JIKO (Jurnal Informatika dan Komputer)
Vol. 7, No. 2, August 2024, pp. 93-100
DOI: 10.33387/jiko.v7i2.8565

Accredited KEMENDIKBUD RISTEK, No.105/E/KPT/2022
p-ISSN: 2614-8897
e-ISSN: 2656-1948

PRIMARY QUERY ANALYSIS ON SQL DATABASE RESTRUCTURING IN GEOGRAPHIC INFORMATION SYSTEMS

Ridwan Ilyas[*1], Wina Witanti[2], Fildzah Syarafina[3]

[1,2,3]Department of Informatics, Faculty of Science and Informatics, Universitas Jenderal Achmad Yani, Cimahi, Indonesia
Email: [1]ilyas@lecture.unjani.ac.id, [2]winawita0406@gmail.com, [3]fildzahsyafarina20@if.unjani.ac.id

(Received: 5 August 2024, Revised: 8 August 2024, Accepted: 17 August 2024)

Abstract

Database restructuring is a crucial process aimed at enhancing data management and access efficiency by modifying the existing data structure. This research focuses on improving a Geographic Information System (GIS) for taxation by migrating and restructuring an inefficient and redundant database. The study conducts a comparative performance evaluation of the old and restructured databases using benchmarking tests with varying numbers of threads and ramp-ups. The results reveal a significant increase in average throughput (24.60%) following the restructuring, indicating a substantial improvement in the database's data processing capacity. However, there is also an average increase in response time (21.65%), suggesting a trade-off between enhanced throughput and slower response times. This increase in response time indicates that while the system can handle more data, it requires more time to process each query. Overall, the restructured database demonstrates enhanced performance and efficiency, though further optimization is necessary to achieve consistent throughput across different workloads and to mitigate the increased response times.

Keywords: *SQL Database Restructuring, Geographic Information Systems (GIS), Taxation, Database Performance, Benchmarking,*

**Corresponding Author: Ridwan Ilyas*

1. INTRODUCTION (UPPERCASE, 10pt, bold)

Database restructuring is the process of altering or modifying the existing data structure to meet new requirements or enhance the efficiency of data management and access. This process involves manipulating data into more structured forms, such as tables or documents, and can include merging, splitting, or reformatting data. Additionally, restructuring encompasses the ability to integrate data from various sources and present it in a more organized format, thereby facilitating easier data access and analysis. This is crucial for addressing issues arising from disorganized data structures and allows for the exploitation of logical relationships between information items. [1]. Data restructuring within the context of databases is crucial because it enhances data accessibility, allows for integration from various sources, and adapts to new requirements. Restructuring also optimizes system performance by accelerating access and storage efficiency, and aids in managing unstructured or semi-structured data. This facilitates the exploitation of logical relationships between information for in-depth analysis. Moreover, it supports innovation and the development of new applications more flexibly, leveraging data in more innovative ways and increasing the value of existing information. [1]. Currently, a geographic information system for taxation has been developed with a database migrated from the previous taxation system, which was integrated with the geographic system. The previous database was inefficient and had many data redundancies, as well as an inefficient data structure.

A Geographic Information System (GIS) is an essential technology used to store, manage, analyze, and display data related to natural conditions. GIS works with attribute data (detailed information) and spatial data (geographical location). This system is typically integrated with computers and other networks to function effectively. [2]. Taxes are contributions paid to the state that are obligatory for those who are required to pay them, in accordance with the law, without receiving any direct benefits. The purpose of taxes is to finance general expenditures

related to the state's duties, such as providing public services, enforcing fair laws, and maintaining national security and order.[3].

The implementation of a Geographic Information System (GIS) enables tax authorities to manage property data more efficiently through clear spatial visualization, helping to identify properties that are unregistered or improperly taxed. Additionally, GIS improves the accuracy of tax assessments by integrating geographical information such as location, land size, and property value, thereby reducing errors and enhancing the fairness of tax collection. This implementation also contributes to increased revenue by detecting properties that should be taxed but have not yet been identified. Furthermore, spatial data from GIS provides valuable information for better urban planning, supporting decision-making related to sustainable and efficient urban development. [4].

