling. However, we used the concept of a data entity as being generic to represent the different data sources. Indeed, several concepts have been proposed in the literature to reference data sources such as data units, datasets, raw data, etc. To standardize them, we have adopted the same concept used by goldMEDAL [28] to denote raw data. It is formalized, as we can see in Eq. (1). Therefore, each data entity must be scaled to support data scalability, which refers to a system's ability to evolve in order to process data of various types in large sizes and at ever quicker speeds.

$$DE = \{di / i \in N\} \quad (1)$$

Eventually, we introduced the categorization concept to denote a set of entities that are grouped according to common points of intersection, such as similar properties between them. We can formalize this concept as follows:

$$G = \{g_i / i \in N\} \quad (2)$$

With:

$$g_i = \{Cluster_{ij} / j \in N \text{ and } Cluster_{ij} \subseteq DE\} \quad (2.1)$$

Afterward, we have defined the concept of grouping as being a collection of homogeneous groups or clusters. It is defined by the formalization below:

$$Gr = \{(g_i, g_j)/g_{i,j} \subseteq G\} \quad (3)$$

However, we cannot talk about all of these concepts without going through the process of defining the relationship between them and the process applicable to the data in order to make them interoperable. These last two concepts, data preparation process and relationship, are formalized as follows:

$$DP = \{(Process_{inp}, Process_{out})/Process_{inp,out} \subseteq DE\} \quad (4)$$

And:

$$Rel: DE, G \rightarrow DE, G \quad (5)$$

Next, Table 3 summarizes the symbols and notations considered in the conceptual model to facilitate its reading.

Further, to illustrate these concepts well and make them more understandable, we have designed a model based on UML class diagrams, as shown in Fig. 2. Thus, it is relevant to point out that we have introduced the concept of scalability. Yet, as we can notice in the conceptual diagram each data entity is linked to a set of metadata entity in which it is characterized by a set of attributes, among the set of attributes we find the size which designates the scalability to grow, evolves with future needs which are directed by data-driven. Data is constantly multiplying; Thinking about scalability from the start

Table 4. Metadata models assessment in terms of concepts

Models	Concepts					
Ravat and Zhao [11]	Dataset	Grouping	Relationship	Process	Multiple zone granularity	-
MEDAL [12]	Version, Presentation	Grouping	Similarity link	Update, Transformation	-	-
HANDLE [27]	Data elements, Data units	Categorization, Zone indicator	Link and/or Relationship	-	-	-
goldMEDAL [28]	Data entity	Grouping	Link	Process	Granularity zone	-
EMEMODL's	Data entity	Grouping	Relationship	Data preparation	Granularity	Scalability

