iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** X **Month of publication:** October 2023

DOI: https://doi.org/10.22214/ijraset.2023.56359

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue X Oct 2023- Available at www.ijraset.com

Migration of Data from RDBMS to NOSQL and Possibility of Implementing a Single Query Language for NOSQL Databases

Amir Guliyev[1], Dr. Fjorda Kazazi[2]
[1]*Landau School, Baku, Azerbaijan,* [2]*UCL, Dept. of Computer Science & Brain Science, London, United Kingdom*

Abstract: ***The migration of data from RDBMS to NoSQL database systems gains increasing popularity nowadays. This research assesses the possibility of implementing an application that would function with all NoSQL databases. Furthermore, it dives into the ways of developing a Single Query Language for all NoSQL databases. Findings suggest that the potential for specialized frameworks within such a language would simplify the usage of this language by programmers. Research also suggests that the language could be bolstered by collaborative efforts between database corporations and beta testing of the language to set a pathway towards achieving this unified language. This literature signifies how important it is to consider user feedback to achieve the expected result. As we can see from previous methods explored, UnQL ceased its existence.***
Keywords: DBS, RDBMS, UNQL, NoSQL, SQL

I. INTRODUCTION

A. An Introduction To Database Systems

A Database System (DBS) is sophisticated software that facilitates data storage and retrieval through a structured mechanism, ensuring data consistency, integrity, and security. DBS is vital across diverse sectors, supporting various applications from banking transaction processing to inventory management in retail and data analytics in research.

Within the realm of DBS, Relational Database Management Systems (RDBMS) stand out, organized around the relational model introduced by Edgar Codd in 1970. Here, data is methodically arranged into tables (relations), with designs adhering to predefined schemas, ensuring data accuracy and integrity. This system supports ACID (Atomicity, Consistency, Isolation, Durability) properties, making it a prime choice for transaction-heavy applications where data consistency is paramount.

To interact with RDBMS, SQL (Structured Query Language) serves as the standardized query language, allowing for structured data requests and tasks such as data insertion, update, deletion, and retrieval.

Contrarily, NoSQL databases offer flexibility, scalability, and efficiency in distributed environments, deviating from the traditional table-based structure and the relational model. These databases are optimal for managing vast volumes of rapidly changing, diverse data types, especially in big data and real-time web applications. However, unlike RDBMS, there is no standardized query language for NoSQL databases, leading to diverse querying methods across different NoSQL systems.

This paper aims to explore the possibility of implementing such a standardized language for NoSQL, ensuring a consistent approach to database interactions, regardless of the underlying database type.

B. Detailed Description And Comparison of RDBMS And NOSQL

In the world of databases, many researchers have dived deep to see how new-age NoSQL databases stack up against traditional ones, especially when dealing with massive amounts of data. Nayak et al. (2013), for instance, aimed to describe all types of nonSQL (NoSQL) databases available nowadays in a literature review. Advantages and disadvantages of using NoSQL over the Relational Database Management System (RDBMS) and vice versa were introduced. It was concluded that NoSQL still has room for improvement as there was an attempt to create a common query language for all NoSQL databases (UnQL), which failed and ceased its existence.

Similar to Nayak et al. (2013), Jatana et al. (2012) also discussed the pros and cons of Relational Database Management Systems (RDBMS) and nonSQL (NoSQL) systems. First, researchers elaborated on the tools of relational databases only. As an example, a comparison was made between Oracle and MySQL database systems. Researchers then introduced non-relational databases, which are elaborated on further in the text.

Results were summarized, and several features that may benefit users were mentioned. For instance: in NoSQL databases, data can be inserted at any time without a schema. Besides, NoSQL databases provide consistency and scalability. However, relational databases are more efficient and quicker in terms of returning requested records, as SQL uses a single primary key for all tables.

