 ISSN:2302-8149
e-ISSN:2540-9719

Peningkatan Performa Laporan Berkala dengan Materialized Views pada Sistem Basis Data Oracle

Performance Improvement of Periodic Reports with Materialized Views on Oracle Database System

[1]Raden Soepriadi*, [2]Gahara Dijerja, [3]Samidi
[1,2,3]Program Studi Magister Ilmu Komputer, Fakultas Teknologi Informasi, Universitas Budi Luhur
[1,2,3]Jl. Ciledug Raya, RT10/RW02, Petukangan Utara, Kec. Pasanggrahan, Jakarta Selatan, Prov. DKI Jakarta, Indonesia
*e-mail: 2311601690@student.budiluhur.ac.id

(***received*:** 2 January 2025, ***revised*:** 21 February 2025, ***accepted*:** 22 February 2025)

Abstrak

Laporan berkala dalam Sistem Informasi Manajemen (SIM) berperan penting dalam pengambilan keputusan organisasi, namun pertumbuhan data yang pesat dapat menurunkan kinerja kueri. Penelitian ini bertujuan untuk meningkatkan performa laporan berkala pada SIM Modul Penerimaan Negara (SIM MPN) Kementerian Keuangan dengan memanfaatkan *Materialized Views* (MVs) dalam basis data Oracle. Sebagai sampel, diambil 294.503.898 baris data dari tahun 2021 hingga 2023. Identifikasi dan analisis dilakukan terhadap delapan tipe kueri laporan berkala yang selama ini menggunakan *views*. Kueri tersebut kemudian dikonversi menjadi MVs untuk mengurangi *execution time*. Pengujian dilakukan dengan membandingkan *execution time* antara *views* dan MVs pada 24 kueri yang dieksekusi di dua lingkungan berbeda. Hasil pengujian menunjukkan bahwa rata-rata *execution time* dengan *views* mencapai 2.047.417 ms, sedangkan dengan MVs hanya 41 ms pada pengujian pertama, 17 ms pada pengujian kedua, dan 12 ms pada pengujian ketiga. Dari hasil ini, dapat disimpulkan bahwa penggunaan MVs secara signifikan meningkatkan performa laporan berkala dengan mempercepat eksekusi kueri. Implikasi praktis dari penelitian ini adalah rekomendasi penerapan MVs untuk sistem yang memiliki volume data besar guna mengoptimalkan kecepatan akses laporan. Penelitian lanjutan dapat fokus pada optimasi waktu *refresh* MVs dan analisis lebih lanjut terkait faktor yang memengaruhi *execution time* pada berbagai skenario penggunaan.

Kata kunci: *execution time*, *materialized views*, *oracle database*, laporan berkala, sistem informasi manajemen

Abstract

Periodic reports in Management Information Systems (MIS) play a crucial role in organizational decision-making. However, rapid data growth can degrade query performance. This study aims to enhance the performance of periodic reports in the State Revenue Collection Module (SIM MPN) of the Ministry of Finance by utilizing Materialized Views (MVs) in the Oracle database system. A dataset comprising 294,503,898 rows from 2021 to 2023 was used as a sample. An analysis was conducted on eight types of periodic report queries that traditionally relied on standard views. These queries were then converted into MVs to reduce execution time. Testing was performed by comparing execution times between views and MVs across 24 queries executed in two different environments. The results showed that the average execution time using views was 2,047,417 ms, whereas with MVs, execution times were reduced to 41 ms in the first test, 17 ms in the second, and 12 ms in the third. These findings confirm that MVs significantly improve the performance of periodic reports by accelerating query execution. The practical implication of this study is the recommendation to implement MVs in systems with large data volumes to optimize report access speed. Future research can focus on optimizing MV refresh times and further analyzing factors affecting execution time under various usage scenarios.

Keywords: *execution time, materialized views, oracle database, periodic reports, management information system*

 ISSN:2302-8149
e-ISSN:2540-9719

1 Pendahuluan

Sejak tahun 2016 sampai 2023, jumlah transaksi penerimaan negara yang dikelola Kementerian Keuangan (Kemenkeu) mengalami pertumbuhan rata-rata sebesar 8,12 persen. Pertumbuhan ini termasuk adanya penurunan jumlah transaksi pada tahun 2020 sebesar minus 15,96 persen akibat pandemi covid-19. Pertumbuhan transaksi penerimaan negara berpengaruh besar terhadap pertumbuhan data pada database sistem penerimaan negara. Merujuk pada *data warehouse* yang dimiliki Kemenkeu, pertumbuhan data atas kenaikan transaksi penerimaan negara dari 2016 sampai 2023 sebesar rata-rata 11,80 persen. Dibuktikan dengan bertambahnya *row data* pada tabel transaksi sebesar rata-rata 89,8 juta baris data per tahun.

Dalam konteks pengelolaan keuangan negara, Kementerian Keuangan memegang peran strategis dalam mengelola dan mengawasi penerimaan serta pengeluaran negara guna memastikan keberlanjutan fiskal dan efektivitas kebijakan ekonomi. Dalam mendukung tugas tersebut, Sistem Informasi Manajemen (SIM) menjadi elemen krusial dalam proses pengambilan keputusan berbasis data. SIM yang digunakan oleh Kemenkeu bertujuan untuk menyediakan akses cepat dan akurat terhadap informasi keuangan negara, sehingga memungkinkan para pengambil kebijakan untuk menganalisis tren penerimaan, melakukan perencanaan anggaran, serta mengidentifikasi potensi penyimpangan atau inefisiensi dalam sistem keuangan negara.

SIM sebagai sistem yang dapat memaparkan informasi yang teranalisis dan berkualitas tinggi untuk organisasi dalam mengambil keputusan yang efisien dan cepat bagi pimpinan organisasi[1], pada Kemenkeu, terintegrasi dengan Oracle *data warehouse* (DWS) dari sistem penerimaan negara, dikenal dengan Modul Penerimaan Negara (MPN). Terhitung mulai diimplementasikannya MPN pada tahun 2007 sampai dengan 2023, jumlah total data telah mencapai lebih dari 1 milyar baris data. Hal ini sangat berdampak dengan semakin tingginya nilai *execution time* pada proses kueri pengambilan data untuk kebutuhan laporan berkala seperti *flash report*, *summary report*, dan monitoring data. *Execution time* didefinisikan sebagai durasi waktu yang diperlukan untuk mengeksekusi program, yang dipengaruhi oleh kondisi input dan perangkat keras[2].

Pemberian indeks pada tabel transaksi penerimaan negara belum dapat memenuhi harapan untuk menurunkan nilai *execution time*. Dengan 294 juta baris data, meskipun indeks membantu mempercepat pencarian, ukuran indeks itu sendiri dapat menjadi sangat besar, menyebabkan *overhead* tambahan dalam proses pencarian[3]. Indeks yang besar juga membutuhkan lebih banyak memori untuk *caching*, dan jika memori tidak cukup, Oracle akan sering mengakses disk, yang memperlambat eksekusi [4]. Dibutuhkan adanya pendekatan baru seperti penggunaan *materialized views* (MVs) untuk mengatasi masalah ini.

MVs didefinisikan sebagai sumber data terdistribusi yang sangat penting bagi banyak aplikasi untuk mendukung *high availability*, akses yang efisien dan kinerja yang andal[5]. MVs juga diartikan sebagai objek skema yang digunakan untuk menyederhanakan dan menggandakan data, baik sebagian maupun seluruhnya. MVs bisa ditempatkan di basis data yang sama atau di basis data yang berbeda dari sumber aslinya. Proses penyegaran hasil replikasi data akan dilakukan secara otomatis oleh sistem setelah transaksi berakhir[6]. MVs telah digunakan dan bermanfaat dalam mengoptimalkan *kueri* selama bertahun-tahun[7]. Penggunaannya pada tabel transaksi akan memberikan peningkatan kinerja yang tinggi dalam sistem manajemen database[8].

Penelitian ini dilakukan untuk membandingkan kinerja *views* dan *materialized views* dari tabel transaksi yang telah memiliki indeks, melalui pengukuran nilai *execution time* menggunakan data riil sebuah organisasi. Sebagaimana rekomendasi penelitian terdahulu yang merekomendasikan adanya riset yang berfokus pada pemeliharaan MVs dan validasi model terhadap *real-world data warehouse* [5].

2 Tinjauan Literatur

Sebagai acuan dan referensi dalam melakukan penelitian ini, peneliti meninjau beberapa literatur yang berkenaan dengan *Materialized Views.* Referensi pertama didapat dari penelitian yang dilakukan oleh Witono dan Parno[9], yang meneliti perbandingan *response time* penggunaan *Indeks*, *Views* dan *Materialized Views* pada database MySQL aplikasi SisfoSPM Badan Penelitian dan Pengembangan Kementerian Pendidikan dan Kebudayaan. Penelitian tersebut memperoleh simpulan bahwa, dari sejumlah 780.212 baris data, *response time* dalam menghitung log data tercepat menggunakan

 ISSN:2302-8149
e-ISSN:2540-9719

Materialized Views yaitu hanya 0,01 detik. Selisih 1,84 detik lebih cepat dari penggunaan *Views* dan 1,94 detik lebih cepat dari penggunaan *Indeks*.