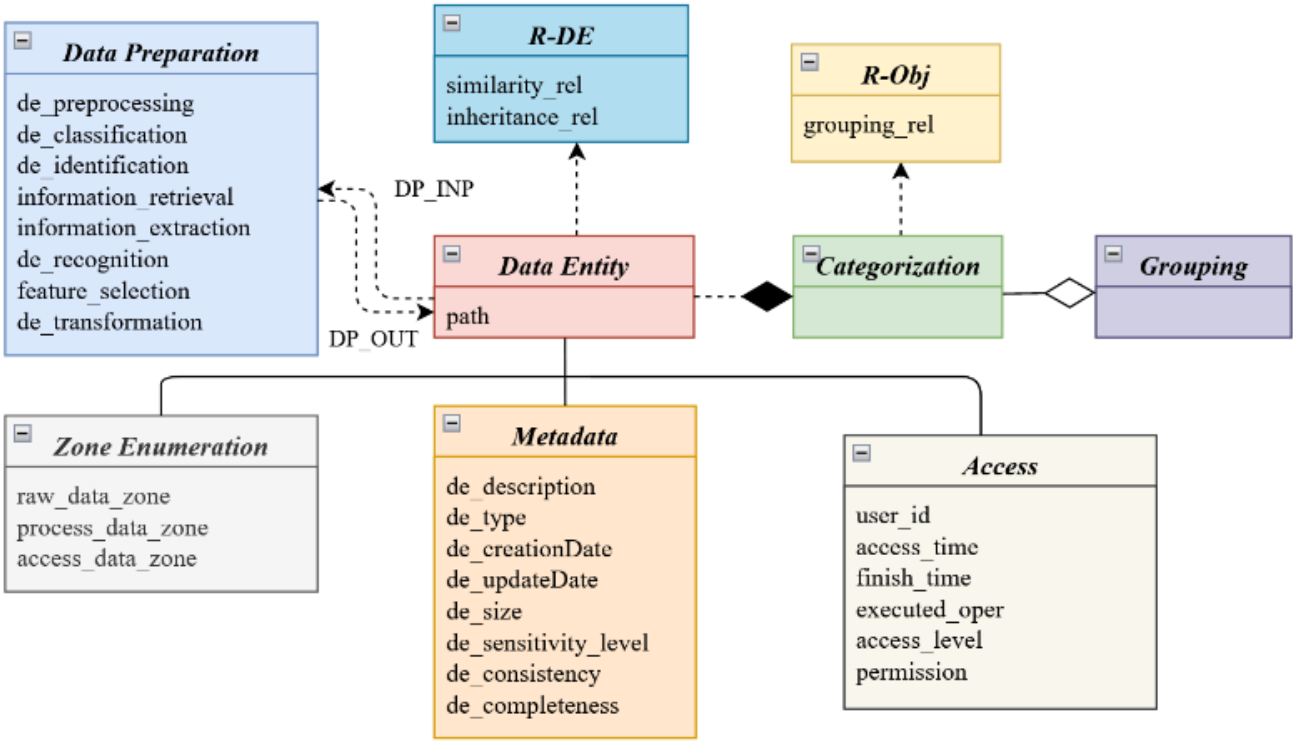


Figure. 2 The UML class diagram for the conceptual metadata model

of a data processing project is very beneficial and adapts with big data projects.

From the conceptual metadata model designed above, data lake users can effectively understand, search, and navigate different data and their relationships. Indeed, the metadata collected from datasets are stored in the "metadata" class and undergoes a pre-treatment process. To enable semantics between heterogeneous data sources, the class "relationship" was created for this need. Based on the "Zone_Enumeration" class, datasets can be assigned to one of the three zones: raw data, process, or access zone.

3.2.2. Logical model

In this sub-subsection, we present a logical model that describes the data in as much detail as possible, regardless of how the data will be physically implemented. Yet, we evaluate our model by comparing it with the complete metadata models in terms of concepts. When a particular concept has no equivalent, in this case, it will be marked by a dash. As illustrated in the Table 4, through the six concepts proposed by our metadata model, we will be able to generalize the different concepts suggested by other models, like [11, 12, 27, 28]. Indeed, for instance, the concept of data entity represents the basic unit of our

model. It is representative, standardized, and flexible in terms of multiple data granularity.

Similarly, for the other concepts such as grouping, relationship, data preparation, multiple data granularity, and scalability. Furthermore, establishing extensible terminologies and determining the relationship between these different concepts is considered one of the major requirements in modeling a metadata model. Without forgetting the scaling of the data, which represents the main asset that favors EMEMODL compared to the proposed models.

However, when working with a relational database, the logical data model refers to a relational model, abbreviated LRDM, which stands for logical relational data model. On the other hand, when working with an oriented graph database, this refers to a graphic model, denoted by LGDM, which stands for logic graph data model. Nonetheless, it is crucial to recognize that the conceptual model cannot be directly implemented in a database; therefore, it is mandatory to transform the data model. Faced with this finding, we define in Table 5, the passage of the transformation rules from the conceptual data model to the logical data model.

3.2.3. Evaluation metrics

The performance of RDMBS and GDBMS is assessed using the following metrics:

- Processing time for a query: a CQL and SQL queries execution time.
- Resources consumption such as RAM and CPU.

We execute each graph and SQL query ten times, and then we calculate the average memory and CPU consumption.

Table 5. Transformation rules from conceptual into logical model.

UML Element	RDBMS	GDBMS
Interface, Class	Table	Vertex
Attributes	Fields	Node properties
Simple Association	Foreign key in the table with minimum cardinalities	Edge
Composition (Or aggregation) relationship	Additional class contains primary keys of related tables	
Inheritance relationship	Foreign key in child classes (Transformation by reference)	

Table 6. Experimental machine specification.