Abdullah et al. (2015) evaluated the performances of both Relational Database Management Systems (RDBMS) and nonSQL (NoSQL) databases (MySQL and MongoDB, as follows) to determine which one of them produced higher speeds. For testing methods, in Abdullah et al. (2015) were used several methodologies, including performance benchmarking, data insertion, etc. As a result of many different tests made by Abdullah et al. (2015), they proved that nonSQL (NoSQL) databases are significantly faster compared to RDBMS databases.

Researchers have also investigated the efficiency of NoSQL with Big Data. Oussous et al. (2015) conducted a review on the efficiency of NoSQL with Big Data. The performance of both NoSQL and RDBMS databases with Big Data was evaluated and concluded that NoSQL database systems are much more suitable for dealing with Big Data. NoSQL ensures better real-time data replication at a lower cost. However, RDBMS systems are better for structured data, complex queries, high integrity, and trustful transactions. Abramova et al. (2014) also evaluated the performance of NoSQL databases with Big Data by utilizing eight workload methods. The same conclusion was reached as in Oussous et al. (2015). NoSQL promised good performance and scalability over Big Data. However, the researchers were limited on a hardware basis, as workloads could be tested on a stronger device.

While the aforementioned studies underscore the potential of NoSQL databases, it's crucial to address the challenges faced during the migration process from RDBMS to NoSQL systems.

C. Research Questions and Hypotheses

Nowadays, databases play a huge role in the technological world, as without them, massive ecosystems like Amazon, Apple, or Microsoft would cease to exist. NoSQL has started to gain popularity among industrial giants. NoSQL offers many features, like good scalability and security systems. However, there is no Single Query Language for these databases. This would give programmers and industrial giants unimaginable benefits in terms of training newcomers faster and in a more comprehensible manner. There was an attempt at creating such a Query Language. The prototype was called UnQL. This attempt, however, failed, as this project tried to implement every single aspect of different types of NoSQL databases, from graph databases to key-value databases (*Is UNQL dead?,* 2012). This idea has potential for growth, but many senior programmers from different database companies should be involved in the development process so that the language will be reliable, fast, and easy to learn.

Furthermore, migrating data from RDBMS to NoSQL from different databases is a tough process. However, with a single application developed for this purpose, it would be much easier to achieve this task. This application would include different data transfer methods that are more suitable for the context of migration. This research literature explores the possibilities of developing a single application for easy data migration from RDBMS systems to NoSQL databases and implementing a Single Query Language for NoSQL databases. Specifically, (1) is it possible to develop a single application for data migration for all NoSQL databases? Also, (2), is it possible to implement a single query language for NoSQL databases?

It was predicted that frameworks for different types of NoSQL databases could be implemented within the query language, so developers could specialize in the field they need (Schmidt et al., 2004). Besides, different database companies could collaborate on a project to create a Single Query Language for NoSQL databases with the help of user feedback from beta testing before its release, as user feedback is an important piece in the puzzle of development (Pagano & Maalej, 2013).

II. LITERATURE REVIEW

A. Migration of Data From RDBMS To NOSQL Systems

Over the past decade, various research efforts have delved into the methodologies and frameworks aimed at facilitating the migration from RDBMS systems to NoSQL systems. One of those examples is a paper written by Hanine et al. (2015). This research literature presents a methodology for migration from RDBMS to NoSQL database systems. The methodology consisted of two steps: loading the logical structure of the source database and mapping between the relational model and the NoSQL model. An application was built on JAVA to demonstrate the proposed methodology. However, one of the limitations the paper had is that the application developed by Hanine et al. (2015) doesn't work with all databases. Another automated algorithm for data migration from RDMBS to NoSQL was proposed by Mahmood (2018). This methodology uses iteration through the rows and maps data from each row to a document. A program was developed on Visual Basic .NET programming language for the proposed algorithm of migration. The paper also has limitations to some extent, as it doesn't discuss the scalability of an algorithm, and the sample database tested is rather simple.

Rocha et al. (2015) introduced a NoSQLayer framework-methodology, supporting developers in migrating automatically from RDMBS to NoSQL systems while preserving the meaning of the original database. NoSQLayer was built on Java (object-oriented language), so it could be used with other database systems as well.