Nugraha *et al*.[10], melakukan analisis atas perbandingan waktu pemrosesan kueri antara penggunaan *where clause, views dan materialized views* pada data laporan transaksi bulanan sebuah sistem penjualan dengan database MySQL[10]. Jumlah data eksperimen yang diteliti terdiri dari 100, 500, 750, 1000 dan 1500 baris data. Waktu yang dibutuhkan dalam menjalankan kueri dengan menggunakan *where clause* atas jumlah data tersebut adalah masing-masing 0.007, 0.0072, 0.0072, 0.008 dan 0.0076 detik. Sedangkan waktu yang dibutuhkan dalam menjalankan kueri dengan menggunakan *views* adalah masing-masing 0.0105, 0.0104, 0.0111, 0.0118 dan 0.0124 detik. Adapun jika menggunakan *materialized views*, waktu pemrosesan *kueri* terbukti lebih cepat yaitu 0.0012, 0.0017, 0.0017, 0.002 dan 0.0027 detik untuk masing-masing data dengan jumlah 100, 500, 750, 1000 dan 1500 baris data.

Literatur lain yang menjadi acuan bagi peneliti adalah tulisan berjudul *Materialized View Effect on Query Performance*[8]. Tulisan ilmiah dari hasil penelitian Ayik dan Kahveci ini mengerucut pada sebuah kesimpulan bahwa penggunaan *Materialized Views* berdampak pada penghematan konsumsi memori, prosesor dan unit input/output.

Tulisan ilmiah Khushairi *et al*.[7], hasil dari penelitian penggunaan *Materialized Views* untuk melakukan penyetelan database pada industri manufaktur, senada dengan ketiga penelitian di atas yang memperoleh sebuah kesimpulan bahwa, MVs secara optimal dapat membantu dalam penyetelan database. Namun demikian, berdasarkan hasil eksperimen yang dilakukan, ada kasus yang menunjukkan bahwa MVs tidak memadai untuk meningkatkan kinerja *kueri* SQL. Sehingga diperlukan penyelidikan lebih lanjut, untuk menemukan cara mengatasi keterbatasan tersebut.

Kumar dan Devi[11] menulis hasil penelitiannya berkenaan dengan penggunaan *Materialized Views Construction Framework* (MVCF) untuk membangun MVs berdasarkan *Query Log* yang didapatkan dari DWS. Kesimpulan yang diperoleh dari penelitiannya adalah MVCF menyediakan sebuah mekanisme untuk memilih kueri yang optimal sehingga dapat membangun sebuah MVs sesuai dengan subjek area/domain yang telah ditentukan.

Terdapat pula peneliti yang memperkenalkan pendekatan algoritma genetika untuk mengotomasi dan mengoptimalkan pemilihan MVs untuk beban kerja basis data. Algoritma ini menggabungkan strategi pembangkitan populasi awal yang cerdas dengan metode *crossover* khusus, mutasi adaptif, dan fungsi kebugaran multi-objektif untuk secara efektif mencari konfigurasi MVs yang optimal[12].

MVS_{vega}, sebuah algoritma dalam pemilihan kueri untuk dijadikan MVs yang dikemukakan Prakash dan Kumar[13] berhasil meningkatkan *response time* dalam kueri analitik. Algoritma lainnya yang ditawarkan Gosain dan Sachdeva[14] adalah SRCSAMVS. Sebuah kerangka kerja yang bermanfaat dalam pemilihan MVs yang layak dan optimal dalam hal meminimalisasi "biaya" pemrosesan kueri.

Dibandingkan dengan penelitian terdahulu, skala data yang digunakan dalam penelitian ini jauh lebih besar. Studi seperti yang dilakukan oleh Witono dan Parno[9] dan Nugraha *et al*.[10] hanya menggunakan dataset dalam skala ribuan hingga ratusan ribu baris data, sementara penelitian ini mengolah lebih dari 294 juta baris data. Hal ini lebih mencerminkan skenario dunia nyata dalam sistem informasi berskala besar seperti yang digunakan oleh Kementerian Keuangan.

Dari segi konteks sistem yang diteliti, penelitian terdahulu sebagian besar dilakukan dalam lingkungan database MySQL dengan fokus pada sistem penjualan atau manufaktur. Sebaliknya, penelitian ini berfokus pada implementasi MVs dalam basis data Oracle yang digunakan oleh Sistem Informasi Manajemen Modul Penerimaan Negara (SIM MPN). Kompleksitas sistem ini lebih tinggi dibandingkan sistem yang digunakan dalam penelitian sebelumnya, karena melibatkan volume transaksi besar dengan keterkaitan data yang luas antar entitas.

Metodologi pengujian yang digunakan dalam penelitian ini juga memiliki keunggulan dibandingkan penelitian terdahulu. Beberapa studi sebelumnya menggunakan eksperimen berbasis simulasi dengan jumlah data terbatas dan parameter yang dikontrol secara ketat. Sementara itu, penelitian ini menguji implementasi MVs dalam sistem yang sedang berjalan, membandingkan *execution time* secara langsung dalam berbagai skenario operasional yang realistis.

Dari aspek optimasi kueri, sebagian besar penelitian sebelumnya hanya membandingkan *execution time* antara *views* dan MVs dalam skala kecil tanpa mempertimbangkan implementasi dalam sistem operasional yang kompleks. Penelitian ini, di sisi lain, membangun MVs berdasarkan

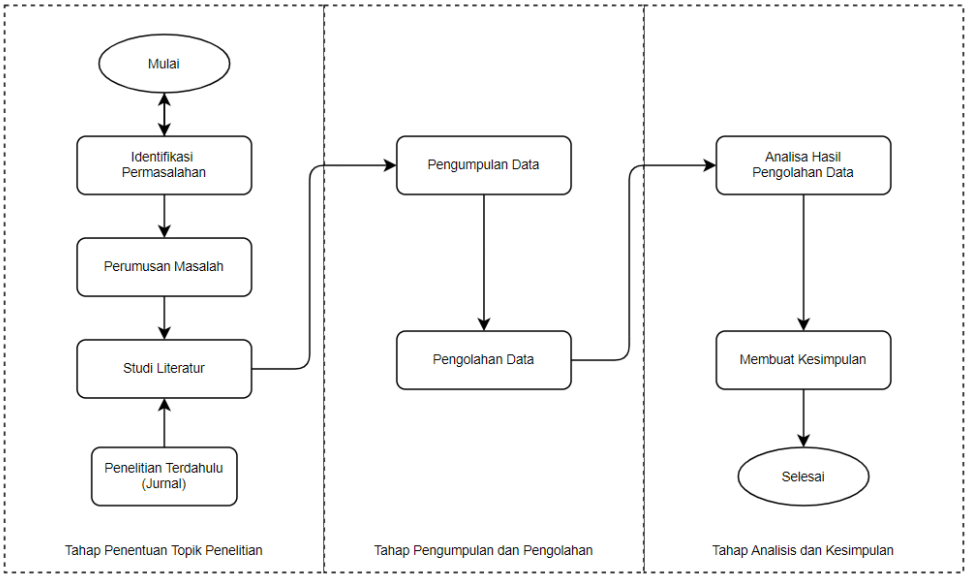
 ISSN:2302-8149
e-ISSN:2540-9719

kueri yang telah digunakan dalam SIM MPN untuk laporan berkala. Dengan pendekatan ini, penelitian memberikan kontribusi praktis yang lebih signifikan terhadap optimalisasi manajemen data dalam institusi pemerintahan.

Dengan adanya perbedaan dalam skala data, konteks sistem, metodologi pengujian, dan fokus penelitian, penelitian ini memperkaya literatur terkait optimasi kueri dengan MVs. Hal ini terutama relevan dalam konteks *big data* pada sistem pemerintahan, di mana efisiensi akses dan kecepatan pemrosesan data menjadi sangat krusial.

3 Metode Penelitian

Berangkat dari sebuah permasalahan meningkatnya pertumbuhan data penerimaan negara, yang berdampak pada proses penayangan laporan berkala pada SIM MPN, penulis melakukan penelitian dengan tahapan: Identifikasi Masalah, Perumusan Masalah, Studi Literatur, Pengumpulan Data, Pengolahan Data, Analisa Hasil Pengolahan Data dan Pengambilan Kesimpulan. Tahapan-tahapan tersebut secara garis besar menjadi tiga tahapan utama yaitu Penentuan Topik Penelitian, Pengumpulan dan Pengolahan, dan Analisis dan Kesimpulan sebagaimana tampak pada Gambar 1.



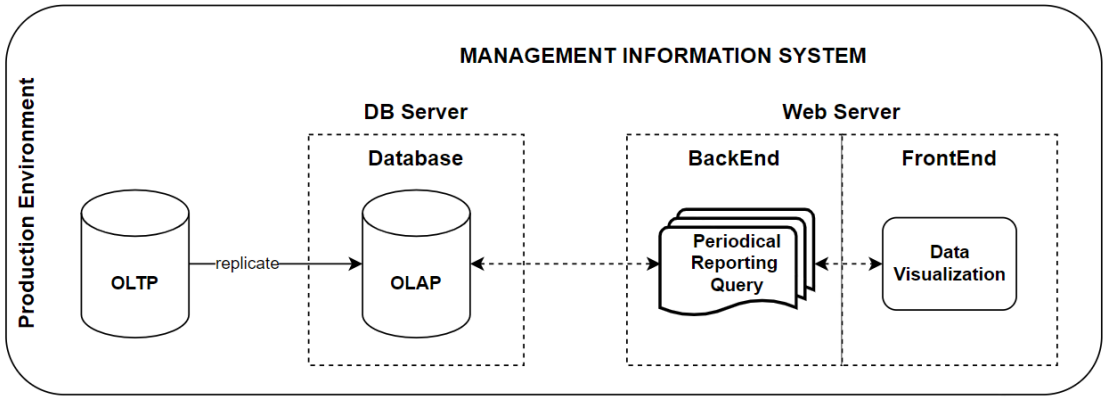
Gambar 1. Metodologi penelitian

Tahapan pertama diinisiasi dengan melakukan proses identifikasi permasalahan, yaitu proses penemuan masalah pada SIM MPN. Permasalahan tersebut adalah adanya perlambatan proses penayangan data pada fitur laporan berkala SIM MPN yang diakibatkan pertumbuhan data yang besar. Perlambatan proses penayangan data ini menjadi tolak ukur penurunan kinerja sistem. Dalam kinerja database yang digunakan, kumpulan data yang besar dapat menyebabkan masalah pada *Database Management System* (DBMS)[15].

