Available online at https://li02.tci-thaijo.org/index.php/ssstj

Suan Sunandha Science and Technology Journal

A Model for Converting Data into NoSQL Data Warehouse for Developing a Real-time Financial Data Warehouse System

Klaokanlaya Silachan[1], Sanya Kuankid[1,2], Thanin Muangpool[1,2*]

[1]Faculty of Science and Technology, Nakhon Pathom Rajabhat University, Malaiman Road, Muang, Nakhon Pathom 73000, Thailand
[2]Creating Innovation for Sustainable Development Research Center, Nakhon Pathom Rajabhat University, Nakhon Pathom, Thailand
***Corresponding author e-mail: signal@npru.ac.th**

Received: *5 February 2024* / **Revised:** *8 May 2024* / **Accepted:** *12 June 2024*

Abstract

This research introduces a novel model, the Financial Data Warehouses API (FDW-API), developed using PHP, Node.js, and Express.js. The model is designed to transform banking credit dataset information into a data warehouse format using a Non-Only SQL (NoSQL) database, stored in JSON format. Three types of databases were employed: MongoDB Node, MongoDB Serverless, and Cassandra. The study includes a comparative analysis of the data retrieval speed from all three databases. The model's applicability was tested in a real-time credit approval web application, demonstrating its effectiveness in transforming and storing data. Testing involved loading datasets ranging from 200, 300, 400, 500, 600, 800, and 1000 entries. Results indicate that the MongoDB serverless database outperformed others in terms of efficiency. Additionally, the FDW-API model streamlines data transformation and storage, facilitating real-time analysis and decision-making for financial institutions and data-driven businesses. Its flexibility integrates seamlessly with existing systems, reducing development time and costs, while its scalability accommodates growing data volumes and evolving business needs, providing a valuable tool for strategic insights and competitive advantage.

Keywords: API, NoSQL, MongoDB node, MongoDB serverless, Cassandra

1. Introduction

The incorporation of database systems into information systems yields benefits for management and facilitates the identification of relationships within business operations. Predominantly, individuals favor the utilization of Online Transactional Processing (OLTP) for data processing, a principle aimed at data storage that reduces redundancy and upholds data accuracy, thus mitigating errors resulting from data editing. Retrieval of available data from various sources is possible (Songsiri & Tamee, 2022). However, OLTP is not conducive to decision support systems requiring specificity. This is due to the necessity for large databases containing more extensive data than usual, segmented into smaller tables based on design principles, rendering it incapable of supporting the query format essential for decision-making support. This includes historical data retrieval to predict potential future trends based on developed models (Harvy, Matitaputty, Girsang, Michael, & Isa, 2019).

The management of large databases experiencing daily data volume increments challenges conventional techniques, posing a risk of errors. The method of segmenting the database into parts enhances its suitability for use. With a database system capable of supporting extensive data entry and simultaneous processing, Online Analytical Processing (OLAP) facilitates immediate service delivery (Wang, Li, Xu, Wang, & Wang, 2021). Consequently, a data warehouse system storing substantial data differs in structure from general databases, optimizing efficiency in data retrieval for end-users. The process of constructing an organization's data warehouse typically involves storing long-term data for at least 5-10 years, utilized in analyzing business operational processes (Garani, Chernov, Savvas, & Butakova, 2019).

The transformation of data from general databases into Online Transactional Processing (OLTP) format for integration into a data warehouse is achieved through the Extract, Transform, and Load (ETL) process (Singsanit, 2021). This process facilitates the extraction of data from various operational sources to execute Data Migration in alignment with the objectives of the software. Furthermore, the development of tools for data transformation through the ETL process encompasses two predominant modalities: tool utilization and code generation-based

programming, typically implemented in batch or offline formats (Yulianto, 2019). The intricate interchangeability inherent in the ETL process may encounter challenges if not appropriately designed, when considering data volume (Barahama & Wardani, 2021).

