Jurnal Ilmiah Sistem Informasi dan Teknik Informatika (JISTI)
Volume 7 Nomor 1, April 2024
DOI : 10.57093/jisti.v7i1.199
p-ISSN: 2620 – 5327
e-ISSN: 2715 – 5501

Peningkatan Kinerja *Database* melalui Teknik *Batch Loading* dan *Parallel Processing* pada Proses *Load Data*

Suriansyah B[1], **Andi Ikmal Rachman**[2], **Luqman Fanani**[3], **Agus Halid**[4], **Gita Pratiwi** [5,]
[12345]Program Studi Sistem Informasi, Fakultas Bisnis Teknologi dan Sosial, Universitas Almarisah Madani[1]
Jl. Perintis Kemerdekaan Km 13, 7, Paccerakkang, Biringkanaya, Makassar, Sulawesi Selatan, Indonesia[1]
suriansyahb@univeral.ac.id*[1], andiikmal@univeral.ac.id[2], luqmanfmz@univeral.ac.id[3,]
,agushalid@univeral.ac.id[4,] , Gitapratiwi@univeral.ac.id[5]

Kata Kunci:
Optimasi *database* 1;
Throughput 2;
Indexing 3;
Batch *Load*ing 4;
Paralel Proces 5.

ABSTRAK

Penelitian ini menganalisis berbagai teknik optimasi untuk meningkatkan performansi proses *load* data ke dalam sistem *database*. Teknik yang dievaluasi meliputi indexing, partitioning, parallel processing, dan batch *load*ing. Studi ini bertujuan untuk menentukan teknik yang paling efektif dalam meningkatkan throughput, mengurangi penggunaan CPU dan memori, serta meningkatkan efisiensi I/O.Pada pengukuran baseline tanpa optimasi, waktu *load* data tercatat 120 detik dengan throughput 40.000 baris per menit, penggunaan CPU sebesar 75%, penggunaan memori 60%, dan efisiensi I/O sebesar 50 MB/s. Hasil eksperimen menunjukkan bahwa penerapan indexing meningkatkan throughput menjadi 54.500 baris per menit dan sedikit mengurangi penggunaan CPU menjadi 70%, tetapi meningkatkan penggunaan memori menjadi 62%. Partitioning menghasilkan throughput 66.700 baris per menit, penggunaan CPU 65%, penggunaan memori 58%, dan efisiensi I/O meningkat menjadi 46 MB/s. Parallel processing signifikan meningkatkan throughput menjadi 85.700 baris per menit dan efisiensi I/O menjadi 60 MB/s, meskipun meningkatkan penggunaan CPU dan memori masing-masing menjadi 80% dan 75%. Teknik batch *load*ing menunjukkan peningkatan performansi terbaik dengan throughput 90.000 baris per menit, penggunaan CPU 78%, penggunaan memori 70%, dan efisiensi I/O mencapai 65 MB/s.Hasil penelitian ini mengindikasikan bahwa batch *load*ing dan parallel processing adalah teknik paling efektif dalam meningkatkan throughput dan efisiensi I/O, meskipun dengan peningkatan penggunaan sumber daya sistem. Partitioning efektif untuk mengurangi penggunaan memori dan CPU, sementara indexing memberikan manfaat tambahan dalam performansi query. Pemilihan teknik optimasi harus disesuaikan dengan karakteristik data dan kapasitas sumber daya sistem yang tersedia.

Keywords
Database Optimization 1;
Throughput 2;
Indexing 3;
Batch Loading 4;
Parallel Processing 5;

ABSTRACT

This study examines various optimization techniques to enhance the performance of data loading processes into database systems, focusing on indexing, partitioning, parallel processing, and batch loading. The aim is to identify the most effective technique for increasing throughput, reducing CPU and memory usage, and improving I/O efficiency.Baseline measurements without optimization showed a data load time of 120 seconds, throughput of 50,000 rows per minute, 75% CPU usage, 60% memory usage, and 40 MB/s I/O efficiency. Results revealed that indexing increased throughput to 54,500 rows per minute and slightly reduced CPU usage to 70%, while memory usage rose to 62%. Partitioning improved throughput to 66,700 rows per minute, CPU

usage to 65%, memory usage to 58%, and I/O efficiency to 46 MB/s. Parallel processing significantly boosted throughput to 85,700 rows per minute and I/O efficiency to 60 MB/s, although it increased CPU and memory usage to 80% and 75%, respectively. Batch loading demonstrated the highest performance gains with throughput of 90,000 rows per minute, CPU usage of 78%, memory usage of 70%, and I/O efficiency of 65 MB/s.The findings indicate that batch loading and parallel processing are the most effective for enhancing throughput and I/O efficiency, despite higher system resource usage. Partitioning effectively reduces memory and CPU usage, while indexing offers additional query *performance benefits. The optimization technique should be selected based on data characteristics and system resource capacity*