Previous research on restructuring has demonstrated that the results of database restructuring include several positive aspects, such as the ability to perform DML operations that were previously impossible, improved database performance through schema optimization, and data alignment with changing business needs. Additionally, restructuring reduces the impact of changes on applications, providing flexibility and adaptability of the system to technological changes and user requirements, and enhances data integrity by ensuring consistency and accuracy. [5].

In previous research focusing on application performance, it was stated that there are several assessment points from the performance index, namely that a low response time indicates good application performance because it reflects the system's ability to respond to user requests quickly. [6]. A high throughput level indicates the system's efficiency in completing more operations within the same amount of time, reflecting the system's capability to handle a high workload. [7]. Scalability evaluation involves assessing the relationship between system resources and workload growth, where a scalable system can handle increased workloads without experiencing significant performance degradation. [8]. By monitoring and evaluating these performance indices, researchers can identify areas that need optimization to improve the performance and efficiency of web applications. [9]. Optimal performance will enhance application access speed. This increased speed will make users feel more comfortable while using the application. [10].

Previous research focused on query optimization in MySQL databases with the aim of improving data retrieval response time through query restructuring. This study evaluated eight clause models in SELECT queries, including SELECT-SYMBOL OPERATOR, IN, INNER JOIN-WHERE, DISTINCT, EXISTS, NOT IN-LEFT JOIN, OR-UNION, and LIKE, to determine their impact on query response time. Additionally, the research examined how query restructuring could be performed to enhance the effectiveness and efficiency of queries compared to the initial queries, and analyzed the response time of optimized queries to assess the efficiency improvements achieved. [11].

The JMeter application is a performance testing tool that Highly flexible and capable of simulating many users simultaneously, allowing for a comprehensive evaluation of application performance. [12]. JMeter's ability to adjust the number of users in testing is highly beneficial for organizations as it can replicate real-world application usage conditions [13]. During testing, JMeter is designed to measure performance both at the user interface level and within the system itself, providing in-depth insights into how the application will function in a production environment. [13]. JMeter provides several metrics for assessment, such as latency, connect time, median, standard deviation, and throughput, which can serve as benchmarks for testing. [14].

Several previous studies have included in-depth analyses of various aspects of database performance in the context of Geographic Information Systems (GIS), focusing on response time, throughput, standard deviation of response time, error rate, and the size of data received and sent. Response time, measured in seconds, is a key metric for assessing how quickly the system responds to service requests, where a short response time indicates high efficiency and better user satisfaction, while a long response time may signal performance issues. [15]. The research also evaluates the minimum and maximum response times in distributed database systems (DDBS) to ensure that all replicas are updated promptly. [16]. Additionally, the standard deviation of response time is measured to assess the variability in response times, which helps in understanding the consistency of system performance. [17]. Error rates, including mean squared error and maximum error rate, are analyzed to assess the accuracy of estimation methods in depicting the statistical profile of the system.

[18] througput, which measure how many queries can be executed per second, serves as a key indicator of the system's efficiency in handling high workloads. [19]. The amount of data received and sent, as well as the average byte size, are also measured to provide an overview of communication activity and data access patterns within the database, as well as to evaluate the efficiency of the SQL queries used.[20][21].

2. RESEARCH METHOD

Describe This research was conducted with testing stages as shown in Figure 1. Prior to testing, several preparation steps were necessary to ensure accuracy and reliability. The final stage involved a thorough analysis of the results, which were obtained from benchmarking outcomes. This comprehensive approach ensured that the data gathered was robust and provided meaningful insights into the database performance.