Addressing challenges in loading and exporting data from large-scale operational databases involves prolonged loading times and necessitates a waiting period before data becomes usable (Nizzad & Irshad, 2021). Conversely, if a real-time data warehouse is operational, capable of instantaneously receiving data from OLTP systems, the data is promptly transmitted from the primary database to the data warehouse. This enhances processing efficiency for concurrent retrieval and analysis, requiring the development of a system to perform real-time data transformation through code scripting into real-time data software, as opposed to batch or offline data transformation. This approach contributes to expediting the loading process, enabling instantaneous data analysis. Moreover, selecting an appropriate database structure is crucial for efficient and swift data loading, facilitating subsequent data analysis and report generation. Generally, structured data is stored in data warehouses, characterized by a meticulous data schema. If data modifications are needed, all structured data must be updated, which is a time-consuming and resource-intensive process (Hassan et al., 2022).

Consequently, non-relational database models have been employed in the development of various systems to accommodate the anticipated surge in data volume. Researchers have introduced non-relational database models as a novel approach with potential enhancements for contemporary data warehouses. Currently, system development incorporates the use of Application Programming Interfaces (APIs) as interfaces for application services or various modules, contributing to accelerated development processes (Jose & Abraham, 2020; Oditisi, Bicevska, Bicevskis, & Karnitis, 2018; Petricioli, Humski, & Vrdoljak, 2021).

In this research endeavor, the proposed approach involves the development of a system model in the form of a web API based on the principles of Node.js and Express JS. The objective is to transform data into the non-relational data warehouse structure in the JSON document format. A performance comparison is conducted to evaluate the efficiency and speed of data retrieval from databases, encompassing MongoDB nodes, MongoDB serverless, and Cassandra (Boonhao, 2020; Bouaziz, Nabli, & Gargouri, 2019; Chauhan, 2019). This comparative analysis aims to identify the most time-efficient and optimal data loading strategy from the data warehouse, providing insights for the subsequent development of a real-time web application credit approval system.

2. Objectives

2.1 To advance the procedures for transforming data from API format to a non-relational database structure.

2.2 To compare and evaluate the results against procedures involving a relational database, measuring the system's impact on data access and efficiency.

3. Research Methodology

The research methodology encompasses five distinct stages, outlined as follows:

3.1 Data structuring for real-time credit systems

This phase involves an in-depth exploration of relevant information. The researcher focused on understanding the development methods and characteristics of data transformation. Due to the complexity of the real-time credit application system, importing data involves intricate personal data, which makes precise measurement challenging. Therefore, a preliminary assessment of data sets was conducted, and the chosen method and non-relational database structure that exhibit optimal responsiveness were implemented. The research utilized a newly generated OLTP data set constructed from the financial dataset structure obtained from the PKDD CUP 99 data source (Alfred & Kazakov, 2006, Al-Mamory & Jassim, 2013; Tavallaee, Bagheri, Lu, & Ghorbani, 2009). This dataset was subsequently refined to align with the credit data structure of banking institutions. A representative data structure is illustrated in Figure 1.

