Jurnal Ilmiah
Teknologi Sistem
Informasi

Jitsi

Vol. 6 (2025) No. 3

https://jurnal-itsi.org/index.php/jitsi; E-ISSN: 2722-4600; ISSN: 2722-4619
DOI: 10.62527/jitsi.6.3.501

Implementasi Metode *Query Optimization* pada Sistem *Point of Sale* Berbasis Web di Toko Eci Dolok

Syahrani Arrahma[#], Suendri[#]

[#] *Jurusan Sistem Informasi, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sumatera Utara, Jl. Lapangan Golf, Desa Durian Jangak, Kec. Pancur Batu Kabupaten Deli Serdang, Provinsi Sumatera Utara, 20353, Indonesia*
E-mail: arrahmasyahrani[at]gmail.com , suendri[at]gmail.com

A B S T R A C T S

This research is motivated by the problem of manual transaction recording at Toko Eci Dolok, which is inefficient and complicates the preparation of sales reports. The purpose of this study is to design a faster and more efficient web-based Point of Sale (POS) system through the application of query optimization. The research applies a quantitative approach by implementing selection, projection, and join skeleton operations on data consisting of 772 product items, sales transactions, cashiers, and suppliers. The experimental results show that query optimization successfully reduced the total execution cost from 60,036 pages to 20,012 pages (a savings of 66.7%) and increased execution speed by three times. In conclusion, query optimization is effective in improving data processing efficiency in POS systems and has the potential to be applied on a larger scale, such as e-commerce or management information systems.

Manuscript received Sep 9, 2025; revised Sep 23, 2025. accepted Sep 24, 2025 Date of publication Sep 30, 2025. International Journal, JITSI : Jurnal Ilmiah Teknologi Sistem Informasi licensed under a Creative Commons Attribution-Share Alike 4.0 International License

A B S T R A K

Penelitian ini dilatarbelakangi oleh permasalahan pencatatan transaksi di Toko Eci Dolok yang masih dilakukan secara manual sehingga kurang efisien dan menyulitkan dalam pembuatan laporan. Tujuan penelitian adalah merancang sistem Point of Sale (POS) berbasis web yang lebih cepat dan efisien melalui penerapan metode query optimization. Metode penelitian menggunakan pendekatan kuantitatif dengan mengimplementasikan operasi seleksi, proyeksi, dan join skeleton pada data yang terdiri dari 772 item barang, transaksi penjualan, kasir, dan supplier. Hasil pengujian menunjukkan bahwa optimasi query mampu menurunkan total biaya eksekusi dari 60.036 pages menjadi 20.012 pages (hemat 66,7%) serta meningkatkan kecepatan eksekusi hingga tiga kali lipat. Kesimpulannya, metode query optimization efektif meningkatkan efisiensi pengolahan data pada sistem POS dan berpotensi diterapkan pada skala yang lebih luas, seperti e-commerce maupun sistem informasi manajemen.

Keywords / Kata Kunci — ***Point of Sale; Query Optimization; Seleksi; Proyeksi; Join Skeleton***

CORRESPONDING AUTHOR

Syahrani Arrahma
Jurusan Sistem Informasi, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sumatera Utara, Indonesia
Email: arrahmasyahrani[at]gmail.com

1. PENDAHULUAN

Perkembangan teknologi informasi memberikan dampak besar dalam meningkatkan efisiensi dan efektivitas pekerjaan manusia. Dalam dunia bisnis, pemanfaatan teknologi memungkinkan transaksi lebih terstruktur, cepat,

dan sistematis [1]. Menurut Ermewaningsih (2023), seiring perkembangan tersebut, pengguna dituntut untuk mampu memanfaatkan teknologi secara optimal [2]. Salah satu peran pentingnya terlihat pada penggunaan sistem point of sale (POS) yang mampu membantu pengusaha melakukan pencatatan transaksi, mengontrol stok, hingga menyusun laporan penjualan [3]. Namun, hasil observasi di Toko Eci Dolok, sebuah toko retail di Pancur Batu, menunjukkan bahwa pencatatan transaksi dan pengelolaan stok masih dilakukan secara manual menggunakan kertas dan kalkulator. Kondisi tersebut menimbulkan berbagai kendala, seperti keterlambatan pelayanan, risiko kesalahan pencatatan, sulitnya penyusunan laporan yang akurat, serta ketiadaan arsip transaksi yang terstruktur.

Studi mengenai query optimization dan sistem point of sale (POS) telah banyak dilakukan oleh peneliti sebelumnya. Widiyanti et al. (2024) mengembangkan POS berbasis web di PT Abercode yang valid dalam pencatatan transaksi namun belum menekankan optimasi query [4], sedangkan Aidil (2024) menerapkan POS pada UMKM Poci Nayla untuk digitalisasi transaksi tetapi masih sebatas otomasi tanpa optimasi query [5]. Tambunan et al. (2025) menerapkan indexing dan stored procedures pada sistem informasi akademik dengan peningkatan performa query hingga 95%, namun terbatas pada domain pendidikan [6], sementara Erkamim et al. (2021) membandingkan hash join dan inner join pada tracer study dan menemukan bahwa hash join lebih cepat untuk data besar tetapi tidak diarahkan pada sistem retail [7]. Selain itu, Alamsyah et al. (2023) membangun POS berbasis web pada toko bangunan dengan hasil positif, tetapi tidak menilai pengaruh optimasi query [8]. Dari berbagai penelitian tersebut, dapat disimpulkan bahwa meskipun query optimization terbukti meningkatkan performa basis data dan POS mendukung digitalisasi usaha, belum ada penelitian yang mengintegrasikan keduanya secara langsung dalam konteks UMKM retail, sehingga penelitian ini hadir untuk mengisi celah tersebut melalui penerapan query optimization pada sistem POS Toko Eci Dolok guna meningkatkan efisiensi transaksi, pengelolaan stok, serta laporan penjualan real-time.

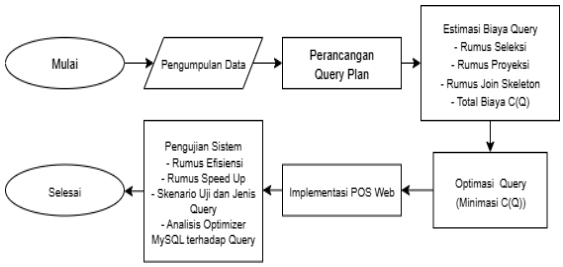
Berdasarkan kesenjangan tersebut, penelitian ini dilaksanakan untuk menjawab kebutuhan nyata di Toko Eci Dolok yang masih menggunakan metode manual dalam pencatatan transaksi dan pengelolaan stok. Sistem yang ada saat ini tidak efisien, rawan kesalahan, dan tidak mampu menghasilkan laporan secara otomatis. Dengan menerapkan sistem kasir digital berbasis POS yang dilengkapi query optimization, diharapkan dapat tercapai peningkatan efisiensi pelayanan, akurasi pencatatan, serta penyediaan laporan penjualan real-time yang dapat mendukung pengambilan keputusan.