Tahapan selanjutnya, yang masih berada dalam klaster Penentuan Topik Penelitian, adalah perumusan masalah. Perumusan masalah yang dilakukan peneliti adalah proses penentuan hipotesis atas permasalahan yang telah teridentifikasi.

Berdasarkan hipotesis yang telah disusun, peneliti meninjau dan mengkaji secara mendalam atas sumber-sumber ilmiah yang telah ditulis peneliti sebelumnya, tentunya yang berkaitan dengan topik MVs sebagai bentuk proses optimasi kueri dalam meningkatkan kinerja sistem.

 ISSN:2302-8149
e-ISSN:2540-9719



Gambar 2. Arsitektur SIM MPN

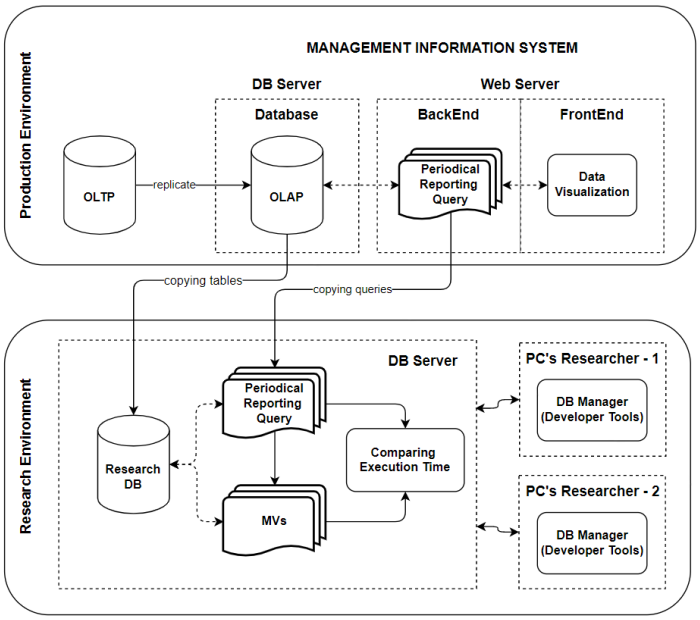
Gambar 2 menunjukkan arsitektur SIM MPN, yang terdiri dari dua komponen utama: database dan *web server*. Database SIM mencakup OLTP dan OLAP, di mana OLAP mereplikasi data transaksi dari OLTP. Transaksi pembayaran penerimaan negara disimpan secara *real-time* dalam OLTP, dan keduanya menggunakan Oracle versi 19c.

OLTP (*Online Transaction Processing*) berorientasi pada pemrosesan transaksi *real-time*[16], dengan model hubungan entitas dan desain database yang sangat rinci. Sebaliknya, OLAP (*Online Analytical Processing*) mengelola data historis dalam jumlah besar, menyediakan fasilitas agregasi, serta digunakan untuk analisis data bagi berbagai pihak, termasuk manajer, eksekutif, dan analis[16].

Web server terdiri dari dua bagian, yaitu *Front-End*, yang berinteraksi langsung dengan pengguna, serta *Back-End*, yang menangani komunikasi dengan database dan menjalankan kueri untuk menghasilkan laporan berkala. Sistem ini berfungsi sebagai penghubung dalam proses visualisasi data dalam bentuk laporan yang diambil dari database OLAP[17].

Dalam konteks eksekusi proses, *execution time* mengacu pada durasi eksekusi sebuah kueri[2] laporan berkala pada *Back-End*, mulai dari proses berjalan hingga keluaran dihasilkan oleh database. Waktu ini telah diuji oleh peneliti. Sementara itu, *response time* mencakup waktu total sejak permintaan diterima hingga respons diberikan, termasuk *execution time*, antrian, *overhead* komunikasi, dan latensi sistem[18]. Dalam arsitektur ini, *response time* dihitung sejak pengguna mengirim permintaan dari *Front-End* ke *Back-End* hingga respons diterima kembali. Namun, aspek ini tidak diuji dalam penelitian.

Arsitektur sistem ini menentukan strategi serta metode pengumpulan dan pengolahan data, yang menjadi bagian inti dari penelitian ini. Tahapan tersebut dijelaskan secara rinci dalam Gambar 3.



Gambar 3. Metodologi pengumpulan dan pengolahan data

 ISSN:2302-8149
e-ISSN:2540-9719

Sebelum melakukan pengumpulan data, peneliti menyiapkan lingkungan riset (*Research Environment*) yang terdiri dari sebuah server database dan dua unit mesin penelitian dengan jenis *Personal Computer* (PC). Server database yang disiapkan memiliki spesifikasi: Sistem Operasi Redhat Enterprise Linux (RHEL) 6.9, RAM 16 GB, CPU 4 core dan kapasitas penyimpanan 1 TB. PC pertama yang digunakan peneliti untuk melakukan penelitian memiliki spesifikasi: sistem operasi Windows 11 Pro, RAM 16 GB, Prosesor Core (TM) i7-11700 @ 2.50GHz, didukung dengan penyimpanan Disk sebanyak 1TB. Sedangkan PC kedua memiliki spesifikasi: sistem operasi Windows 11 Pro, RAM 16 GB, prosesor 11th Gen Intel® Core™ i7-1165G7 @ 2,80 GHz, dan kapasitas penyimpanan sebesar 500 GB. Dalam server database, peneliti telah menyiapkan database *Oracle* versi 11g. Server ini terhubung dalam jaringan LAN yang sama dengan PC peneliti yang telah dipasang sebuah *database manager* sebagai alat pengolahan data, yaitu Toad for Oracle versi 14.0.75.662 untuk PC pertama, dan DBeaver versi 23.3.202312241705 pada PC kedua. PC kedua disiapkan untuk menguji konsistensi hasil penelitian.

Tahapan berikutnya adalah menganalisis berbagai format laporan berkala pada SIM MPN. Dari hasil analisis tersebut, peneliti mengumpulkan sejumlah kueri *views* yang digunakan oleh SIM MPN dalam mengambil dan mengolah data OLAP. Kumpulan kueri *views* tersebut direplikasi untuk disiapkan dan disimpan ke dalam mesin pada lingkungan penelitian.

Berdasarkan kueri *views* yang telah tersimpan, peneliti melakukan identifikasi atas tabel-tabel yang digunakan atau diperlukan. Selanjutnya dilakukan proses replikasi/duplikasi tabel transaksi dan tabel referensi dari database OLAP ke *research DB* (database penelitian) melalui mekanisme *DBLink* dari database penelitian ke database OLAP. Data yang diperoleh dari tabel transaksi terbatas untuk transaksi tahun 2021 sampai 2023. Jumlah data transaksi hasil proses replikasi terhimpun sebagaimana tampak pada Tabel 1.

Tabel 1. Data transaksi penerimaan negara per tahun

No.	Tahun	Jumlah Data (*row*)
1	2021	89.461.110
2	2022	99.689.824
3	2023	105.352.964
Jumlah		**294.503.898**

Peneliti menyusun sejumlah MVs berdasarkan kumpulan *views* sesuai dengan jenis laporan. Setelah adanya MVs ini, maka tahapan pengujian dalam proses penelitian dilakukan dengan membandingkan *execution time* antara *views* dengan MVs. Tahapan ini dilakukan dengan mengeksekusi keduanya melalui alat pengolahan data yang telah disiapkan dalam mesin penelitian dan terhubung dengan database penelitian. Proses pengujian dilakukan menggunakan dua mesin dengan spesifikasi berbeda untuk menguji konsistensi hasil.

4 Hasil dan Pembahasan

Pada bagian ini, akan dibahas mengenai hasil penelitian yang dilakukan untuk menganalisis performa laporan berkala dalam SIM MPN dengan menggunakan MVs. Pengujian dilakukan dengan membandingkan *execution time* antara kueri yang menggunakan *views* dan kueri yang menggunakan MVs. Seluruh pengolahan data dan analisis hasil akan dijelaskan secara bertahap, dimulai dari metode pengumpulan data, perancangan MVs hingga hasil pengujian dan pembahasan. Pengumpulan data dilakukan untuk mengidentifikasi jenis-jenis laporan berkala sekaligus menemukan kueri-kueri *views* pembentuk laporan berkala, yang dijelaskan pada bagian pengumpulan data. Sedangkan tahapan perancangan MVs dari sekumpulan kueri-kueri *views*, sekaligus pengujian dan pembahasan disampaikan pada bagian pengolahan data.

4.1 Pengumpulan Data

SIM MPN memiliki fitur andalan berupa laporan yang digunakan dalam pengambilan keputusan bagi *top management* atau pimpinan organisasi. Laporan tersebut berupa laporan berkala bulanan dan tahunan sebagaimana tampak pada Tabel 2.