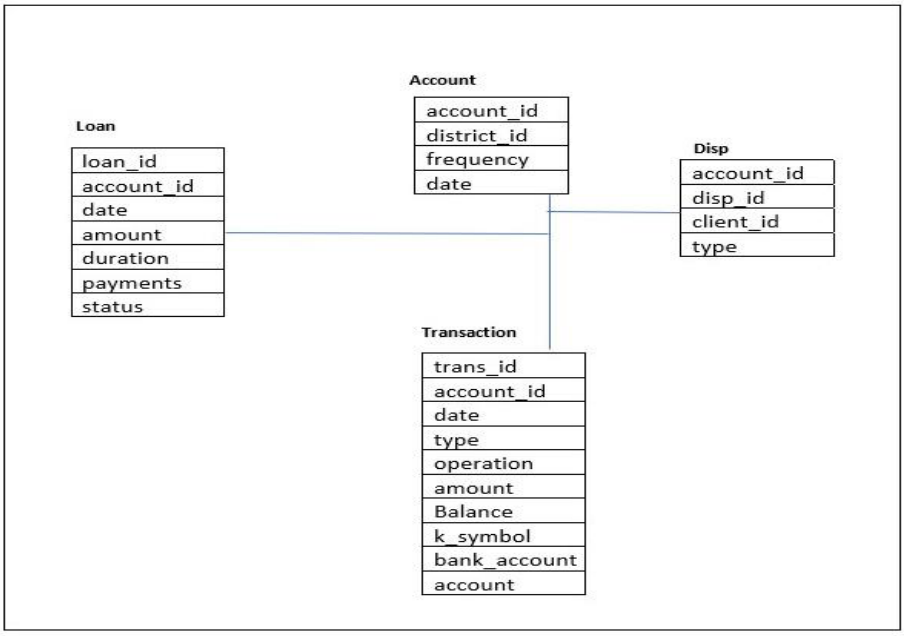

Figure 1. Dataset structure.

3.2 System analysis

The system analysis phase is a critical step where the studied system is examined to define its scope. This entails conducting a comprehensive review of the system's structure and the methodologies employed in its development and testing. Specifically, the analysis focused on the development and testing of three data transformation processes designed to import data into a non-relational database. These processes included MongoDB nodes, MongoDB serverless, and Cassandra, as depicted in Figure 2. The primary objective was to evaluate the efficiency of these processes in terms of response time for data transformation into an optimized database structure.

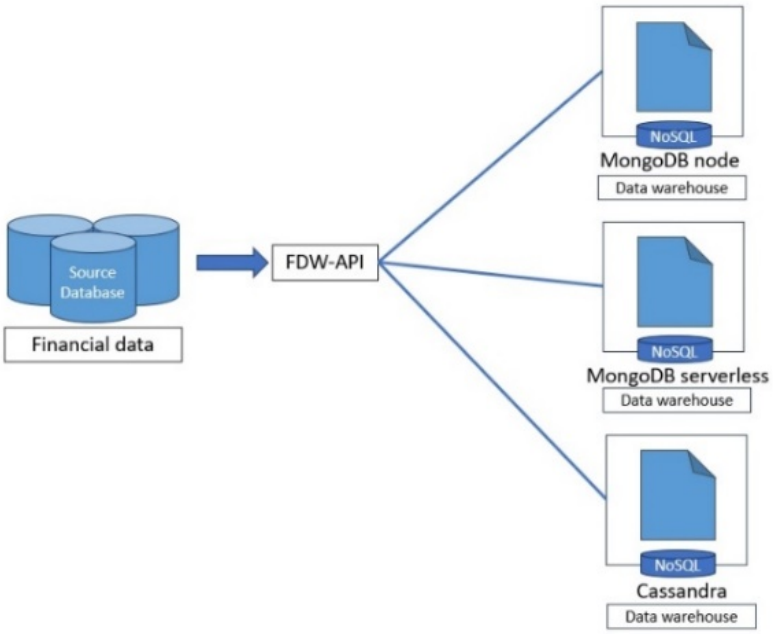

Figure 2. Diagram of data transformation into non-relational database using API.

3.3 System design

In system design, the approach is divided into two parts based on usage scenarios:

3.3.1 Design for data warehouse

In this segment of the system design, the focus was on the transformation of data to the data warehouse. The process involved converting data from the analytical online processing of the credit dataset through FDW-API to MongoDB nodes, MongoDB serverless, and Cassandra. Notably, Cassandra was served as the non-relational database with a distinct data structure compared to MongoDB nodes. The design adopted a hybrid approach, blending table-based data storage with key-value pairs. This allows for the conversion and utilization of JSON documents for storage in the Cassandra table structure, as illustrated in the representative data structure shown in Figure 3.