Sejalan dengan latar belakang dan tinjauan literatur, penelitian ini mengajukan pertanyaan utama: (1) Bagaimanakah mengimplementasikan metode query optimization pada proses sistem point of sale di Toko Eci Dolok berbasis web? (2) Bagaimanakah membangun sistem point of sale dengan mengimplementasikan metode query optimization di Toko Eci Dolok berbasis web? Sehingga tujuan dari penelitian ini adalah merancang dan mengimplementasikan sistem kasir digital berbasis POS dengan optimasi query di Toko Eci Dolok, mempercepat pemrosesan transaksi, meminimalkan kesalahan pencatatan, serta menghasilkan laporan penjualan otomatis. Selain itu, penelitian ini diharapkan memberikan kontribusi akademis sebagai referensi penerapan query optimization pada sistem POS, sekaligus menjadi rujukan bagi bisnis retail lain dengan permasalahan serupa

2. METODOLOGI PENELITIAN

Penelitian ini menggunakan metode kuantitatif untuk menganalisis penerapan *query optimization* pada sistem *Point of Sale* (POS) berbasis web di Toko Eci Dolok. Data dikumpulkan melalui pengujian langsung dengan membandingkan waktu eksekusi *query* sebelum dan sesudah optimasi. Hasil pengujian kemudian dianalisis menggunakan perhitungan persentase efisiensi guna mengukur peningkatan performa sistem secara objektif.

Penjelasan lebih rinci mengenai setiap tahap tersebut disajikan pada bagian berikut :



GAMBAR 1. Tahapan Penelitian

1. Pengumpulan Data

Dalam penelitian ini, pengumpulan data dilakukan melalui tiga teknik utama, yaitu observasi, wawancara, dan studi pustaka.

a. Observasi dilakukan secara langsung di Toko Eci Dolok untuk memperoleh data dan variabel yang selama ini dicatat secara manual, antara lain:
   (1) Dari Nota / Buku Penjualan Harian: Nomor Nota/Kode Transaksi, Tanggal & Waktu Transaksi, Nama Barang yang Dibeli, Jumlah Barang (Qty), Harga Satuan, Total Harga, serta Nama Kasir (jika dicatat).
   (2) Dari Buku Daftar Barang / Produk: Nama Barang, Stok, Satuan, Kategori Barang (misalnya sembako, minuman, makanan), Harga Jual, serta Harga Beli (opsional).
   (3) Dari Buku Kas / Catatan Kasir: Nama Kasir dan Jumlah Setoran Kas Harian.

(4) Dari Catatan *Supplier*: Nama *Supplier*, Barang yang Disuplai, serta Kontak atau Catatan Pembayaran.

b. Wawancara dilakukan dengan pemilik toko untuk menggali informasi mengenai permasalahan yang dihadapi serta kebutuhan terhadap sistem yang diinginkan.

c. Studi pustaka dilakukan dengan menelaah berbagai literatur dan penelitian terdahulu yang relevan, khususnya mengenai penerapan metode *query optimization* dan sistem *point of sale* berbasis web. Kajian literatur ini digunakan sebagai landasan teoritis sekaligus referensi dalam merancang solusi sistem yang lebih optimal.

2. Perancangan Query Plan

Perancangan query plan merupakan tahap penting dalam mengoptimalkan kinerja sistem basis data, khususnya pada sistem *Point of Sale* (POS) berbasis web di Toko Eci Dolok. Pada tahap ini, *query* yang diajukan oleh pengguna tidak langsung dieksekusi, melainkan terlebih dahulu diterjemahkan oleh *query optimizer* ke dalam bentuk rencana eksekusi (*execution plan*). Proses ini mencakup analisis struktur *query*, pemilihan strategi akses data, serta penentuan urutan operasi seperti seleksi, proyeksi, dan join skeleton.

a. Seleksi
Seleksi merupakan operasi pada *query* yang berfungsi mengambil *tuple*(σ) dari suatu relasi dengan syarat tertentu, sehingga hanya data yang sesuai kondisi yang akan diteruskan ke tahap berikutnya. Dalam proses optimasi *query*, seleksi biasanya ditempatkan sedekat mungkin dengan sumber data (*push down selection*) agar volume data yang diproses berkurang dan eksekusi menjadi lebih efisien. Seleksi juga digunakan untuk mengambil seluruh kolom yang ada dalam sebuah tabel. Namun, apabila hanya kolom tertentu saja yang diperlukan, sebaiknya dituliskan secara spesifik nama kolom yang ingin diakses [9].

b. Proyeksi
Proyeksi(π) adalah operasi dalam aljabar relasional yang memilih subset kolom (atribut) tertentu dari sebuah relasi atau tabel, menghilangkan atribut yang tidak diperlukan dalam hasil *query*. Dalam optimasi *query*, proyeksi berfungsi mengurangi ukuran data (jumlah kolom) yang diproses dan dikembalikan, sehingga mempercepat eksekusi dan mengurangi penggunaan sumber daya. Lakukan proyeksi sedini mungkin untuk membuat tupel yang lebih kecil dan mengurangi hasil antara (penurunan) dan proyeksikan semua atribut kecuali yang diminta atau diperlukan .

c. Join Skeleton
Join Skeleton adalah representasi kerangka dasar dari *query join* yang hanya menunjukkan tabel-tabel (relasi) yang terlibat dan hubungan join antar tabel tersebut, tanpa memperhatikan kondisi seleksi(σ), proyeksi(π), ataupun atribut detail lainnya. Join skeleton membuat tabel besar cukup dibaca sekali saja untuk semua *query* sehingga menghemat biaya I/O banyak sekali.

3. Estimasi Biaya Query

Estimasi biaya *query* adalah proses memperkirakan sumber daya yang diperlukan untuk mengeksekusi sebuah rencana *query* dalam sistem basis data. Estimasi ini digunakan oleh *query optimizer* untuk melakukan optimasi terhadap biaya dari berbagai alternatif yang tersedia, kemudian memilih salah satu dengan nilai biaya paling rendah.

a. Rumus Seleksi
Dibawah ini merupakan query seleksi dengan kondisi berbeda dapat digabung menjadi satu *query* seleksi dengan kondisi logika OR, menggunakan rumus:[10]

$$\sigma_{\theta_1}(r)\psi\sigma_{\theta_2}(r)\psi \ldots \psi\sigma_{\theta_n}(r) =_{\triangleright} \sigma_{\theta_1 \vee \theta_2 \ldots \theta_n}(r) \quad (1)$$