Hardware Specifications	Software Specifications
CPU: Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz	Operating System (OS) : Windows 10
RAM: 8GB	DBMS: MYSQL & Neo4j
DISK (HDD): 1TO	Query language: SQL & CQL
GPU: NVIDIA GeForce MX250	Category: (Row & Graph) Storage engine

Table 7. TPC-H dataset description

Datasets Name	Size	Description
TPCH-SF1	100M	Consists of the base row size (several million elements).
TPCH-SF10	500M	Consists of the base row size x 10.
TPCH-SF100	1G	Consists of the base row size x 100 (several hundred million elements).

4. Experimental results and discussions

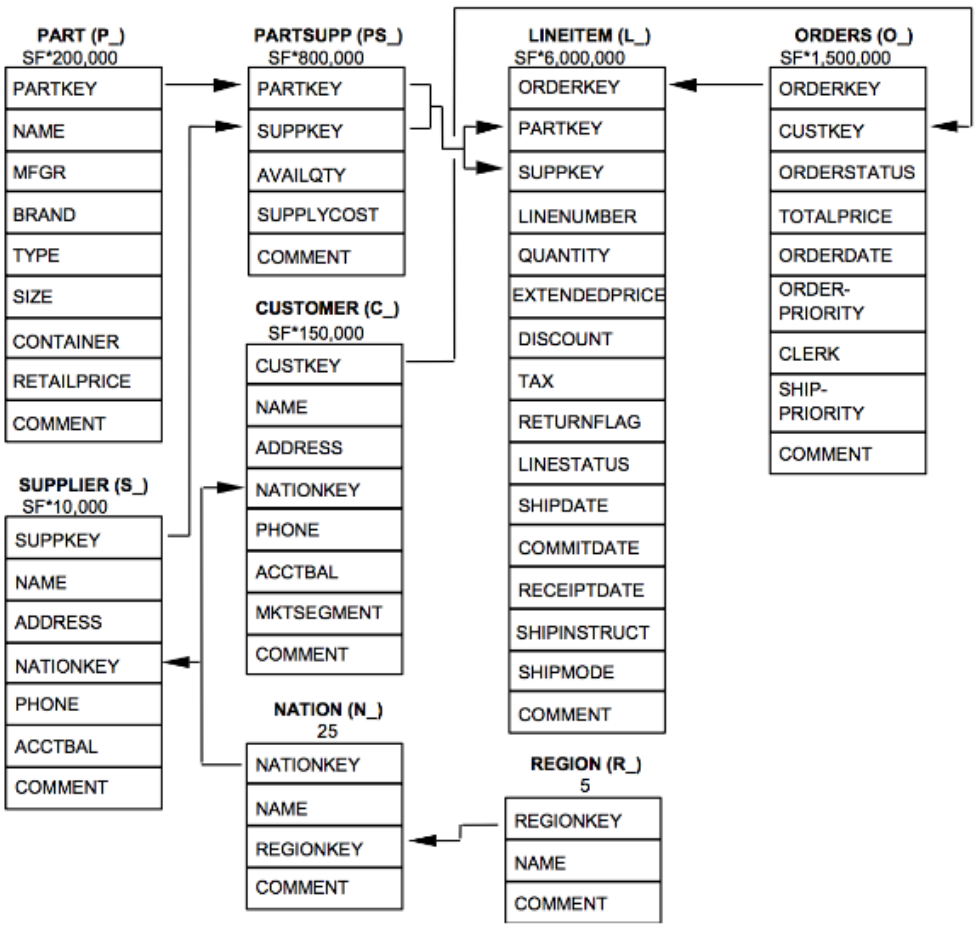
In this section, we begin by describing the experimental setup. Then, we assess EMEMODL relevance based on RDBMS and GDBMS over the TPC-H datasets under the evaluation metrics defined in Section 3.2.3. Finally, we'll discuss our findings.

