Vol. 6 No. 2, 2025, pp. 113-119 **Jurnal Pepadun**

Performance Comparison of Object-Relational Mapping (ORM) and SQL Query in Developing the Booking Service API at PT Tunas Dwipa Matra

[1] Bambang Hermanto ,[2] M. Iqbal Parabi, [3] Dwi Sakethi, and *[4] Nadaa Azhar

[1,2,3,4]Department of Computer Science, Faculty of Mathematics and Natural Sciences, Universitas Lampung
Jl. Prof. Sumantri Brojonegoro No. 1, Gedong Meneng, Rajabasa District, Bandar Lampung City, Lampung, Indonesia

e-mail: [1]bambang.hermanto@fmipa.unila.ac.id, [2]iqbal.parabi@fmipa.unila.ac.id, [3]dwijim@fmipa.unila.ac.id,
[*4]**nadaa.azhar14@gmail.com**

Abstract *- In this digital era, information technology plays a pivotal role across various industries, particularly in the service and reservation sector, where companies like PT Tunas Dwipa Matra are compelled to adopt advanced tools to remain competitive and enhance customer experiences. A critical component of this transformation is the use of Application Programming Interfaces (APIs), which enable seamless communication between systems and improve operational efficiency. A key decision in system design revolves around choosing between Object Relational Mapping (ORM) and SQL queries for database management. ORM simplifies development processes and enhances maintainability, making it well-suited for routine tasks and scenarios where ease of use and security are priorities. On the other hand, SQL queries excel in handling complex, performance-critical operations, offering greater control and faster execution for specialized or large-scale tasks. Performance testing highlights a trade-off between the two approaches: ORM tends to be slower in execution but more memory-efficient, while SQL delivers faster query processing at the cost of higher memory consumption. By strategically leveraging the strengths of both ORM and SQL, PT Tunas Dwipa Matra can optimize system performance, scalability, and maintainability, ultimately ensuring high-quality service delivery and improved customer satisfaction.*

Keywords**:** *Object Relational Mapping; SQL Query; Performance Testing; Execution Speed; Memory Efficiency.*

1. INTRODUCTION

In this digital era, information technology plays a vital role across various industries, including the service and reservation sector [1]. PT Tunas Dwipa Matra, a prominent company in the service sector, recognizes the need for cutting-edge solutions to enhance customer experience and operational efficiency. A crucial aspect of this digital transformation is the integration of Application Programming Interfaces (APIs), which facilitate seamless data exchange and enable better scalability. APIs have become essential tools in ensuring that systems remain interconnected, allowing for smoother transactions and services that meet customer expectations [2]. Previous studies have demonstrated that APIs, particularly those integrated with frameworks such as Odoo ERP and Couchbase, significantly improve data management and enhance overall system performance [3][4][5]. This underscores the importance of implementing APIs to create a fast, reliable, and user-friendly Booking Service for PT Tunas Dwipa Matra.

A key decision when developing systems that rely on APIs is the choice between Object-Relational Mapping (ORM) and Structured Query Language (SQL) queries for database management. ORM simplifies interactions with databases by using object-oriented programming principles, which are typically easier to understand and implement for developers. This approach abstracts the database schema into object models, making it particularly advantageous for developers who prefer rapid development and ease of maintenance [6][7]. On the other hand, SQL queries provide direct access to databases, offering greater control over data retrieval and manipulation. While SQL queries often deliver superior performance, especially for complex or large-scale tasks, they can be harder to maintain and require more in-depth knowledge of the database's underlying

 | doi: 10.23960/pepadun.v6i3.263

structure [8]. For PT Tunas Dwipa Matra, selecting the right method is essential to ensure the system is both efficient and scalable.

This paper aims to investigate and compare ORM and SQL queries, specifically in the context of implementing a Booking Service API at PT Tunas Dwipa Matra. The comparison focuses on understanding the strengths and weaknesses of each approach, with particular attention to performance metrics, including execution speed, memory efficiency, and scalability. ORM is often perceived as slower in execution but more memory-efficient, whereas SQL queries typically offer faster performance at the cost of higher memory usage [9][10]. By exploring these factors, the study seeks to provide practical recommendations for PT Tunas Dwipa Matra, aiding the company in making an informed decision about which approach to adopt for their booking service system.