 ISSN:2302-8149
e-ISSN:2540-9719

Tabel 2. Daftar laporan berkala pada SIM MPN

No.	Kode Laporan	Nama Laporan
1	MR-01	Jumlah Transaksi per CA per Mata Uang (parameter Tahun)
2	YR-01	Jumlah Transaksi per CA per Mata Uang (parameter Tahun dan Bulan)
3	MR-02	Jumlah Transaksi per CA per Kanal per Mata Uang (parameter Tahun)
4	YR-02	Jumlah Transaksi per CA per Kanal per Mata Uang (parameter Tahun dan Bulan)
5	MR-03	Jumlah Transaksi per Biller per Mata Uang (parameter Tahun)
6	YR-03	Jumlah Transaksi per Biller per Mata Uang (parameter Tahun dan Bulan)
7	MR-04	Jumlah Transaksi per Mata Uang (parameter Tahun)
8	YR-04	Jumlah Transaksi per Mata Uang (parameter Tahun dan Bulan)

Daftar laporan berkala sebagaimana pada Tabel 2, secara umum terbagi menjadi tiga bagian besar yaitu Laporan Jumlah Transaksi per CA, per Biller dan per Mata Uang. CA merupakan kependekan dari *Collecting Agent*, yaitu lembaga keuangan dan non-keuangan yang ditunjuk dan ditetapkan Kemenkeu untuk dapat melakukan penyetoran penerimaan negara. CA juga biasa disebut Bank/Pos Persepsi/Lembaga Persepsi Lainnya. Sedangkan Biller adalah unit eselon I Kementerian Keuangan yang mengelola data tagihan penerimaan negara.

Berdasarkan daftar laporan berkala yang disajikan dalam SIM MPN, peneliti mengambil *periodic reports query* yang tersimpan dalam *back-end* aplikasi. Kueri-kueri ini yang digunakan untuk menjalankan aplikasi sehingga dapat menayangkan data sebagaimana format laporan yang telah ditetapkan. Dari delapan jenis laporan sebagaimana pada Tabel 2 peneliti memperoleh delapan format kueri pembentuk laporan berkala. Adapun penulisan delapan format kueri tersebut, disusun dengan notasi dan simbol-simbol yang mengacu pada *Fundamentals of Database System*[19].

Kueri SQL dasar untuk mengambil informasi dari database disebut SQL *statement.* Statement ini dibentuk dengan struktur SELECT-FROM-WHERE[19]. Klausa SELECT-FROM yang digunakan dalam kueri mengikuti notasi (1).

$$\pi_{<attribute\ list>}(R) \quad (1)$$

Notasi (1) ini digunakan untuk mengambil atribut (klausa SELECT) dari sebuah relasi *R* (klausa FROM), dimana π (phi) merupakan simbol yang merepresentasikan PROJECT *operation*, dan <*attribute list*> merupakan daftar nama atribut yang nilainya akan diambil oleh kueri.

Untuk pemilihan kondisi dalam kueri, yang merupakan representasi dari klausa WHERE menggunakan notasi (2).

$$\sigma_{<selection\ condition>}(R) \quad (2)$$

Dimana σ (sigma) merupakan simbol yang digunakan dalam pemilihan kondisi, yaitu penetapan atribut-atribut yang digunakan dalam *where condition* dalam kueri. Atribut-atribut dengan kondisi tertentu tersebut dimasukkan ke dalam <*selection condition*> dari relasi *R*.

Fungsi agregat matematis pada kumpulan nilai, seperti SUM, COUNT, AVERAGE, MINIMUM dan MAXIMUM tidak diekspresikan dalam aljabar relasional dasar, namun dapat didefinisikan menggunakan symbol ℑ[19] sehingga dalam kueri diekspresikan dengan notasi (3).

$$<grouping\ attributes>\Im_{<function\ list>}(R) \quad (3)$$

Kueri pembentuk laporan berkala memiliki parameter yang digunakan secara berulang. Parameter tersebut adalah "Tahun Anggaran" yang dalam penelitian ini diberi nama variabel "vta" dengan nilai "2021" atau "2022" atau "2023". Sehingga formula pemilihan kondisi dengan parameter tahun yang merupakan klausa WHERE, dibuat variabel menggunakan notasi (4).

$$\text{P\_TRXvta} \leftarrow \sigma_{(selection\ condition)}(TB\_TRX) \quad (4)$$

Variabel "P_TRXvta" merupakan variabel kondisi dengan nilai "vta" adalah tahun, dan TB_TRX merupakan nama relasi yang menyimpan data transaksi penerimaan negara. Bagian <*selection condition*> diisi dengan notasi $\sigma_{(ta=2021)}$ atau $\sigma_{(ta=2022)}$ atau $\sigma_{(ta=2023)}$.

 ISSN:2302-8149
e-ISSN:2540-9719

Terdapat 42 atribut dalam relasi TB_TRX yang dapat dilihat pada Tabel 3.

Tabel 3. Daftar atribut relasi TB_TRX

No.	Field Name	Type	Alias
1	STAN	VARCHAR2 (6 Byte)	A01
2	LOCAL_DATE_TIME	TIMESTAMP(6)	A02
3	GMT	TIMESTAMP(6)	A03
4	CAID	VARCHAR2 (12 Byte)	A04
5	SDATE	VARCHAR2 (4 Byte)	A05
6	CHANNELTYPE	VARCHAR2 (4 Byte)	A06
7	TERMINAL_ID	VARCHAR2 (16 Byte)	A07
8	TERMINAL_LOCATION	VARCHAR2 (40 Byte)	A08
9	BILLCODE	VARCHAR2 (15 Byte)	A09
10	CUSTOMER_NAME	VARCHAR2 (50 Byte)	A10
11	BUDGET_ID	VARCHAR2 (20 Byte)	A11
12	AMOUNT	NUMBER	A12
13	RESPONSE_CODE	VARCHAR2 (2 Byte)	A13
14	NTPN	VARCHAR2 (16 Byte)	A14
15	PAYMENTSTATUS	VARCHAR2 (2 Byte)	A15
16	BRANCH_CODE	VARCHAR2 (20 Byte)	A16
17	NTB	VARCHAR2 (12 Byte)	A17
18	LOCATION_CODE	VARCHAR2 (4 Byte)	A18
19	BILLING_INFO1	VARCHAR2 (50 Byte)	A19
20	BILLING_INFO2	VARCHAR2 (50 Byte)	A20
21	BILLING_INFO3	VARCHAR2 (50 Byte)	A21
22	BILLING_INFO4	VARCHAR2 (50 Byte)	A22
23	BILLING_INFO5	VARCHAR2 (50 Byte)	A23
24	BILLING_INFO6	VARCHAR2 (50 Byte)	A24
25	BILLING_INFO7	VARCHAR2 (50 Byte)	A25
26	BILLING_INFO8	VARCHAR2 (50 Byte)	A26
27	BILLING_INFO9	VARCHAR2 (50 Byte)	A27
28	BILLING_INFO10	VARCHAR2 (50 Byte)	A28
29	TRX_DATE	TIMESTAMP(6)	A29
30	SWITCHER_ID	VARCHAR2 (3 Byte)	A30
31	CURRENCY	VARCHAR2 (3 Byte)	A31
32	BILLERID	VARCHAR2 (5 Byte)	A32
33	BANK_ACCOUNT_NUMBER	VARCHAR2 (20 Byte)	A33
34	BILLER_NOTIFICATION	INTEGER	A34
35	SPAN_NOTIFICATION	INTEGER	A35
36	ISREVERSAL	INTEGER	A36
37	BILLER_NOTIFICATION_RESPONSE	VARCHAR2 (2 Byte)	A37
38	BILLER_NOTIFICATION_FAILED	INTEGER	A38
39	OLAP_NOTIFICATION	INTEGER	A39
40	TA	VARCHAR2 (4 Byte)	A40
41	UPDATED_DATE	TIMESTAMP(6)	A41
42	TRACE_ID	VARCHAR2 (50 Byte)	A42

 ISSN:2302-8149
e-ISSN:2540-9719

Empat puluh dua atribut yang ada dalam relasi TB_TRX terhimpun ke dalam dua bagian jenis data yaitu data tagihan dan data pembayaran.

Data tagihan merupakan data yang berasal dari biller pemilik tagihan yang menginformasikan setidaknya nama penyetor dan nilai setoran. Data tagihan ini terungkap pada atribut: BILLCODE (A09) menyimpan informasi kode billing/tagihan, CUSTOMER_NAME (A10) yang menyimpan informasi nama penyetor, AMOUNT (A12) yang merupakan nominal pembayaran, BILLING_INFO1 sampai dengan BILLING_INFO10 (A19, A20, A21, A22, A23, A24, A25, A26, A27, A28) yang menyimpan informasi detail tagihan yang berbeda-beda sesuai dengan jenis penerimaan seperti pajak, bea dan cukai serta penerimaan negara bukan pajak dan BILLERID (A32) yang merupakan kode biller.

Data pembayaran merupakan data yang berasal dari hasil proses pembayaran yang dilakukan melalui berbagai kanal pembayaran. Beberapa atribut penting adalah: CAID (A04) yang merupakan atribut kode *Collecting Agent* atau tempat melakukan pembayaran, LOCAL_DATE_TIME yang mengandung informasi tanggal dan waktu pembayaran di sisi *Collecting Agent*, SDATE (A05) yang merupakan tanggal dan bulan pembukuan transaksi, CHANNEL_TYPE (A06) yang menggambarkan informasi jenis kanal pembayaran (teller/ATM/*internet banking*, *mobile banking* dan beberapa kanal lainnya), NTPN (A14) yang merupakan kode unik atas transaksi yang berhasil dibayarkan, TRX_DATE (A29) yang mengandung informasi tanggal dan jam terbentuknya NTPN, CURRENCY (A31) yang memuat informasi kode mata uang, dan TA (A40) yang merupakan kode tahun anggaran atau tahun pembukuan.

Atribut-atribut penting yang dipilih dalam kueri pembentuk laporan berkala terbatas pada A40, A05, A04, A06, A32 dan A31 untuk jenis non-agregasi. Sedangkan atribut yang diagregasi adalah A09 dan A12.

Dari hasil pemilihan atribut yang mengacu jenis laporan berkala sebagaimana pada Tabel 2, maka teridentifikasi delapan kueri pembentuk laporan berkala sebagaimana Tabel 4 berikut.

