Creation of column-oriented NoSQL databases automatically in Big Data environments and its impact on energy consumption.

Fouad ELOTMANI[1], *Redouane* ESBAI[1], and *Mohamed* ATOUNTI[1]

[1]University Mohammad Premier, Oujda, Morocco

Abstract. This study investigates the automatic creation of column-oriented NoSQL databases in Big Data environments and their impact on energy consumption. Traditional row-oriented databases face limitations in handling large volumes of data, resulting in slower query response times and energy inefficiencies. In contrast, column-oriented NoSQL databases store data in columns, enabling efficient compression, retrieval, and query processing. Innovative techniques are employed to automatically create these databases, optimizing performance and minimizing manual intervention. Storing data in a columnar format reduces storage requirements and power consumption while improving data locality and reducing I/O operations. This study emphasizes the benefits of adopting column-oriented NoSQL databases, including improved performance, scalability, and energy efficiency in Big Data environments.

Index Terms— Big DATA**,** SQL relational database, NoSQL, energy efficiency, The transformation rules, BigData, CQL

1 Introduction

As the scale and complexity of Big Data continue to expand rapidly, efficient data management strategies are crucial to ensure optimal performance and reduced energy consumption. This study focuses on the automatic creation of column-oriented NoSQL databases in Big Data environments and explores their potential impact on energy consumption. Traditional row-oriented databases face limitations when handling vast volumes of data due to their inherent structure, leading to increased query response times and energy inefficiencies. In contrast, column-oriented NoSQL databases leverage a unique data organization approach, storing data in columnar fashion rather than traditional row-wise manner[1].

Big data will play a crucial role in facilitating the implementation of the Global Energy Internet (GEI). Firstly, the GEI entails a vast array of measurements, monitoring devices, and control systems, resulting in the generation of substantial data throughout the entire energy production, transmission, transaction, and consumption processes. Additionally, the GEI necessitates extensive data mining due to its reliance on renewable energy sources and

the active participation of consumers. Given the uncertainties and complexities associated with the GEI, traditional physical model-based approaches are inadequate. In contrast, big data analytics, as a data-driven methodology, proves highly effective. By leveraging big data analysis, scientific predictions can be made regarding all aspects of energy production, distribution, transformation, and consumption. This approach enables both decentralized and centralized coordination of energy management, while also facilitating the identification of potential risks at each stage.

The automated creation process optimizes database performance and minimizes manual intervention, ensuring faster deployment and reducing the burden on data management teams. The impact of utilizing column-oriented NoSQL databases on energy consumption is a key aspect of this study. By storing data in a columnar format, these databases can enhance energy efficiency in several ways. Firstly, the columnar data organization enables efficient data compression techniques, resulting in reduced storage requirements and lower power consumption. Secondly, the query execution process benefits from improved data locality and reduced I/O operations, which contribute to decreased energy usage. These combined effects can lead to significant energy savings in Big Data environments[2]. The findings of this study provide valuable insights into the benefits of automatically creating column-oriented NoSQL databases in Big Data environments. The adoption of such databases can contribute to improved performance, enhanced scalability, and reduced energy consumption. As organizations strive to optimize their data management processes while minimizing environmental impact, the automatic creation of column-oriented NoSQL databases emerges as a promising approach in the era of Big Data[3].

This paper provides an overview of the methods used to transform from a UML class diagram model to the column-oriented NoSQL database, referred to as MDA approach[4]. First, the related works are discussed in the second section. The third section provides an explanation of the MDA approach, followed by the fourth section which examines the structure of NoSQL databases, of which column-oriented is one. Afterwards, in the fifth section, the source and target meta-models are presented. The sixth section focuses on the transformation process from UML class diagram model to the column-oriented NoSQL database, which is demonstrated by both model-to-model (M2M) and model-to-text (M2T) transformation. Finally, the last section summarizes the paper and presents possible further directions of the research.

2 Related works

Recent years have seen a surge in research on MDA (Model-driven Architecture) and the process of transforming traditional relational databases into a NoSQL model. The most relevant are : [6-9]:

This paper [5]evaluates the energy efficiency of NoSQL databases, specifically MongoDB and Apache Cassandra, in Big Data applications. It presents experimental results and discusses the impact of different factors on energy consumption.

This paper[6] investigates the energy consumption of NoSQL databases, including MongoDB, Cassandra, and HBase, in Big Data environments. It presents a comparative study and analyzes the energy efficiency of these databases.