Furthermore, this study is expected to contribute to the broader field of database management and optimization, particularly in industries undergoing digital transformation. The research will provide valuable insights into how ORM and SQL queries impact system efficiency, which is crucial for businesses aiming to enhance service delivery through API integration. It is anticipated that the findings will not only assist PT Tunas Dwipa Matra but also serve as a reference for other companies in similar industries that rely on robust, scalable, and efficient database solutions. The increasing reliance on APIs for business operations further underscores the need for businesses to carefully evaluate their technology choices, ensuring they meet the demands of modern service-oriented systems [11].

2. RESEARCH METHODOLOGY

The research stages for comparative analysis of Object-Relational Mapping (ORM) and Structured Query Language (SQL) query in the implementation of booking service API at PT Tunas Dwipa Matra are illustrated in Figure 1.

2.1. Problem Identification

The study focuses on issues within the booking service module at PT Tunas Dwipa Matra, where the current database processing code shows signs of inefficiency, leading to performance bottlenecks and a suboptimal user experience. The main challenge revolves around choosing between the Object-Relational Mapping (ORM) and SQL Query frameworks to determine which one is more effective for the module's operations. While ORM frameworks offer advantages in terms of code maintainability and developer productivity, they often introduce inefficiencies such as poor execution plans and scalability issues. On the other hand, traditional SQL queries provide greater control over performance tuning but may increase complexity and maintenance overhead [12][13]. To address this, the research evaluates various factors, including maintenance speed, scalability, and the impact on user experience. By analyzing these aspects, the study aims to provide a clear direction for improving the system's functionality and efficiency, ensuring a more responsive and user-friendly booking service module.

2.2. Literature Review

A literature review is the process of collecting information from various sources, such as journals, books, and related studies, to build a deep understanding of the research topic. This stage helps shape the key elements of the study, ensuring they align with the research objectives. By analyzing existing knowledge, the review serves as a foundation for exploring the chosen subject and guiding the direction of the research [14].

 | doi: 10.23960/pepadun.v6i3.263

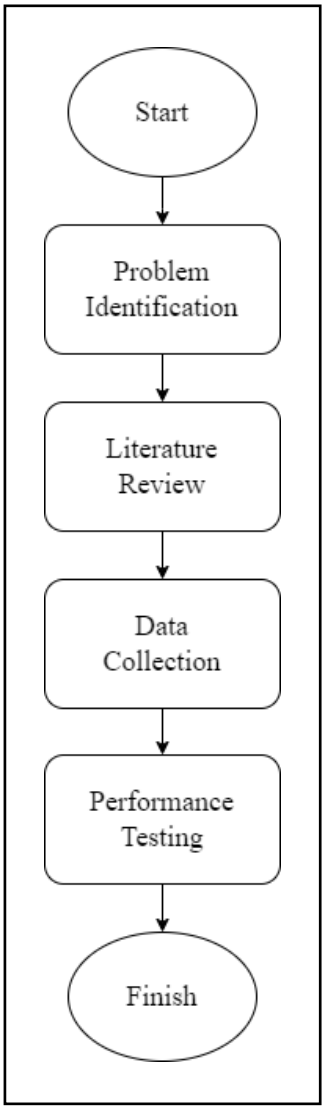

Figure 1. Research flowchart.

2.3. Data Collection

The data for this research is collected through two primary methods: literature review and observation. Initially, a literature review is conducted to gather relevant information on APIs, booking services, and the Odoo framework, including written materials, images, and electronic documents. Following this, observation is carried out to gain a clearer understanding of the current state of the booking service, particularly by examining the database structure of the system to be integrated.

2.4. Performance Testing

This research compares the execution time of Object-Relational Mapping (ORM) and SQL Query to evaluate their performance. Given the differences in their code structure and data access techniques, the comparison focuses on speed performance and memory usage. To conduct this analysis, three key components are required: a data dictionary, a design for performance testing comparison, and a system testing scenario [15].