Another Cloud-based NoSQL data migration method showcased by Bansel et al. (2016) consists of three main algorithms: document to graph translation, document to columnar translation, and columnar to graph translation. A special framework was created and then tested for efficiency with different databases.

Similarly, Mpinda et al. (2015) proposed a methodology for migrating from a relational database to a NoSQL (column-oriented) database. The methodology encompasses translating a source data schema into a target data schema, involving a series of translation rules. This process is outlined in three stages.

Different methods of migrating data from RDBMS to NoSQL databases were introduced by Ghotiya et al. (2017). The study highlighted various approaches and evaluated each for efficiency, advantages, and limitations. The study concluded that the adoption of NoSQL databases is essential due to growing data management needs.

Kuderu and Kumari (2016) proposed a Schema-Migration and Mapping Framework for efficiently transitioning from relational databases to NoSQL. This autonomous framework maintains the original database's semantics and allows seamless data access without modifying application code. However, there are some limitations to the approach.

Glenn (2018) discussed challenges for database administrators and developers throughout the migration process and also highlighted the best features of modeling techniques for NoSQL databases.

Methodologies utilized for assessing the software architectures of SQL and NoSQL databases were part of a systematic literature review conducted by Khan et al. (2023). The study discussed various factors and limitations of the two database types, emphasizing the need for careful consideration when choosing between SQL and NoSQL systems.

Mapanga et al. (2013) conducted a comprehensive analysis of different database management systems (DBMS). The review highlighted the strengths and weaknesses of each system based on real-world applications and theoretical models. The researchers concluded that both RDBMS and NoSQL have their unique applications and are not absolute substitutes for each other.

In a recent research paper by Padhy et al. (2023), various non-relational databases (NoSQL databases) were analyzed and compared, illuminating their respective architectures, data models, and specific features. Despite its insights, the study primarily relied on theoretical analyses without real-world experimental validation. The paper suggests that while traditional RDBMS databases still have a place, NoSQL databases provide valuable alternatives for modern applications.

B. Implementing A Single Query Language For NOSQL Databases

Nowadays, technology develops rapidly. So there is a need for developers to learn complex structures and syntaxes to catch up with trends in the technological field. However, there are many distinct query languages for different NoSQL databases that will take time to utilize. A Single Query Language for this type of database would boost the efficiency of a developer and save the time required to master the language. The research paper by Bach & Werner (2014) deeply explores this topic. It also delves into the burgeoning realm of NoSQL databases, tracing their origins from the term's inception in 1998 by Carlo Strozzi and its resurgence at a 2009 conference in San Francisco. As the modern interpretation of "NoSQL" emphasizes a departure from the traditional relational model rather than the SQL language itself, the paper explores the challenges faced in early NoSQL iterations, notably their lack of support for declarative languages akin to SQL. This absence posed hurdles for potential users accustomed to relational databases' declarative language. The researchers highlight the increasing calls for language standardization and the creation of user-friendly languages for non-relational data access. The myriad of NoSQL systems, each demanding unique programming knowledge, suggests a need for standardized interfaces. However, given the vast diversity of NoSQL solutions (key-value, column family, graph, and document databases), the authors conclude that crafting a universal standard might be challenging. Instead, a more feasible approach might involve developing standards for each database category independently. The paper serves as a comprehensive guide for those navigating the NoSQL landscape and weighing its potential against traditional RDBMS. The limitations mentioned revolve around the inherent diversity in NoSQL solutions, which complicates the creation of a unified standard.

In the quest to establish a unified query language for NoSQL, UNQL has been introduced with considerable anticipation. However, critical assessments have illuminated potential drawbacks in its design and approach. Foremost, UNQL's ambitious aim to provide a comprehensive solution across the spectrum of NoSQL—from key-value stores to document-stores and graph databases—might inadvertently dilute its specialized efficacy. This overarching approach essentially leads UNQL to predominantly function as a basic key-value access system, which, while foundational, overlooks the nuances and capabilities of more specialized NoSQL databases.