4.1 Experiment setup

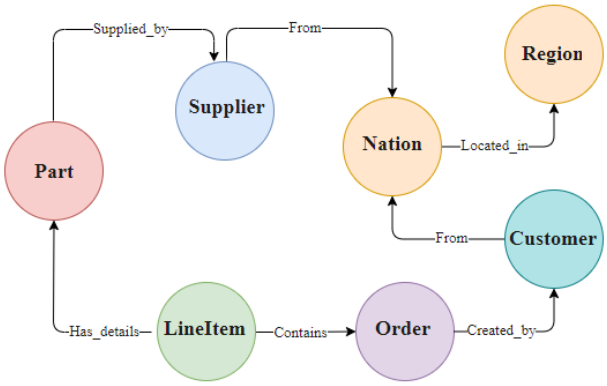
The experiments are carried out on a machine with the hardware and software specifications summarized in Table 6.

4.2 Experiment datasets

To assess the performance of our proposed metadata model, we performed the test on a suite of business-oriented ad-hoc TPC-H datasets of different sizes, varying from 100 MB to 1 GB, as we can see in Table 7. The TPC-H datasets consists of concurrent data modifications and a suite of business ad-hoc queries. The database's queries and data have been chosen for their broad industry relevance. This benchmark exemplifies decision support systems that analyze substantial amounts of data, carry out queries with a high level of complexity, and provide solutions



(a)



(b)

Figure. 3 MySQL and Neo4j TPC-H Schema

to important business concerns. Thus, it allows us to test our SQL and CQL queries with a small set of records, and then execute the same queries on a bigger set of records to test scalability and flexibility capabilities for large-scale datasets as part of showing the strength of EMEMODL. There are eight tables that make up the TPC-H dataset. The relational and graph schema are illustrated in Fig. 3.

Following Fig. 3 (a), the prefix of the table's column names is shown in the figure by the parenthesis after each table name. Thus, the arrows pointing in the direction reflect the one-to-many link between tables. Yet, the number or formula under each table name indicates the number of records in that table. Some are scaled by SF, or scale factor, which is used to identify the appropriate database based on its size.

Even though the TPC-H data model contains eight entities. In Fig. 3 (b), we have only defined seven node labels (customer, supplier, nation, region,

Table 8. Mapping between MySQL and Neo4j Queries

	MySQL RDBMS (SQL)	Neo4j GDBMS (CQL)
LDD	CREATE TABLE table_name (property type [constraint], …);	CREATE (node:label {key: value});
	DROP TABLE table_name;	MATCH (node:label {key: value}) DELETE node;
	ALTER TABLE table_name [ADD, MODIFY, or DROP](property type [constraint], …);	MATCH (node:label {properties}) SET node.property1 = node.property2 RETURN node
LMD	INSERT INTO table_name VALUES(value1, …);	MATCH (node1:label1), (node2:label2) WHERE node1.property1 = valueX AND node2.property2=valueY CREATE (node1)-[r: relation_type]->(node2) RETURN node1, node2;
	DELETE FROM table_name WHERE Condition;	MATCH (node:label{properties}) DETACH DELETE node;
	UPDATE table_name SET attribut = value WHERE Condition;	MATCH (node {key: value}) SET node:newLabel RETURN node.key, labels(node) AS labels
LID	SELECT column1, ... FROM table_name WHERE key=value ORDER BY column1, ...;	MATCH (node:label) WHERE node.key = value RETURN node.property AS propertyLable ORDER BY propertyLable;

order, lineitem, and part) because the Partsupp entity has been replaced by a relationship. Thus, we have defined six relationship types (Contains, Supplied_By, Has_details, Created_By, Located_in, and From) to capture all of the relationships between the various entities in the TPC-H. The Has_details relationship type in the TPC-H data model substitutes the Partsupp entity, and we store the availqty and supplycost values from the Partsupp entity inside the relationship.

4.3 Experiment results analysis

In this section, we summarize the experimental findings and analyze them. Based on the experimental setup and datasets, we perform four experimental scenarios. Before starting the different performance scenarios of our model, we have summarized some of the correspondence queries between the relational and graphical models, as illustrated in Table 8.