A data dictionary is used to define and identify the tables that will be utilized in creating data queries. These queries will be sent to the API applications, such as Motoran, and vice versa. For the performance testing comparison design, three tables from the data dictionary will be the focus: "dms_uang_titipan_customer," "dms_account_invoice," and "dms_booking_service." In this study, query testing is conducted through

 | doi: 10.23960/pepadun.v6i3.263

performance testing, followed by a comparative analysis of the results. The testing scenarios aim to evaluate the use of ORM and SQL Query within the Booking Service API. Performance tests will be performed using datasets of 10,000, 50,000, and 100,000 entries, ensuring optimal internet network conditions for accurate results. Table 1 illustrates the system testing scenario.

Table 1. System testing scenario.

List of Tests	Description	Expected Outcome
SELECT Data	Retrieve data that meets several specified criteria on '*dms.titipan.customer*' model.	Successfully retrieve data that meets the specified criteria.
WRITE Data	Write (update) object from the '*dms.account.invoice*' model.	The object updates successfully.
INSERT Data	Create a new record in the '*dms.booking.service*' table.	Successfully create a new record.

3. RESULTS AND DISCUSSION

3.1. Select Data for Table 'dms.uang.titipan.customer'

This study evaluates the efficiency of Object-Relational Mapping (ORM) and Structured Query Language (SQL) by conducting data selection tests on the 'dms.uang.titipan.customer' table, which stores down payment data from transactions in the Motoran application. This step is critical for comparing the performance of both methods. Additionally, query testing is performed using Python's `psycopg2` and `memory_profiler` modules. While `psycopg2` facilitates PostgreSQL integration and measures query execution speed, `memory_profiler` monitors memory usage during data processing, providing insights into the performance and resource efficiency of ORM and SQL.

Testing is then conducted on one module using table 'dms.uang.titipan.customer' with data sizes of 10,000, 50,000, and 100,000 records, performing 10 iterations for each data size. The results of the testing for the three data sizes are presented in Table 2.

Table 2. Execution results from 'dms.uang.titipan.customer'.

Data Size	Framework	Average		Maximum		Minimum		Median	
		Time (s)	Memory Usage (MiB)	Time (s)	Memory Usage (MiB)	Time (s)	Memory Usage (MiB)	Time (s)	Memory Usage (MiB)
10.000	**ORM**	0.042	1364.0	0.07	1415.4	0.018	1226.4	0.045	1410.9
	SQL Query	0.007	1819.0	0.012	1887.2	0.003	1637.6	0.007	1881.6
50.000	**ORM**	0.105	1402.8	0.133	1433.6	0.083	1187.3	0.105	1425.8
	SQL Query	0.018	1893.3	0.023	1899.2	0.013	1886.3	0.018	1892.8
100.000	**ORM**	0.572	1473.5	0.586	1487.2	0.547	1457.6	0.575	1471.1
	SQL Query	0.051	1951.7	0.058	1967.4	0.042	1935.1	0.052	1949.6

 | doi: 10.23960/pepadun.v6i3.263

The analysis of the three tables reveals that ORM consistently requires more time to execute queries compared to direct SQL, as reflected in its higher average, maximum, minimum, and median execution times across all datasets. However, ORM proves to be more memory-efficient, exhibiting lower memory consumption in all measured aspects. This pattern highlights a trade-off between execution speed and memory usage while ORM operates more slowly, it utilizes memory more efficiently, whereas SQL executes queries faster but at the cost of higher memory consumption.

3.2. Update Data for Table 'dms_account_invoice'

Following the same method used for testing table 'dms.uang.titipan.customer' with 'psycopg2' and 'memory_profiler,' table 'dms.account.invoice' was evaluated. The primary focus of this test was updating data, specifically modifying the input invoice and output invoice values. Table 3 shows the execution results.

Table 3.Execution results from Table 'dms.account.invoicer'.