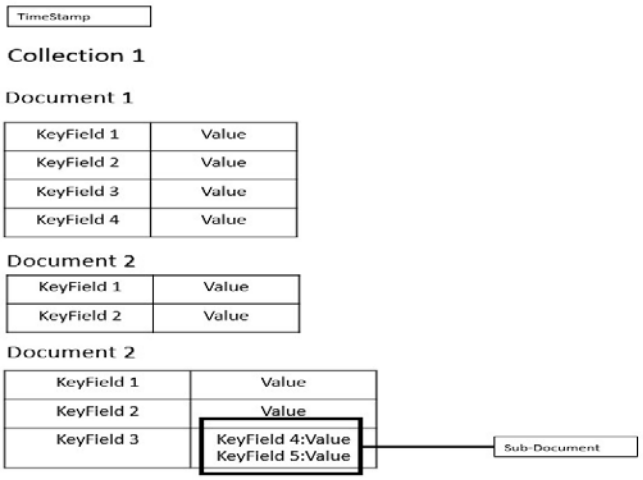

Figure 3. Structure of the document-oriented database.

3.3.2 System design

In the system design pertaining to the API-based model, the overall process for data transformation was outlined to identify the most efficient performance for immediate integration into the real-time system. The following steps elucidate the FDW API (Financial Data Warehouses API) model:

Step 1: FDW API transformation process

Utilizes Python Flask for the conversion of data from .asc (1) format to JSON documents conforming to the JSON Schema. Subsequently, the transformed data is stored in three distinct databases, namely MongoDB nodes, MongoDB serverless, and Cassandra.

Step 2: Node.js and Express.js API creation

- Reads ASCII files with semicolon delimitation from the dataset, assigning field names based on the file headers.
- Detects the data type and maximum length of the data in each field.
- Identifies the central table that has the highest number of foreign key connections to other tables.
- Embeds the data in JSON document format using the identified primary table.
- Displays the results as JSON documents through node.js + express.js, employing specified endpoints in the form of GET requests:
    - / warehouse/ json/ <table_name> for retrieval from MongoDB nodes, MongoDB serverless, and Cassandra.
    - /warehouse/mysql/<table_name> for retrieval from MySQL with full table join.
- Conducts load testing to evaluate and compare the performance of the designed API-based model.

3.4 System development

The system development phase focused on the implementation of the data transformation and management processes into the data warehouse. The FDW API (Financial Data Warehouses API) was developed using PHP, Node.js, Express.js, and in the form of a RestAPI. This API was designed to interact with non-relational databases, including MongoDB, MongoDB Serverless, and Cassandra. Additionally, for the relational database segment handling financial credit dataset, MariaDB was employed.

3.5 System testing and evaluation

This phase involves the comprehensive testing and evaluation of the system's functionality. The system demonstrates its ability to process, transform, and extract data from the data warehouse for accurate and timely presentation. Comparative speed assessments for data transformation have been conducted across non-relational database formats, including MongoDB, MongoDB Serverless, and Cassandra. The test results have been analyzed to identify the optimal transformation time, forming the basis for establishing a non-relational database for real-time credit data warehousing development.

4. Results

4.1 System development results

From the FDW API methodology outlined in Section 3.2.2, the designed process was executed to convert the dataset into a document-oriented data store format known as JSON. The details conform to the structure of JSON documents, facilitating data storage in a non-relational database format characterized by document properties akin to transaction timestamps, as depicted in Figure 4.

Figure 4. Example of converting JSON document structure for data storage in a JSON-oriented database.

4.2 Performance evaluation results

In this section, we present the performance evaluation results, specifically focusing on the processing speed comparison between non-relational database formats. The comparison encompasses MongoDB, MongoDB serverless, Cassandra, and is benchmarked against the average processing time. These results correspond to the second objective, which aims to compare and evaluate the results against procedures involving a relational database, measuring the system's impact on data access and efficiency.

The software components include MongoDB, MongoDB serverless, and Cassandra for database management. The hardware utilized for the evaluation comprises an Intel Core i7 8700 processor (6 cores / 12 threads), 16GB of RAM, Western Digital Black M.2 Solid State Drive, 10/100/1000 Gbps Ethernet, and the operating system employed is Microsoft Windows 11.

The system was developed using the PHP, Node.js, and Express.js programming languages in the form of RESTful APIs. The experimentation involved the Financial Dataset to assess and compare the response times of the proposed models and the non-relational database.

The results are analyzed by comparing the response times against the quantity of data entries used in the testing phase. The evaluation considers both the presented model and the non-relational database, as depicted in Figure 5.