Sedangkan, kondisi seleksi tiap *query* terdiri dari bagian khusus query (θ_i') dan bagian umum (θ_I) yang dipakai bersama antar *query* dengan kondisi logika AND, menggunakan rumus:

$$\theta_i = \wedge_{all}(\theta_i) = (\wedge_{all}(\theta_i')\wedge (\wedge_{all}(\theta_I) = \theta_i' \wedge \theta_I \quad (2)$$

b. Rumus Proyeksi
Proyeksi(π) merupakan salah satu operasi dasar dalam aljabar relasional yang digunakan untuk mengambil atribut tertentu dari suatu relasi. Operasi ini berfungsi untuk menyederhanakan hasil *query* dengan hanya menampilkan kolom (*field*) yang relevan sesuai kebutuhan, sehingga data yang dihasilkan lebih ringkas dan efisien. Operasi proyeksi dapat dituliskan sebagai berikut:

$$\Pi_{L_1}(\Pi_{L_2}\left(\ldots \Pi_{L_n}(r)\right)) \equiv \Pi_{L_1}(r) \quad (3)$$

Keterangan:
- Π = Simbol operasi proyeksi dalam aljabar relasional.
- $L_1, L_2, \ldots L_n$ = Daftar atribut (kolom) yang dipilih dari relasi r.
- r = Relasi (tabel) asal.

c. Rumus Join Skeleton
Dalam *query optimization*, Join Skeleton memegang peran penting karena menjadi dasar bagi sistem untuk menentukan urutan join (*join order*) yang paling efisien. Dengan mengetahui skeleton dari suatu join, DBMS dapat mengevaluasi berbagai kemungkinan rencana eksekusi, menghitung estimasi biaya $C(Q)$, lalu memilih strategi yang menghasilkan biaya terendah. Operasi proyeksi dapat dituliskan sebagai berikut:

$$q_1 \psi q_2 \dots \psi q_n =_{\triangleright} \psi_c \forall T_j \in T\ T_j \quad (4)$$

Keterangan:
- $q_1 q_2 \dots q_n$ = *query-query* awal,
- $T = \{T_1, T_2, \dots\}$ = himpunan tabel yang terlibat dalam *query*,
- $C = \theta_1 \vee \theta_2 \vee \dots \vee \theta_n$ = kondisi gabungan dari seluruh *query*.

d. Rumus Total Biaya C(Q)
Total biaya *query* $C(Q)$ adalah jumlah keseluruhan biaya eksekusi yang timbul dari setiap operator dalam rencana *query*. Rumusnya dapat dinyatakan sebagai:

$$C(Q) = \sum_{i=1}^{n} C(q_i) \quad (5)$$

Keterangan:
- $C(Q)$: total biaya eksekusi dari sebuah *query* Q.
- n : jumlah operator yang terlibat dalam rencana eksekusi *query*.
- q_i : operator ke- i dalam rencana *query* (misalnya: scan, join, sort, proyeksi).
- $C(q_i)$: estimasi biaya dari operator ke- i, biasanya dihitung berdasarkan sumber daya yang digunakan.

4. Optimasi Query (Minimasi C(Q))

Optimasi *query* merupakan proses yang dilakukan oleh sistem basis data untuk menentukan rencana eksekusi query yang paling efisien. Tujuan utama dari optimasi *query* adalah meminimalkan total biaya eksekusi *query* $C(Q)$ [11], sehingga waktu pemrosesan lebih singkat dan penggunaan sumber daya menjadi lebih hemat. Optimasi query bermanfaat dalam mengurangi penggunaan CPU dan memori sehingga dapat mencegah terjadinya hambatan kinerja sistem. *Query* yang tidak dioptimalkan sering menyebabkan penggunaan CPU berlebih akibat operasi join yang kurang efisien, proses penyortiran yang tidak diperlukan, dan pemindaian tabel secara menyeluruh. Dengan menggunakan strategi optimasi berbasis biaya, *query* dapat dijalankan melalui rencana eksekusi yang lebih efisien sehingga beban kerja sistem menjadi lebih ringan [12]. Rumusnya dapat dinyatakan sebagai:

$$C(Q)_{optimasi} = \sum_{T_j \in T} B_{T_j} \quad (6)$$

Keterangan:
- $C(Q)_{optimasi}$: total biaya *query* setelah dilakukan optimasi.
- $\sum$: penjumlahan semua komponen biaya.
- T : himpunan tabel/relasi yang terlibat dalam *query*.
- T_j : tabel atau relasi ke-jjj di dalam himpunan T.
- B_{T_j} : biaya (cost) yang diasosiasikan dengan tabel/relasi T_j, misalnya biaya akses (I/O), pemrosesan (CPU), atau komunikasi (pada basis data terdistribusi).

5. Implementasi Sistem

Mengimplementasikan hasil *query* yang telah dioptimasi ke dalam sistem *Point of Sale* berbasis web yang bertujuan untuk menemukan jalur akses dengan biaya eksekusi terendah agar waktu pemrosesan *query* lebih singkat dan respons sistem lebih cepat. Dengan pemilihan jalur akses yang tepat, sistem menjadi lebih efisien, beban server berkurang, dan kualitas layanan meningkat, sekaligus tetap menghasilkan output yang benar dengan cara paling optimal.

6. Pengujian Sistem

Tahap terakhir adalah melakukan pengujian untuk mengevaluasi kinerja sistem sebelum dan sesudah optimasi *query*. Pengujian dilakukan dengan membandingkan waktu eksekusi *query*.

a. Rumus Efesiensi (Perbandingan Cost)
Efisiensi diukur menggunakan rumus:

$$Efesiensi(\%) = \frac{X_{before} - X_{after}}{X_{before}} \times 100\% \quad (7)$$

b. Rumus Speedup (Waktu Eksekusi)
Tingkat percepatan dihitung dengan:

$$Speedup = \frac{X_{before}}{X_{after}} \quad (8)$$

c. Skenario Uji dan Jenis Query
Pengujian dirancang untuk tiga skenario beban, yaitu rendah (50–100 transaksi), sedang (500–1000 transaksi), dan tinggi (>5000 transaksi). Ketiga skenario tersebut dipilih untuk merepresentasikan kondisi operasional sistem POS mulai dari toko sepi hingga kondisi sibuk. *Query* yang diuji difokuskan pada menu laporan, yaitu laporan penjualan, laporan stok, laporan transaksi, laporan kasir, dan laporan produk terlaris.

d. Analisis Optimizer MySQL terhadap Query
Analisis terhadap *optimizer* MySQL dilakukan dengan membandingkan rencana eksekusi *query* antara mode *single* dan *batch*. Perbandingan ini mencakup pemilihan indeks, urutan join, dan biaya yang dihasilkan. Dengan pendekatan ini, dapat diketahui mengapa eksekusi *batch* lebih efisien.

3. Hasil Dan Pembahasan

1. Pengumpulan Data

Data yang digunakan dalam penelitian ini berasal dari Toko Eci Dolok dan mencakup informasi mengenai nama barang, satuan, stok, harga beli, serta harga jual. Jumlah keseluruhan data yang diperoleh adalah 772 item barang. Proses pengumpulan data dilakukan secara langsung melalui rekapan buku milik toko dan pencatatan manual dari barang yang ada di rak, kemudian data tersebut diolah lebih lanjut untuk keperluan pencatatan.

a. Data Penjualan Harian
Dalam artikel jurnal ini, ditampilkan 5 sampel data penjualan harian sebagai representasi dari keseluruhan populasi data.