Tabel 4. Daftar kueri laporan berkala dalam SIM MPN

Kode Kueri	Sintaks
Q-01	Q01TRXvta ← $\pi_{A40,A04,A31,}\Im_{COUNT\ A09,}\Im_{SUM\ A12}(P\_TRXvta)$ R01TRXvta ← $\tau_{A40,A04,A31}(Q01TRXvta)$
Q-02	Q02TRXvta ← $\pi_{A40,substr(A05,1,2),A04,A31,}\Im_{COUNT\ A09,}\Im_{SUM\ A12}(P\_TRXvta)$ R02TRXvta ← $\tau_{A40,substr(A05,1,2),A04,A31}(Q02TRXvta)$
Q-03	Q03TRXvta ← $\pi_{A40,A04,A06,A31,}\Im_{COUNT\ A09,}\Im_{SUM\ A12}(P\_TRXvta)$ R03TRXvta ← $\tau_{A40,A04,A06,A31}(Q03TRXvta)$
Q-04	Q04TRXvta ← $\pi_{A40,substr(A05,1,2),A04,A06,A31,}\Im_{COUNT\ A09,}\Im_{SUM\ A12}(P\_TRXvta)$ R04TRXvta ← $\tau_{A40,substr(A05,1,2),A04,A06,A31}(Q04TRXvta)$
Q-05	Q05TRXvta ← $\pi_{A40,A32,A31,}\Im_{COUNT\ A09,}\Im_{SUM\ A12}(P\_TRXvta)$ R05TRXvta ← $\tau_{A40,A32,A31}(Q05TRXvta)$
Q-06	Q06TRXvta ← $\pi_{A40,substr(A05,1,2),A32,A31,}\Im_{COUNT\ A09,}\Im_{SUM\ A12}(P\_TRXvta)$ R06TRXvta ← $\tau_{A40,substr(A05,1,2),A32,A31}(Q06TRXvta)$
Q-07	Q07TRXvta ← $\pi_{A40,A31,}\Im_{COUNT\ A09,}\Im_{SUM\ A12}(P\_TRXvta)$ R07TRXvta ← $\tau_{A40,A31}(Q07TRXvta)$
Q-08	Q08TRXvta ← $\pi_{A40,substr(A05,1,2),A31,}\Im_{COUNT\ A09,}\Im_{SUM\ A12}(P\_TRXvta)$ R08TRXvta ← $\tau_{A40,substr(A05,1,2),A32,A31}(Q08TRXvta)$

Delapan kueri sebagaimana tabel di atas, dapat diklasifikasikan menjadi empat bagian. Bagian pertama adalah kueri untuk mendapatkan transaksi per CA per Mata Uang. Kueri tersebut adalah Q-01 dan Q-02. Persamaan keduanya adalah nilai pada <*attribute list*> yang diisi dengan A40, A04 dan

 ISSN:2302-8149
e-ISSN:2540-9719

A31 serta agregasi dari atribut A09 dan A12 ($\Im_{COUNT\ A09}$,$\Im_{SUM\ A12}$). Sedangkan pembeda keduanya adalah adanya tambahan atribut A05 untuk mendapatkan informasi "kode bulan" pada Q-02.

Bagian kedua adalah kueri untuk mendapatkan transaksi per CA per Kanal per Mata Uang. Kueri tersebut adalah Q-03 dan Q-04. Persamaan keduanya adalah nilai pada <*attribute list*> yang diisi dengan A40, A04, A06 dan A31 serta agregasi dari atribut A09 dan A12 ($\Im_{COUNT\ A09}$,$\Im_{SUM\ A12}$). Sama halnya dengan bagian pertama, pembeda keduanya adalah adanya tambahan atribut A05 untuk mendapatkan informasi "kode bulan" pada Q-04.

Bagian ketiga adalah kueri untuk mendapatkan transaksi per Biller per Mata Uang. Kueri tersebut adalah Q-05 dan Q-06. Persamaan keduanya adalah nilai pada <*attribute list*> yang diisi dengan A40, A32 dan A31 serta agregasi dari atribut A09 dan A12 ($\Im_{COUNT\ A09}$,$\Im_{SUM\ A12}$). Pembeda kedua kueri tersebut adalah adanya tambahan atribut A05 untuk mendapatkan informasi "kode bulan" pada Q-06.

Dan bagian terakhir kueri untuk mendapatkan transaksi per Mata Uang. Kueri tersebut adalah Q-07 dan Q-08. Persamaan keduanya adalah nilai pada <*attribute list*> yang diisi dengan A40 dan A31 serta agregasi dari atribut A09 dan A12 ($\Im_{COUNT\ A09}$,$\Im_{SUM\ A12}$). Pembeda kedua kueri tersebut pun sama dengan ketiga bagian lainnya, yaitu adanya tambahan atribut A05 untuk mendapatkan informasi "kode bulan" pada Q-08.

Berikut adalah Tabel 5 yang merupakan matriks hubungan antara jenis laporan dengan kueri-kueri laporan berkala.

Tabel 5. Matriks relasi jenis laporan berkala dengan kueri penyusun laporan

No.	Kode Laporan	Kode Kueri	Periodisasi/Parameter
1	MR-01	Q-01	Tahunan
2	YR-01	Q-02	Bulanan
3	MR-02	Q-03	Tahunan
4	YR-02	Q-04	Bulanan
5	MR-03	Q-05	Tahunan
6	YR-03	Q-06	Bulanan
7	MR-04	Q-07	Tahunan
8	YR-04	Q-08	Bulanan

Selain relasi TB_TRX yang menjadi relasi utama yang mengandung data transaksi, terdapat relasi M_CA yang merupakan relasi jenis referensi yang digunakan untuk proses "JOIN". Atas dasar hal ini, penyiapan data pada database riset dilakukan dengan menyalin atau menduplikasi kedua relasi tersebut. Proses *copy* relasi itu sendiri dilakukan dengan membuat relasi baru pada database riset, dengan mengambil data pada database OLAP melalui mekanisme DBLink. Sintaks yang digunakan adalah "CREATE TABLE" dimana data transaksi pada TB_TRX dibatasi dengan parameter A40 bernilai "2021, 2022 dan 2023". Sedangkan untuk data pada relasi M_CA dilakukan penyalinan seluruhnya.

4.2 Pengolahan Data

Dengan memerhatikan atribut dan struktur data yang digunakan dalam membentuk kueri, tidak semua kueri harus membentuk MVs masing-masing. Artinya, delapan kueri pada Tabel 4 tidak mengharuskan terbentuk juga delapan MVs. Kueri-kueri yang menggunakan beberapa atribut yang sama, dapat membentuk satu MVs. Untuk melihat sejauh mana atribut-atribut digunakan dalam *periodic reports query*, telah disusun matriks penggunaan atribut dalam kueri pembentuk laporan berkala sebagaimana pada Tabel 6 di bawah ini.

Tabel 6. Penggunaan atribut dalam *periodic reports query*

No.	Kode Kueri	Atribut								Jumlah Atribut
		A40	A05	A04	A06	A32	A31	A09	A12	
1	Q-01	√	-	√	-	-	√	√	√	5
2	Q-02	√	√	√	-	-	√	√	√	6

 ISSN:2302-8149
e-ISSN:2540-9719

3	Q-03	√	-	√	√	-	√	√	√	6
4	Q-04	√	√	√	√	-	√	√	√	7
5	Q-05	√	-	-	-	√	√	√	√	5
6	Q-06	√	√	-	-	√	√	√	√	6
7	Q-07	√	-	-	-	-	√	√	√	4
8	Q-08	√	√	-	-	-	√	√	√	5
	Jumlah	**8**	**4**	**4**	**2**	**2**	**8**	**8**	**8**	

Tabel di atas menunjukan ada 8 atribut yang dimanfaatkan dalam menyusun *periodic reports query* yaitu A40, A05, A04, A06, A32, A31, A09 dan A12, walaupun tidak ada satu pun kueri yang menggunakan keseluruhan atribut tersebut. Urutan atribut ini menggambarkan urutan dalam sintaks "SELECT" dalam kueri.

Tabel tersebut juga memperlihatkan bahwa ada 4 atribut yang selalu digunakan dalam setiap kueri yaitu A40, A31, A09 dan A12. Sedangkan keempat atribut lainnya tidak selalu digunakan dalam kueri.

Langkah awal dalam perancangan MVs adalah dengan memilih kueri yang menggunakan atribut terbanyak.

Kandidat MVs pertama dibentuk untuk memenuhi kueri pada Q-04 yang memiliki 7 atribut: A40, A05, A04, A06, A31, A09 dan A12. MVs yang disusun berdasarkan Q-04 akan memenuhi kueri Q01, Q02 dan Q03. Bahkan, kueri Q-04 juga bisa meliputi kebutuhan kueri Q-07 dan Q-08.

MVs selanjutnya dapat disusun untuk memenuhi kueri Q-06 dengan atribut yang dipilih yaitu A40, A05, A32, A31, A09 dan A12. Enam atribut yang dipilih oleh kueri Q-06, 5 atribut diantaranya dipilih oleh kueri Q-05, sehingga MVs yang kedua bisa memenuhi kebutuhan kueri Q-05.

Berdasarkan hal tersebut, maka setidaknya 2 MVs yang dapat dirancang dan dibangun untuk mewakili kedelapan kueri pembentuk laporan berkala. Hubungan MVs dengan kueri dapat dilihat dalam Tabel 7 berikut.