Furthermore, the absence of robust support for intricate features intrinsic to NoSQL, such as joins, paths, and sub-structures, is palpable. This one-size-fits-all approach may not be entirely advantageous, as databases with varying complexities demand tailored solutions. For instance, while a rudimentary key-value store could suffice with a fluent-interface, graph traversals or multifaceted map-reduces warrant a more bespoke approach. One of UNQL's paramount criticisms, however, is its potential oversight of NoSQL's innate advantages, particularly its adeptness at managing lists and sub-objects—areas where traditional SQL stumbles. The desideratum, as voiced by many, is for a declarative paradigm, reminiscent of SQL's clarity and user-friendliness, which UNQL in its present incarnation might not wholly encapsulate.

Name	Pros	Cons	Unique Features
N1QL (Couchbase)	- SQL-like syntax - Integrates well with JSON data	- Specific to Couchbase - Slightly steeper learning curve than raw SQL	- Natively supports JSON - Array and Object operations
MongoDB Query Language	- Schema-free - Flexible and powerful	- Unique syntax may be unfamiliar to SQL users - Less straightforward for complex JOIN-like operations	- Deeply nested document queries - Native operators for document manipulation
Hypertable Query Language	- Built for high-performance scenarios - Simple and familiar syntax for those used to SQL	- Specific to Hypertable - Less community support	- Supports real-time applications - Integrates well with big data ecosystems
Cypher Query Language	- Graph-centric - Intuitive syntax for graph traversal	- Learning curve for those new to graph databases - Specific to Neo4j	- Pattern-based syntax - Rich graph operations natively supported
CQL (Cassandra)	- Designed for distributed data - Familiar SQL-like syntax	- Some SQL features are missing	- Native support for tables and clustering - Built for scalability and fault tolerance
SPARQL	- Designed for querying RDF data - Powerful for semantic web queries	- Steeper learning curve - Specific to RDF stores	- Can pull data from diverse sources into unified views - Supports graph patterns and optional matching

Fig. 1. Table with pros, cons and unique features of different NoSQL Query Languages

III. CONCLUSIONS

The seismic shifts in database management paradigms, exemplified by the increased adoption of NoSQL databases over traditional RDBMS, have galvanized the technological world. This transition underscores the quest for scalability, flexibility, and adaptability in addressing modern-day data challenges. Amid this, the migration from RDBMS to NoSQL emerges as a significant theme, with numerous methodologies being proposed, as seen from the works of Hanine et al. (2015), Mahmood (2018), and others. These methodologies, while valuable, also bear inherent limitations, underscoring the need for comprehensive solutions that address the multifaceted challenges of migration, such as scalability, data consistency, and heterogeneity of data types.