TABEL 1. Data Penjualan Harian

Kode	Tanggal & Waktu	Nama Barang	Qty	Harga Satuan	Total Harga	Kasir
T-012	07-08-2025 16:08	Indomie Kaldu Antangin Cair Gaga Besar	10 12 5	Rp. 2.500 Rp.3.000 Rp.21.000	Rp. 166.000	Chika
T-003	12-08-2025 12:05	Altra Ziga Sampoerna Besar Sampoerna Kecil	10 5 10 10	Rp. 8.300 Rp. 12.900 Rp. 21.500 Rp. 15.000	Rp. 512.500	Chika
T-038	24-08-2025 17:51	Soklin Lantai SST SOS Sereh Besar	15 6	Rp. 5.000 Rp. 13.000	Rp. 205.000	Putri
T-054	25-08-2025 12:11	Ale-Ale Aqua Galon	5 5	Rp. 20.000 Rp. 20.000	Rp. 200.000	Chika
T-058	29-08-2025 18:59	Dettol Kecil Energen Hansaplas	12 3 5	Rp. 3.500 Rp.17.000 Rp. 29.000	Rp. 238.000	Putri
...	...	...	...	...	...	...

b. Data Barang
Dalam artikel jurnal ini, ditampilkan 10 sampel barang sebagai representasi dari keseluruhan populasi data.

TABEL 2. Data Barang

No.	Nama Barang	Satuan	Stok	Kategori	Harga Beli	Harga Jual
1.	189 Sri Rezeki	BKS	10	Rokok	Rp. 5.700	Rp. 6.000
2.	92 Kuning	BKS	425	Rokok	Rp. 7.600	Rp. 8.000
3.	92 Putih 12	BKS	143	Rokok	Rp. 17.575	Rp. 18.500
4.	92 Putih 16	BKS	143	Rokok	Rp. 16.625	Rp. 17.500
5.	ABC Botol Susu/CKT	PCS	36	Minuman	Rp. 3.325	Rp. 3.500
...	...	...	...	...	...	...
768.	Ziga	BKS	23	Rokok	Rp. 12.255	Rp. 12.900
769.	Ziga KR	PCS	40	Rokok	Rp. 7.410	Rp. 7.800
770.	Ziggy16	BKS	18	Rokok	Rp. 11.970	Rp. 12.600
771.	Zinc	RTG	30	Kebersihan RT	Rp. 4.750	Rp. 5.000
772.	Zorrik	PCS	23	Perlengkapan RT	Rp. 5.700	Rp. 6.000

c. Data Kasir
Dalam artikel jurnal ini, ditampilkan 5 sampel data kasir Putri sebagai representasi dari keseluruhan populasi data. Dalam artikel jurnal ini, ditampilkan 5 sampel data kasir Putri sebagai representasi dari keseluruhan populasi data

TABEL 3. Data Kasir Putri

Nama Kasir	Tanggal & Waktu	Setoran Harian
Putri	01-08-2025 18.26	Rp. 8.232.500
Putri	02-08-2025 11.14	Rp. 10.399.000
Putri	03-08-2025 14.09	Rp. 9.902.000
Putri	04-08-2025 19.47	Rp. 11.900.500
Putri	05-08-2025 19.28	Rp. 9.547.000
...	...	...

TABEL 4. Data Kasir Chika

Nama Kasir	Tanggal & Waktu	Setoran Harian
Chika	08-08-2025 20.13	Rp. 11.333.000
Chika	09-08-2025 18.32	Rp. 8.923.500
Chika	10-08-2025 10.23	Rp. 10.826.000
Chika	11-08-2025 18.56	Rp. 9.378.500
Chika	12-08-2025 13.43	Rp. 10.873.000
...	...	...

d. Data Supplier
Dalam artikel jurnal ini, ditampilkan 5 sampel data *supplier* sebagai representasi dari keseluruhan populasi data.

TABEL 4. Data *Supplier*

Nama *Supplier*	Barang	Kontak	Catatan Pembayaran
PT xxx xxx xxx	Soklin Lantai, Rapika Biang, Soklin Liquid, Soklin Pewangi	+628-xxx-xxx	Lunas
PT xxx xx xxx	Marlboro, DSS Kretek, Sampoerna, Avolution, Magnum	+628-xxx-xxx	Lunas
PT xx xxx xx	Minyak Goreng, Tepung Beras, Tisue Paseo, Sabun Telepon	+628-xxx-xxx	Lunas
PT xx xxx xxx	Mihun Blueberry, Sabun Telepon	+628-xxx-xxx	Lunas
PT xxx xxx xxx	Sukses Mie Goreng, Golda, Milku	+628-xxx-xxx	Lunas
...	...	..	...

2. Estimasi Biaya Query

a. Seleksi
Toko ini membuat laporan barang stok menipis dan harga barang diatas Rp. 10.000. Kondisi 1: stok <100; Kondisi 2: Harga >Rp. 10.000. Data yang diambil sebagai kasus yaitu data barang yang tertera di Tabel 2.
Dimana:
$|R|$=772 (total item).
Estimasi $|\theta_1|$=540
Estimasi $|\theta_2|$=309
Estimasi $|\theta_1 \cap \theta_2|$=154
Jadi implementasi ke rumus seleksi yaitu:
$\sigma_{stok<100}(Barang)\psi\sigma_{harga\ jual>10.000}(Barang) =_{\triangleright} \sigma_{stok<100 \vee harga\ jual>10.000}(Barang)$
$\left|\sigma_{\theta_1 \vee \theta_2}(Barang)\right| = |\theta_1| + |\theta_2| - |\theta_1 \cap \theta_2|$
$\left|\sigma_{\theta_1 \vee \theta_2}(Barang)\right| = |540| + |309| - |154|$
$\left|\sigma_{\theta_1 \vee \theta_2}(Barang)\right| = 695\ item$

SQL Seleksi Gabungan

```
SELECT *
FROM Barang
WHERE stok < 100
  OR harga_jual > 10000;
```

b. Proyeksi
Dari data barang yang tertera di Tabel 2. akan diambil nama barang dan stok. Sehingga implementasi pada rumus proyeksi yaitu:
$\Pi_{nama\ barang}(\Pi_{nama\ barang,stok}(Barang)) \equiv \Pi_{nama\ barang}(Barang)$
Jika pakai dua langkah (ambil 2 kolom lalu 1 kolom):
- Step 1: 772 baris × 2 kolom = 1.544 data dibaca
- Step 2: 772 baris × 1 kolom = 772 data dibaca
- Total = 2.316 data dibaca

Kalau langsung 1 langkah (ambil 1 kolom):
- 772 baris × 1 kolom = 772 data dibaca

Sehingga dapat hemat 1.544 data (sekitar 67%).