Tabel 7. Daftar kandidat *materialized views*

No.	Kode MVs	Nama MVs	Kode Kueri
1	MV-01	MV1	Q-04, Q-01, Q-02, Q-03, Q-07, Q-08
2	MV-02	MV2	Q-06, Q-05

Kueri-kueri yang ada dalam kolom "Kode Kueri" disusun sesuai urutan yang ditentukan oleh jumlah atribut yang digunakan. Semakin banyak atribut yang digunakan, maka menempati posisi paling kiri. Begitu pula sebaliknya, semakin sedikit atribut yang digunakan dalam kueri, maka posisinya semakin ke kanan. Atribut penyusun MVs pertama (MV1) mengacu pada Q-04 yaitu A40, A05, A04, A06, A31, A09 dan A12. Sedangkan atribut penyusun MVs kedua (MV2) mengikuti atribut yang digunakan dalam kueri Q-06 yaitu A40, A05, A32, A31, A09 dan A12.

Berdasarkan kebutuhan penggunaan atribut, maka MVs (MV1 dan MV2) yang disusun untuk meningkatkan efisiensi pemrosesan kueri dalam laporan berkala, dirancang melalui sintaks yang ditampilkan pada Tabel 8 berikut.

Tabel 8. Sintaks pembentuk *materialized views*

MV1

```
CREATE MATERIALIZED VIEW LOG ON TB_TRX
    WITH ROWID, SEQUENCE
    (
        A01,A02,A03,A04,A05,A06,A07,A08,A09,A10,
        A11,A12,A13,A14,A15,A16,A17,A18,A19,A20,
        A21,A22,A23,A24,A25,A26,A27,A28,A29,A30,
        A31,A32,A33,A34,A35,A36,A37,A38,A39,A40,
        A41,A42
    ) INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW MV1
    BUILD IMMEDIATE
    REFRESH FAST ON COMMIT AS
        SELECT
                A40,
```

MV2

```
CREATE MATERIALIZED VIEW MV2
    BUILD IMMEDIATE
    REFRESH FAST ON COMMIT AS
        SELECT
                A40,
                SUBSTR(A05,1,2) AS A05,
                A32,
                A31,
                COUNT(A09) AS TRX,
                SUM(A12) AS TOTAMOUNT
            FROM
                TB_TRX
            GROUP BY
                A40,
                SUBSTR(A05,1,2),
```

 ISSN:2302-8149
e-ISSN:2540-9719

```
    SUBSTR(A05,1,2) AS A05,                    A32,
    A04,                                       A31;
    A06,
    A31,
    COUNT(A09) AS TRX,
    SUM(A12) AS TOTAMOUNT
FROM
    TB_TRX
GROUP BY
    A40,
    SUBSTR(A05,1,2),
    A04,
    A06,
    A31;
```

Sintaks dalam menyusun MVs melibatkan pembuatan MV Log pada tabel utama "TB_TRX" melalui perintah "CREATE MATERIALIZED VIEW LOG". Setelah pembuatan log berhasil, selanjutnya peneliti membangun 2 MVs masing-masing dengan nama "MV1" untuk MV-01 dan "MV2" untuk MV-02. Atribut-atribut pembentuk MVs mengacu pada kueri Q-04 untuk MV1 dan kueri Q-06 untuk MV2. Dua atribut yang dilakukan proses agregasi baik pada MV1 maupun pada MV2 adalah atribut A09 dan A12.

Dengan terbentuknya MV1 dan MV2, maka kueri-kueri yang akan dilakukan pengujian dibagi menjadi 2 bagian: pengujian melalui *views* dengan menggunakan relasi "TB_TRX" sebagai *R*, dan bagian yang lain menguji melalui MVs dengan menggunakan relasi "MV1" atau "MV2". Masing-masing kueri diuji dengan 3 parameter atribut A40 dengan *value* "2021", "2022" dan "2023". Sehingga, setiap kueri akan diuji sebanyak 3 kali. Sedangkan untuk kueri yang diujikan melalui MVs, selain diuji sebanyak 3 kali melalui 3 parameter atribut A40, juga diuji melalui mesin yang berbeda, dengan rincian pengujian pertama dan ketiga dilakukan di PC ke-1 (PC1) dan pengujian kedua dilakukan di PC ke-2 (PC2). Tabel 9 berikut adalah daftar kueri yang dilakukan pengujian, baik yang menggunakan *views* maupun MVs.

Tabel 9. Daftar kueri pengujian

No.	Kode Kueri	Nilai Parameter A40	Kode Kueri Eksperimen	Relasi *R* yang digunakan dalam Views	Relasi *R* yang digunakan dalam MVs
1	Q-01	2021	Q01-21	TB_TRX	MV1
2	Q-02	2021	Q02-21	TB_TRX	MV1
3	Q-03	2021	Q03-21	TB_TRX	MV1
4	Q-04	2021	Q04-21	TB_TRX	MV1
5	Q-05	2021	Q05-21	TB_TRX	MV2
6	Q-06	2021	Q06-21	TB_TRX	MV2
7	Q-07	2021	Q07-21	TB_TRX	MV1
8	Q-08	2021	Q08-21	TB_TRX	MV1
9	Q-01	2022	Q01-22	TB_TRX	MV1
10	Q-02	2022	Q02-22	TB_TRX	MV1
11	Q-03	2022	Q03-22	TB_TRX	MV1
12	Q-04	2022	Q04-22	TB_TRX	MV1
13	Q-05	2022	Q05-22	TB_TRX	MV2
14	Q-06	2022	Q06-22	TB_TRX	MV2
15	Q-07	2022	Q07-22	TB_TRX	MV1
16	Q-08	2022	Q08-22	TB_TRX	MV1
17	Q-01	2023	Q01-23	TB_TRX	MV1
18	Q-02	2023	Q02-23	TB_TRX	MV1
19	Q-03	2023	Q03-23	TB_TRX	MV1
20	Q-04	2023	Q04-23	TB_TRX	MV1
21	Q-05	2023	Q05-23	TB_TRX	MV2
22	Q-06	2023	Q06-23	TB_TRX	MV2
23	Q-07	2023	Q07-23	TB_TRX	MV1
24	Q-08	2023	Q08-23	TB_TRX	MV1

 ISSN:2302-8149
e-ISSN:2540-9719

Tabel 9 memperlihatkan bahwa setiap kueri pada tabel 6 dilakukan pengujian sebanyak 3 kali dengan parameter A40 yang berbeda-beda dan membentuk Kode Kueri Eksperimen yang unik. Kode Kueri Eksperimen tersebut secara berurutan mulai Q01-21 yang merupakan kueri pertama dengan nilai A40 '2021', Q02-21 yang merupakan kueri kedua dengan nilai A40 '2021' sampai dengan Q08-23, yaitu kueri ke-8 dengan nilai A40 '2023'.

Pengujian dibagi menjadi 2 bagian: pengujian melalui *views* dengan menggunakan relasi TB_TRX sebagai *R*, dan bagian yang lain menguji melalui MVs dengan menggunakan relasi MV1 atau MV2. Kueri yang diujikan melalui MVs, selain diuji sebanyak 3 kali dengan 3 parameter atribut A40 yang berbeda, juga diuji melalui mesin yang berbeda, dengan rincian pengujian pertama dan ketiga dilakukan di PC1 dan pengujian kedua dilakukan di PC2.

Hasil pengujian dari kueri eksperimen sebagaimana pada Tabel 9 menghasilkan nilai-nilai *execution time* sebagaimana pada Tabel 10 berikut ini.

Tabel 10. Perbandingan *execution time* antara views dan MVs

Kode Kueri Eksperimen	Total Data (*rows*)	Jumlah Data per parameter A40 (*rows*)	Views (ms) (PC1)	Materialized Views (ms)		
				Uji ke-1 (PC1)	Uji ke-2 (PC2)	Uji ke-3 (PC1)
Q01-21	294.503.898	89.461.110	2.092.000	35	17	14
Q01-22	294.503.898	99.689.824	1.913.000	37	19	15
Q01-23	294.503.898	105.352.964	2.159.000	42	21	20
Q02-21	294.503.898	89.461.110	1.843.000	38	28	19
Q02-22	294.503.898	99.689.824	2.145.000	38	29	20
Q02-23	294.503.898	105.352.964	2.172.000	40	33	23
Q03-21	294.503.898	89.461.110	1.891.000	33	23	18
Q03-22	294.503.898	99.689.824	2.658.000	37	23	19
Q03-23	294.503.898	105.352.964	2.493.000	41	25	21
Q04-21	294.503.898	89.461.110	1.568.000	39	15	11
Q04-22	294.503.898	99.689.824	2.295.000	41	17	12
Q04-23	294.503.898	105.352.964	2.174.000	47	24	17
Q05-21	294.503.898	89.461.110	1.690.000	36	14	4
Q05-22	294.503.898	99.689.824	2.006.000	38	14	5
Q05-23	294.503.898	105.352.964	1.827.000	43	15	8
Q06-21	294.503.898	89.461.110	2.126.000	41	9	8
Q06-22	294.503.898	99.689.824	1.954.000	43	11	9
Q06-23	294.503.898	105.352.964	1.807.000	48	13	11
Q07-21	294.503.898	89.461.110	1.838.000	43	3	3
Q07-22	294.503.898	99.689.824	1.910.000	43	8	3
Q07-23	294.503.898	105.352.964	2.240.000	46	8	6
Q08-21	294.503.898	89.461.110	1.868.000	31	6	4
Q08-22	294.503.898	99.689.824	2.120.000	43	5	6
Q08-23	294.503.898	105.352.964	2.349.000	49	16	11
	Total		**49.138.000**	**972**	**396**	**287**
	Rata-rata		**2.047.417**	**41**	**17**	**12**