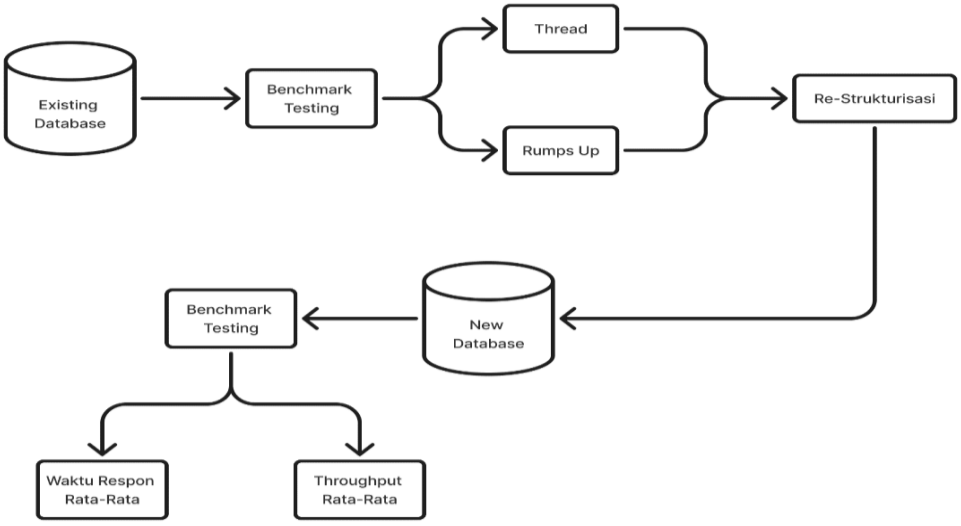

Figure 1. database testing flow

2.1 Existing Databse

The existing database serves as the primary focus of this research (see Table 1). This database, which is already in use, becomes the target for analysis and evaluation to identify deficiencies and potential performance improvements. At the initial stage of the research, the existing database will be migrated to a local host environment to separate it from the active host, ensuring that the analysis can proceed without disrupting the ongoing system operations. Once the migration is complete, the structure and performance of the database will be thoroughly examined to uncover issues such as redundancy, inconsistencies, and undesirable data dependencies. This detailed analysis aims to highlight areas needing enhancement and to provide a foundation for optimizing database performance.

Table 1. Data Existing Database

No	Component	Number
1	Table	51
2	Row data	625,589
3	Row data/table	12266.45
4	Table null	3
5	Attribute	247
6	Average: Attribute/table	4.84

2.2 Benchmarking testing

In the benchmarking phase, researchers use pre-designed scenarios to evaluate the database performance based on threadsand ramp-up variables. A threadsrefers to the smallest unit of execution in processing that allows multiple tasks to be performed simultaneously within an application. Meanwhile, ramp-up refers to the amount of workload or transactions sent to the database at a given time. Researchers conduct the evaluation with various combinations of threadscounts and ramp-up levels, specifically 100, 200, and 300 threads, and 200, 400, and 600 ramp-ups. The testing utilizes a single primary query employed in the system.

The query is used to calculate the total Tax Object Selling Value (NJOP) for each tax object by considering both the land value and the building value. This process is carried out through several subqueries that separately compute the land NJOP and the building NJOP based on area and the corresponding NJOP per square meter. The results from these two subqueries are then combined and summed to obtain the total NJOP for each tax object.

Query1: Total NJOP per Property Query

```
SELECT nomor_objek_pajak, SUM(njop) AS njop FROM (
      SELECT nomor_objek_pajak, njop FROM (
            SELECT nomor_objek_pajak,
            luas_bumi * njop_bumi.njop_per_meter AS njop FROM objek_pajak,
            njop_bumi
            WHERE kode_kelas_tanah = njop_bumi.kode_kelas_bumi) as njbumi
            UNION ALL SELECT nomor_objek_pajak, njop FROM (
                  SELECT nomor_objek_pajak,
                  luas_bangunan * njop_bangunan.njop_per_meter AS njop FROM objek_pajak,
                  njop_bangunan
                  WHERE objek_pajak.kode_kelas_bangunan = njop_bangunan.kode_kelas_bangunan)
                  AS njopbangunan
) AS jj GROUP BY nomor_objek_pajak;
```

Re-Structuring Database

In the database restructuring phase, this stage involves improving and optimizing the existing database structure to enhance efficiency, performance, and scalability. This process includes reorganizing tables and indexes to ensure that the database meets the needs effectively. Through this stage, redundancy can be eliminated, data integrity improved, and query response times accelerated, thereby supporting existing operations.