Data Size	Framework	Average		Maximal		Minimal		Median	
		Time (s)	Memory Usage (MiB)	Time (s)	Memory Usage (MiB)	Time (s)	Memory Usage (MiB)	Time (s)	Memory Usage (MiB)
10.000	ORM	0.008	1683.7	0.018	1695.4	0.004	1674.4	0.007	1682.5
	SQL Query	0.168	1683.9	0.301	1695.4	0.111	1674.4	0.161	1682.8
50.000	ORM	0.015	1698.9	0.030	1730.4	0.009	1683.5	0.014	1689.1
	SQL Query	0.956	1699.1	1.337	1730.4	0.768	1683.5	0.926	1689.5
100.000	ORM	0.037	1712.3	0.086	1789.9	0.023	1677.9	0.031	1680.0
	SQL Query	1.952	1712.5	2.591	1790.6	1.667	1677.9	1.854	1680.0

The tests conducted on Table 3 consistently show that ORM outperforms direct SQL in execution speed, while maintaining nearly identical memory usage. In various scenarios, ORM's ability to process queries faster—whether the difference is significant or slight—positions it as a more efficient choice, especially in situations where rapid query execution is essential. Despite its speed advantages, ORM maintains comparable memory efficiency to SQL. As a result, ORM emerges as a highly effective and dependable alternative to direct SQL, offering optimized query performance without compromising memory resources.

3.3. Insert Data for Table 'dms.booking.service'

The objective of testing Table 'dms.booking.service' is to produce a set number of data entries and measure the memory usage and time needed to generate multiple entries simultaneously. Execution results are available on Table 4.

Table 4. Execution results from Table 'dms.booking.service'.

Data Size	Framework	Average		Maximal		Minimal		Median	
		Time (s)	Memory Usage (MiB)	Time (s)	Memory Usage (MiB)	Time (s)	Memory Usage (MiB)	Time (s)	Memory Usage (MiB)
10	ORM	0.704	13339.1	0.984	14008.5	0.559	13106.5	0.687	13194.5
	SQL Query	0.058	13339.2	0.190	14008.5	0.017	13106.5	0.039	13194.5
50	ORM	3.796	13225.2	4.258	13370.1	3.278	13156.0	3.753	13202.7
	SQL Query	0.116	13223.1	0.158	13365.0	0.076	13156.0	0.114	13200.0

 | doi: 10.23960/pepadun.v6i3.263

Data Size	Framework	Average		Maximal		Minimal		Median	
		Time (s)	Memory Usage (MiB)	Time (s)	Memory Usage (MiB)	Time (s)	Memory Usage (MiB)	Time (s)	Memory Usage (MiB)
100	ORM	10.114	13229.4	10.690	13276.5	9.552	13183.5	10.091	13232.6
	SQL Query	0.330	13225.3	0.429	13277.0	0.135	13183.5	0.392	13222.0

In all three experiments comparing direct SQL and ORM, SQL consistently demonstrates faster execution speeds while utilizing a similar amount of memory. Although ORM occasionally shows longer maximum execution times in specific scenarios, SQL generally performs significantly faster, with consistently shorter execution times. Despite these speed advantages, both ORM and SQL exhibit nearly identical memory usage, with only minor differences. These results highlight SQL as the more efficient option for performance optimization, providing a quicker and more effective approach to query execution without compromising memory efficiency.

4. CONCLUSIONS

This research compares Object-Relational Mapping (ORM) and Structured Query Language (SQL) queries in the development of a Booking Service API at PT Tunas Dwipa Matra. The findings reveal that ORM is advantageous for routine, low-complexity tasks, offering ease of development and maintainability. However, ORM tends to be slower in execution, especially with larger datasets, which can impact performance in high-throughput applications. In contrast, SQL queries excel in execution speed and are better suited for complex, performance-critical operations, though they require more intricate management and can be more memory-intensive. These differences highlight the need for careful selection based on the specific requirements of each system.

The study suggests a hybrid approach, combining the strengths of both ORM and SQL to optimize system performance, scalability, and maintainability. Using ORM for routine tasks and SQL for complex operations allows PT Tunas Dwipa Matra to balance ease of development with high performance. This research contributes to the broader field of database management, offering practical insights for businesses seeking to enhance service delivery through API integration. By making informed decisions about database technologies, companies can improve their operational efficiency and customer satisfaction in the evolving digital landscape.