This paper [7]proposes an energy-efficient NoSQL data management framework for Big Data workloads. It introduces techniques such as workload-aware data partitioning and consolidation to reduce energy consumption. Experimental results demonstrate the effectiveness of the proposed framework.

This research[8] investigates energy-efficient backup and recovery mechanisms specifically designed for column-oriented NoSQL databases. The authors propose a novel approach that intelligently schedules backup operations based on energy availability and workload patterns, reducing energy consumption during data backup and recovery processes.

This article[9] highlights the increasing development of the Internet of Things (IoT) and big data, which has led to the adoption of non-relational (NoSQL) databases to address workload complexity and scalability issues. However, the performance, scalability, and availability of NoSQL databases face challenges as workloads grow. Energy consumption by NoSQL databases has become a concern due to rising energy costs and environmental sustainability. Despite NoSQL's popularity, there is limited understanding of its energy footprint, and comprehensive studies analyzing energy consumption are lacking. This article presents a comprehensive survey on energy consumption analysis of NoSQL and discusses the limited proposals to reduce energy consumption. It also outlines research opportunities for improving energy conservation in NoSQL systems.

3 Column-Oriented NoSQL Database

Column-oriented NoSQL databases are a type of database system that store and manage data in a columnar format. Unlike traditional relational databases, which organize data in rows, column-oriented databases store data vertically, with each column stored separately. This allows for efficient storage and retrieval of specific columns, making them ideal for analytical workloads and data-intensive application[10].

In a column-oriented NoSQL database, data is grouped together based on column values rather than individual rows. This design provides advantages for analytical queries that typically involve aggregating data from specific columns. By storing data in a columnar format, these databases can perform operations like data filtering, aggregation, and compression more efficiently, resulting in faster query execution times[11].

The column-oriented approach offers several benefits, including improved data compression, as columnar storage allows for better compression ratios compared to row-oriented databases. This leads to reduced storage requirements and potentially lower costs. Additionally, column-oriented NoSQL databases can provide high performance for read-heavy workloads, as retrieving specific columns can be done more efficiently, even when dealing with large volumes of data[12].

Another advantage of column-oriented NoSQL databases is their suitability for big data analytics. With the increasing volume and complexity of data, these databases excel at handling large-scale analytical queries by leveraging distributed computing architectures

and parallel processing. They can scale horizontally by adding more nodes to the system, accommodating the growing demands of data-intensive applications.

Overall, column-oriented NoSQL databases offer a specialized approach to data storage and retrieval, optimizing performance for analytical workloads and data-intensive applications. By leveraging columnar storage, these databases provide efficient data compression, faster query execution, and scalability for handling big data analytics[13].

In this paper, the focus is on the principal aspects of a column-oriented database such as Cassandra. Cassandra is a partitioned row store, which means that it can distribute data across multiple machines. Rows are organized into tables with a required primary key. This is different from a table-based relational database in that one can omit columns, or add arbitrary columns at any time[14].

4 Source and Target Meta-Models

We opted for the modeling and template approaches for our MDA approach to generate a column-oriented NoSQL database. These approaches involve the use of two meta-models; a source meta-model and a target meta-model. In this section, we present the various meta-classes which form the UML class diagram, the source meta-model, and the column-oriented NoSQL target meta-model.

4.1 UML source Meta-model

The figure 1 outlines a simplified UML source meta-model based on packages that consist of data types and classes. These classes are comprised of properties that are marked by multiplicities (lower and upper bounds), and they also include operations that have typed parameters.