2.3 New Database

In the new database phase, researchers have made improvements from the previous stage, having undergone a restructuring process. The metadata for the new database is provided in Table 2.

This table outlines the key characteristics of the new database. It shows that the database consists of 51 tables with a total of 626,148 rows. On average, each table contains approximately 12,277 rows. The database includes 247 attributes, with an average of about 4.84 attributes per table. Additionally, there are 3 tables that contain null values. These metrics provide insights into the structure and complexity of the restructured database.

Table 2. Data New Database

No	Component	Number
1	Table	51
2	Row data	626,148
3	Row data/table	12277.41176
4	Table null	3
5	Attribute	247
6	Average: attribute/table	4.84

2.3 Database testing

In the new database benchmarking phase, researchers tested the database that had been improved through data cleaning and normalization processes. This testing was conducted using the same scenarios as in the previous stage to ensure result consistency. The two main outcomes measured in this phase are the average response time and average throughput. The average response time measures how quickly the database can respond to a given query, while the average throughput measures the number of transactions or operations that the database can process within a specific period.

3. RESULT AND DISCUSSION

This section presents the results of the research conducted to evaluate the performance of the database after undergoing the improvement process. The data obtained from the benchmarking phase are used to measure and compare the average response time and throughput before and after optimization. These results are detailed in the following section, providing a clear picture of the performance improvements achieved and identifying areas that still require further attention.

3.1 Existing Database

In this phase of testing the existing database, the researchers obtained results from the tests conducted using the pre-designed scenarios. The results of the testing under Scenario 1 are detailed in Table 3. This table shows various performance metrics for different threadscounts: 100, 200, and 300 threads. For instance, the average response time, minimum and maximum response times, standard deviation of response times, error percentage, average throughput, amount of data received, and average byte size are provided for each threads count. The data in Table 3 to 5 provide insights into the database's performance under the specified conditions, allowing for a comparison of how different threads configurations impact performance metrics.

Table 3. Testing of the Old Database Scenario 1

Aspect	**skenario 1 (200 rumps up)**		
	100 threads	**200 threads**	**300 threads**
Database Performance			
Average Response Time	1953	1344	1342
Minimum Response Time	1580	737	737
Maximum Response Time	3137	3137	3137
Standard Deviation of Response Time	153.33	429.95	355.78
Error Percentage	0.00%	0.00%	0.00%
Throughput			
Average Throughput	0.50034	0.17315	0.2464
Amount of Data Received	907.1	313.92	446.72
Amount of Data Sent	0	0	0
Average Byte Size	1856479	1856479	1856479

Table 4. Testing Result of the Old Database Scenario 2

Aspek	**skenario 2 (400 rumps up)**		
	100 threads	**200 threads**	**300 threads**
Database Performance			
Average Response Time	1328	1298	1266
Minimum Response Time	737	737	737
Maximum Response Time	3137	3137	3137
Standard Deviation of Response Time	342.83	315.29	288.15
Error Percentage	0.00%	0.00%	0.00%
Throughput			
Average Throughput	0.21967	0.24192	0.27113
Amount of Data Received	398.26	438.59	491.55
Amount of Data Sent	0	0	0
Average Byte Size	1856479	1856479	1856479