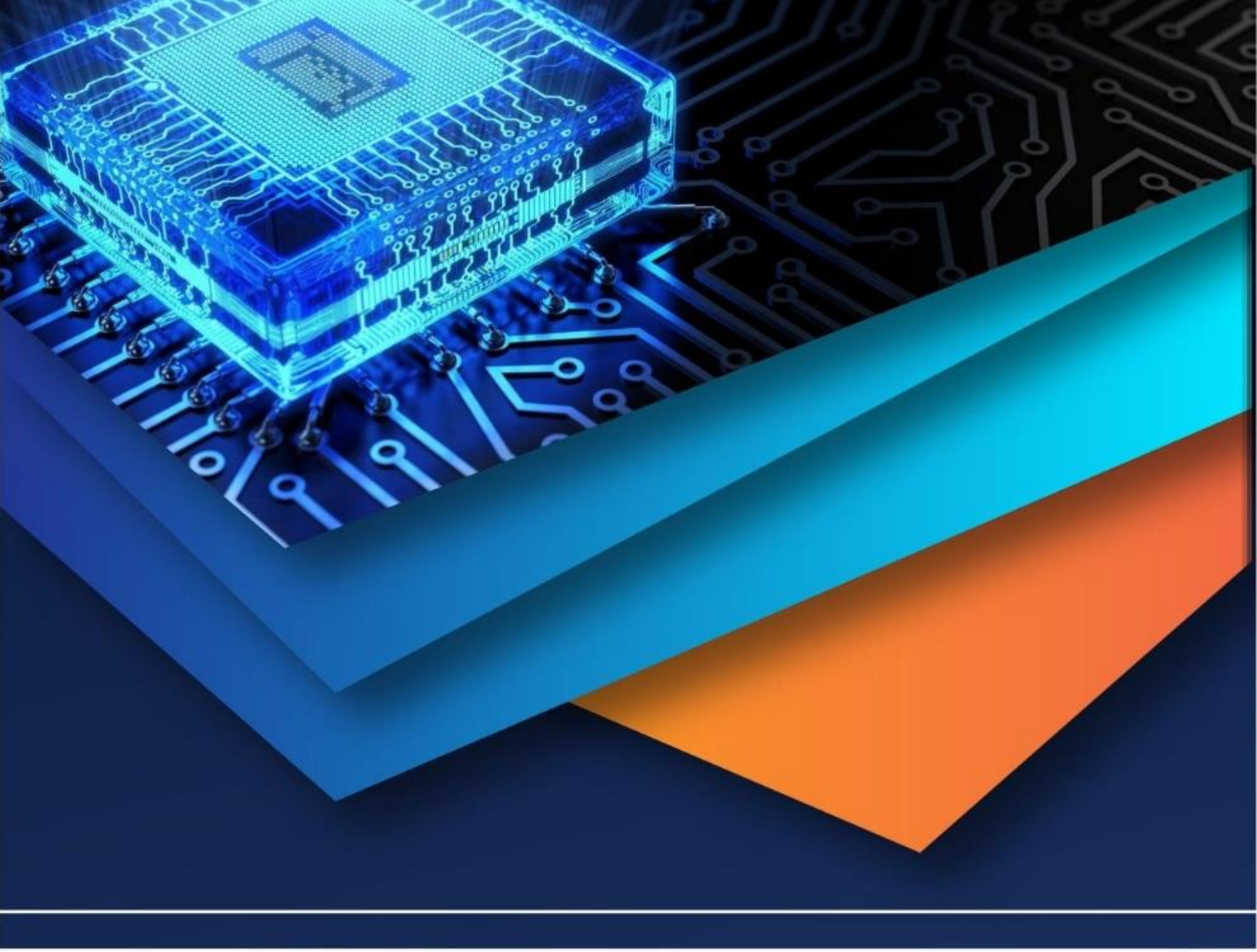
Concurrently, there's a palpable demand for a unified query language for NoSQL databases. The evolution of UNQL, despite its promising premise, elucidates the intricacies and challenges in standardizing a singular language across the diverse NoSQL spectrum. The pros, cons, and unique features of various NoSQL query languages, as presented in Fig. 1, further accentuate the richness and specialization of these languages, making a "one-size-fits-all" approach challenging.

While NoSQL databases undeniably present an alluring prospect for handling massive, unstructured datasets, they're not without their challenges, chief among them being the lack of a standardized query language. The dream of a universal NoSQL query language, capturing the intuitiveness and flexibility of SQL, remains just that—a dream, albeit one with immense potential and promise.

In closing, while the technological world marches towards embracing NoSQL's capabilities, the balance between innovation and standardization will remain a pivotal aspect of future database research. The nexus between RDBMS and NoSQL, typified by the migration methodologies and the quest for a unified language, represents fertile ground for future exploration, with the hope of achieving optimal synergies in data management and retrieval.