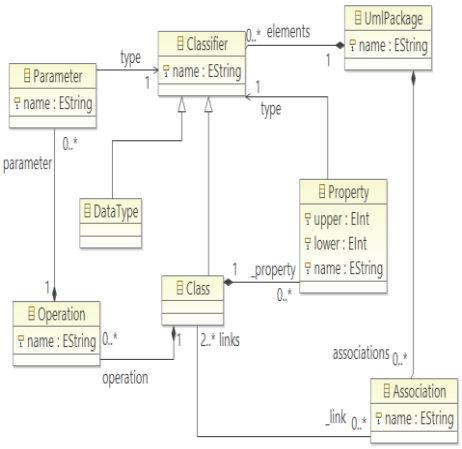

Fig. 1. Simplified UML source meta-model

4.2 Column-Oriented target meta-model

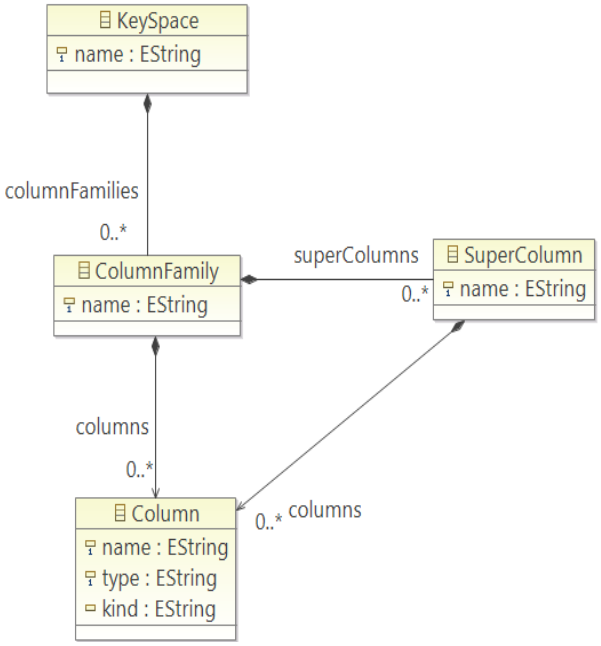

Fig. 2. Simplified column-oriented target meta-model

The database is stored in a single Keyspace by default, which is composed of column-families. Every column-family has its own distinct identifier, known as "PrimaryKey", and consists of a set of columns or super-columns that must be declared when the schema is created.

5 The Process of Transforming UML Source Model to column-oriented target code

In order to create the column-oriented database, we first developed ECORE models that correspond with our source and target meta-models. This required multiple model transformations. Our implementation of the M2M transformation algorithm (see section 6.1was) done using the QVT Operational Mappings language. Afterwards, we used the Acceleo[15] language for the second M2T transformation.

5.1 The Transformation Rules M2M

The SQL to UML transformation uses a UML type model as an input, and produces a column-oriented database model in output.The first transformation rule maps the UML package elements to the Keyspace type elements of the column-oriented database. The second rule transforms each UML class and association into a family of columns, creating

columns and referential links for each column-family. This involves transforming each class property into a column, making sure to assign names and types to the various columns.

UML2NoSQL.qvto

```
modeltype UML uses "http:///umlMM.ecore";
modeltype NoSQL uses "http://nosqlmm/1.0";
transformation UML2NoSQL(in umlModel: UML, out noSQLModel:NoSQL);
main() {
    umlModel.objects()[UmlPackage]->map UmlPackage2keySpace();
}
mapping UmlPackage::UmlPackage2keySpace () : KeySpace {
    name:= self.name;
    columnFamilies:=umlModel.objects()[Class] ->
            map classToColumnsFamily();
    columnFamilies+=umlModel.objects()[Association] ->
            map AssociationToColumnsFamily();
}
mapping Class::classToColumnsFamily():ColumnFamily{
name:=self.name;
columns:= object Column {
    name:= "id"+self.name;
    type:="uuid";
    kind:="PrimaryKey";
    };
columns+= self._property -> map propertyToColumn();
}
mapping Property::propertyToColumn():Column{
    name:=self.name;
    type:= self.type.name;
}
mapping Association::AssociationToColumnsFamily():ColumnFamily{
name:=self.name;
var cs : Sequence (Column);
self.links-> forEach(ls){
                cs+=object Column {
                    name:=ls.name+".Reference";
                    type:="uuid";
                    kind:="PrimaryKey";
                }
            };
columns:=cs;
}
```

Fig. 3. M2M transformation with QVT From UML to NoSQL model

5.2 The Transformation Rules M2T

The transformation from model to code for the development of a column-oriented database like Cassandra is doable with the Acceleo transformation language. Generally, it does not pose any challenges in terms of implementation, as it simply involves writing the transformation rules in a text file. With the help of this language and the text file, one can easily create the desired code for the database.