LITERATURE

[1] Y. L. Pattinama, Ferdiansyah, I. Susanti, and Painem “Implementasi Rest API Web Service dengan Otentifikasi JSON Web Token untuk Aplikasi Properti,” *INFORMATIK*, vol. 19, no. 1, 2023, https://doi.org/10.52958/iftk.v19i1.5724.

[2] A. Filiana, M. N. A. Rini, A. G. Prabawati, and R. A. Samat, “Pengembangan Rest API Untuk Informasi Pasar Tradisional di Kota Yogyakarta dengan Metode Incremental,” *SINTECH Journal*, vol. 5, no. 1, pp. 10–23, 2022.

[3] R. D. Permatasari, N. F. Ariyani, and A. Munif, “Rancang Bangun API untuk Odoo ERP pada Modul CRM (Customer Relationship Management),” *Jurnal Teknik* ITS, vol. 8, no. 2, 2019, https://doi.org/10.12962/j23373539.v8i2.49403.

[4] Wu, J. Y., and L. T. Chen. "Odoo ERP with business intelligence tool for a small-medium enterprise: a scenario case study." *Proceedings of the 2020 11th International Conference on E-Education, E-Business, E-Management, and E-Learning*. 2020.

 | doi: 10.23960/pepadun.v6i3.263

[5] M., K. Niza and A. I. Ahlunnazak. "Odoo-erp as a solution to business performance in supply chain of the coffe industry: A systematic literature review." *Jurnal Manajemen Stratejik dan Simulasi Bisnis*, vol. 4, no. 2, 2023, pp. 69-81.

[6] M. Gorodnichev, M. Moseva, K. Poly, K. Dzhabrailov, and R. Gematudinov, "Exploring Object-Relational Mapping (ORM) Systems and How to Effectively Program a Data Access Model," *PalArch's Journal of Archaeology of Egypt / Egyptology*, vol. 17, no. 3, pp. 615–627, Nov. 2020.

[7] B. Alshemaimri, R. Elmasri, T. Alsahfi, and M. Almotairi, "A survey of problematic database code fragments in software systems," *Engineering Reports*, vol. 3, no. 10, Oct. 2021.

[8] D. Colley, C. Stanier, and M. Asaduzzaman, "The Impact of Object-Relational Mapping Frameworks on Relational Query Performance," in *2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, Southend, United Kingdom: IEEE, Aug. 2018, pp. 47–52.

[9] F. Akhdani and D. Wijayanto, "Comparison of Eloquent ORM with Query Builder in Work Management System (Case Study: Muhammadiyah Lamongan Hospital)," *Senatik*, vol. 7, Mar. 2022.

[10] M. Filip, and M. Borkovcova. "Design of Data Access Architecture Using ORM Framework." *2023 34th Conference of Open Innovations Association (FRUCT)*. IEEE, 2023.

[11] D. V. Sivakumar, T. Balachander, Logu, and R. Jannali, "Object Relational Mapping Framework Performance Impact," *Turkish Journal of Computer and Mathematics Education*, vol. 12, no. 7, 2021, pp. 2516-2519, Jan. 2021.

[12] D. Colley, C. Stanier, and M. Asaduzzaman, "Investigating the Effects of Object-Relational Impedance Mismatch on the Efficiency of Object-Relational Mapping Frameworks," *Journal of Database Management*, vol. 31, no. 4, 2020.

[13] B. Alexandre, C. Quinton, and R. Rouvoy. "Understanding the performance-energy tradeoffs of object-relational mapping frameworks." *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2024.

[14] J. A. Luft, S. Jeong, R. Idsardi, and G. Gardner, "Literature Reviews, Theoretical Frameworks, and Conceptual Frameworks: An Introduction for New Biology Education Researchers," *LSE*, vol. 21, no. 3, Sep. 2022.

[15] G. A. Eren and B. Avenoglu. "Performance analysis of object-relational mapping (orm) tools in. net 6 environment." *Bilişim Teknolojileri Dergisi*, vol. 15, no. 4, 2022, pp. 453-465.