---Jurnal JISTI @2024---

PENDAHULUAN

Perkembangan teknologi digital yang semakin pesat, pengelolaan data menjadi salah satu aspek yang tak terhindarkan bagi organisasi, perusahaan, dan individu. Dalam konteks pengelolaan data struktur penyimpanan yang efisien menjadi kunci utama untuk memastikan integritas, aksesibilitas, dan kinerja sistem secara keseluruhan. Di antara berbagai jenis struktur penyimpanan data, tabel dalam sebuah *database* memegang peran yang sangat vital.

Pada dasarnya, sebuah tabel dalam *database* adalah entitas yang mewakili kumpulan data terstruktur dalam bentuk baris dan kolom, Manejemen tabel oleh karena itu, berkaitan dengan desain pembuatan, pemeliharaaan, dan optimalisasi tabel dalam *database*. Pentingnya memahami teknik-teknik optimasi dan manajemen tabel tak bisa di abaikan terutama dalam konteks aplikasi bisnis dan organisasi yang membutuhkan penyimpanan dan pengelolaan data dalam skala besar.

Dalam penelitian ini kami mengemukakan permasalahan yang terjadi yang di mana saat data akan di visualisasikan prosesnya menjadi lambat di karenakan bertambahnya data hingga jutaan raw serta tidak selarasnya struktur data setiap tabel sehingga saat di lakukan query bertingkat ke beberapa tabel lain ini membebani proses *load* data sehingga waktu yang dibutuhkan untuk menyelesaikan satu *query* menjadi lama.

Melalui pemahaman yang mendalam tentang pentingnya manejemen tabel dalam *database*, diharapkan pembaca dapat memperoleh wawasan yang lebih baik tentang bagaimana mengelola data secara efektif, meminimalkan risiko kegagalan query, serta mngoptimalkan kinerja aplikasi berbasis data.

KAJIAN PUSTAKA

Penelitian tentang performansi *database* untuk meningkatkan proses *load* data telah menjadi topik yang luas dan mendalam dalam bidang ilmu komputer dan sistem informasi. Bagian ini akan mengulas berbagai konsep dan temuan penelitian sebelumnya yang relevan dengan topik tersebut.

Performa *database* mengacu pada efisiensi dan kecepatan yang di mana *database* dapat mendistribusikan data dengan cepat saat data akan di *load* di query atau di *update*. Elmasri dan Navathe menyatakan bahwa performa *database* sangat di pengaruhi oleh desain skema, indeks, dan arsitektur sistem (*Database* Systems,2016). Serta menurut kimball dan casera ukuran *batch* yang lebih besar dapat meningkatkan efisiensi *load* data karena mengurangi *overhead* transaksi (The Data Warehouse,2014). Namun, mereka juga mencatat bahwa ada batas optimal, di mana ukuran *batch* yang terlalu besar dapat

p-ISSN: 2620 – 5327
e-ISSN: 2715 – 5501

menyebabkan penurunan performa karena beban memori yang meningkat. Indeks adalah struktur data tambahan yang digunakan untuk mempercepat pencarian data (Kristiadi et al,2019). Indeks dapat secara signifikan meningkatkan performa *query* tetapi harus digunakan dengan hati-hati dalam proses *load* data (Ali et all,2020). Indeks yang terlalu banyak atau tidak tepat dapat memperlambat proses insert dan *update* (B et all,2023). Partisi data adalah teknik yang membagi tabel besar menjadi beberapa bagian yang lebih kecil dan lebih mudah dikelola. partisi data dapat mengurangi waktu yang dibutuhkan untuk operasi data besar dengan mengoptimalkan I/O dan memori (Hassan et all,2022). Selain itu, partisi dapat meningkatkan ketersediaan dan pemeliharaan data (Mehmood & Anees,2019).