UML2NoSQL.mtl

```
[comment encoding = UTF-8 /]
[module UML2NoSQL('http://nosqlmm/1.0', 'http:///umlMM.ecore')]

[template public generateElement(aKeySpace : KeySpace)]
[comment @main/]
[file (aKeySpace.name.concat('.cql'), false, 'UTF-8')]
create KEYSPACE  [aKeySpace.name/] WITH REPLICATION =
{ 'class' : 'SimpleStrategy', 'replication_factor' : 1 };

USE [aKeySpace.name/] ;

[for (cf:ColumnFamily|aKeySpace.columnFamilies)]
[ColumnFamilyToCQL(cf)/]
[/for]

[/file]
[/template]

[template public ColumnFamilyToCQL(cf: ColumnFamily)]
[let s:String='' ]
CREATE TABLE [cf.name/] (
[for (c:Column|cf.columns) separator(',')][c.name/] [c.type/] [/for],
PRIMARY KEY ([for (c:Column|cf.columns->
    select(f:Column|f.kind='PrimaryKey')) separator(',')]
        [c.name/]
[/for])
);
[/let]
[/template]
```

Fig. 4. M2T transformation with Acceleo to Generate a CQL code

5.3 Result

In order to validate our conversion rules, we performed several tests.

As an example, we consider a class diagram consisting of Department, Employee, and City classes (see Figure 5).

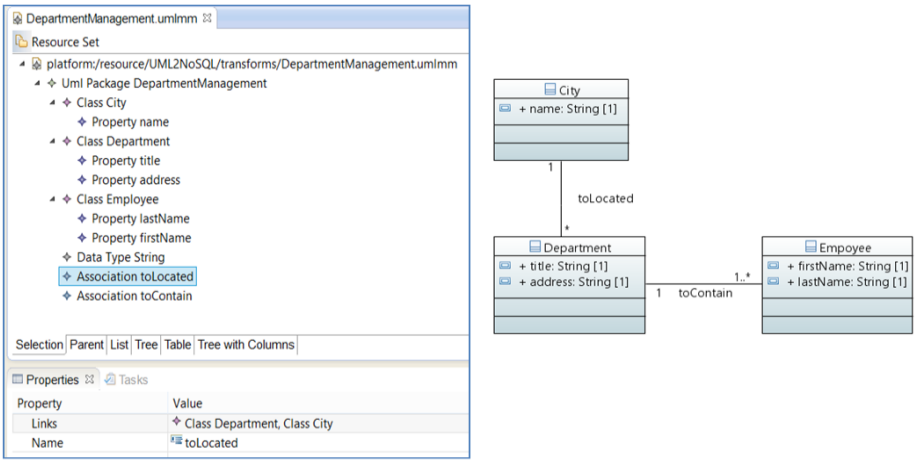

Fig. 5. UML source model: Class diagram EMF model and Class diagram instance model

After applying the transformation to the UML source model, we generated a column-oriented PSM target model (see Figure 6).

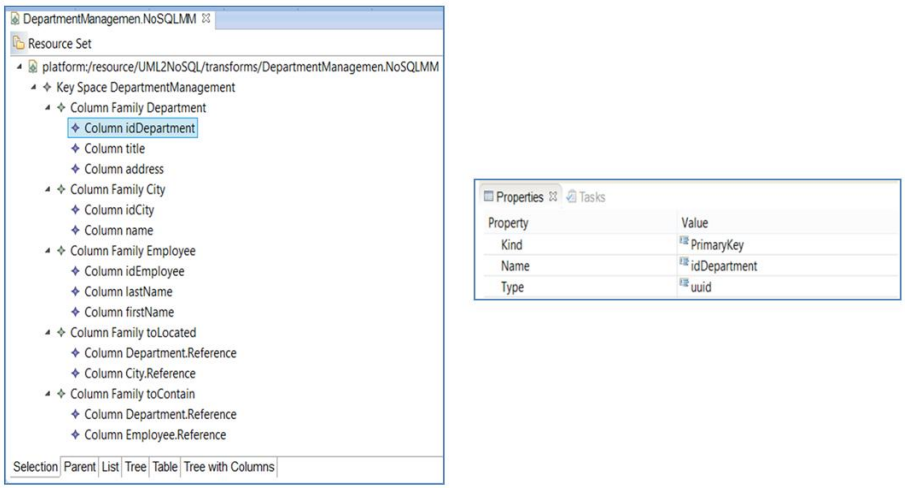

Fig. 6. Column-Oriented Cassandra PSM: Resource Set and their Properties

6 Conclusion and perspectives

the implementation of column-oriented NoSQL databases in Big Data environments has a significant impact on energy consumption. These databases offer efficient data storage and retrieval, reducing the amount of data accessed and processed during queries. This targeted retrieval approach improves energy efficiency and allows for faster data analysis. The scalability and flexibility of column-oriented databases also contribute to reduced hardware requirements and energy consumption. Overall, utilizing these databases in Big Data environments leads to lower energy consumption, improved performance, and enhanced sustainability in energy systems.