Table 5. Testing Result of the Old Database Scenario 3

Aspek	skenario 3 (600 rumps up)		
	100 threads	200 threads	300 threads
Database Performance			
Average Response Time	2085	1344	2082
Minimum Response Time	1098	737	1098
Maximum Response Time	3004	3137	3004
Standard Deviation of Response Time	200.93	429.95	146.99
Error Percentage	0.00%	0.00%	0.00%
Throughput			
Average Throughput	0.16777	0.17315	0.21411
Amount of Data Received	304.16	313.92	388.18
Amount of Data Sent	0	0	0
Average Byte Size	1856479	1856479	1856479

Based on the test results, the data indicates that the average response time generally improves with an increase in the number of threads, suggesting enhanced system efficiency. Specifically, as the number of threads increased from 100 to 300, the average response time decreased from 1953 milliseconds to 1342 milliseconds, reflecting better performance. However, the average throughput exhibits variability and does not consistently increase with the number of threads, with values ranging from 0.50034 for 100 threads to 0.2464 for 300 threads. This inconsistency suggests that there is room for further optimization to handle the number of operations per second more consistently across different workload levels. Despite improvements in response time, the variation in throughput highlights the need for additional adjustments to achieve stable performance across various threads counts.

3.2 Re-structuring Database

The database restructuring phase has produced a new database structure that aligns with the system's requirements (see Table 6 to 8). The following are some of the tables that have been restructured. This restructuring ensures that the database is optimized for performance, efficiency, and scalability, addressing previous inefficiencies and improving overall data management

Table 6. Information Schema Table Kelurahan

No	Attribute	Data Type			
		Before	Length	After	Length
1	kode_kelurahan	varchar	200	char	3
2	nama_kelurahan	char	50	varchar	14
3	kecamatan	char	50	varchar	14

Table 7. Information Schema Table Kecamatan

No	Attribute	Data Type			
		Before	Length	After	Length
1	kode_kecamatan	varchar	150	char	3
2	nama_kecamatan	varchar	150	varchar	14

Table 8. formation Schema Table Faktor Pengurang

No	Attribute	Data Type			
		Before	Length	After	Length
1	id_faktor	int	11	int	11
2	nomor_objek_pajak	varchar	50	char	20
3	faktor_pengurang	double		double	
4	date_created	datetime		datetime	
5	date_updated	datetime		datetime	

3.3 New Database dan Benchmark Testing

After completing the previous stages, the database is re-evaluated in this phase following the changes. The following are the results obtained after these modifications (see Table 9 to 11). This evaluation provides insights into the impact of the changes on the database's performance and efficiency. It highlights improvements as well as any areas that may still require further adjustment. The results offer a comprehensive view of the database's current status post-optimization.

From the evaluation results, the average response time decreased as the number of threads increased across all scenarios, indicating improved system efficiency in handling larger workloads. Meanwhile, the average throughput generally increased with the number of threads in some scenarios, although it was not consistent in all scenarios. This suggests that the system becomes faster in responding to requests with increased workloads. This analysis was conducted using the following equation 1:

$$\%change = \frac{new\ value - old\ value}{old\ value}\ x100 \qquad (1)$$

Table 9. Testing of the New Database Scenario 1

Aspect	Scenario 1 (200 rumps up)		
	100 threads	200 threads	300 threads
Database Performance			
Average Response Time	1824	2102	2143
Minimum Response Time	1335	1098	1098
Maximum Response Time	3137	3004	3004
Standard Deviation of Response Time	173.5	161.1	162.57
Error Percentage	0.00%	0.00%	0.00%
Throughput			
Average Throughput	0.2428	0.27009	0.35694
Amount of Data Received	440.18	489.67	647.13
Amount of Data Sent	0	0	0
Average Byte Size	1856479	1856479	1856479

Table 10. Testing Result of the New Database Scenario 2

Aspek	skenario 2 (400 rumps up)		
	100 threads	200 threads	300 threads
Database Performance			
Average Response Time	2093	1885	1695
Minimum Response Time	1098	703	703
Maximum Response Time	3004	3004	3004
Standard Deviation of Response Time	215.9	485.96	580.51
Error Percentage	0.00%	0.00%	0.00%
Throughput			
Average Throughput	0.24646	0.26439	0.30277
Amount of Data Received	446.83	479.34	548.91
Amount of Data Sent	0	0	0
Average Byte Size	1856479	1856479	1856479