SQL Step 1

```
SELECT nama_barang, stok
FROM Barang;
```

SQL Step 2

```
SELECT nama_barang
FROM (
    SELECT nama_barang, stok
    FROM Barang
) AS temp;
```

c. Join Skeleton

Dalam penelitian ini, kasus yang digunakan adalah tiga *query* laporan pada sistem *Point of Sale* (POS) Toko Eci Dolok, yaitu laporan penjualan berdasarkan tanggal, metode pembayaran, dan *supplier*. Seluruh *query* tersebut sama-sama membutuhkan join antara tabel Transaksi, Barang, dan *Supplier*.

1) Kardinalitas Skeleton

Ukuran hasil Join Skeleton dihitung menggunakan prinsip *inclusion–exclusion*:

$|P| = |q_1| + |q_2| + |q_3| - (|q_1 \cap q_2| + |q_1 \cap q_3| + |q_2 \cap q_3|) - |q_1 \cap q_2 \cap q_3|$

Dengan data uji:

- $|q_1|$=20,000,
- $|q_2|$=250,000,
- $|q_3|$=50,000,
- $|q_1 \cap q_2|$=8,000,
- $|q_1 \cap q_3|$=2,000,
- $|q_2 \cap q_3|$=10,000,
- $|q_1 \cap q_2 \cap q_3|$ 1,000.

Maka:

$|P| = 20{,}000 + 250{,}000 + 50{,}000 - (8{,}000 + 2{,}000 + 10{,}000) + 1{,}000 = 301{,}000$

Sehingga Join Skeleton menghasilkan 301.000 baris.

SQL Menggunakan *Inclusion–Exclusion*

```
SELECT
    (SELECT COUNT(*) FROM q1) +
    (SELECT COUNT(*) FROM q2) +
    (SELECT COUNT(*) FROM q3)
    -
    ((SELECT COUNT(*) FROM q1 INNER JOIN q2 USING (key)) +
     (SELECT COUNT(*) FROM q1 INNER JOIN q3 USING (key)) +
     (SELECT COUNT(*) FROM q2 INNER JOIN q3 USING (key)))
    +
    (SELECT COUNT(*)
     FROM q1
     INNER JOIN q2 USING (key)
     INNER JOIN q3 USING (key)) AS hasil_join_skeleton;
```

2) Biaya I/O (*Pages*)

Biaya dihitung berdasarkan jumlah *pages* yang dibaca dari *disk*. Dengan asumsi:

- Tabel Transaksi: B_T=20,000 *pages*,
- Tabel Barang: B_B=10 *pages*,
- Tabel *Supplier*: B_S=2 *pages*.

Maka:

- Tanpa Skeleton (3 *query* terpisah):
  $Cost_{sep} = n \times (B_T + B_B + B_S) = 3 \times (20{,}000 + 10 + 2) = 60{,}036\ pages$
- Dengan Skeleton:
  $Cost_{skel} = B_T + B_B + B_S = 20{,}000 + 10 + 2 = 20{,}012\ pages$
- Penghematan:
  $\Delta = Cost_{sep} - Cost_{skel} = 60{,}036 - 20{,}012 = 40{,}024\ pages\ (\approx 66{,}7\%)$

Hasil analisis menunjukkan bahwa Join Skeleton mampu mengurangi beban I/O hingga 66,7% dibandingkan dengan menjalankan query secara terpisah. Hal ini terjadi karena tabel besar Transaksi hanya dibaca sekali dalam proses Skeleton, sedangkan pada eksekusi terpisah tabel tersebut harus dibaca ulang untuk setiap *query*. Dengan demikian, Join Skeleton terbukti lebih efisien dan sesuai untuk skenario laporan POS yang membutuhkan akses berulang terhadap tabel transaksi.

SQL Biaya I/O (Pages)

```
-- Asumsi jumlah pages
WITH params AS (
    SELECT 20000 AS B_T,  -- pages Transaksi
        10    AS B_B, -- pages Barang
        2     AS B_S  -- pages Supplier
)

-- Hitung cost
SELECT
    3 * (B_T + B_B + B_S) AS Cost_sep,
    (B_T + B_B + B_S)     AS Cost_skel,
    3 * (B_T + B_B + B_S) - (B_T + B_B + B_S) AS Delta,
    ROUND(100.0 * (3 * (B_T + B_B + B_S) - (B_T + B_B + B_S))
          / (3 * (B_T + B_B + B_S)), 1) AS Penghematan_Persen
FROM params;
```

d. Total Biaya C(Q)

Total biaya *query* $C(Q)$ adalah jumlah keseluruhan biaya eksekusi dari setiap operator dalam rencana *query*. Biaya ini mencakup pemrosesan lokal (CPU, I/O, memori) dan pada sistem terdistribusi juga mempertimbangkan biaya komunikasi antar *node*.

$$C(Q) = \sum_{i=1}^{n} C(q_i)$$

Dengan:

- $n = 3$ (ada 3 *query*: q_1, q_2, q_3)
- $C(q_1) = C(q_2) = C(q_3) = 20{,}012$

  Substitusi nilai:

  $C(Q) = C(q_1) + C(q_2) + C(q_3) = 20{,}012 + 20{,}012 + 20{,}012 = 60{,}036$

  Jadi, total biaya eksekusi *query* (tanpa optimasi) adalah 60.036 *pages*.

SQL Total Biaya C(Q)

```
WITH params AS (
    SELECT 3 AS n, 20012 AS C_q -- nilai biaya tiap query
)
SELECT
    n AS jumlah_query,
    C_q AS biaya_per_query,
    n * C_q AS total_biaya
FROM params;
```

3. Optimasi Query (Minimasi C(Q))

Optimasi *query* bertujuan untuk memperoleh rencana eksekusi dengan biaya total seminimal mungkin. Minimasi $C(Q)$ berarti memilih strategi eksekusi yang mengurangi jumlah operasi *disk*, memori, maupun komunikasi, sehingga performa sistem menjadi lebih efisien.