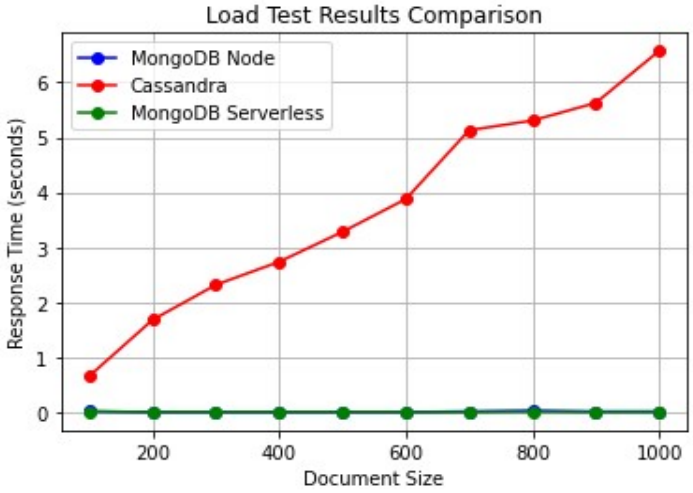

Figure 5. Response time evaluation results.

The response times of the MongoDB Node and MongoDB Serverless databases exhibit closely aligned values, prompting a dedicated comparison as shown in Figure 6.

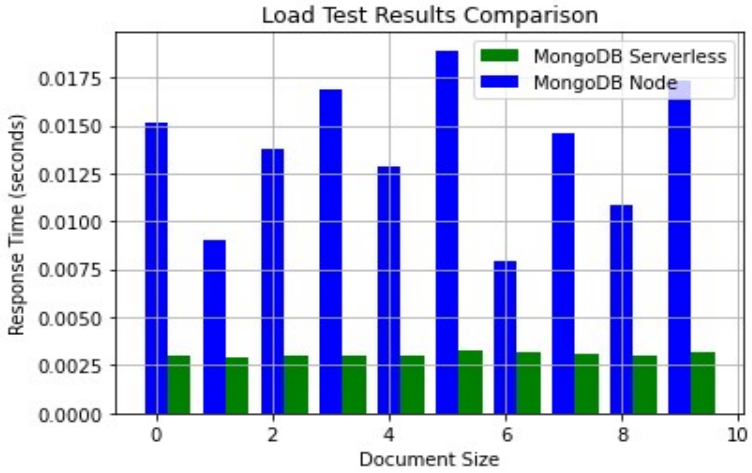

Figure 6. Comparative analysis of response times between MongoDB Node and MongoDB Serverless databases.

The average load times resulting from the assessment of the initial 100 dataset entries, followed by increments of 200, 300, 400, 500, 600, 800, and concluding with the last set of 1000, are tabulated below in Table 1.

Table 1. Average load times for testing datasets.

Number of Record Loaded	MongoDB Node (seconds)	Cassandra (seconds)	MongoDB Serverless (seconds)
100	0.015157	0.707098	0.003006
200	0.009023	1.311107	0.002936
300	0.013725	1.881718	0.003028
400	0.016828	2.448808	0.002962
500	0.012839	3.087998	0.002991
600	0.018908	3.733704	0.003286
700	0.007931	4.332826	0.003177
800	0.014600	5.038835	0.003118
900	0.010808	5.391285	0.003006
1000	0.017298	5.989958	0.003162

The experimentation results clearly demonstrate the successful development of the FWD-API model for establishing a non-relational data warehouse. Moreover, upon comparing the response time outcomes, it was evident that the MongoDB serverless operates more efficiently within the system, outperforming both MongoDB and Cassandra. This superiority is attributed to its faster data loading capabilities, resulting in enhanced performance, stability, and faster overall processing when compared to alternative models.

4.3 Development of real-time web-enabled credit data warehouse system

The outcome of the performance evaluation in developing a real-time banking credit web application, utilizing the FDW-API model to transform credit approval requests into the MongoDB serverless data warehouse, is presented. This section reflects the development and testing phase, showcasing the system's functionality in real-world application scenarios. When a credit applicant submits the request through the data entry form, the information is stored in the online analytical processing database. Upon approval, detailed data regarding the credit applicant and the approval transaction are stored in two segments: the MariaDB database and the MongoDB serverless data warehouse. This design enables further analysis and reporting based on the structure outlined in Figure 7.

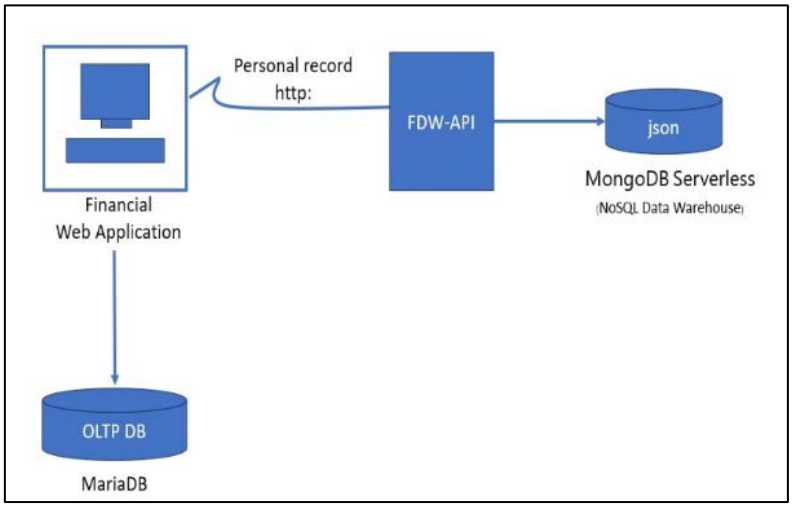

Figure 7. System development structure.

5. Discussion

This research endeavors to develop a real-time credit data warehouse system, employing API principles to manage data integration into a non-relational database structure. The chosen document-oriented JSON structure, known as FDW-API, was implemented using PHP, Node.js, Express.js, and utilized across three database formats: MongoDB, MongoDB serverless, Cassandra. The objective was to identify the most responsive format for real-time credit application systems.

Upon testing, it was observed that the MongoDB serverless data warehouse exhibited the best overall performance. This conclusion stems from its superior data retrieval speed and efficient operational outcomes. These findings align with the research conducted by Jaratsantijit, Y. (Jaratsantijit, 2022), which explored data storage strategies in NoSQL databases, specifically comparing the performance of relational databases with NoSQL databases for information systems. The choice of MongoDB as the NoSQL database for this research is substantiated by its optimal responsiveness and efficiency in handling real-time credit application systems.

It is recommended that future research further explores the scalability and performance of the FDW-API model across diverse industry sectors. Investigating the implementation of the model in sectors beyond finance could provide valuable insights into its adaptability and efficacy in different contexts. Furthermore, guidelines should be developed for agencies and stakeholders interested in implementing the FDW-API model. These guidelines could outline best practices for data integration, system architecture design, and performance optimization to maximize the model's benefits. Additionally, educational resources and training programs could be developed to support organizations in effectively leveraging the model for real-time data analysis and decision-making.

6. Conclusion

This study focuses on the design and development of a procedural model for transforming data from a credit database transaction set (OLTP) to a non-relational data warehouse in the form of a document-oriented model, specifically leveraging the MongoDB database. The study aims to accommodate both data storage and retrieval needs. Through performance testing involving 100 initial datasets, followed by increments of 200, 300, 400, 500, 600, 800, and finally 1000 entries, the procedural model and the data warehouse, based on the non-relational MongoDB serverless architecture, demonstrated efficient response times, indicating rapid data processing. This suggests that the developed model is well-suited for application as a data warehouse.

Acknowledgements

This research was made possible by the kind support of the Research and Development Institute of Nakhon Pathom Rajabhat University.

Conflict of Interest

No conflict of interest.

ORCID

Thanin Muangpool
https:// orcid.org/0000-0002-1088-5725
Sanya Kuankid
https:// orcid.org/0000-0003-0918-1402

Publication Ethic

Submitted manuscripts must not have been previously published by or be under review by another print or online journal or source

References

Alfred, R., & Kazakov, D. (2006). Pattern-based transformation approach to relational domain learning using dynamic aggregation for relational attributes. *Proceedings of the 2006 International Conference on Data Mining* (pp. 118-124). Las Vegas, Nevada.

Al-Mamory, S., & Jassim, F. S. (2013). Evaluation of different data mining algorithms with KDD CUP 99 data set. *Journal of University of Babylon*, *21*.

Barahama, A. D., & Wardani, R. (2021). Utilization extract, transform, load for developing data warehouse in education using Pentaho Data Integration. *Journal of Physics: Conference Series*, *2111*. doi:10.1088/1742-6596/2111/1/012030

Boonhao, P. (2020). NewSQL databases and usability trends. *Mahidol R2R e-Journal*, *8*(2), 38-52.

Bouaziz, S., Nabli, A., & Gargouri, F. (2019). Design a data warehouse schema from document-oriented database. *Procedia Computer Science*, *159*, 221-230. doi:10.1016/j.procs.2019.09.177

Chauhan, A. (2019). A review on various aspects of MongoDB databases. *International Journal of Engineering Research & Technology (IJERT)*, *8*(05), 90-92.

Garani, G., Chernov, A., Savvas, I., & Butakova, M. (2019). A data warehouse approach for business intelligence. *Proceedings of the 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)* (pp. 70-75). Napoli, Italy.

Harvy, I., Matitaputty, G. A., Girsang, A. S., Michael, S., & Isa, S. M. (2019). The use of book store GIS data warehouse in implementing the analysis of most book selling. *Proceedings of the 7th International Conference on Cyber and IT Service Management (CITSM)* (pp. 1-5). Jakarta, Indonesia.

Hassan, C. A. U., Hammad, M., Uddin, M., Iqbal, J., Sahi, J., Hussian, S., & Ullah, S. S. (2022). Optimizing the performance of data warehouse by query cache mechanism. *IEEE Access*, *10*, 13472-13480. doi:10.1109/ACCESS.2022.3148131

Jaratsantijit, Y. (2022). *Comparative study of query performance between relational database and NoSQL database for information system: A case study of the asset database of information technology service center.* Chiang Mai: Chiang Mai University.

Jose, B., & Abraham, S. (2020). Performance analysis of NoSQL and relational databases with MongoDB and MySQL. *Materials Today: Proceedings*, *24*(3), 2036-2043. doi:10.1016/j.matpr.2020.03.634

Nizzad, A. R. M., & Irshad, M. B. M. (2021). Data warehouse implementation: Cost effective approach for small businesses. *Journal of Information Systems & Information Technology (JISIT)*, *6*(2), 62-71.

Oditisi, I., Bicevska, Z., Bicevskis, J., & Karnitis, G. (2018). Implementation of NoSQL-based data warehouses. *Baltic J. Modern Computing*, *6*(1), 45-55.

Petricioli, L., Humski, L., & Vrdoljak, B. (2021). The challenges of NoSQL data warehousing. *Proceedings of International Conference on E-business Technologies* (pp. 44-48). Serbia.

Singsanit, K. (2021). The development of executive information system for managing research in university by data integration techniques on ontology on business intelligence. *Journal of Buddhist Education and Research: JBER*, *7*(1), 157-174.

Songsiri, K., & Tamee, K. (2022). Development of data warehouse for financial report in Faculty of Science, Naresuan University. *Journal of Applied Informatics and Technology*, *4*(2), 99-113. doi:10.14456/jait.2022.8

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009)* (pp. 1-6). Ottawa, Canada.

Wang, D., Li, Q., Xu, C., Wang, P., & Wang, Z. (2021). Research of data warehouse for science and technology management system. *Proceedings of the International Conference on Service Science (ICSS)* (pp. 65-69). Xi'an, China.

Yulianto, A. A. (2019). Extract transform load (ETL) process in distributed database academic data warehouse. *Journal on Computer Science and Information Technologies*, *4*(2), 64-71. doi:10.11591/APTIKOM.J.CSIT.36