BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Tabel 2. 1 Tabel Penelitian Terdahulu

no	Authors	State of the art	Objective	Main Critical Factors	Methodology	Deficiencies	Research Gap	Conclusion	Future Research
1	[4]	Penelitian ini menyoroti bahwa benchmark HTAP yang ada belum mampu merepresentasikan kondisi nyata karena tidak mempertimbangkan *real time* query, skema yang konsisten secara semantic antara OLTP dan OLAP, dan beban kerja	Merancang dan mengevaluasi OLxPBench sebagai *benchmark* HTAP yang mampu mengukur dampak interaksi beban kerja OLTP dan OLAP secara realistis melalui *hybrid transaction,* skema semantic konsisten, dan	*Real-time query* yang disisipkan dalam transaksi OLTP, penggunaan skema basis data yang konsisten secara semantic antara OLTP dan OLAP, serta perbedaan karakteristik	Metode eksperimen berbasis *benchmark* dengan menjalankan beban kerja OLTP,OLAP, dan *hybrid* secara bersamaan pada beberapa sistem HTAP, serta mengukur metrik kinerja seperti *throughput,* rata	Perancangan dan evaluasi *benchmark* lintas sistem HTAP dan tidak secara khusus mengkaji sistem ERP, dan basis data tertentu. Dan tidak ada pemisahan arsitektur basis data.	Tidak secara spesifik mengkaji pengaruh OLAP terhadap OLTP. Dan juga penelitian ini tidak berfokus pada ERP.	*Benchmark* HTAP yang mempertimbangkan *real-time query,* skema semantic konsisten, dan beban kerja berbasis domain lebih mampu mengungkap interferensi nyata antara OLTP dan OLAP dibandingkan	Pengembangan *benchmark* lebih lanjut untuk berbagai domain aplikasi, eksplorasi Teknik peningkatan isolasi kinerja OLTP dan OLAP, serta evaluasi sistem HTAP pada skala dan arsitektur

no	Authors	State of the art	Objective	Main Critical Factors	Methodology	Deficiencies	Research Gap	Conclusion	Future Research
		spesifik domain, sehingga hasil evaluasi kinerja OLTP dan OLAP dapat menyesatkan.	*workload* berbasis domain.	beban kerja umum dan domain-spesifik terhadap kinerja sistem.	rata *latency,* dan *tail latency.*			*benchmark* konvensional.	yang lebih beragam.
2	[3]	Kebutuhan pemrosesan OLTP dan OLAP secara bersamaan dalam satu sistem basis data tanpa menurunkan kinerja transaksi, karena solusi ETL tradisional menimbulkan latensi data dan kompleksitas sinkronisasi yang tinggi.	Merancang dan mengevaluasi sistem basis data HTAP berbasis *cloud-native* yang mampu memberikan kinerja OLAP tinggi dengan dampak minimal terhadap kinerja OLTP serta menjaga kesegaran data secara near *real-time.*	Isolasi sumber daya, replikasi berbasis REDO log, dan *in-memory column index,* sistem HTAP dapat menjalankan OLTP dan OLAP secara bersamaan dengan degradasi kinerja OLTP kurang dari 5%.	Eksperimen kinerja menggunakan benchmark standar seperti TPC-H, TPC-C, dan CH-Benchmark pada sistem PolarDB-IMCI, disertai pengujian *workload* campuran untuk mengukur *throughput, latency, visibility delay,* dan elastisitas sumber daya.	Tidak mengkaji perilaku interferensi OLTP dan OLAP pada sistem ERP konvensional yang menggunakan satu DBMS relasional umum seperti PostgreSQL tanpa mekanisme atau optimasi khusus.	Belum ada kajian yang mengevaluasi secara eksperimental pengaruh beban kerja OLAP terhadap kinerja OLTP pada sistem ERP berbasis PostgreSQL dalam satu basis data tanpa penerapan solusi HTAP.	PolarDB-IMCI mampu menangani beban kerja *hybrid* OLTP dan OLAP secara efisien dengan kinerja OLAP tinggi, gangguan minimal pada OLTP, serta kesegaran data yang rendah latensi dalam lingkungan *cloud-native.*	Pengembangan lebih lanjut pada model optimasi *cost-based routing hybrid,* peningkatan adaptivitas eksekusi query, serta eksplorasi desain HTAP yang lebih fleksibel untuk berbagai pola beban kerja dan skenario aplikasi.
3	[5]	Pemanfaatan GPU pada DBMS umumnya terbatas pada	Merancang DBMS berbasis memori utama yang diakselerasi	Pemetaan OLTP ke eksekusi paralel berbasis	Pendekatan *co-execution* CPU-GPU dengan pemrosesan OLTP berbasis	Hanya berfokus pada OLTP murni, sehingga tidak membahas	Belum membahas interaksi OLAP-OLTP	Akselerasi GPU efektif untuk OLTP dan mampu melampaui	Pengembangan mekanisme otomatis dan dukungan