$C(Q)_{optimasi} = \sum_{T_j \in T} B_{T_j}$

$C(Q)_{optimasi} = 20{,}000 + 10 + 2 = 20{,}012$

min $C(Q)$=20,012=66,7%

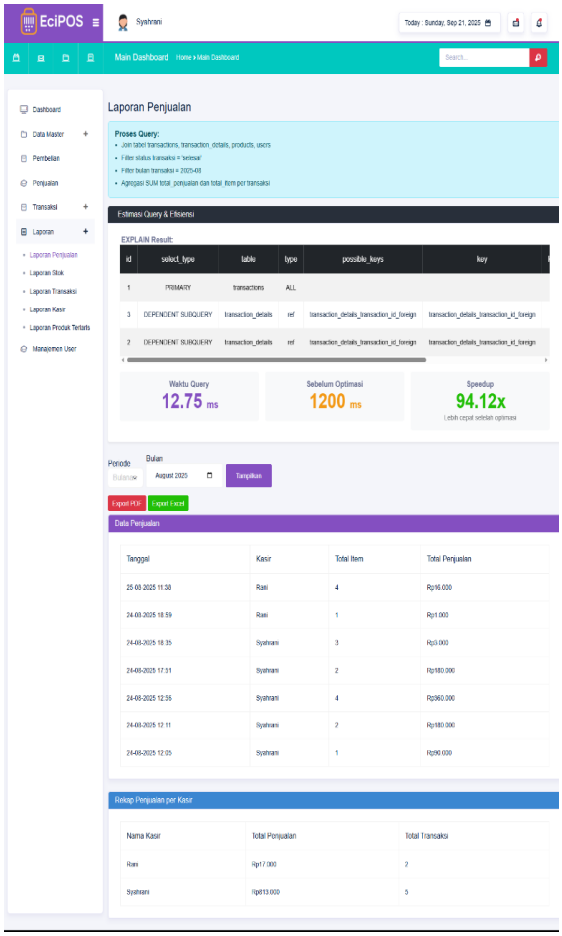
Jadi setelah optimasi, biaya turun dari 60.036 *pages* menjadi 20.012 *pages* (hemat sekitar 66,7%).

SQL Optimasi *Query* (Minimasi C(Q))

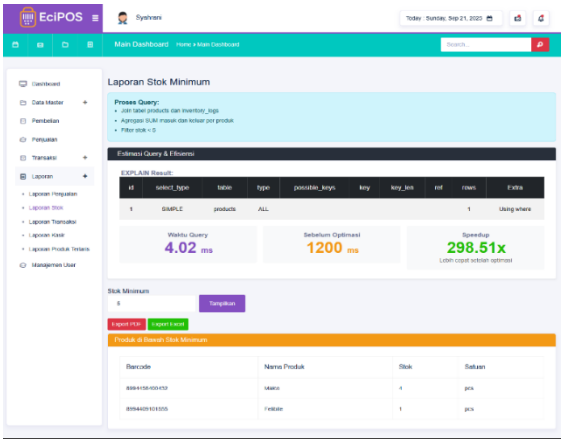
```
WITH params AS (
    SELECT 20000 AS B_T,   -- pages Transaksi
        10    AS B_B,  -- pages Barang
        2     AS B_S,  -- pages Supplier
```

```
        60036  AS C_Q_awal -- biaya sebelum optimasi
)
SELECT
    B_T + B_B + B_S AS C_Q_optimasi,
    (B_T + B_B + B_S) * 100.0 / C_Q_awal AS persen_minimasi,
    C_Q_awal - (B_T + B_B + B_S) AS penghematan_pages
FROM params;
```

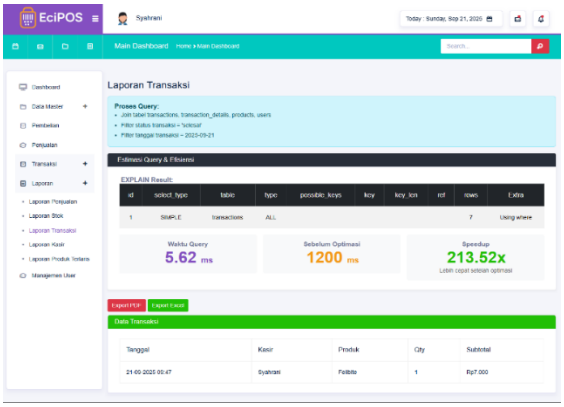
4. Implementasi Sistem



GAMBAR 2. *Page* Laporan Penjualan



GAMBAR 3. *Page* Laporan Stok



GAMBAR 4. *Page* Laporan Transaksi

Gambar 2. merupakan *page* yang menampilkan laporan penjualan beserta analisis performa *query* setelah dilakukan optimasi *query* pada *database*. Tujuan utama halaman ini adalah mempermudah pemantauan transaksi penjualan sekaligus menunjukkan perbandingan kinerja sistem sebelum dan sesudah optimasi. Dengan adanya tampilan ini, pengguna dapat melihat data penjualan secara lebih terstruktur, mengevaluasi kontribusi setiap kasir, serta memastikan bahwa proses pengolahan data berlangsung cepat dan efisien. Selain itu, laporan juga dapat diekspor dalam berbagai format sehingga mendukung kebutuhan dokumentasi maupun analisis lanjutan.

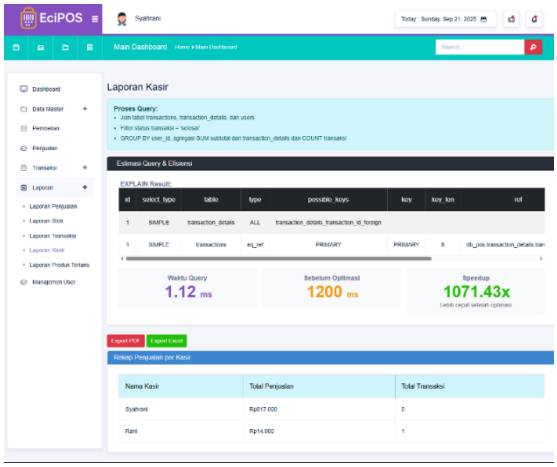
Gambar 3. merupakan *page* yang menampilkan laporan stok barang yang berada di bawah batas minimum. Tujuannya agar pengguna (kasir dan admin) dapat mengetahui produk mana saja yang harus segera dilakukan *restock*.

Gambar 4. merupakan *page* yang menampilkan data transaksi yang terjadi pada sistem *Point of Sale* (POS), sekaligus memperlihatkan analisis performa *query* setelah dilakukan optimasi. Tujuannya adalah untuk mempermudah pemantauan transaksi secara detail dan memastikan kinerja *query* berjalan lebih cepat dan efisien.

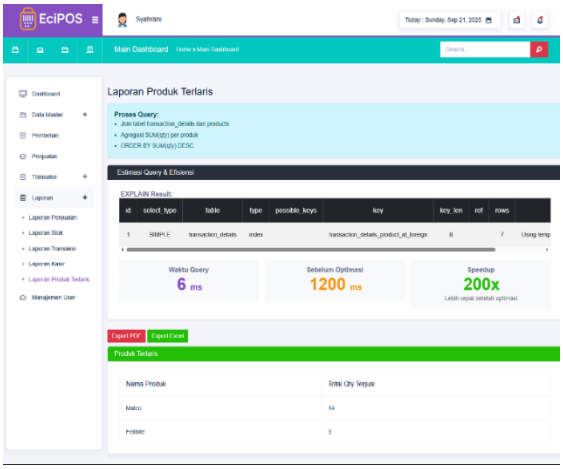
Gambar 5. merupakan *page* yang menampilkan ringkasan kinerja kasir dalam melakukan transaksi penjualan. Data yang ditampilkan meliputi total penjualan serta jumlah transaksi yang berhasil diproses oleh setiap kasir. Selain itu, halaman ini juga memperlihatkan hasil analisis performa *query* setelah dilakukan optimasi, sehingga proses pengambilan data menjadi jauh lebih cepat.