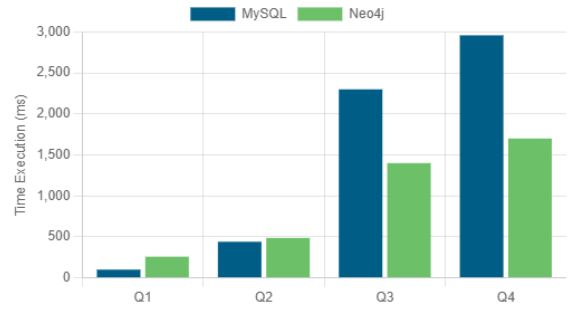
After having defined the correspondence between the SQL and CQL query languages of the two database management systems, we have chosen four queries from the TPCH benchmarks to test the scalability performance of our metadata model in MySQL, which ensures the requesting of data via an easy-to-implement navigation language, thus ensuring a high-security level. In addition, we implemented the queries on Neo4j, which ensures scalability and flexibility.

To test our hypothesis, we setup an experiment on three different TPC-H datasets, using MySQL and Neo4j. For the preliminary evaluation, four different queries are run on the objects of the mentioned schema using both tools, i.e., two SGBDs. The queries are listed in Table 9. The queries in MySQL were implemented using SQL, whereas the queries in Neo4j were implemented using CQL. The results of our experiment assess the performance of our queries in terms of the evaluation criteria defined in Section 3.2.3. Show the mean response time of MySQL and Neo4J in milliseconds (ms) for each of the four queries across the three datasets used; also, resource consumption and the ability of the model to be scalable and flexible.

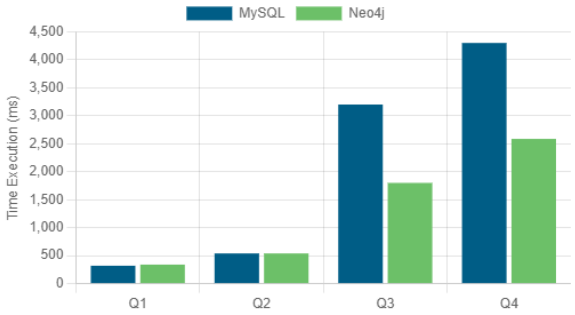
Experiments have shown that Neo4j performs well in most queries for the TPC-H data set in a reasonable amount of time, especially for large datasets, i.e., scaling. Fig. 4 shows the time taken by each query in milliseconds. In simple queries cases such as query 1 and 2, MySQL outperforms Neo4J. But, Neo4J outperforms MySQL when multiple joins between tables are involved. Indeed, query 4 is likely the most complex in terms of data that needs to be related, as it requires five joins across six tables as well as an ordering operation (order by clause), indicating that data size has an impact on query performance when queries are complex. Because joins are known to be expensive operations in relational databases, we can expect performance to suffer as data (table) size grows in the presence of multiple joins. Similarly, for query 3, MySQL performs worse than Neo4j, and this query also has four joins and an ordering operation. Thus, Neo4j eliminates expensive computation like

Table 9. Tested Queries

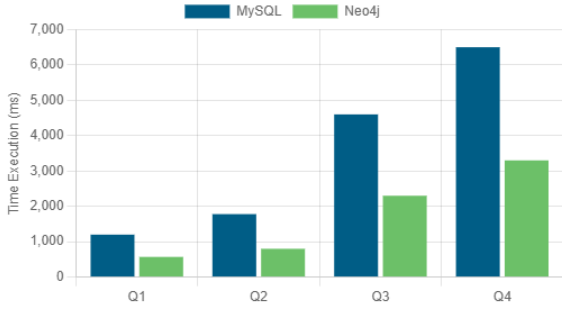
Query ID	Business Queries	Query With Join	Query Without Join	No# of tables joins	uses primitive clauses (order by, group by, ...)
Q1	**Revenue Change Forecasting:** quantifies the amount of revenue increase that would have occurred if certain companywide discounts were eliminated in a given percentage range in a given year		X		
Q2	**Pricing Summary Report:** provides a pricing summary report for line items shipped in a specified date		X		X
Q3	**Supplier with the Lowest Cost:** determines which supplier should be chosen to place an order for a given part in a given region at the lowest possible cost	X		5	X
Q4	**Volume of Local Suppliers:** lists the revenue volume generated by local suppliers	X		6	X



(a)



(b)



(c)

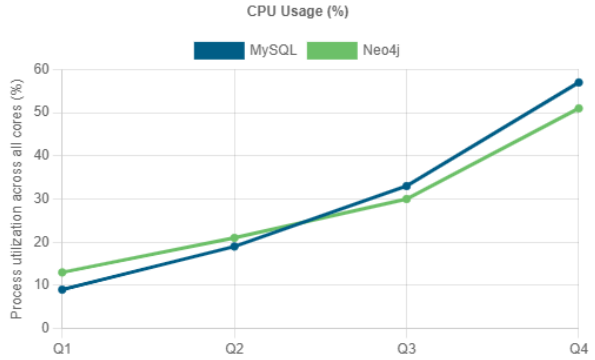
Figure. 4 Queries execution time (ms): (a) TPCH_1 dataset, (b) TPCH_10, and (c) TPCH_100

join operations. Furthermore, increasing the number of join tables implies an exponential decrease in execution time for neo4j. When the dataset size grows, the graph database outperforms relational databases.

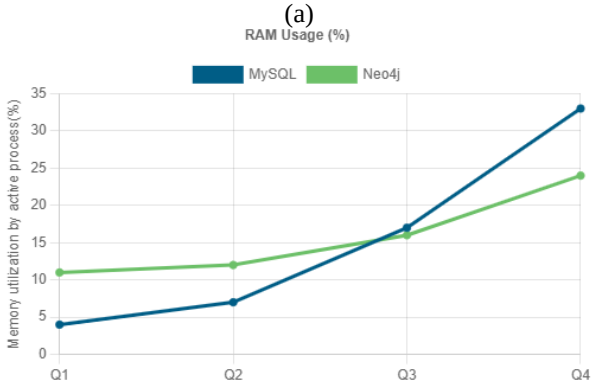
Fig. 5 (a) and 5 (b) show the average CPU and main memory usage rates. From the figure, we can observe that Neo4j also outperforms MySQL in terms of CPU and main memory usage. When the data size increases, the RDBMS requires a lot of resources to consume. Furthermore, it remains inefficient in terms of resource consumption.

4.4 Discussion

Neo4j database management systems typically process data faster than relational ones, due mainly to their simpler data models and the fact that they are not required to commit to certain restrictions imposed by the ACID properties. Furthermore, relationship modeling is not appropriate in relational databases; because, they use foreign keys to link one piece of information to another. Further, Neo4j is used to efficiently manage and process large amounts of unstructured data. As the size of the dataset grows, graph databases outperform relational databases. Due to constraints, a relational database follows a rigid schema structure and it is difficult to manage changes when dealing with multiple tables. Certainly, graph databases are a good choice for applications that involve a large number of data relationships. The retrieval times for big data queries give us a conclusion that graph databases are suitable for EMEMODL.



(a)



(b)

Figure. 5 Resource consumption for query execution (%): (a) CPU Usage and (b) RAM Usage

Our contribution demonstrates the importance of high flexibility and better scalability as major features, giving more extensibility and genericity, which is not supported by recent models such as [9-13, 26-28]. Although the NoSQL paradigm will never completely replace the relational paradigm, it may become a better option for projects that require scalability and work with unstructured data. NoSQL databases are horizontally scalable databases that can be expanded by adding new servers to a cluster environment. In the cluster, commodity hardware is used to store big data. When selecting a database management system for our software application, we must consider the DBMS's scalability. Relational databases use vertical scalability, which allows the existing node's storage and processing capabilities to grow as the volume of data does. This type of scalability is costly due to increased hardware failure risk and hardware costs for future upgradability. As a result, the overall implementation cost will rise as data volumes increase. Whereas, NoSQL databases use horizontal scalability, which allows the system to grow by adding more nodes for data storage and processing power when the volume of data is high. As a result, horizontal scalability is a less expensive solution than vertical scalability. Thus, NoSQL databases support auto-sharding by distributing data across multiple servers, which improves database performance.

Additionally, the ability to change the database schema during the development or evolution of a software application is not a feature provided by all DBMS. Then, the static database schema for SQL databases must be pre-defined before data injection. Therefore, modification of the database schema should be considered precisely, because frequent changes may result in performance degradation, service failure, or even call for maintenance and further investments to adjust application components. However, NoSQL databases do not require a pre-defined schema because they have a dynamic one. Thanks of their dynamic schema design, NoSQL databases can quickly adapt to changes in the data structure. The data structure is another issue that affects the database's flexibility. SQL databases only handle well-structured data. As data volume grows, this can have an impact on database performance. While NoSQL databases can handle all types of data, including structured, semi-structured, and unstructured data; they are used for agile and scalable environments that are constantly developing and evolving due to their data modeling. When compared to relational databases, NoSQL databases have a more flexible model, making it easier to organize large amounts of data in various formats and with flexible growth over time. NoSQL is the ideal solution for large datasets, the need for constant schema change, and the need for performance and flexibility.

5. Conclusion

In this contribution, we investigate the limits of recent metadata models by introducing EMEMODLs as a new extensible, generic metadata model for data lakes that supports data flexibility and scalability to fit the challenges of big data projects such as volume, variety, value, veracity, etc. Indeed, our model is based on a set of features and a list of concepts. Moreover, the proposed concepts encompass almost all the concepts suggested by the literature, as shown in subsection 3.2.2. Eventually, EMEMODLS supports all the ideal features identified in the comparison between the different metadata models. If this indicates something, it means the extensibility and genericity aspects induced by our model. However, analyzing the issue of different data formats and their effectiveness in storing and processing requires a database with a flexible and scalable DBMS. Thus, Neo4j proved to be one of the best solutions for dealing big data with an adjustable schema, in contrary to MySQL, which requires a

predefined schema. Therefore, Neo4j is less expensive in computation for the case of joins. It is capable of handling large amounts of data at a low cost and with minimal overhead. Furthermore, GDBMS is more suitable for EMEMODL. Finally, to the best of our knowledge, designing a metadata model for data lakes remains a very active research topic open to all researchers. An essential future perspective concerns comparison with other NoSQL database types, such as column-oriented HBase, document-oriented Cassandra, and key-value databases like Redis.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

"Conceptualization, MC and AE; methodology, MC and AE; software, MC; validation, AE; formal analysis, AE; resources, MC and AE; data curation, MC; writing—original draft preparation, MC; writing—review and editing, AE; visualization, MC; supervision, AE".

References

[1] I. Suriarachchi and B. Plale, "Crossing Analytics Systems: A Case for Integrated Provenance in Data Lakes", In: *Proc. of International Conf. On eScience*, 2017.

[2] M. A. Hussein and E. K. Hamza, "Secure Mechanism Applied to Big Data for IIoT by Using Security Event and Information Management System (SIEM)", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 6, pp. 667-681, 2022, doi: 10.22266/ijies2022.1231.59

[3] C. Quix and R. Hai, "Data Lake", In: *Proc. of International Conf. on Big Data Technologies*, 2018.

[4] C. Madera and A. Laurent, "The next information architecture evolution: the data lake wave", In: *Proc. of International Conf. on Management of Digital EcoSystems*, pp. 174-180, 2016.

[5] N. G. Miloslavskaya and A. Tolstoy, "Big Data, Fast Data and Data Lake Concepts", *International Journal of Procedia Computer Science*, Vol. 88, pp. 300-305, 2016.

[6] B. Inmon, "Designing the Data Lake and Avoiding the Garbage Dump", *Academic Press*, pp. 169-231, 2016.

[7] P. P. Khine and Z. S. Wang, "Data lake: a new ideology in big data era", In: *Proc. of International Conf.* On *ITM Web Conferences*, Vol. 17, pp. 203-225, 2018.

[8] P. Sawadogo and J. Darmont, "On data lake architectures and metadata management", *International Journal of Intelligent Information Systems*, Vol. 56, pp. 97-124, 2021.

[9] I. Megdiche, F. Ravat, and Y. Zhao, "Metadata Management on Data Processing in Data Lakes", In: *Proc: of International Conf. on Theory and Practice of Computer Science*, 2021.

[10] J. M. Hellerstein, V. Sreekanti, J. E. Gonzalez, J. Dalton, A. Dey, S. Nag, K. Ramachandran, S. Arora, A. Bhattacharyya, S. Das, M. Donsky, G. Fierro, C. She, C. Steinbach, V. Subramanian, and E. Sun, "Ground: A Data Context Service", In: *Proc. of International Conf. on Innovative Data Systems Research*, 2017.

[11] F. Ravat and Y. Zhao, "Metadata Management for Data Lakes", In: *Proc. of International Conf. on Databases and Information Systems, Communications in Computer and Information Science*, pp. 37-44, 2019.

[12] P. Sawadogo, E. Scholly, C. Favre, E. Ferey, S. Loudcher, and J. Darmont, "Metadata Systems for Data Lakes: Models and Features", In: *Proc. of International Conf. on BI and Big Data Application*s, pp.440-451, 2019.

[13] C. Diamantini, P. L. Giudice, L. Musarella, D. Potena, E. Storti, and D. Ursino, "A New Metadata Model to Uniformly Handle Heterogeneous Data Lake Sources", In: *Proc. of International Conf. on Databases and Information Systems*, pp. 165–177, 2018.

[14] A. Beheshti, B. Benatallah, R. Nouri, and A. Tabebordbar, "CoreKG: a knowledge lake service", In: *Proc. of International Conf. on The VLDB Endowment*, Vol. 11, No. 12, pp. 1942-1945, 2018.

[15] M. Cherradi and A. E. Haddadi "Grover's Algorithm for Data Lake Optimization Queries", *International Journal of Advanced Computer Science and Applications*, Vol. 13, No. 8, pp. 568-576, 2022.

[16] R. Hai, C. Quix, and D. Wang, "Relaxed Functional Dependency Discovery in Heterogeneous Data Lakes", In: *Proc. of International Conf. on Computer Science*, pp. 225-239, 2019.

[17] J. Riley, "Understanding metadata", *Information Standards Organization*, Vol. 2, 2017.

[18] H. Mehmood, E. Gilman, M. Cortés, P. Kostakos, A. Byrne, K. Valta, S. Tekes, and J. Riekki, "Implementing Big Data Lake for Heterogeneous Data Sources", In: *Proc. of*

International Conf. on Data Engineering, pp. 37-44, 2019.

[19] M. Manur, A. K. Pani, and P. Kumar, "Big Data Analysis Using Fuzzy Deep Convolution Network Based Model for Heart Disease Classification", *International Journal of Intelligent Engineering and Systems*, Vol. 14, No. 2, 2021, doi: 10.22266/ijies2021.0430.13.

[20] A. Alserafi, A. Abelló, O. Romero, and T. Calders, "Towards Information Profiling: Data Lake", In: *Proc. of International Conf. on Data Mining Workshops*, pp. 178-185, 2017.

[21] M. Cherradi, A. E. Haddadi, and H. Routaib, "Data Lake Management Based on DLDS Approach", In: *Proc: of International Conf. on Networking, Intelligent Systems and Security*, pp. 679-690, 2022.

[22] M. Cherradi and A. E. Haddadi, "Data Lakes: A Survey Paper", In: *Proc. of International Conf. On Innovations in Smart Cities Applications*, Vol. 5, pp. 823-835, 2022.

[23] M. R. Llave, "Data lakes in business intelligence: reporting from the trenches", *International Journal of Procedia Computer Science*, Vol. 138, pp. 516-524, 2018.

[24] D. Loshin, "Data governance for big data analytics", *Morgan Kaufmann, 1st Edition*, 2013.

[25] M. Cherradi and A. E. Haddadi, "A Scalable framework for data lakes ingestion", *International Journal of Procedia Computer Science*, Vol. 215, pp. 809-814, 2022.

[26] C. Quix, R. Hai, and I. Vatov, "GEMMS: A Generic and Extensible Metadata Management System for Data Lakes", In: *Proc. of International Conf, on Advanced Information Systems Engineering*, pp. 130-136, 2016.

[27] R. Eichler, C. Giebler, C. Groger, H. Schwarz, and B. Mitschang, "HANDLE - A Generic Metadata Model for Data Lakes", In: *Proc. on International Conf. on Big Data Analytics and Knowledge Discovery*, pp. 7-16, 2020.

[28] E. Scholly, P. Sawadogo, P. Liu, J. A. E. Oviedo, C. Favre, S. Loudcher, J. Darmont, and C. Noûs, "Coining goldMEDAL: A New Contribution to Data Lake Generic Metadata Modeling", In: *Proc. of International Conf. on Design, Optimization, Languages and Analytical Processing of Big Data*, pp. 31-40, 2021.