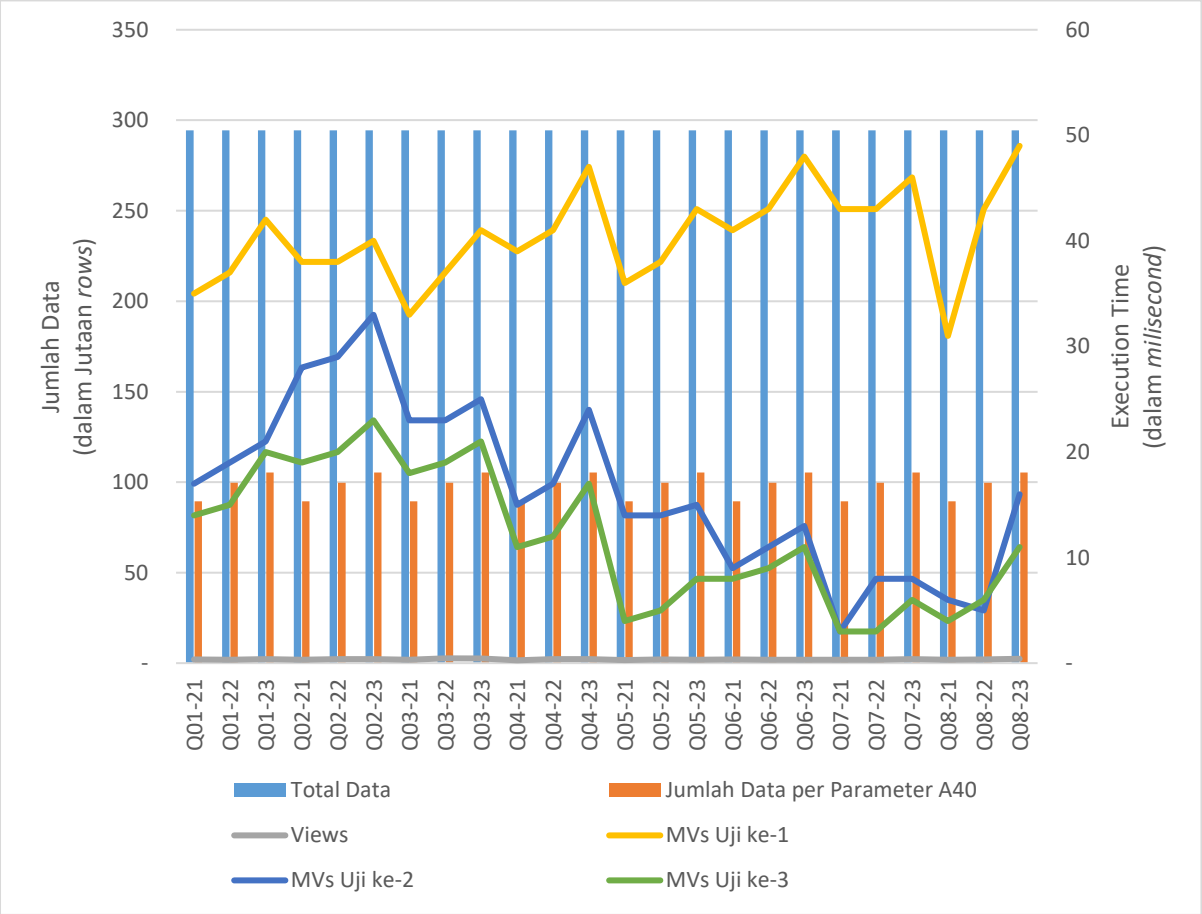
Tabel di atas menunjukkan perbandingan *execution time* antara *views* dan MVs dalam pengujian yang dilakukan dengan berbagai parameter data. Setiap eksperimen diberi kode unik, seperti Q01-21, Q01-22, dan seterusnya hingga Q08-23. Jumlah total data yang diuji tetap sama, yaitu

 ISSN:2302-8149
e-ISSN:2540-9719

sebanyak 294.503.898 baris. Jumlah data per parameter A40 bervariasi antara 89.461.110 hingga 105.352.964 baris.

Hasil pengujian menunjukkan bahwa *execution time* menggunakan *views* dapat berkisar antara 1.568.000 ms hingga 2.658.000 ms, dengan rata-rata waktu eksekusi sebesar 2.047.417 ms. Sementara itu, *execution time* menggunakan MVs jauh lebih cepat. Pada pengujian pertama (PC1), rata-rata waktu eksekusi adalah 41 ms, pada pengujian kedua (PC2) adalah 17 ms, dan pada pengujian ketiga (PC1) adalah 12 ms.

Total waktu eksekusi *views* secara keseluruhan mencapai 49.138.000 ms, jauh lebih tinggi dibandingkan total waktu eksekusi MVs yang hanya 972 ms pada pengujian pertama, 396 ms pada pengujian kedua, dan 287 ms pada pengujian ketiga.



Gambar 4. Perbandingan *execution time* antara *views* dan MVs dengan jumlah data

Gambar 4 ini adalah grafik yang menampilkan hubungan antara jumlah data dan *execution time* dalam konteks uji coba perbandingan antara *views* dan MVs untuk berbagai kueri (Q01-21 hingga Q08-23). Sumbu horizontal (X-Axis) merepresentasikan kueri-kueri yang diuji. Grafik ini juga memiliki dua sumbu vertikal dengan fungsi berbeda: sumbu kiri (Y-Axis) menunjukkan jumlah data dalam satuan jutaan baris (*rows*), sedangkan sumbu kanan (Y-Axis) menunjukkan *execution time* dalam milidetik (ms).

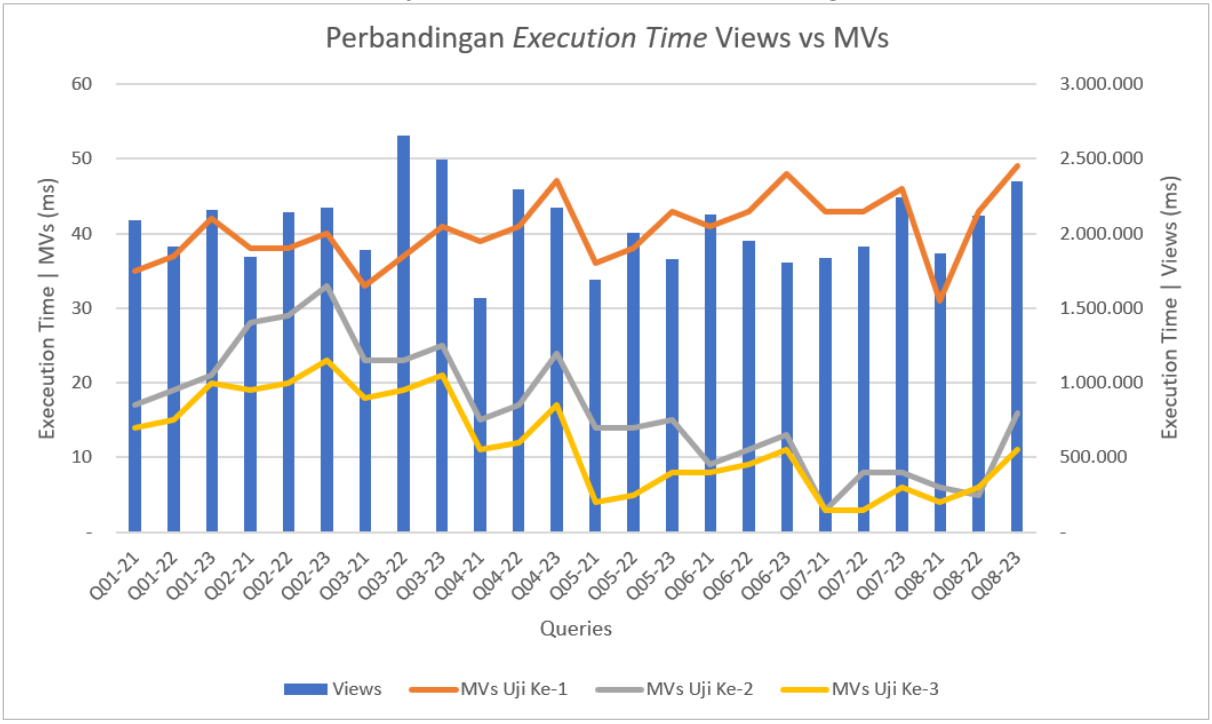
Pada sumbu vertikal kiri, grafik menampilkan dua elemen utama terkait jumlah data: total jumlah data (ditampilkan dengan bar biru) dan jumlah data yang diproses untuk parameter spesifik A40 (ditunjukkan oleh bar oranye).

Sumbu vertikal kanan menggambarkan *execution time* untuk *views* (abu-abu) dan tiga uji coba MVs (kuning untuk uji ke-1, biru tua untuk uji ke-2 dan hijau untuk uji ke-3). Dengan skala *execution time* hanya dalam satuan puluhan, menyebabkan *line charts* untuk *views* tidak dapat terdeteksi pada grafik tersebut.

Untuk memenuhi kebutuhan penayangan *line charts* untuk *views*, peneliti menampilkan grafik pada Gambar 5 dengan mengganti sumbu kiri (Y-Axis) yang semula menunjukan jumlah data menjadi

 ISSN:2302-8149
e-ISSN:2540-9719

execution time untuk MVs. Sedangkan sumbu kanan (Y-Axis) semula menunjukan *execution time* untuk *views* dan MVs, diubah menjadi *execution time* untuk *views* dengan skala lebih besar.



Gambar 5. Perbandingan *execution time* antara *views* dan MVs

Dari grafik yang diperlihatkan oleh Gambar 5, dapat diamati bahwa *execution time* untuk *views* (dengan bar biru) secara konsisten lebih tinggi dibandingkan *execution time* MVs pada semua uji coba. MVs dalam tiga uji coba (ditunjukkan dengan garis oranye, abu-abu, dan kuning) memiliki variasi nilai yang lebih rendah, yang menunjukkan efisiensi lebih baik dalam waktu eksekusi dibandingkan *views*. Hal ini menunjukkan bahwa MVs dapat secara signifikan mengurangi *execution time* kueri, terutama pada uji coba kedua dan ketiga (abu-abu dan kuning).

Salah satu alasan utama mengapa MVs lebih cepat dibandingkan *views* adalah karena hasil kueri yang membentuk MVs telah disimpan secara fisik dalam bentuk tabel terpisah. Berbeda dengan *views* yang harus menjalankan kueri secara *real-time* setiap kali diakses, MVs menyimpan hasil eksekusi kueri tersebut sehingga saat diakses, sistem hanya mengambil data dari tabel hasil MVs tanpa perlu melakukan pemrosesan ulang. Hal ini sejalan dengan temuan bahwa penggunaan MVs mampu menghemat konsumsi memori dan prosesor karena hasil agregasi telah tersedia secara langsung dalam tabel yang sudah terstruktur[8].

MVs juga memungkinkan penggunaan indeks yang lebih optimal karena data yang disimpan dalam MVs telah berupa hasil agregasi atau hasil pengolahan tertentu yang lebih ringkas dibandingkan tabel sumber aslinya. Dengan adanya indeks yang sesuai pada kolom-kolom hasil agregasi tersebut, pencarian data menjadi jauh lebih cepat dibandingkan kueri yang menggunakan *views* yang melibatkan pencarian pada tabel sumber dengan volume data yang jauh lebih besar[6].

Penurunan *execution time* secara signifikan pada ujicoba kedua dan ketiga disebabkan adanya mekanisme *caching* pada basis data Oracle menggunakan *Oracle Database In-Memory* (DBIM) yang secara otomatis menyimpan hasil kueri dalam *In-Memory Expression Units* (IMEUs)[20][21][22]. Ini memungkinkan kueri yang berulang dan dengan hasil yang sama, akan mengambil langsung dari memori tanpa perlu melakukan eksekusi ulang, sehingga mempercepat pengambilan data secara signifikan.

Dari hasil ini, terlihat bahwa MVs menawarkan keunggulan performa yang sangat signifikan dibandingkan *views*. MVs menjadi pilihan yang lebih efisien untuk menurunkan nilai *execution time*, terutama untuk skenario dengan jumlah data yang besar dan kebutuhan akses yang cepat. Dan ini memberikan bukti kuat tentang efektivitas MVs dalam meningkatkan performa sistem basis data dan kebutuhan pembentukan laporan berkala.

 ISSN:2302-8149
e-ISSN:2540-9719

5 Kesimpulan

Pengujian terhadap 24 jenis kueri pada 8 tipe *periodic reports query* yang menggunakan 294.503.898 baris data pada database Oracle SIM MPN, menunjukkan bahwa pemanfaatan *Materialized Views* (MVs) dapat mempercepat *execution time* dibandingkan *views*, dengan rata-rata 41 ms, 17 ms, dan 12 ms dalam tiga tahap pengujian, sementara *views* memerlukan 2.047.417 ms. Hasil ini mendukung teori bahwa MVs mengurangi konsumsi sumber daya sistem dengan menyimpan hasil agregasi dalam tabel fisik[8] serta meningkatkan efisiensi pengambilan data melalui indeks[6]. Implementasi MVs terbukti sebagai solusi optimal dalam menangani volume data besar untuk mempercepat penyajian laporan berkala dan pengambilan keputusan berbasis data. Namun, penelitian ini memiliki keterbatasan dalam mempertimbangkan waktu pembentukan awal dan proses *refresh* MVs, serta belum menguji dampak beban sistem di lingkungan transaksi *real-time*. Oleh karena itu, penelitian lanjutan disarankan untuk mengevaluasi *execution time* secara menyeluruh, termasuk proses pembentukan awal, waktu *refresh*, serta analisis faktor yang memengaruhi penurunan nilai *execution time* pada berbagai tahap pengujian guna memahami performa MVs dalam jangka panjang.

Referensi

[1] M. R. Islam, "*Impacts of Management Information System on Decision Making of the Organization*," *Int. J. Bus. Soc. Sci. Res*, vol. 6, no. 2, pp. 56–61, Mar. 2018, doi: 10.18034/gdeb.v8i2.100.

[2] S. Hahn, J. Reineke, and R. Wilhelm, "*Towards Compositionality in Execution Time Analysis-Definition and Challenges*," *Association for Computing Machinery*, vol. 12, no. 1, p. 28, Mar. 2015, doi: 10.1145/2752801.2752805.

[3] H. Zhang, G. Chen, B. Chin Ooi, W.-F. Wong, S. Wu, and Y. Xia, "'*Anti-Caching'-based Elastic Memory Management for Big Data*," 2015. doi: 10.1109/ICDE.2015.7113375.

[4] K. Bok, S. Yoo, D. Choi, J. Lim, and J. Yoo, "*In-Memory Caching for Enhancing Subgraph Accessibility*," *Applied Sciences (Switzerland)*, vol. 10, no. 16, pp. 1–18, Aug. 2020, doi: 10.3390/app10165507.

[5] S. D. Choudari and R. Agrawal, "*Optimization Design of Query Processing Performance using Appropriate Materialized View Selection & Preservation*," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 3, no. 12, pp. 8893–8896, Dec. 2014, doi: 10.17148/ijarcce.2014.31249.

[6] R. Lumbantoruan, L. Siringoringo, E. Y. Sinaga, and E. Sitanggang, "Penerapan *Materialized View* yang berindeks pada Basis Data Studi Kasus: *Online Library Information System* Politeknik Del," in *National Conference of Information System*, Surabaya: National Conference of Information System, Dec. 2010. [Online]. Available: https://www.researchgate.net/publication/260339724

[7] N. M. Khushairi, N. A. Emran, and A. K. Menon, "*Database Tuning using Oracle Materialized View for Manufacturing Industry*," *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 10, pp. 38–46, 2018, [Online]. Available: www.mirlabs.net/ijcisim/index.html

[8] Y. Ziya Ayık and F. Kahveci, "*Materialized View Effect on Query Performance," International Journal of Computer and Information Engineering*, vol. 11, no. 9, 2017.

[9] E. Witono and Parno, "Perbandingan *Response Time* Penggunaan *Index*, *Views*, dan *Materialized Views Database MySQL*," *Jurnal Sains Komputer & Informatika (J-SAKTI*, vol. 6, no. 1, pp. 499–506, 2022, doi: 10.30645/j-sakti.v6i1.463.

[10] F. Surya Nugraha, F. H. Purwanto, and F. E. N. Saputro, "Optimasi *Query* pada Laporan Transaksi Penjualan menggunakan *Materialized View* (Studi Kasus : Moonly café)," *CCIT (Creative Communication and Innovative Technology) Journal*, vol. 11, no. 1, pp. 1–14, 2018.

[11] T. V. V. Kumar and K. Devi, "*An Architectural Framework for Constructing Materialized Views in a Data Warehouse*," *International Journal of Innovation, Management and Technology*, 2013, doi: 10.7763/ijimt.2013.v4.390.

 ISSN:2302-8149
e-ISSN:2540-9719

[12] M. Manavi, "*Multi-Objective Genetic Algorithm for Materialized View Optimization in Data Warehouses*," *Interdisciplinary Conference on Electrics and Computer (INTCEC 2024)*, Jun. 2024.
[13] J. Prakash and T. V. V. Kumar, "*A Multi-Objective Approach for Materialized View Selection*," *International Journal of Operations Research and Information Systems*, vol. 10, no. 2, pp. 1–19, 2019, doi: 10.4018/IJORIS.2019040101.
[14] A. Gosain and K. Sachdeva, "*Materialized View Selection for Query Performance Enhancement using Stochastic Ranking based Cuckoo Search Algorithm*," *International Journal of Reliability, Quality and Safety Engineering*, vol. 27, no. 3, Jun. 2020, doi: 10.1142/S0218539320500084.
[15] N. Emran, "*Database Performance Tuning Methods for Manufacturing Execution System*," *World Appl SCI J*, vol. 30, Jan. 2014, doi: 10.5829/idosi.wasj.2014.30.icmrp.14.
[16] A. Kour, "*Data Warehousing, Data Mining, OLAP and OLTP Technologies are Indispensable Elements to Support Decision-Making Process in Industrial World*," *International Journal of Scientific and Research Publications*, vol. 5, no. 5, May 2015, [Online]. Available: www.ijsrp.org
[17] A. Turnip *et al.*, "*Backend Design of Web-based ECG Signal Monitoring System*," Scitepress, Mar. 2021, pp. 317–324. doi: 10.5220/0010371403170324.
[18] M. H. Moghadam, M. Saadatmand, M. Borg, M. Bohlin, and B. Lisper, "*Learning-based Response Time Analysis in Real-Time Embedded Systems: A Simulation-based Approach*," in *Proceedings - International Conference on Software Engineering*, IEEE Computer Society, May 2018, pp. 21–24. doi: 10.1145/3194095.3194097.
[19] R. Elmasri and S. B. Navathe, *Database Systems SEVENTH EDITION*. 2016.
[20] A. Mishra *et al.*, "*Accelerating Analytics with Dynamic In-Memory Expressions*," Sep. 2016. doi: 10.14778/3007263.3007280.
[21] S. Shoyab and V. Sriharsha, "*Real-Time Fault-Tolerant Analytics using Distributed Database in-Memory*," *International Journal of Research*, vol. 5, no. 17, Jul. 2018, doi: 10.1109/ICDE.2016.7498333.
[22] N. Mukherjee *et al.*, "*Distributed Architecture of Oracle Database in-memory*," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, Aug. 2015, doi: 10.14778/2824032.2824061.