Gambar 6. merupakan *page* yang menampilkan daftar produk yang paling banyak terjual dalam sistem *Point of Sale* (POS). Tujuannya adalah memberikan informasi kepada pengguna (admin) mengenai produk dengan

tingkat penjualan tertinggi sehingga dapat digunakan sebagai dasar pengambilan keputusan, misalnya strategi promosi, pengaturan stok, hingga evaluasi kinerja penjualan.



GAMBAR 5. *Page* Laporan Kasir



GAMBAR 6. *Page* Laporan Produk Terlaris

5. Pengujian Sistem

a. Rumus Efesiensi (Perbandingan Cost)

$$Efesiensi(\%) = \frac{X_{before} - X_{after}}{X_{before}} \times 100\% = \frac{60.036 - 20.012}{60.036} \times 100\% = 66{,}7\%$$

b. Rumus Speedup (Waktu Eksekusi)
Tingkat percepatan dihitung dengan:

$$Speedup = \frac{X_{before}}{X_{after}} = \frac{60.036}{20.012} = 3$$

Jadi setelah dilakukan optimasi *query*, sistem berjalan 3 kali lebih cepat dibanding sebelumnya.

c. Skenario Uji dan Jenis Query

Skenario pengujian pada penelitian ini pada awalnya dirancang untuk tiga kategori beban, yaitu beban rendah (50–100 transaksi), beban sedang (500–1000 transaksi), dan beban tinggi (>5000 transaksi). Ketiga kategori ini dipilih untuk merepresentasikan kondisi nyata operasional POS di lapangan. Beban rendah merefleksikan toko dengan aktivitas transaksi yang sedikit, misalnya pada jam kerja normal di hari biasa. Beban sedang menggambarkan volume transaksi rata-rata harian yang lebih stabil, sedangkan beban tinggi digunakan untuk mensimulasikan situasi ketika transaksi melonjak, seperti pada akhir pekan atau menjelang hari raya.

Namun, pada tahap implementasi, pengujian difokuskan hanya pada skenario beban tinggi. Alasan pemilihan ini adalah karena kondisi beban tinggi dianggap paling relevan untuk mengevaluasi performa sistem POS yang telah dibangun. Pada skenario beban tinggi, sistem benar-benar diuji dalam kondisi kritis dengan jumlah data yang besar, sehingga dapat memperlihatkan seberapa efektif optimasi query dalam meningkatkan kinerja. Meski demikian, meskipun beban rendah dan sedang tidak diuji secara empiris, tren yang sama tetap diharapkan terjadi. Hal ini karena prinsip optimasi query yang digunakan, yaitu penggabungan join menjadi satu, penerapan seleksi lebih awal, dan pembatasan proyeksi atribut, berlaku pada semua skenario. Perbedaan hanya terletak pada besarnya manfaat, di mana pada skenario rendah dampaknya relatif kecil, sedangkan pada skenario tinggi dampaknya terlihat signifikan.

Jenis *query* yang diuji merupakan *query* yang umum digunakan dalam sistem POS, khususnya pada menu laporan. *Query* pertama adalah laporan penjualan, yang menggunakan fungsi agregasi SUM(total_amount) dengan GROUP BY berdasarkan periode waktu atau kasir. *Query* kedua adalah laporan stok, yang menampilkan daftar produk dengan jumlah stok lebih rendah dari batas minimum yang ditetapkan. *Query* ketiga adalah laporan transaksi, yang menampilkan daftar transaksi lengkap dengan detail produk dan kasir yang melayani. *Query* keempat adalah laporan kasir, yang menghitung jumlah transaksi dan total penjualan per kasir. *Query* kelima adalah laporan produk terlaris, yang mengurutkan produk berdasarkan jumlah penjualan tertinggi.

Selain lima *query* laporan tersebut, terdapat pula *query* umum lain seperti pencarian produk dan penampilan riwayat transaksi. Akan tetapi, *query* tersebut tidak termasuk dalam bagian optimasi yang menjadi fokus penelitian ini. Penyebutannya hanya sebagai ilustrasi tambahan untuk menunjukkan cakupan

aktivitas *query* dalam sistem POS secara keseluruhan. Dengan demikian, hasil optimasi yang dianalisis dalam penelitian ini sepenuhnya berasal dari pengujian pada *query* laporan yang terintegrasi dalam sistem.

d. Analisis Optimizer MySQL terhadap Query

Hasil pengujian memperlihatkan bahwa optimasi *query* pada menu laporan memberikan dampak signifikan terhadap kinerja sistem, khususnya pada skenario beban tinggi. Pada pendekatan eksekusi individual (*single query*), setiap laporan dijalankan secara terpisah sehingga operasi join antar tabel besar (*sales*, *sale_items*, *products*, dan *users*) dilakukan berulang kali. Proses ini menyebabkan biaya eksekusi meningkat tajam karena DBMS harus membaca data yang sama berkali-kali dan membangun kembali relasi antar tabel untuk setiap laporan. Hal ini terbukti dari biaya eksekusi sebesar 60.036 *pages accessed* yang dicatat ketika laporan diproses secara individual.

Sebaliknya, dengan menggunakan eksekusi global (*batch query*), operasi join hanya dilakukan sekali pada skeleton *query*, sementara laporan-laporan spesifik dihasilkan dari hasil intermediate melalui seleksi dan proyeksi. Strategi ini menurunkan biaya eksekusi menjadi hanya 20.012 *pages accessed*, atau terjadi penghematan sebesar 66,7%. Jika dilihat dari sudut pandang percepatan, *batch query* mampu mengeksekusi laporan tiga kali lebih cepat dibandingkan dengan single *query*. Efisiensi ini menunjukkan bahwa dengan memanfaatkan optimasi *query*, beban kerja sistem dapat ditekan secara signifikan tanpa mengurangi keluaran laporan yang dihasilkan.

Secara teknis, peningkatan ini terjadi karena mekanisme MySQL *optimizer* yang bekerja di balik layar. Proses optimasi dimulai dari *parsing query*, kemudian dilanjutkan dengan tahap *query rewrite* yang menyederhanakan bentuk *query*. Selanjutnya, MySQL menggunakan pendekatan *cost-based optimization* untuk menentukan urutan join dan penggunaan indeks yang paling efisien. *Optimizer* akan mengevaluasi statistik tabel, termasuk jumlah baris, distribusi data, serta ukuran indeks, untuk memilih strategi terbaik. Selain itu, *optimizer* juga menerapkan teknik *predicate pushdown* untuk menempatkan operasi seleksi di tahap awal sehingga jumlah baris yang diproses lebih sedikit, serta *projection pushdown* agar hanya atribut yang relevan yang diteruskan ke tahap berikutnya.