Table 11. Testing Result of the New Database Scenario 3

Aspek	skenario 3 (600 rumps up)		
	100 threads	200 threads	300 threads
Database Performance			
Average Response Time	1633	1432	1328
Minimum Response Time	647	646	644
Maximum Response Time	3004	3004	3004
Standard Deviation of Response Time	608.45	666.28	671.76
Error Percentage	0.00%	0.00%	0.00%
Throughput			
Average Throughput	0.27354	0.25636	0.27364
Amount of Data Received	495.92	464.77	496.1
Amount of Data Sent	0	0	0
Average Byte Size	1856479	1856479	1856479

The evaluation was conducted using two primary metrics: average response time and average throughput. Performance tests were carried out in three different scenarios, each with varying numbers of threads (100, 200, and 300) and ramp-up periods (200, 400, and 600). These tests aimed to measure the databases' efficiency and capacity to handle different levels of workload. By simulating various real-world conditions, the tests provided insights into how the databases perform under different stress levels. The following section provides a comprehensive comparative analysis of the average response time and throughput across the different scenarios, highlighting the performance improvements and trade-offs observed after the restructuring process. The specific calculations for each scenario are shown to provide a clear understanding of the performance changes. This analysis is crucial for identifying the benefits of the restructuring process, such as enhanced data processing capacity and potential drawbacks, including increased response times under certain conditions. Overall, the evaluation underscores the importance of database optimization for achieving balanced and efficient data management systems.

a. Average Response Time

Skenario 1:
- 100 threads: $\left(\frac{1824-1953}{1953}\right) \times 100\% = -6.61\ \%$
- 200 threads: $\left(\frac{2102-1344}{1344}\right) \times 100\% = 56.40\ \%$
- 300 threads: $\left(\frac{2143-1342}{1342}\right) \times 100\% = 59.69\ \%$

Skenario 2:
- 100 threads: $\left(\frac{2093-1328}{1328}\right) \times 100\% = 57.61\ \%$
- 200 threads: $\left(\frac{1885-1298}{1298}\right) \times 100\% = 45.22\ \%$
- 300 threads: $\left(\frac{1695-1266}{1266}\right) \times 100\% = 33.89\ \%$

Skenario 3:
- 100 threads: $\left(\frac{1633-2085}{2085}\right) \times 100\% = -21.68\ \%$
- 200 threads: $\left(\frac{1432-1344}{1344}\right) \times 100\% = -36.22\ \%$
- 300 threads: $\left(\frac{1328-2082}{2082}\right) \times 100\% = -51.47\ \%$

b. Average Throughput

Skenario 1:
- 100 threads: $\left(\frac{0.2428-0.50034}{0.50034}\right) \times 100\% = -51.47\ \%$
- 200 threads: $\left(\frac{0.27009-0.17315}{0.17315}\right) \times 100\% = 55.99\ \%$
- 300 threads: $\left(\frac{0.35694-0.2464}{0.2464}\right) \times 100\% = 44.86\ \%$

Skenario 2:
- 100 threads: $\left(\frac{0.24646-0.21967}{0.21967}\right) \times 100\% = 12.20\ \%$
- 200 threads: $\left(\frac{0.26439-0.24192}{0.24192}\right) \times 100\% = 9.29\ \%$
- 300 threads: $\left(\frac{0.30277-0.27113}{0.27113}\right) \times 100\% = 11.67\ \%$

Skenario 3:
- 100 threads: $\left(\frac{0.27354-0.16777}{0.16777}\right) \times 100\% = 63.04\ \%$
- 200 threads: $\left(\frac{0.25636-0.17315}{0.17315}\right) \times 100\% = 48.06\ \%$
- 300 threads: $\left(\frac{0.27364-0.21411}{10.21411}\right) \times 100\% = 27.80\ \%$

c. Average Response Time Change

Percent Average = $\left(\frac{-6.61+56.40+59.69+57.61+45.22+33.89-21.68+6.55-36.22}{9}\right)$ = 21.65%

d. Average Throughput Change:

Percent Average = $\left(\frac{-51.47+55.99+44.86+12.20+9.29+11.67+63.04+48.06+27.80}{9}\right)$ = 24.60%

From the tests conducted, comparing the performance of the old database with the new database using two main metrics: average response time and average throughput. Performance tests were carried out in three different scenarios with varying numbers of threads (100, 200, and 300) and ramp-ups (200, 400, and 600). The following is the comparative analysis:

Average Response Time

Since the average percentage change in response time is positive (21.65%), this indicates that, overall,

the average response time increased after the database restructuring. This means that the time required to respond to requests increased on average across all scenarios and threads.

Average Throughput

With the average percentage change in throughput also being positive (24.60%), this indicates that the average throughput overall increased after the database restructuring. This means that the amount of data that can be processed in a given time period increased on average across all scenarios and threads.

4. CONCLUSION

From the analysis results, it can be concluded that the database restructuring resulted in a significant increase in throughput with an average increase of 24.60%, indicating that the system's capacity to process data increased overall. However, the restructuring also caused an average increase in response time of 21.65%, indicating that the time required to process requests increased. Overall, while the restructured database can process more data, it comes with the compromise of increased time needed to respond to requests.

5. REFERENCE

[1] G. O. Arocena and A. O. Mendelzon, "Restructuring documents, databases, and webs," *Theory Pract. Object Syst.*, vol. 5, no. 3, pp. 127–141, 1999, doi: 10.1002/(SICI)1096-9942(1999)5:3<127::AID-TAPO2>3.0.CO;2-X.

[2] G. Wiro Sasmito, "Penerapan Metode Waterfall Pada Desain Sistem Informasi Geografis Industri Kabupaten Tegal," *J. Inform. J. Pengemb. IT*, vol. 2, no. 1, pp. 6–12, 2017, doi: 10.30591/jpit.v2i1.435.

[3] D. P. S. M. A. Yosi Yulia, Ronni Andri Wijaya, "Pengaruh Pengetahuan Perpajakan, Kesadaran Wajib Pajak, Tingkat Pendidikan Dan Sosialisasi Perpajakan Terhadap Kepatuhan Wajib Pajak Pada Umkm Dikota Padang," *Sist. Inf.*, vol. 1, no. September, pp. 60–69, 2018, doi: 10.31933/JEMSI.

[4] A. Singh *et al.*, "Designing Geographic Information System Based Property Tax Assessment in India," *Smart Cities*, vol. 5, no. 1, pp. 364–381, 2022, doi: 10.3390/smartcities5010021.

[5] E. Domínguez, J. Lloret, Á. L. Rubio, and M. A. Zapata, "When and how to restructure a view-based relational database (extended version)," *CEUR Workshop Proc.*, vol. 639, no. September, pp. 137–150, 2010, doi: 10.1007/978-3-642-15576-5.

[6] I. Y. Andhica and D. Irwan, "Performa Kinerja Web Server Berbasis Ubuntu Linux Dan Turnkey Linux," *PIKSEL Penelit. Ilmu Komput. Sist. Embed. Log.*, vol. 5, no. 2, pp. 68–78, 2018, doi: 10.33558/piksel.v5i2.269.

[7] Z. X. and X. W. and Y.-C. Tu, "Power-Aware Throughput Control for Database Management Systems," *Icac*, pp. 315–324, 2013, [Online]. Available: https://www.usenix.org/conference/icac13/technical-sessions/presentation/xu_zichen

[8] I. Gorton, J. Klein, and A. Nurgaliev, "Architecture Knowledge for Evaluating Scalable Databases," *Proc. - 12th Work. IEEE/IFIP Conf. Softw. Archit. WICSA 2015*, pp. 95–104, 2015, doi: 10.1109/WICSA.2015.26.

[9] Q. Wu and Y. Wang, "Performance testing and optimization of J2EE-based web applications," *2nd Int. Work. Educ. Technol. Comput. Sci. ETCS 2010*, vol. 2, pp. 681–683, 2010, doi: 10.1109/ETCS.2010.583.

[10] M. Mardiana, "Implementasi User Satisfaction Model Dalam Mengukur Kualitas Website," *MATRIK J. Manajemen, Tek. Inform. dan Rekayasa Komput.*, vol. 19, no. 2, pp. 266–272, 2020, doi: 10.30812/matrik.v19i2.711.

[11] O. M. I. Tavares, S. M. Rangkoly, S. B. Desy Bawan, E. Utami, and M. S. Mustafa, "Analisis Perbandingan Performansi Waktu Respons Kueri antara MySQL PHP 7.2.27 dan NoSQL MongoDB," *J. Teknol. Inf.*, vol. 4, no. 2, pp. 303–313, 2020, doi: 10.36294/jurti.v4i2.1695.

[12] Apache, "Apache JMeter - Apache JMeterTM," 2014. https://jmeter.apache.org/#:~:text=Apache JMeter may be used,performance under different load types.

[13] J. Wang and J. Wu, "Research on performance automation testing technology based on JMeter," *Proc. - 2019 Int. Conf. Robot. Intell. Syst. ICRIS 2019*, pp. 55–58, 2019, doi: 10.1109/ICRIS.2019.00023.

[14] Apache Software Foundation, "Apache JMeter Glossary." https://jmeter.apache.org/usermanual/glossary.html#:~:text=Throughput is calculated as requests,)%2F (total time).

[15] K. D. Kang, J. Oh, and S. H. Son, "Chronos: Feedback control of a real database system performance," *Proc. - Real-Time Syst. Symp.*, no. January 2008, pp. 267–276, 2007, doi: 10.1109/RTSS.2007.16.

[16] H. W. Maalouf and M. K. Gurcan, "Minimisation of the update response time in a distributed database system," *Perform. Eval.*, vol. 50, no. 4, pp. 245–266, 2002, doi: 10.1016/S0166-5316(02)00085-8.

[17] S. Sen, A. Dutta, A. Cortesi, and N. Chaki, "A new scale for attribute dependency in large database systems," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7564 LNCS, pp. 266–277, 2012, doi: 10.1007/978-3-642-33260-9_23.

[18] M. V. Mannino, P. Chu, and T. Sager, "Statistical Profile Estimation in Database Systems," *ACM*

Comput. Surv., vol. 20, no. 3, pp. 191–221, 1988, doi: 10.1145/62061.62063.

[19] O. W. Purbo, Sriyanto, Suhendro, R. A. Aziz, and R. Herwanto, "Benchmark and comparison between hyperledger and MySQL," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 18, no. 2, pp. 705–715, 2020, doi: 10.12928/TELKOMNIKA.v18i2.13743.

[20] N. Khamphakdee, N. Benjamas, and S. Saiyod, "Performance Evaluation of Big Data Technology on Designing Big Network Traffic Data Analysis System," *Proc. - 2016 Jt. 8th Int. Conf. Soft Comput. Intell. Syst. 2016 17th Int. Symp. Adv. Intell. Syst. SCIS-ISIS 2016*, no. November 2017, pp. 454–459, 2016, doi: 10.1109/SCIS-ISIS.2016.0103.

[21] J. L. Lo, L. A. Barroso, S. J. Eggers, K. Gharachorloo, H. M. Levy, and S. S. Parekh, "Analysis of database workload performance on simultaneous multithreaded processors," *Conf. Proc. - Annu. Int. Symp. Comput. Archit. ISCA*, pp. 39–50, 1998, doi: 10.1145/279361.279367.