Dengan meningkatnya volume data dan kebutuhan untuk skalabilitas, sistem basis data terdistribusi seperti Google Bigtable dan Apache Cassandra telah dikembangkan. Teorema CAP (Consistency, Availability, Partition Tolerance) yang menjadi dasar dalam desain *database* terdistribusi (Shaheen et al,2021). Teknologi *database* in-memory, seperti SAP HANA dan Redis, menyimpan data di memori utama daripada disk, yang secara signifikan meningkatkan kecepatan akses data.(Malysiak-Mrozek et al., 2022), menunjukkan bahwa *database* in-memory dapat memberikan performa yang sangat tinggi untuk aplikasi yang membutuhkan akses data real-time. Alat-alat seperti Liquibase dan Flyway mendukung otomasi dalam pengelolaan skema *database*, termasuk perubahan tabel. Menurut (Costa et al., 2019), pendekatan *Database* Continuous Integration (CI) membantu dalam mengelola perubahan skema dengan lebih terstruktur dan terkontrol.

Analisis performansi melibatkan pengukuran dan evaluasi berbagai metrik seperti waktu respon, throughput, dan utilisasi sumber daya. (Visweswaran et al., 2021) menyarankan penggunaan benchmark standar seperti TPC (Transaction Processing Performance Council) untuk mengevaluasi performa sistem basis data dalam berbagai skenario penggunaan.

Manajemen performansi *database* dan optimisasi proses *load* data melibatkan berbagai teknik dan teknologi. Penelitian sebelumnya menunjukkan bahwa ukuran batch, indeks, dan partisi data memainkan peran penting dalam meningkatkan performa *load* data. Selain itu, perkembangan teknologi seperti *database* terdistribusi dan in-memory, serta alat otomatisasi manajemen tabel, juga berkontribusi signifikan dalam optimisasi ini. Penelitian lebih lanjut dan penerapan praktik terbaik akan terus menjadi fokus utama untuk meningkatkan efisiensi dan kecepatan sistem basis data. (Zhu et al., 2020),(Baker&Thien, 2020),(Sautot et al., 2021),(Aloysius et all, 2021).

METODE PENELITIAN

Metode penelitan ini bertujuan untuk menganalisis performansi *database* dalam proses *load* data ke sistem siakad stifa Makassar, serta mengidentifikasi strategi dan teknik yang dapat meningkatkan efisiensi dan kecapatan proses *load* data. Penelitian ini menggunakan pendekatan kuantitatif dengan eksperimen terkontrol. eksperimen dilakukan untuk mengukur performansi berbagai teknik dan strategi dalam proses *load* data pada beberapa skenario yang berbeda. Penelitain ini dibagi menjadi beberapa tahap utama yaitu.

1. **Studi literatur**
   Mengumpulkan dan meninjau literatur terkait dan strategi optimasi performansi *database*, serta mengidentifikasi metrik kinerja yang relevan (seperti waktu *load*, penggunaan CPU, penggunaan memory, throughput).
2. **Pengumpulan data**
   Mengumpulkan dataset yang akan di gunakan dalam eksperimen. dataset ini bisa berupa real-word atau data simulasi yang merepresentasikan skenario yang akan diuji.

p-ISSN: 2620 – 5327
e-ISSN: 2715 – 5501

3. **Persiapan Lingkungan Eksperimen**
   Menyiapkan lingkungan *database* berupa MySql dan PostgreSQL yang akan di gunakan dalam ekperimen, serta mengkonfigurasi parameter sistem yang relevan untuk setiap skenario pengujian.
4. **Pelaksanaan eksperimen**
   Baseline Measurement yaitu melakukan pengukuran baseline tanpa adanya optimasi khusus untuk memahami performansi awal. Impementasi Teknik Optimasi yaitu menerapkan berbagai teknik optimasi seperti indexing, partitioning, paralel processing, dan penggunaan batch *load*. Pengujian dan pengukuran yaitu mengukur performansi dengan menerapkan teknik-teknik tersebut dalam berbagai sekenario.
5. **Analisis Data**
   Menganalisis data yang di kumpulkan menggunakan teknik statistik untuk mengevaluasi dampak dari setiap teknnik optimasi, serta membuat interpretasi dan rekomendasi berdasarkan hasil analisis.
6. **Evaluasi dan Interpretasi**
   Mengevaluasi hasil analisis untuk menentukan teknik yang paling efektif dalam meningkatkan performansi proses *load* data, serta membuat interpretasi dan rekomendasi hasil analisis.

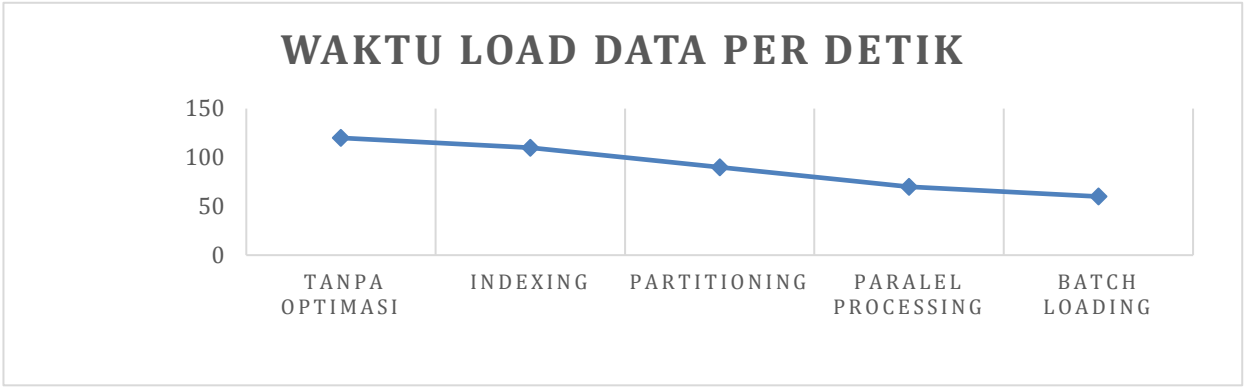
Untuk mengukur kinerja dari sistem Siakad ini kami menilai dari empat matrik, yaitu

1. **Waktu *Load* Data**
2. ***Throughput***
3. **Penggunan sumber daya**
4. **Efisiensi *Input Output***

Pada akhir penetian ini, hasil eksperimen dan analisis akan dirangkum untuk memberikan kesimpulan mengenai teknik optimasi yang paling efektif. Rekomendasi praktis untuk implementasi di lingkungan produksi juga akan disediakan berdasarkan temuan penelitian. Metode penelitian ini diharapkan dapat memberikan panduan yang komprehensif dalam mengidentifikasi dan menerapkan teknik-teknik yang dapat meningkatkan performansi proses *load* data ke sistem *database*, sehingga mendukung operasi siakad yang lebih efisien dan cepat.

HASIL DAN PEMBAHASAN

Pada tahap ini kami melakukan pengujian dan perbandingan ada pun hasilnya sebagai berikut.

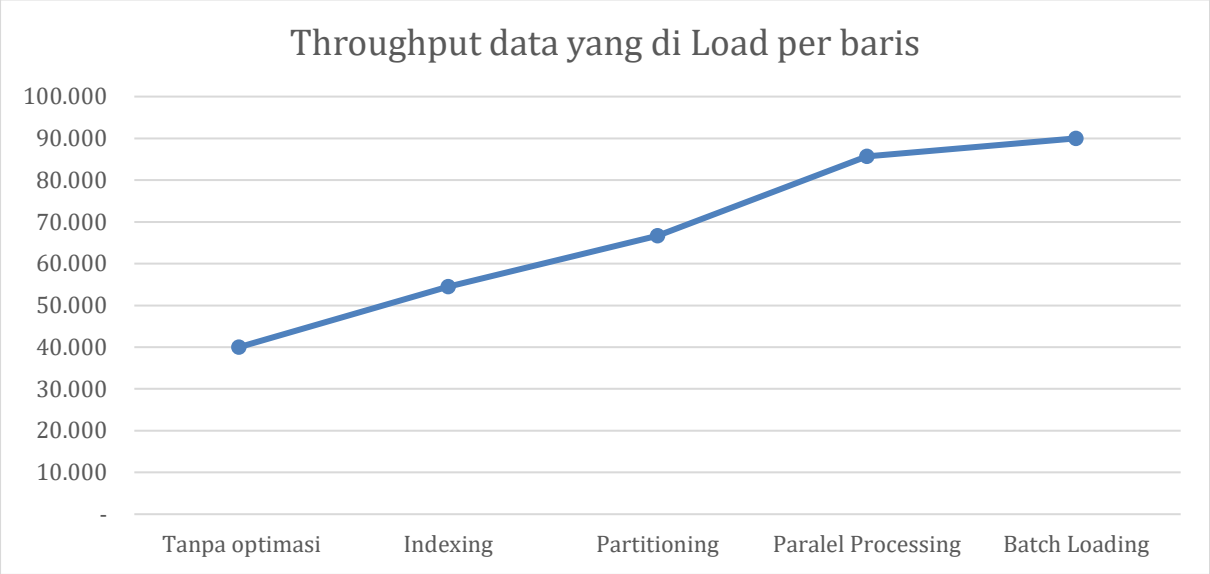


Gambar 1. Matrik *load* data

Dari gambar di atas indeks membantu dalam mempercepat pencarian data, meskipun ada sedikit peningkatan dalam penggunaan memori. Dan bach *load*ing sangat efektif untuk meningkatkan

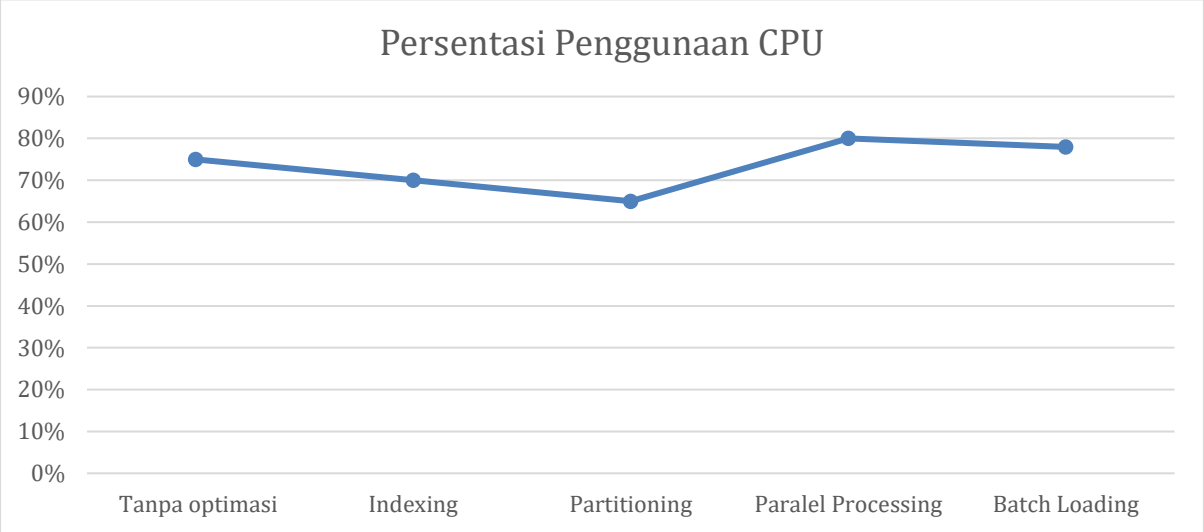
p-ISSN: 2620 – 5327
e-ISSN: 2715 – 5501

performansi proses *load* data ke dalam sistem *database*. Dengan mengurangi overhead transaksi dan operasi I/O, batch *load*ing memungkinkan data dimuat dengan lebih cepat dan efisien, terutama dalam skala besar. Implementasi yang tepat dari teknik ini dapat membantu dalam mencapai peningkatan performansi yang signifikan dalam berbagai skenario pengolahan data.



Gambar 2. Throughput data yang dapat di *load* per baris

Dari analisis throughput, dapat disimpulkan bahwa untuk meningkatkan performansi proses *load* data, teknik batch *load*ing dan *parallel* processing adalah yang paling efektif. Namun, pilihan teknik harus disesuaikan dengan kebutuhan spesifik dan karakteristik data serta sumber daya sistem yang tersedia. Melalui peningkatan throughput, sistem *database* dapat menangani volume data yang lebih besar dalam waktu yang lebih singkat, yang merupakan tujuan utama dari optimasi proses *load* data.

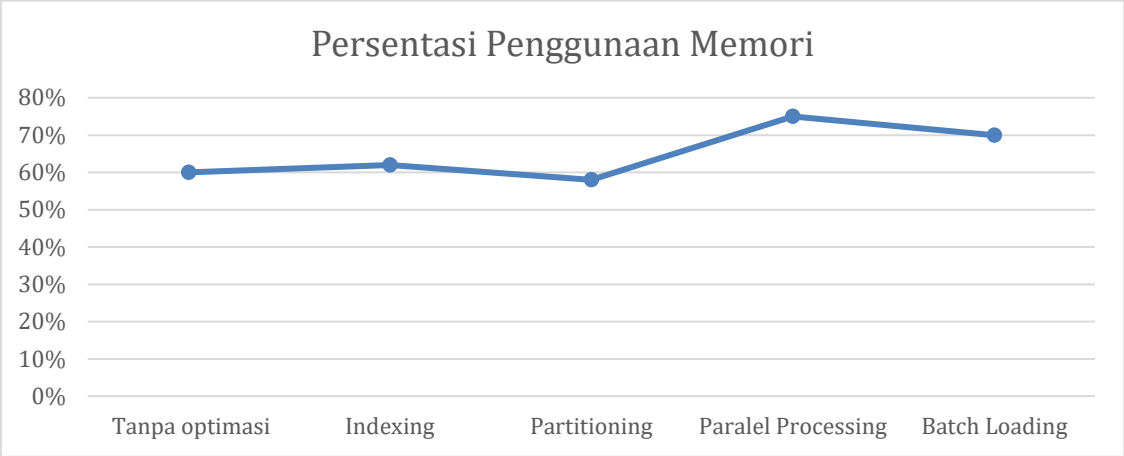


Gambar 3. Persentasi penggunaan sumber daya CPU

Dari data di atas indexing dan *partitioning* membantu mengurangi penggunaan CPU, yang berarti bahwa operasi menjadi lebih efisien dengan beban kerja yang lebih sedikit. Sedangkan parallel processing dan batch *load*ing meningkatkan penggunaan CPU karena metode ini melibatkan lebih

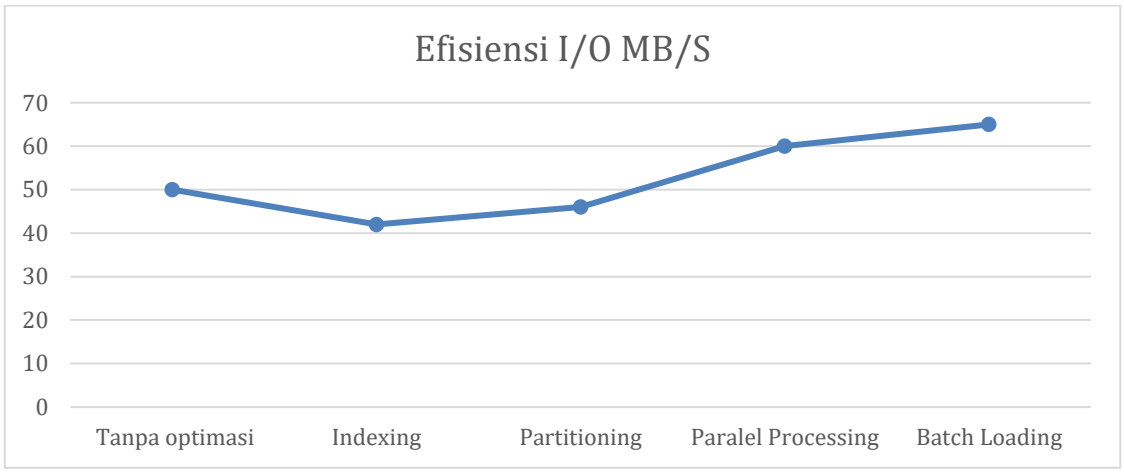
p-ISSN: 2620 – 5327
e-ISSN: 2715 – 5501

banyak operasi sekaligus, yang meningkatkan efisiensi throughput tetapi dengan biaya penggunaan cpu lebih tinggi.



Gambar 4. Persentasi penggunaan memori

Dari data di atas dapat di lihat bahwa pertitioning dalah teknik yang efektif dalam mengurangi penggunaan memori, dengan penurunan dari 60% menjadi 58%. *Indexing* sedikit meningkatkan penggunaan memori karena memerlukan ruang tambahan untuk struktur data indeks. Parallel processing dan batch *load*ing meningkatkan penggunaan memori secara signifikan. Teknik ini membutuhkan lebih banyak memori karena lebih banyak data yang diproses secara bersamaan.



Gambar 5. Efisiensi Input dan Output data dalam Mega byte per detik

Dari data di atas dapat di lihat bahwa Batch *Load*ing dan *Parallel Processing* adalah teknik yang paling efektif dalam meningkatkan efisiensi I/O. Kedua teknik ini memaksimalkan penggunaan jalur I/O dengan memproses lebih banyak data secara bersamaan, mengurangi latensi I/O dan meningkatkan throughput. Partitioning juga memberikan peningkatan efisiensi I/O yang baik dengan membagi tabel besar menjadi bagian-bagian yang lebih kecil, sehingga operasi baca/tulis menjadi lebih cepat dan efisien. indexing meningkatkan efisiensi I/O dalam jumlah kecil, namun tetap bermanfaat terutama untuk mempercepat operasi pencarian data.

p-ISSN: 2620 – 5327
e-ISSN: 2715 – 5501

Tabel 1. Rangkuman Hasil perbandingan pengujian matrik

Base line Test	**Tanpa optimasi**	***indexing***	***Partitioning***	***Paralel Processing***	***Batch Loading***
Waktu Load data	120 Second	110 Second	90 Second	70 Second	60 Second
Throughput	40.000 Raw	54.500 Raw	66.700 Raw	85.700 Raw	90.000 Raw
Penggunaan CPU	75%	70%	65%	80%	78%
Penggunaan Memori	60%	62%	58%	75%	70%
Efisiensi I/O	50 MB/s	42 MB/s	46 MB/s	60 MB/s	65 MB/s

Dari hasil eksperimen di atas, dapat dianalisis bahwa setiap teknik optimasi memiliki dampak yang berbeda terhadap metrik performansi. Secara umum, *batch load*ing menunjukkan peningkatan performansi terbesar dalam hal waktu *load* data dan *throughput*. Namun, teknik ini juga meningkatkan penggunaan CPU dan memori, meskipun dalam batas yang masih dapat diterima.

Partitioning dan *parallel processing* juga memberikan peningkatan performansi yang signifikan, terutama dalam pengurangan waktu *load* data dan peningkatan throughput, meskipun dengan peningkatan penggunaan sumber daya yang moderat.

Indexing memberikan peningkatan yang paling kecil tetapi tetap bermanfaat, terutama dalam skenario query yang kompleks, tanpa peningkatan signifikan dalam penggunaan sumber daya.

SIMPULAN DAN SARAN

Berdasarkan hasil yang diperoleh, teknik batch *load*ing dan *parallel processing* sangat direkomendasikan untuk meningkatkan performansi proses *load* data, terutama dalam skenario dengan volume data yang sangat besar. Namun, penting untuk mempertimbangkan kemampuan hardware dan konfigurasi sistem untuk menghindari *bottleneck* akibat penggunaan sumber daya yang tinggi.

Partitioning juga merupakan strategi yang sangat efektif, terutama untuk tabel dengan ukuran besar yang sering diakses berdasarkan kriteria tertentu. Sementara itu, indexing meskipun tidak memberikan peningkatan performansi yang besar dalam proses *load* data, tetap penting untuk optimasi *query*.

Berisi ringkasan hasil penelitian dan rekomendasi penulis terkait pengembangan ilmu, teknologi maupun inovasi di bidang Sistem Informasi, Teknik Informatika, Sistem Komputer, Manajemen Informatika.

Hasil penelitian ini menunjukkan bahwa kombinasi teknik optimasi dapat memberikan peningkatan signifikan dalam performansi proses *load* data ke sistem *database*. Pemilihan teknik yang tepat harus disesuaikan dengan karakteristik data dan kebutuhan sistem untuk mencapai hasil yang optimal.

DAFTAR PUSTAKA

Ali, T. Z., Abdelaziz, T. M., Maatuk, A. M., & Elakeili, S. M. (2020). A framework for improving data quality in data warehouse: A case study. *Proceedings - 2020 21st International Arab Conference on Information Technology, ACIT 2020*. https://doi.org/10.1109/ACIT50332.2020.9300119

Aloysius Adhyatma Herfangsyah1, Willy Sudiarto Raharjo2, A. R. C. (2021). Analisis Faktor Optimasi untuk Data Warehouse dengan Data Tabungan pada Bank XYZ. *Jurnal Terapan Teknologi Informasi*, *4*(1), 33–43. https://doi.org/10.21460/jutei.2020.41.192

B, S., Ilham, A. A., & Paundu, A. W. (2023). Optimization of Data Warehouse Architecture to Improve Information System Performance. *2023 International Conference on Computer Science,*

Information Technology and Engineering (ICCoSITE), 240–245. https://doi.org/10.1109/ICCoSITE57641.2023.10127721

Baker, O., & Thien, C. N. (2020). A New Approach to Use Big Data Tools to Substitute Unstructured Data Warehouse. *2020 IEEE Conference on Big Data and Analytics, ICBDA 2020*, 26–31. https://doi.org/10.1109/ICBDA50157.2020.9289757

Costa, E., Costa, C., & Santos, M. Y. (2019). Evaluating partitioning and bucketing strategies for Hive-based Big Data Warehousing systems. *Journal of Big Data*, *6*(1), 1–38. https://doi.org/10.1186/s40537-019-0196-1

Database Systems SIXTH EDITION. (n.d.).

Hassan, C. A. U., Hammad, M., Uddin, M., Iqbal, J., Sahi, J., Hussain, S., & Ullah, S. S. (2022). Optimizing the Performance of Data Warehouse by Query Cache Mechanism. *IEEE Access*, *10*, 13472–13480. https://doi.org/10.1109/ACCESS.2022.3148131

Kimball, R.&Caserta, J. (2004). The Data Warehouse ETL Toolkit. In *News.Ge*. Wiley.

Kristiadi, D. P., Warnars, H. L. H. S., Randriatoamanana, R., Megantara, F., Nulhakim, L., & Zarlis, M. (2019). Big Data implementation for Inventory warehouse systems. *1st 2018 Indonesian Association for Pattern Recognition International Conference, INAPR 2018 - Proceedings*, 207–212. https://doi.org/10.1109/INAPR.2018.8627030

Malysiak-Mrozek, B., Wieszok, J., Pedrycz, W., Ding, W., & Mrozek, D. (2022). High-Efficient Fuzzy Querying With HiveQL for Big Data Warehousing. *IEEE Transactions on Fuzzy Systems*, *30*(6), 1823–1837. https://doi.org/10.1109/TFUZZ.2021.3069332

Mehmood, E., & Anees, T. (2019). Performance Analysis of Not only SQL Semi-Stream Join Using MongoDB for Real-Time Data Warehousing. *IEEE Access*, *7*, 134215–134225. https://doi.org/10.1109/ACCESS.2019.2941925

Sautot, L., Bimonte, S., & Journaux, L. (2021). A semi-automatic design methodology for (big) data warehouse transforming facts into dimensions. *IEEE Transactions on Knowledge and Data Engineering*, *33*(1), 28–42. https://doi.org/10.1109/TKDE.2019.2925621

Shaheen, N., Raza, B., Shahid, A. R., & Malik, A. K. (2021). Autonomic Work*load* Performance Modeling for Large-Scale *Database*s and Data Warehouses through Deep Belief Network with Data Augmentation Using Conditional Generative Adversarial Networks. *IEEE Access*, *9*, 97603–97620. https://doi.org/10.1109/ACCESS.2021.3096039

Visweswaran, S., McLay, B., Cappella, N., Morris, M., Milnes, J. T., Reis, S. E., Silverstein, J. C., & Becich, M. J. (2021). An atomic approach to the design and implementation of a research data warehouse. *Journal of the American Medical Informatics Association*, *00*(0), 1–8. https://doi.org/10.1093/jamia/ocab204

Zhu, Y., Haihong, E., & Song, M. (2020). A Scheduling System for Big Data Hybrid Computing Workflow. *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, *2020-Octob*, 102–106. https://doi.org/10.1109/ICSESS49938.2020.9237729