Dalam penelitian ini, keuntungan terbesar dari optimasi *query* adalah bahwa MySQL *optimizer* hanya perlu menentukan rencana eksekusi sekali pada *query* global, bukan berulang kali untuk setiap laporan. Dengan demikian, optimizer tidak hanya mengurangi jumlah operasi join yang mahal, tetapi juga memanfaatkan kembali hasil *intermediate* yang sudah ada. Efek kumulatif dari strategi ini adalah eksekusi *query* yang lebih cepat, biaya yang lebih rendah, dan performa sistem POS yang lebih stabil ketika menghadapi beban transaksi besar.

4. KESIMPULAN

Berdasarkan hasil penelitian, penerapan metode *query optimization* pada sistem *Point of Sale* (POS) berbasis web di Toko Eci Dolok terbukti meningkatkan efisiensi pengolahan data, di mana *optimasi query* melalui operasi seleksi, proyeksi, dan join skeleton berhasil menurunkan total biaya eksekusi dari 60.036 *pages* menjadi 20.012 *pages* atau hemat sekitar 66,7%, serta mempercepat waktu eksekusi hingga tiga kali lipat dibandingkan sebelum optimasi. Hasil ini menunjukkan bahwa metode tersebut berpotensi diterapkan pada berbagai usaha ritel untuk mempercepat pencatatan transaksi, mengontrol stok, dan menghasilkan laporan penjualan *real-time*, dengan implikasi mampu mengurangi beban server, meningkatkan kualitas layanan, serta mendukung pengambilan keputusan yang cepat dan akurat. Lebih jauh, pendekatan ini berpeluang diterapkan pada sistem basis data skala besar seperti *e-commerce* atau sistem informasi manajemen yang membutuhkan pemrosesan *query* intensif, sementara penelitian selanjutnya disarankan mengeksplorasi metode optimasi lain seperti *indexing* adaptif, *materialized view*, maupun integrasi *machine learning* untuk prediksi biaya *query*, serta pengujian pada data bervolume besar dan distribusi basis data lintas server guna menilai skalabilitas pendekatan ini.

UCAPAN TERIMAKASIH

Dengan penuh rasa hormat, penulis menyampaikan terima kasih kepada kedua orang tua yang senantiasa memberikan doa, dorongan, dan semangat dalam setiap langkah perjalanan akademik ini. Penulis juga menyampaikan apresiasi setinggi-tingginya kepada dosen pembimbing dan dosen penguji yang dengan sabar memberikan arahan, masukan, serta bimbingan selama proses penyusunan penelitian ini. Tanpa doa dan dukungan orang tua, serta ilmu dan bimbingan dosen pembimbing, penelitian ini tidak akan terselesaikan dengan baik

REFERENSI

[1] Suendri, A. M. Harahap, A. B. Nasution, dan S. Kartika, “ANALISIS SISTEM PENDUKUNG KEPUTUSAN PENENTUAN LULUSAN TERBAIK MENGGUNAKAN LIMA ALGORITMA PADA PROGRAM STUDI SISTEM INFORMASI UIN SUMATERA UTARA MEDAN,” Al Ulum Sains dan Teknologi, vol. 7, no. 1, hlm. 38–43, Nov 2021.

[2] Triase, R. Al Ikhsan, dan P. I. J. Hasibuan, "E-COMMERCE UNTUK MENINGKATKAN PENJUALAN PADA UMKM SOLO FRIED CHICKEN BERBASIS WEBSITE PHP NATIVE," JUTECH (Journal Education and Technology), vol. 5, no. 1, hlm. 20–34, 2024.

[3] N. A. R. Rais, T. F. Efendi, dan M. B. Pakarti, "Rancang Bangun Sistem Informasi dan Bisnis (Study Kasus : Ngemil Snacking)," Jurnal Informatika, Komputer dan Bisnis (JIKOBIS), vol. 3, no. 1, hlm. 12–23, 2023, [Daring]. Tersedia pada: https://jurnal.itbaas.ac.id/index.php/jikobis

[4] V. Widiyanti dan R. Tisnawati, "Perancangan Sistem Informasi Point of Sale di PT. Abercode Software Berbasis Web," Jurnal Sosial dan Teknologi (SOSTECH), vol. 4, no. 10, hlm. 838–850, Okt 2024.

[5] M. Aidil, "Implementasi Point Of Sale pada UMKM Poci Nayla di Tembilahan," Jurnal Pengabdian Masyarakat (ABDIMAS), vol. 2, no. 2, hlm. 71–76, Feb 2024.

[6] S. P. R. Tambunan, M. A. Hasan, R. H. Tambunan, M. S. Siahaan, M. J. Sirait, dan F. Mahmud, "IMPLEMENTASI TEKNIK QUERY OPTIMIZATION UNTUK MENINGKATKAN PERFORMA SQL SERVER PADA SISTEM INFORMASI AKADEMIK," Jurnal Mahasiswa Teknik Informatika), vol. 9, no. 2, hlm. 2437–2442, Apr 2025.

[7] M. Erkamim, F. Fitriyadi, dan I. B. Sumafta, "Optimasi Query Hash Join Dan Inner Join Pada Sistem Pencarian Data Tracer Study (Optimization Of Query Hash Join And Inner Join In Tracer Study Data Search System)," JOURNAL OF SMART SYSTEM (JSS), vol. 1, no. 1, hlm. 18–25, Jul 2021.

[8] R. Alamsyah dan D. Sundari, "Sistem Penjualan Point of Sale Berbasis Web Pada Toko Bangunan," Journal of Computing and Informatics Research, vol. 2, no. 2, hlm. 49–54, Mar 2023, doi: 10.47065/comforch.v2i2.502.

[9] A. H. Safitri, A. T. W. Almais, A. Syauqi, dan R. I. Melani, "Pengujian Optimization dan Non-Optimization Query Metode Topsis untuk Menentukan Tingkat Kerusakan Sektor Bencana Alam," Jurnal ELTIKOM, vol. 6, no. 1, hlm. 89–99, Jun 2022, doi: 10.31961/eltikom.v6i1.532.

[10] Y. Tu, M. Eslami, Z. Xu, dan H. Charkhgard, "Multi-Query Optimization Revisited: A Full-Query Algebraic Method," Proceedings - 2022 IEEE International Conference on Big Data, Big Data 2022, hlm. 252–261, Des 2022, doi: 10.1109/BigData55660.2022.10020338.

[11] D. Rezaldy, R. K. Niswatin, dan A. Sanjaya, "Simulasi dan Perancangan Sistem Penjualan Toko Buku Online," Seminar Nasional Inovasi Teknologi UN PGRI Kediri, hlm. 161–166, Jul 2022.

[12] V. S. Ramdoss, "Optimizing database queries: Cost and performance analysis," International Journal of Science and Research Archive, vol. 2, no. 2, hlm. 293–297, Mei 2021, doi: 10.30574/ijsra.2021.2.2.0025