REFERENCES

[1] Nayak, A. Poriya, and D. Poojary, "Type of NOSQL Databases and its Comparison with Relational Databases," International Journal of Applied Information Systems, vol. 5, no. 4, pp. 1–4, Mar. 2013.
[2] N. Jatana, D. Gosain, I. Kathuria, M. Ahuja, and S. Puri, "A Survey and Comparison of Relational and Non-Relational Database," International Journal of Engineering Research & Technology (IJERT), vol. 1, no. 6, pp. 1–5, Aug. 2012.
[3] A. Abdullah and Q. Zhuge, "From relational databases to NoSQL databases: Performance evaluation," Research Journal of Applied Sciences, Engineering and Technology, vol. 11, no. 4, pp. 434–439, 2015. https://doi.org/10.19026/rjaset.11.1799
[4] M. Hanine, A. Bendarag, and O. Boutkhoum, "Data Migration Methodology from Relational to NoSQL Databases," International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol. 9, no. 12, pp. 1–5, Dec. 2015.
[5] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih, "Comparison and Classification of NoSQL Databases for Big Data," 2015
[6] V. Abramova, J. Bernardino, and P. Furtado, "Experimental evaluation of nosql databases," International Journal of Database Management Systems, vol. 6, no. 3, pp. 01–16, 2014. https://doi.org/10.5121/ijdms.2014.6301
[7] M. Bach and A. Werner, "Standardization of nosql database languages," Communications in Computer and Information Science, pp. 50–60, 2014. https://doi:10.1007/978-3-319-06932-6_6
[8] A. A. Mahmood, "Automated algorithm for data migration from relational to NoSQL databases," Al-Nahrain Journal for Engineering Sciences, vol. 21, no. 1, p. 60, 2018. https://doi.org/10.29194/njes21010060
[9] L. Rocha, F. Vale, E. Cirilo, D. Barbosa, and F. Mourão, "A framework for migrating relational datasets to nosql 1," Procedia Computer Science, vol. 51, pp. 2593–2602, 2015. https://doi.org/10.1016/j.procs.2015.05.367
[10] S. Ghotiya, J. Mandal, and S. Kandasamy, "Migration from relational to nosql database," IOP Conference Series: Materials Science and Engineering, vol. 263, p. 042055, 2017. https://doi.org/10.1088/1757-899x/263/4/042055
[11] A. Bansel, H. Gonzalez-Velez, and A. E. Chis, "Cloud-based NoSQL data migration," 2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 2016. https://doi.org/10.1109/pdp.2016.111
[12] Pepito, Glenn. (2018). RDBMS to NoSQL Migration: Challenges and Strategies.
[13] S. A. T. Mpinda, P. A. Bungama, and L. G. Maschietto, "From relational database to column-oriented nosql database: Migration process," International Journal of Engineering Research and, vol. V4, no. 05, 2015.
[14] N. Kuderu and V. Kumari, "Relational database to nosql conversion by schema migration and mapping," International Journal of Computer Engineering in Research Trends, vol. 3, no. 9, p. 506, 2016. https://doi.org/10.22362/ijcert/2016/v3/i9/48900
[15] W. Khan et al., "SQL and NoSQL database software architecture performance analysis and assessments—A systematic literature review," Big Data and Cognitive Computing, vol. 7, no. 2, p. 97, 2023. doi:10.3390/bdcc7020097
[16] Mapanga, Innocent & Kadebu, Prudence. (2013). Database Management Systems: A NoSQL Analysis. International Journal of Modern Communication Technologies & Research (IJMCTR). Volume-1. 12-18.
[17] Padhy, Rabi & Ranjan, Manas & Suresh, Patra & Satapathy, Chandra & India, Oracle. (2023). RDBMS to NoSQL: Reviewing Some Next-Generation Non-Relational Database's.
[18] "Is UNQL dead?," ArangoDB, https://arangodb.com/2012/04/is_unql_dead/ (accessed Oct. 26, 2023).
[19] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," 2013 21st IEEE International Requirements Engineering Conference (RE), 2013. doi:10.1109/re.2013.6636712
[20] D. C. Schmidt, A. Gokhale, and B. Natarajan, "Leveraging application frameworks," Queue, vol. 2, no. 5, pp. 66–75, 2004. doi:10.1145/1016998.1017005



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY