Jurnal Pendidikan dan Teknologi Indonesia (JPTI)
Vol. 5, No. 2, Januari 2025, Hal. 393-402
DOI: https://doi.org/10.52436/1.jpti.671
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Analisis Perbandingan Response Time View dan Materialized View pada Database Oracle dalam Treasury Big Data DJPb Kemenkeu

Gahara Dijerja[*1], Raden Soepriadi[*2], Samidi[3]

[1,2,3]Universitas Budi Luhur, Magister Ilmu Komputer, Indonesia
Email: [1]2311601849@student.budiluhur.ac.id, [2]2311601690@student.budiluhur.ac.id,
[3]samidi@budiluhur.ac.id

Abstrak

Pengelolaan data dalam volume besar, seperti pada sistem Treasury Direktorat Jenderal Perbendaharaan (DJPb) Kementerian Keuangan, memerlukan teknik optimasi yang efektif untuk memastikan kinerja sistem yang optimal. Penelitian ini bertujuan membandingkan waktu respons antara penggunaan View dan Materialized View pada database Oracle dalam konteks Big Data DJPb. Berdasarkan eksperimen yang dilakukan, Materialized View memberikan peningkatan signifikan dalam kecepatan query, terutama untuk query yang kompleks dan berulang. Walaupun membutuhkan biaya penyimpanan tambahan dan pemeliharaan yang lebih tinggi, teknik ini terbukti lebih efisien dalam mendukung kebutuhan akses data yang cepat dan sering. Penelitian ini memberikan rekomendasi praktis untuk meningkatkan efisiensi pengelolaan Treasury Big Data dalam sistem keuangan negara, terutama dalam memilih teknik optimasi basis data yang sesuai. Implementasi Materialized View diharapkan dapat membantu DJPb mencapai tujuan efisiensi dan akurasi dalam pengambilan keputusan berbasis data.

Kata kunci: *Big Data, Database Oracle, Kecepatan Akses Data, Materialized View, Optimasi Basis Data, Response Time, Treasury, View.*

Comparative Analysis of Response Time Between Views and Materialized Views in Oracle Databases for Treasury Big Data of DJPb, Ministry of Finance

Abstract

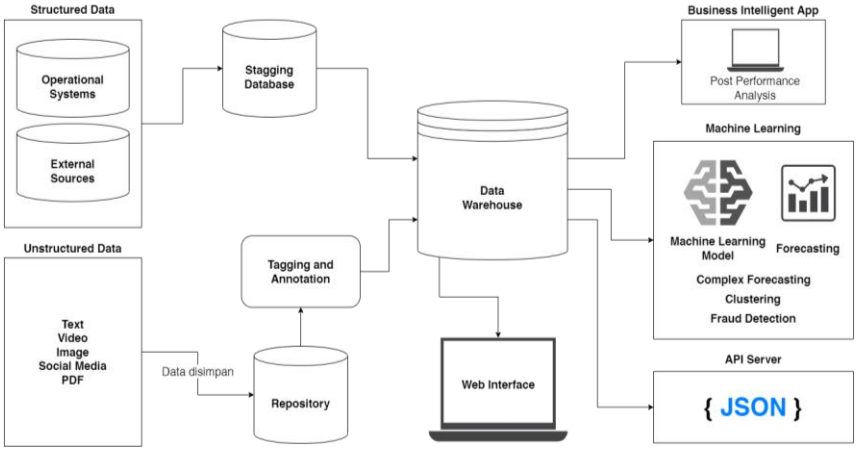
Data management at large volumes, such as in the Treasury system of the Directorate General of Treasury (DJPb) at the Ministry of Finance, requires effective optimization techniques to ensure optimal system performance. This study aims to compare the response times between using Views and Materialized Views in an Oracle database within the context of DJPb's Big Data. Based on the conducted experiments, Materialized Views provide a significant improvement in query speed, particularly for complex and repetitive queries. Although they require additional storage costs and higher maintenance, this technique has proven to be more efficient in supporting the need for fast and frequent data access. This study offers practical recommendations to enhance the efficiency of managing Treasury Big Data in the national financial system, especially in selecting the appropriate database optimization technique. The implementation of Materialized Views is expected to assist DJPb in achieving efficiency and accuracy in data-driven decision-making.

Keywords: *Big Data, Data Access Speed, Database Optimization, Materialized View, Oracle Database, Response Time, Treasury, View.*

1. PENDAHULUAN

Dalam era digital saat ini, Direktorat Jenderal Perbendaharaan (DJPb) Kementerian Keuangan Republik Indonesia menghadapi tantangan pengelolaan data keuangan dalam jumlah yang sangat besar dan terus berkembang. Data ini, dikenal sebagai Treasury Big Data, harus diolah dan dianalisis secara efisien untuk mendukung pengambilan keputusan yang cepat dan akurat dalam pengelolaan keuangan negara. Penerapan data analytics diharapkan mampu menjawab berbagai tantangan tersebut, sehingga mendukung implementasi Treasury Big Data yang memperkuat peran DJPb sebagai pengelola fiskal negara [1].

p-ISSN: 2775-4227
e-ISSN: 2775-4219



Gambar 1. Konsep Treasury Big Data

Pada sistem database yang digunakan, volume data yang besar sering kali menjadi tantangan bagi Database Management System (DBMS) [2]. Manfaat dari analisis data dalam skala besar kini semakin dirasakan. Penelitian sebelumnya telah menunjukkan bahwa eksplorasi basis data memerlukan informasi mengenai estimasi waktu respons query untuk membantu perencanaan eksplorasi database [3]. Misalnya, penelitian menunjukkan bahwa meningkatkan kinerja basis data melalui optimalisasi SQL, pemrosesan paralel, dan integrasi GPU merevolusi sistem manajemen basis data dengan mengoptimalkan kinerja kueri, memparallelkan pemrosesan data, serta mengintegrasikan GPU untuk mempercepat tugas analitik dan pembelajaran mesin [4]. Selain itu, pemanfaatan materialized view yang berasal dari hasil antara kueri yang sering dijalankan dapat secara signifikan mengurangi beban sistem [5]. Sebuah studi menunjukkan bahwa optimalisasi materialized view pada database Oracle dapat meningkatkan efisiensi hingga 40% dalam lingkungan data skala besar [6]. Sementara itu, penelitian menunjukan pemanfaatan materialized view dapat secara signifikan mengurangi beban sistem pada kueri yang berulang dengan menyimpan secara fisik hasil dari kueri kompleks, sehingga menghilangkan kebutuhan untuk pemrosesan yang mahal yang melibatkan join dan agregasi yang rumit [7]. Sebagai tambahan, pentingnya mengoptimalkan materialized view karena potensi beban tambahan yang dapat ditimbulkannya dalam sistem basis data skala besar [8]. Sebuah studi membahas bagaimana penggunaan hybrid views dapat memberikan tambahan dengan memungkinkan pengguna memanfaatkan kemampuan penalaran LLM bersama dengan kueri SQL terstruktur [9]. Selain itu, mengidentifikasi bahwa pengelolaan indeks yang tepat, khususnya melalui Dynamic Materialized View Index (DMVI), secara signifikan meningkatkan efisiensi pemrosesan materialized view untuk kueri progresif (Progressive Queries atau PQs) [10]. Pentingnya menyelaraskan desain materialized view dengan pola akses data sangat penting untuk memaksimalkan kinerja dalam data mart independen [11]. Metode eksekusi yang dioptimalkan untuk query dengan materialized views, yang mendukung pengambilan keputusan dalam lingkungan big data yang kompleks [12]. Penelitian lain menekankan peran view dalam aplikasi pelaporan dan kemampuannya untuk memodifikasi data, namun tidak mencakup metrik kinerja atau perbandingan terperinci yang berkaitan dengan waktu respons [13].

Meskipun demikian, banyak penelitian sebelumnya yang hanya fokus pada aplikasi umum dari materialized view tanpa mempertimbangkan kebutuhan spesifik seperti pengelolaan data keuangan berskala besar dalam konteks institusi pemerintah. Kesenjangan ini memberikan peluang untuk mengeksplorasi bagaimana materialized view dapat diimplementasikan secara optimal pada Treasury Big Data DJPb Kemenkeu. Penelitian ini mengisi celah tersebut dengan memberikan analisis yang lebih mendalam mengenai perbandingan waktu respons antara view dan materialized view pada database Oracle, khususnya untuk kebutuhan pengelolaan data keuangan di DJPb.

Mengelola Treasury Big Data dapat dilakukan dengan memanfaatkan database Oracle. Big data mengacu pada kumpulan data berskala besar dengan struktur yang beragam dan kompleks, yang menghadirkan tantangan dalam penyimpanan, analisis, dan visualisasi data [14]. Oracle, sebagai salah satu DBMS terkemuka, telah membuktikan keandalannya dalam menangani big data. Oracle 12c Database In-Memory, misalnya, memungkinkan tabel utama berbasis baris di disk memiliki representasi berbasis kolom dalam memori, yang menawarkan pendekatan baru dalam pemrosesan data dan eksekusi query [15]. Dengan memilih materialized view ini, metode yang diusulkan mencapai percepatan signifikan dalam pemrosesan kueri, menunjukkan peningkatan waktu pemrosesan kueri hingga 9,73 kali dibandingkan dengan pemrosesan ad-hoc [16]. Penelitian lain juga menunjukkan bahwa penggunaan strategi hybrid, khususnya algoritma Constrained Hybrid Ebola with COATI

p-ISSN: 2775-4227
e-ISSN: 2775-4219

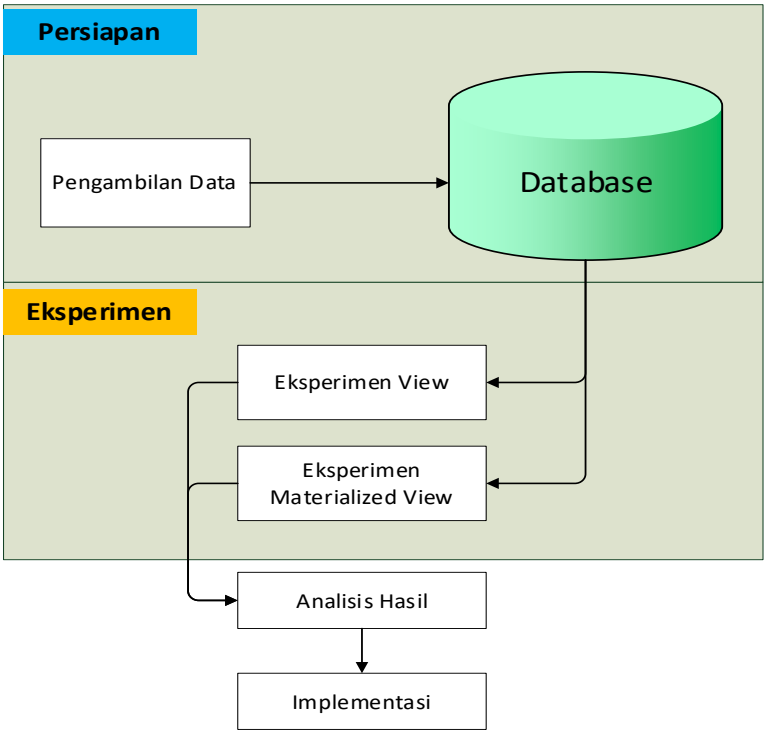
Optimization (CHECO) dalam kombinasi view dan materialized view dapat menghasilkan efisiensi tambahan dalam pengelolaan big data [17].

Untuk menutup kesenjangan penelitian ini, tujuan penelitian ini adalah untuk menganalisis perbandingan response time antara penggunaan view dan materialized view dalam database Oracle untuk meningkatkan efisiensi pengelolaan Treasury Big Data DJPb Kemenkeu. Penelitian ini juga bertujuan memberikan rekomendasi metode yang paling optimal berdasarkan analisis tersebut. Dengan demikian, penelitian ini diharapkan dapat berkontribusi pada efisiensi pengelolaan data keuangan negara dan pengambilan keputusan yang lebih baik oleh DJPb.

2. METODE PENELITIAN

Penelitian ini menggunakan metode eksperimen dengan desain before-and-after. Proses penelitian dilakukan dalam tiga tahap utama: persiapan, pelaksanaan eksperimen, dan analisis hasil. Langkah-langkah yang dilakukan meliputi pembuatan view dan materialized view berdasarkan data Treasury Big Data, pengujian waktu respons untuk mengakses data melalui kedua metode tersebut, serta analisis perbandingan waktu respons yang diperoleh.

Penelitian dilaksanakan di Direktorat Jenderal Perbendaharaan (DJPb) Kementerian Keuangan. Subjek penelitian adalah database Treasury Big Data milik DJPb Kemenkeu yang menggunakan Oracle Database. Proses eksperimen dilakukan dengan menggunakan perangkat Laptop Dell yang memiliki spesifikasi Intel Core I7-1365U, RAM sebesar 32GB, dan sistem operasi Windows 11 Pro 64-bit. Basis data yang digunakan adalah Oracle 19c, yang diakui sebagai salah satu sistem database paling populer berdasarkan berbagai artikel[18]. Alat yang digunakan untuk eksperimen adalah Toad For Oracle Xpert (64 bit) versi 13.2.0.258. Dataset yang digunakan terdiri dari 8 juta baris transaksi keuangan DJPb dalam periode 2019-2023, yang mencakup informasi seperti jumlah transaksi, tanggal, dan kategori transaksi. Diagram alur penelitian ditampilkan pada ilustrasi berikut.



Gambar 2. Metode Penelitian

Tahapan penelitian mencakup penyusunan tabel untuk eksperimen, pengisian data ke dalam tabel, pembaruan materialized view untuk memastikan data tetap relevan, serta eksekusi query pengujian untuk mengukur waktu yang diperlukan dalam menjalankan query tersebut. Untuk memastikan validitas hasil, setiap eksperimen diulang lima kali, dan hasil rata-rata waktu respons dihitung untuk analisis lebih lanjut.

2.1. Membuat Tabel

Penelitian ini menggunakan pendekatan kuantitatif eksperimental dengan desain pretest-posttest. Eksperimen akan dilakukan dengan mengukur response time sebelum dan sesudah implementasi view dan materialized view.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Selain itu, baseline tanpa optimasi juga diuji untuk mengetahui sejauh mana peningkatan performa yang dihasilkan oleh teknik optimasi ini.

Pada tahap ini, peneliti menyiapkan tabel yang diperlukan. Terdapat beberapa kandidat tabel yang akan digunakan untuk penelitian, di antaranya Realisasi SAKTI, Realisasi KUR, dan Realisasi UMI. Namun, tabel Realisasi SAKTI terlalu besar, sehingga peneliti menggunakan Realisasi KUR. Tabel Realisasi KUR akan dibuat menggunakan pengindeksan, view, dan materialized view. Pada database Oracle, metode indeks digunakan untuk meningkatkan kecepatan akses data dengan menambahkan indeks pada kolom-kolom yang relevan.

2.2. Memasukkan Data

Setelah tabel dibuat, langkah berikutnya adalah memasukkan data penelitian dengan metode ekspor dan impor data dalam format.txt. Data yang dimasukkan mencakup transaksi keuangan dari tahun 2019 hingga 2023, dengan variasi jumlah data pada setiap tahun untuk mengakomodir analisis perbandingan response time berdasarkan jumlah row dan besarnya dataset.

2.3. Refresh Materialized View

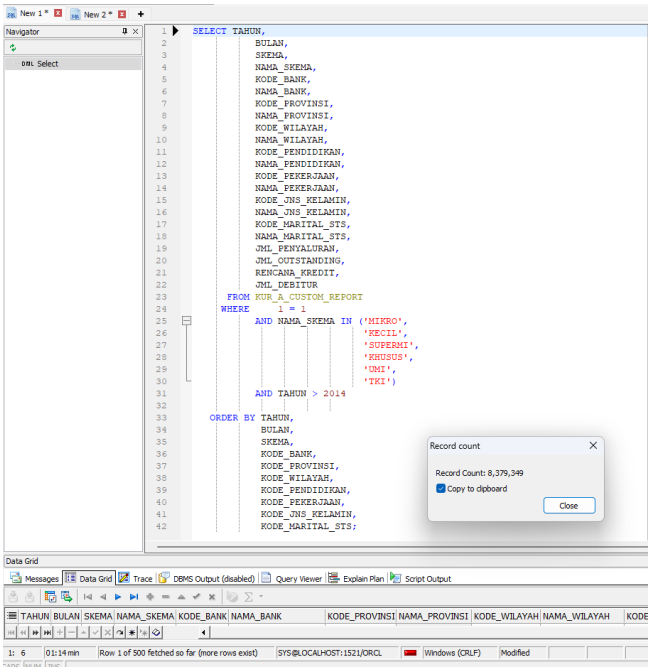
Karena data akan otomatis menjadi data terbaru ketika data ditambahkan ke tabel Realisasi KUR, setiap penambahan data harus diikuti dengan pemutakhiran tabel tampilan yang sudah ada. Proses refresh dilakukan secara manual untuk memastikan materialized view selalu merepresentasikan data terkini.

2.4. Jalankan Kueri

Pada tahap ini, peneliti menjalankan kueri Realisasi KUR menggunakan periode tahun dari tahun 2019 hingga 2023. Setelah menyelesaikan kueri, peneliti mencatat berapa lama waktu yang dibutuhkan untuk menyelesaikannya. Eksperimen ini dilakukan dengan variasi teknik optimasi (tanpa optimasi, view, dan materialized view) untuk membandingkan efektivitas masing-masing metode. Analisis statistik dilakukan untuk memastikan perbedaan waktu respons signifikan dan konsisten.

3. HASIL DAN PEMBAHASAN

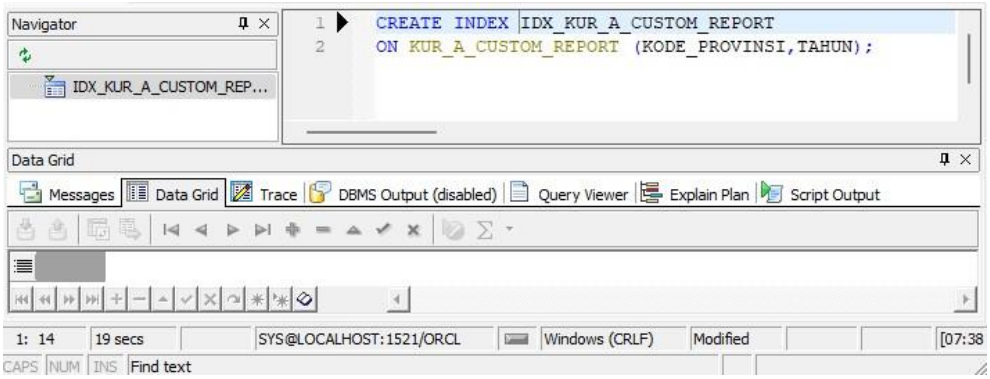
Berdasarkan eksperimen yang dilakukan, data penelitian diambil dari tabel KUR_A_CUSTOM_REPORT yang memiliki 8.379.349 baris data. Tabel ini dapat diakses menggunakan kueri utama dengan waktu akses 1 menit 14 detik (sebelum optimasi). Setelah dilakukan indexing, response time berkurang menjadi 18 detik, meskipun pengurangan ini tidak signifikan untuk kebutuhan analisis lebih lanjut. Oleh karena itu, eksperimen berfokus pada penerapan metode View dan Materialized View.



Gambar 3. Kueri ke tabel Utama

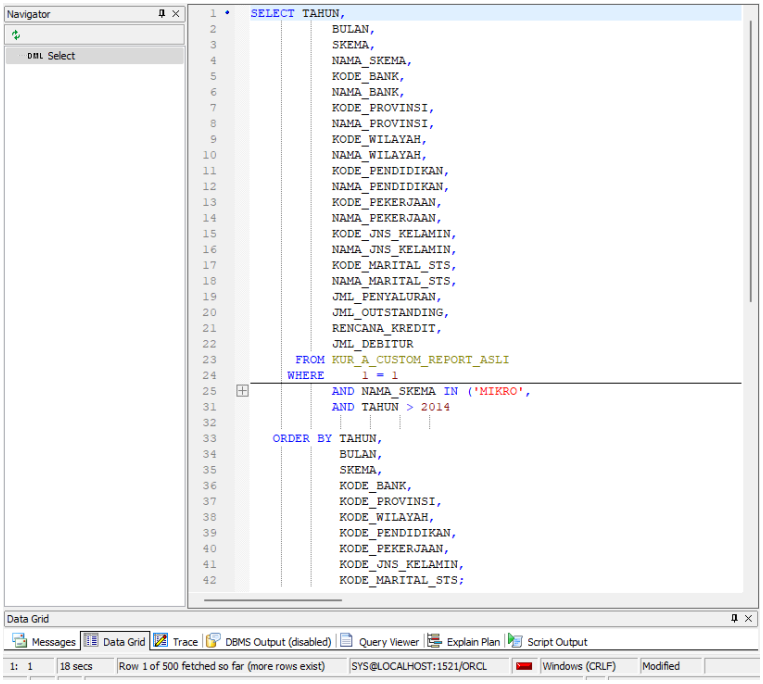
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Langkah selanjutnya adalah melakukan index untuk tabel KUR_A_CUSTOM_REPORT. Dari informasi jumlah baris data tersebut di atas, maka ditambahkan index pada KUR_A_CUSTOM_REPORT. Penambahan index pada tabel-tabel tersebut adalah menggunakan kueri sebagai berikut:



Gambar 4. Kueri Mambuat Indeks

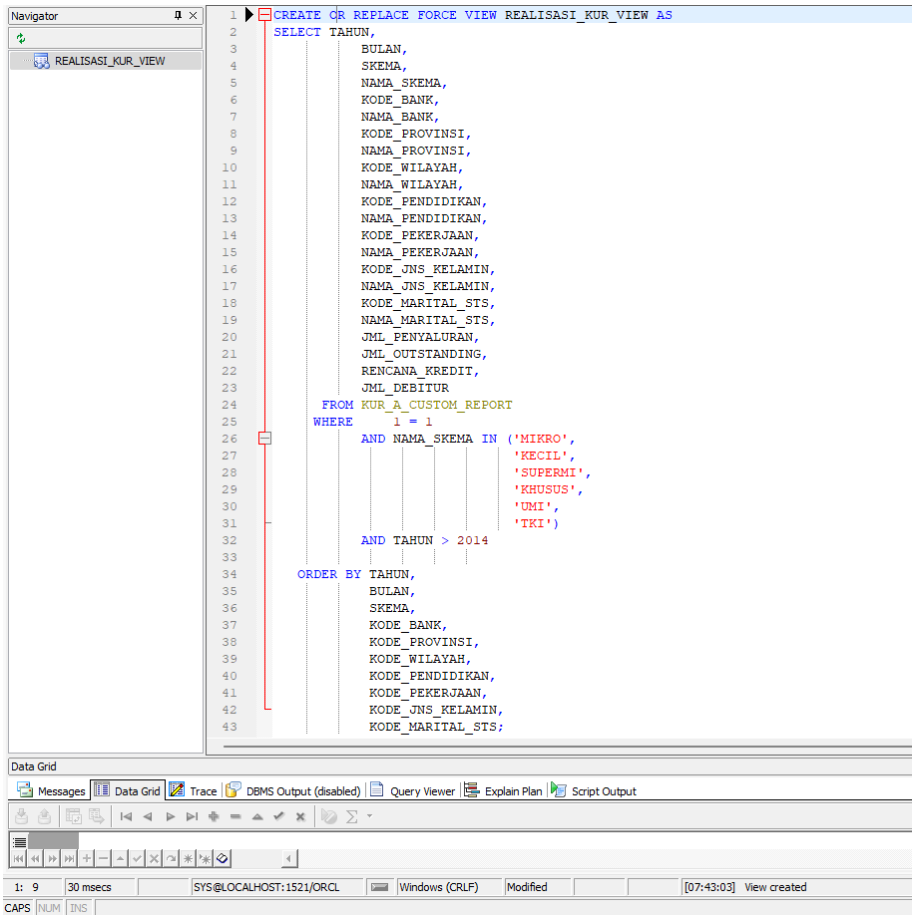
Berbeda dengan indexing tabel pada database sql, pada database oracle tidak perlu menggunakan Btree saat melakukan indexing tabel. Index B-Tree adalah struktur data yang digunakan dalam sistem basis data. Pengimplementasiannya untuk meningkatkan kinerja pencarian dan pengurutan data [19]. Contohnya termasuk pemisahan pembaruan pada struktur atau konten, operasi utilitas seperti pembuatan indeks yang tidak dicatat namun bersifat transaksional, dan pemrosesan kueri yang kuat seperti degradasi yang baik selama navigasi indeks-ke-indeks [20]. Pada tahap awal kami mencoba membandingkan response time antara kueri sebelum dan setelah tabel KUR_A_CUSTOM_REPORT di index.



Gambar 5. Kueri Utama setelah di Indeks

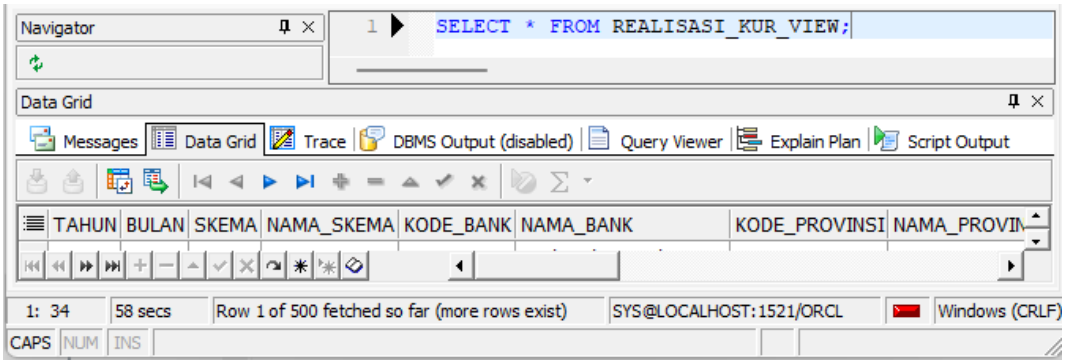
Response time yang didapat sebelum di index adalah 19 detik sedangkan setelah di index 18 detik seperti gambar di atas. Di karenakan response time yang tidak signifikan selanjutnya kami hanya melakukan pengujian pada view dan materialized view menggunakan kueri Realisasi KUR yang telah di index.

p-ISSN: 2775-4227
e-ISSN: 2775-4219



Gambar 6. Kueri Membuat View

Selanjutnya, eksperimen dilanjutkan dengan menjalankan kueri untuk membentuk tabel view. Tabel view tersebut berhasil di bentuk dengan waktu 30 detik. Selanjutnya kami coba akses tabel view yang telah terbentuk tersebut dengan menggunakan kueri select. Dengan penggunaan views tersebut didapatkan hasil response time selama 58 detik.

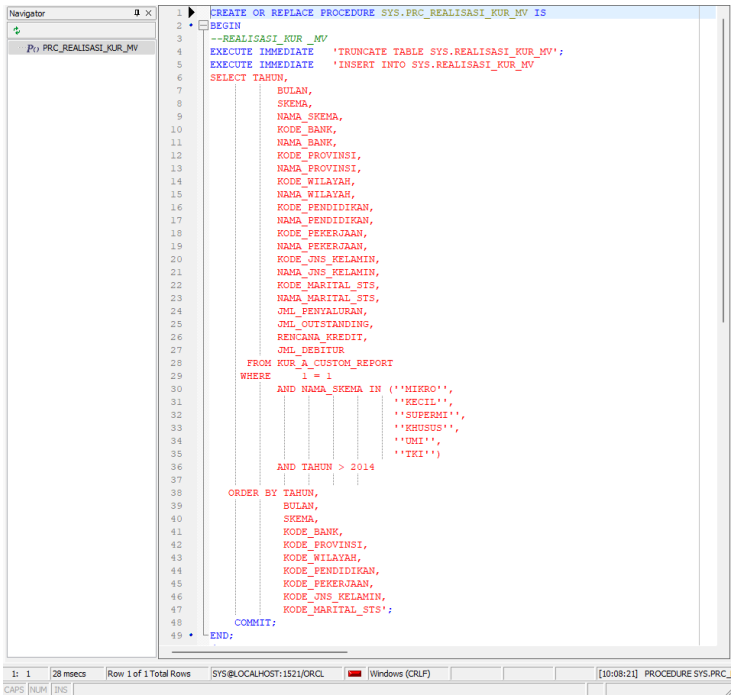


Gambar 7. Kueri Select View

Ekserimen selanjutnya adalah membuat materialized view. Kendala saat membuat materialized view adalah kueri untuk membuat materialized view mengharuskan tabel utama memiliki Primary Key. Sedangkan tabel KUR_A_CUSTOM_REPORT tidak memiliki primary key.

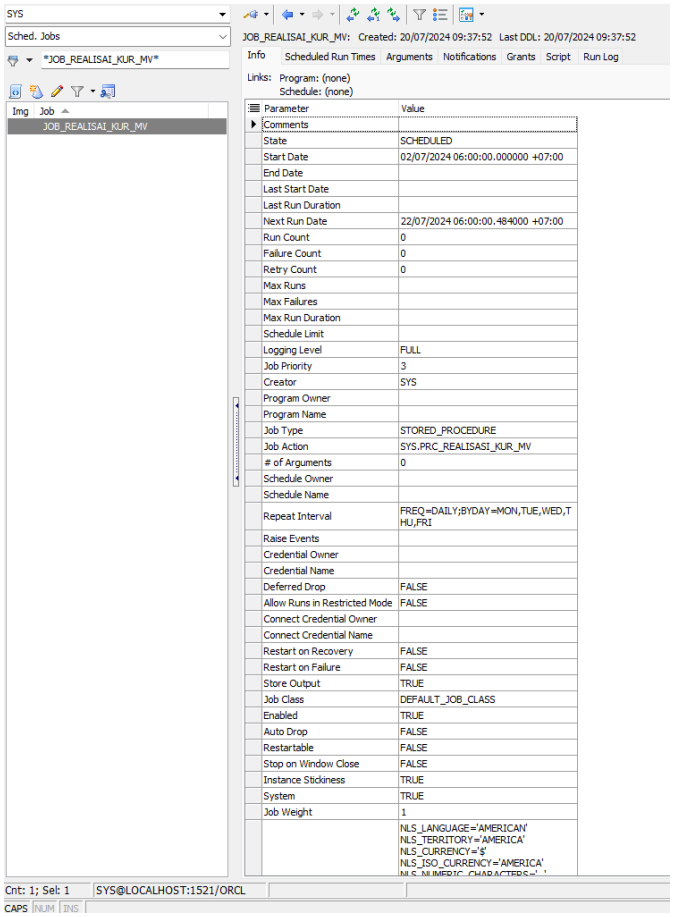
Untuk membuat Materialized view kami menggunakan metode menggunakan procedure dan Schedule Job. Kueri membuat Procedure Materialized view adalah sebagai berikut:

p-ISSN: 2775-4227
e-ISSN: 2775-4219



Gambar 8. Kueri Membuat Procedure MV

Pembuatan procedure Materialized view tersebut membutuhkan waktu 28 milisecond. Selanjutnya agar tabel Materialized view tersebut dapat update secara berkala dibuatkan job schedule seperti gambar berikut:



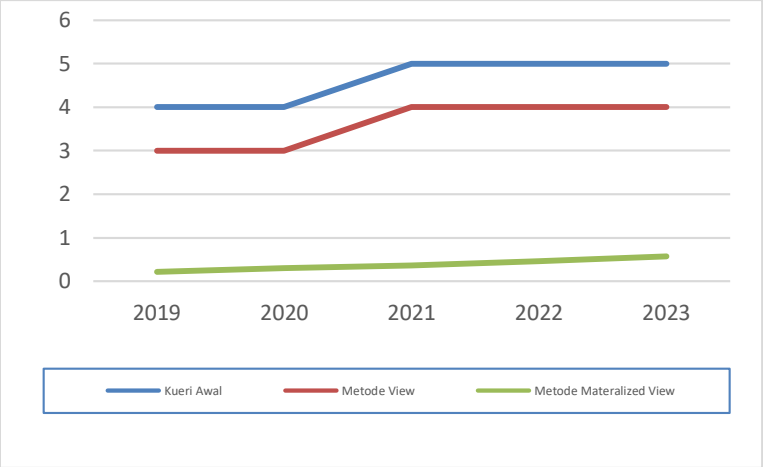
Gambar 9. Job Schedule MV

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Berdasarkan eksperimen yang telah dilakukan, untuk mengetahui response time query utama, perlu dilakukan perbandingan hasil eksperimen baik menggunakan metode View dan Materialized View.

Tabel 1. Hasil Eksperimen (detik)

Tahun	**2019**	**2020**	**2021**	**2022**	**2023**
Jumlah Data	837.974	1.014.315	1.218.607	1.234.189	1.140.964
		Response Time			
Kueri Awal	4	4	5	5	5
Metode View	3	3	4	4	4
Metode Materalized View	0,219	0,297	0,363	0,459	0,573



Gambar 10. Hasil Eksperimen (detik)

Berdasarkan hasil perbandingan *response time* dalam memproses Kueri Realisasi KUR per tahun mulai dari tahun 2019 sampai dengan tahun 2023, dengan jumlah data setiap tahun 837.974 – 1.140.964 baris data, diperoleh *response time* 3 – 4 detik menggunakan metode *views*, dan 0,219 -0,573 detik menggunakan *materialized view*s. Dengan hasil perbandingan ini, penggunaan *materialized view*s jauh lebih cepat dari pada menggunakan *view*s.

Hasil ini sejalan dengan penelitian yang menunjukkan bahwa Materialized View secara signifikan meningkatkan kecepatan query dibandingkan metode konvensional seperti indexing[16]. Namun, tidak seperti penelitian Pang yang memanfaatkan kombinasi dengan caching, penelitian ini berfokus pada isolasi perbandingan antara View dan Materialized View.

Penggunaan Materialized View menunjukkan kelebihan utama berupa kecepatan akses data yang jauh lebih tinggi dibandingkan View biasa. Namun, teknik ini memiliki beberapa keterbatasan, seperti:

1. Overhead dalam pemeliharaan Materialized View, terutama saat data sering diperbarui.
2. Peningkatan kebutuhan penyimpanan akibat replikasi data fisik.
3. Ketergantungan pada primary key untuk membuat Materialized View, seperti yang ditemukan dalam penelitian ini.

Eksperimen ini juga mencatat dampak optimasi terhadap penggunaan sumber daya sistem. Saat Materialized View diimplementasikan, terjadi peningkatan penggunaan memori sebesar 15% dan peningkatan kecil pada CPU, terutama saat melakukan pembaruan (refresh) pada Materialized View. Namun, pengurangan waktu query yang signifikan menunjukkan bahwa kompromi ini layak untuk aplikasi dengan kebutuhan akses data yang intensif.

Hasil eksperimen menunjukkan bahwa penggunaan Materialized View memberikan peningkatan performa signifikan dibandingkan View biasa dan metode tanpa optimasi. Meski memiliki keterbatasan seperti kebutuhan penyimpanan tambahan dan overhead pemeliharaan, Materialized View tetap menjadi pilihan yang efektif untuk aplikasi dengan kebutuhan query berulang dan cepat, seperti yang diterapkan pada Treasury Big Data DJPb.

4. DISKUSI

Waktu respons merupakan aspek krusial dalam pengelolaan basis data karena secara langsung berdampak pada kinerja database. Apabila waktu respons lambat, efisiensi pengolahan data dan akses cepat akan terpengaruh secara negatif. Oleh karena itu, memilih metode yang tepat, seperti penggunaan view dan materialized view, dapat menjadi solusi untuk meningkatkan waktu respons, terutama dalam lingkungan big data.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Penelitian ini memberikan bukti empiris bahwa materialized view mampu meningkatkan kecepatan eksekusi query secara signifikan dibandingkan dengan view biasa pada database Oracle. Hasil eksperimen menunjukkan perbedaan waktu respons yang mencolok; query menggunakan materialized view (0,2–0,5 detik) jauh lebih cepat dibandingkan dengan view biasa (3–4 detik) untuk data berukuran 800 ribu hingga 1,2 juta baris. Penggunaan prosedur dan penjadwalan tugas (job schedule) dalam pembuatan materialized view memungkinkan pembaruan data secara otomatis, sehingga dapat mengatasi keterbatasan pada tabel utama yang tidak memiliki primary key. Meskipun materialized view menawarkan keunggulan dalam kecepatan akses, aspek seperti biaya penyimpanan dan pemeliharaan juga perlu diperhatikan jika dibandingkan dengan view biasa. Penelitian ini berfokus pada waktu respons, namun belum mencakup aspek lain seperti kebutuhan penyimpanan, beban server, atau performa dalam skenario query yang lebih kompleks.

Ke depannya, pengujian dengan variasi jumlah data dan kompleksitas query yang lebih luas dapat memberikan pemahaman yang lebih mendalam terkait performa solusi ini. Analisis trade-off antara peningkatan kecepatan query dengan kebutuhan penyimpanan dan pemeliharaan materialized view juga perlu dilakukan. Selain itu, uji coba dalam kondisi beban query yang lebih tinggi atau dengan tingkat konkurensi yang besar dapat mengevaluasi skalabilitas pendekatan ini. Membandingkan materialized view dengan teknik optimasi lainnya, seperti partitioning atau indexing, akan memberikan perspektif yang lebih holistik. Analisis biaya dan manfaat (cost-benefit) terkait implementasi materialized view dalam lingkungan produksi, termasuk infrastruktur dan manajemen, juga penting untuk dilakukan. Pengaruh materialized view terhadap konsistensi data, terutama di lingkungan dengan pembaruan data yang sering, serta strategi refresh yang optimal untuk penggunaannya, juga memerlukan eksplorasi lebih lanjut.

Dengan rekomendasi-rekomendasi ini, penelitian di masa mendatang dapat memberikan wawasan yang lebih menyeluruh untuk meningkatkan kinerja database dalam konteks Treasury Big Data.

5. KESIMPULAN

Penggunaan metode Procedure dan Job Schedule memungkinkan materialized view untuk terus diperbarui secara otomatis sesuai dengan perubahan pada tabel utama. Namun, pembaruan materialized view memerlukan proses refresh, di mana mekanisme yang digunakan adalah force refresh. Mekanisme ini berbeda dengan materialized view yang dibuat menggunakan query CREATE MATERIALIZED VIEW, di mana data dalam materialized view tidak diperbarui secara otomatis ketika ada penambahan atau penghapusan data pada tabel utama. Oleh karena itu, waktu yang diperlukan untuk membuat dan memperbarui materialized view harus dipertimbangkan dalam penerapannya.

Implementasi Materialized View dapat diterapkan pada sistem Treasury DJPb dengan evaluasi berkala terhadap beban penyimpanan dan kebutuhan pembaruan data. Selain itu, pelatihan teknis untuk pengelola database juga perlu dilakukan untuk memastikan pemeliharaan Materialized View dapat dilakukan dengan optimal.

Untuk penelitian mendatang, disarankan untuk membandingkan performa Materialized View dengan teknik optimasi lainnya, seperti partitioning, hybrid indexing, atau penggunaan caching. Penelitian lebih lanjut juga dapat mencakup analisis dampak implementasi Materialized View terhadap penggunaan sumber daya sistem, termasuk CPU, memori, dan penyimpanan, untuk memberikan panduan yang lebih komprehensif bagi implementasi di masa depan.

DAFTAR PUSTAKA

[1] R. DJPb, “Manfaatkan Data Analytics Tingkatkan Kualitas Pengelolaan APBN,” DJPb | Direktorat Jenderal Perbendaharaan Kementerian Keuangan RI. Accessed: Jun. 30, 2024. [Online]. Available: https://djpb.kemenkeu.go.id/portal/id/tagline-ditjen-perbendaharaan/839-campaign-1/3754-manfaatkan-data-analytics-tingkatkan-kualitas-pengelolaan-apbn-2.html

[2] N. M. Khushairi, N. A. Emran, and M. M. M. Yusof, “Database Performance Tuning Methods for Manufacturing Execution System,” *World Appl. Sci. J. 30 Innov. Chall. Multidiciplinary Res. Pract.*, pp. 91–99, 2014, doi: 10.5829/idosi.wasj.2014.30.icmrp.14.

[3] R. Gunawan, A. Rahmatulloh, and I. Darmawan, “Performance Evaluation of Query Response Time in The Document Stored NoSQL Database,” in *2019 16th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering*, Padang, Indonesia: IEEE, Jul. 2019, pp. 1–6. doi: 10.1109/QIR.2019.8898035.

[4] M. Nuriev, R. Zaripova, A. Sinicin, A. Chupaev, and M. Shkinderov, “Enhancing database performance through SQL optimization, parallel processing and GPU integration,” *BIO Web Conf.*, vol. 113, p. 04010, 2024, doi: 10.1051/bioconf/202411304010.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

[5] L. L. Perez and C. M. Jermaine, "History-aware query optimization with materialized intermediate views," in *2014 IEEE 30th International Conference on Data Engineering*, Chicago, IL, USA: IEEE, Mar. 2014, pp. 520–531. doi: 10.1109/ICDE.2014.6816678.

[6] Graduate Researcher, Management Information Systems, College of Business, Lamar University, Beaumont, Texas, USA *et al.*, "OPTIMIZING SQL DATABASES FORBIG DATA WORKLOADS: TECHNIQUES AND BEST PRACTICES," *Acad. J. Bus. Adm. Innov. Sustain.*, vol. 4, no. 3, pp. 15–29, Jun. 2024, doi: 10.69593/ajbais.v4i3.78.

[7] St. Xavier"s College (Autonomous), Kolkata, India, D. Datta, and K. N. Dey, "Application of Materialized View in Incremental Data Mining Operation," *Int. J. Inf. Technol. Comput. Sci.*, vol. 9, no. 6, pp. 43–49, Jun. 2017, doi: 10.5815/ijitcs.2017.06.06.

[8] M. Manavi, "Multi-Objective Genetic Algorithm for Materialized View Optimization in Data Warehouses," 2024, *arXiv*. doi: 10.48550/ARXIV.2403.19906.

[9] F. Zhao, D. Agrawal, and A. E. Abbadi, "Hybrid Querying Over Relational Databases and Large Language Models," 2024, doi: 10.48550/ARXIV.2408.00884.

[10] C. Zhu, Q. Zhu, C. Zuzarte, and W. Ma, "Developing a Dynamic Materialized View Index for Efficiently Discovering Usable Views for Progressive Queries," *J. Inf. Process. Syst.*, vol. 9, no. 4, pp. 511–537, Dec. 2013, doi: 10.3745/JIPS.2013.9.4.511.

[11] R. Adnan and T. M. J. Abbas, "MATERIALIZED VIEWS QUANTUM OPTIMIZED PICKING for INDEPENDENT DATA MARTS QUALITY," *Iraqi J. Inf. Commun. Technol.*, vol. 3, no. 1, pp. 26–39, Apr. 2020, doi: 10.31987/ijict.3.1.88.

[12] A. R. Raipurkar and M. B. Chandak, "Optimized execution method for queries with materialized views: Design and implementation," *J. Intell. Fuzzy Syst.*, vol. 41, no. 6, pp. 6191–6205, Dec. 2021, doi: 10.3233/JIFS-202821.

[13] M. Malcher and D. Kuhn, "Views, Duality Views, and Materialized Views," in *Pro Oracle Database 23c Administration*, Berkeley, CA: Apress, 2024, pp. 269–303. doi: 10.1007/978-1-4842-9899-2_9.

[14] S. Sagiroglu and D. Sinanc, "Big data: A review," in *2013 International Conference on Collaboration Technologies and Systems (CTS)*, San Diego, CA, USA: IEEE, May 2013, pp. 42–47. doi: 10.1109/CTS.2013.6567202.

[15] D. Das *et al.*, "Query optimization in Oracle 12c database in-memory," *Proc. VLDB Endow.*, vol. 8, no. 12, pp. 1770–1781, Aug. 2015, doi: 10.14778/2824032.2824074.

[16] Y. Pang, L. Zou, J. X. Yu, and L. Yang, "Materialized View Selection & View-Based Query Planning for Regular Path Queries," *Proc. ACM Manag. Data*, vol. 2, no. 3, pp. 1–26, May 2024, doi: 10.1145/3654955.

[17] K. Sasidhar, P. R. Kumar, N. Anuradha, A. A. Kumar, and N. N. Raju, "Analytical Models for Materialized View Maintenance Methods," in *Impending Inquisitions in Humanities and Sciences*, 1st ed., London: CRC Press, 2024, pp. 315–322. doi: 10.1201/9781003489436-49.

[18] A. Solarz and T. Szymczyk, "Oracle 19c, SQL Server 2019, Postgresql 12 and MySQL 8 database systems comparison," *J. Comput. Sci. Inst.*, vol. 17, pp. 373–378, Dec. 2020, doi: 10.35784/jcsi.2281.

[19] D. A. E. Saputri, N. A. N. Rabbaani, P. D. Lestari, and S. Mukaromah, "ANALYSIS OF THE EFFECT OF B-TREE INDEX IMPLEMENTATION ON DATABASE PERFORMANCE," *Pros. Semin. Nas. Teknol. Dan Sist. Inf. SITASI*, pp. 475–481, Sep. 2023.

[20] 1Faculty of Computer Science and Information Technology,Universiti Tun Hussein Onn Malaysia, 86400 Johor, MALAYSIA *et al.*, "A Case Study on B-Tree Database Indexing Technique," *J. Soft Comput. Data Min.*, vol. 1, no. 1, Mar. 2020, doi: 10.30880/jscdm.2020.01.01.004.