References

[1] A. Arif, T. A. Alghamdi, Z. A. Khan, and N. Javaid, "Towards Efficient Energy Utilization Using Big Data Analytics in Smart Cities for Electricity Theft Detection," *Big Data Res.*, vol. 27, p. 100285, Feb. 2022, doi: 10.1016/j.bdr.2021.100285.

[2] H. Liao, E. Michalenko, and S. C. Vegunta, "Review of Big Data Analytics for Smart Electrical Energy Systems," *Energies*, vol. 16, no. 8, p. 3581, Apr. 2023, doi: 10.3390/en16083581.

[3] O. Alotaibi and E. Pardede, "Transformation of Schema from Relational Database (RDB) to NoSQL Databases," *Data*, vol. 4, no. 4, p. 148, Nov. 2019, doi: 10.3390/data4040148.

[4] J. Xie, F. Xu, Z. Li, and X. Li, "Data Mining Method under Model-Driven Architecture (MDA)," *Secur. Commun. Networks*, vol. 2022, pp. 1–10, Mar. 2022, doi: 10.1155/2022/5806829.

[5] D. Mahajan, C. Blakeney, and Z. Zong, "Improving the energy efficiency of relational and NoSQL databases via query optimizations," *Sustain. Comput. Informatics Syst.*, vol. 22, pp. 120–133, Jun. 2019, doi:

10.1016/j.suscom.2019.01.017.

[6] T. Li, G. Yu, X. Liu, and J. Song, "Analyzing the Waiting Energy Consumption of NoSQL Databases," in *2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*, 2014, pp. 277–282, doi: 10.1109/DASC.2014.56.

[7] F. Mehdipour, H. Noori, and B. Javadi, "Energy-Efficient Big Data Analytics in Datacenters," 2016, pp. 59–101.

[8] A. H. Abed, "Recovery and Concurrency Challenging in Big Data and NoSQL Database Systems," *Int. J. Adv. Netw. Appl.*, vol. 11, no. 04, pp. 4321–4329, 2020, doi: 10.35444/IJANA.2020.11041.

[9] M. Shah, A. Kothari, and S. Patel, "A Comprehensive Survey on Energy Consumption Analysis for NoSQL," *Scalable Comput. Pract. Exp.*, vol. 23, no. 1, pp. 35–50, Apr. 2022, doi: 10.12694/scpe.v23i1.1971.

[10] N. Shehata and A. H. Abed, "Big Data With Column Oriented NOSQL Database To Overcome The Drawbacks Of Relational Databases," *Int. J. Adv. Netw. Appl.*, vol. 11, no. 05, pp. 4423–4428, 2020, doi: 10.35444/IJANA.2020.11057.

[11] M. J. Suárez-Cabal, P. Suárez-Otero, C. de la Riva, and J. Tuya, "MDICA: Maintenance of data integrity in column-oriented database applications," *Comput. Stand. Interfaces*, vol. 83, p. 103642, Jan. 2023, doi: 10.1016/j.csi.2022.103642.

[12] A. Hillenbrand, U. Storl, M. Levchenko, S. Nabiyev, and M. Klettke, "Towards Self-Adapting Data Migration in the Context of Schema Evolution in NoSQL Databases," in *2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW)*, 2020, pp. 133–138, doi: 10.1109/ICDEW49219.2020.00013.

[13] S. Bjeladinovic, Z. Marjanovic, and S. Babarogic, "A proposal of architecture for integration and uniform use of hybrid SQL/NoSQL database components," *J. Syst. Softw.*, vol. 168, p. 110633, Oct. 2020, doi: 10.1016/j.jss.2020.110633.

[14] T. Fouad and B. Mohamed, "Model Transformation From Object Relational Database to NoSQL Column Based Database," in *Proceedings of the 3rd International Conference on Networking, Information Systems & Security*, 2020, pp. 1–5, doi: 10.1145/3386723.3387881.

[15] "acceleo." [Online]. Available: https://www.eclipse.org/acceleo/.