no	Authors	State of the art	Objective	Main Critical Factors	Methodology	Deficiencies	Research Gap	Conclusion	Future Research
		OLAP, sedangkan OLTP masih bergantung pada CPU dan sulit mencapai *throughput* tinggi secara efisien.	GPU untuk meningkatkan *throughput* OLTP melalui eksekusi bersama CPU-GPU	batch, desain kontrol konkurensi yang efisien, serta dukungan eksekusi OLTP skala besar.	batch di GPU dan kontrol konkurensi hibrida.	interaksi beban hibrida (OLTP dan OLAP).	pada DBMS relasional ERP.	kinerja sistem OLTP berbasis CPU.	HTAP (OLAP dan OLTP)
4	[7]	Optimasi performa data warehouse dibandingkan database OLTP pada PostgreSQL cloud melalui star schema, indexing, pre-aggregated table, dan EXPLAIN plan.	Membandingkan performa query antara database OLTP dan data warehouse untuk meningkatkan kecepatan retrieval data pelanggan.	Desain data warehouse, pre-aggregation, indexing, EXPLAIN plan, query execution time, cache, ukuran storage.	Studi kasus eksperimental pada PostgreSQL; membandingkan OLTP dan DW menggunakan tujuh query berat, optimasi query, indexing, pre-aggregated table, dan analisis EXPLAIN plan.	Fokus pada perbandingan OLTP vs DW, bukan pengaruh OLAP terhadap OLTP dalam satu database; tidak mengukur mixed workload, response time OLTP, throughput OLTP, atau interferensi workload.	Belum ada analisis pengaruh beban kerja OLAP terhadap kinerja OLTP ketika transaksi dan analitik dijalankan bersamaan pada sistem ERP berbasis PostgreSQL.	DW dengan pre-aggregated table dan index meningkatkan performa query 1–3 orde magnitudo, bahkan hingga 2.493 kali lebih cepat dari OLTP, serta menggunakan storage 24% lebih kecil.	Mengembangkan optimasi lanjutan melalui bitmap index, partitioning, parallel query execution, materialized view, dan integrasi dengan machine learning/NLP.
5	[2]	Adanya permasalahan interferensi beban kerja pada sistem *hybrid transactional/*	Mengkarakterisasi dan menganalisis dampak interferensi antara beban	OLTP dan OLAP saling mengganggu ketika dijalankan bersamaan,	Eksperimental dengan *micro-benchmrk* dan *CH-benchmark* untuk mengevaluasi	Implementasi sistem ERP nyata atau DBMS relasional umum seperti	Dampak beban kerja OLAP terhadap OLTP pada sistem ERP berbasis PostgreSQL	Interferensi antara OLTP dan OLAP pada sistem HTAP dapat menyebabkan	Penelitian lanjutan pengembangan mekanisme manajemen sumber daya,

no	Authors	State of the art	Objective	Main Critical Factors	Methodology	Deficiencies	Research Gap	Conclusion	Future Research
		analytical processing (HTAP), di mana OLTP dan OLAP dijalankan secara bersamaan pada data dan sumber daya perangkat keras yang sama, yang menyebabkan penurunan kinerja terutama pada beban kerja OLTP.	kerja OLTP dan OLAP pada sistem HTAP, khususnya untuk memahami bagaimana berbagi sumber daya perangkat keras dan memengaruhi *throughput* dan waktu eksekusi masing masing beban kerja	*throughput* OLTP dapat menurun 42% akibat kontensi cache dan *bandwidth* memori, sementara kinerja OLAP dapat meningkat secara eksponensial ketika propagasi data tertinggi dari laju transaksi OLTP.	kinerja OLTP dan OLAP secara bersamaan pada berbagai arsitektur HTAP, serta melakukan analisis mendalam terhadap penggunaan *cache, bandwidth* memori, dan mekanisme propagasi data.	PostgreSQL. Dan juga tidak merepresentasikan skenario operasional ERP yang menggunakan satu basis data tanpa optimasi atau mekanisme HTAP khusus.	dalam satu basis data tanpa penerapan arsitektur HTAP atau optimasi sumber daya khusus.	penurunan signifikan pada kinerja OLTP dan peningkatan waktu eksekusi OLAP, sehingga diperlukan mekanisme isolasi sumber daya dan manajemen propagasi data untuk meminimalkan dampak tersebut.	isolasi workload, dan penjadwalan adaptif untuk mengurangi interferensi OLTP dan OLAP, serta eksplorasi desain sistem HTAP yang lebih efisien dalam menangani propagasi data transaksi.
6	[8]	Tantangan sistem basis data modern dalam lingkungan *cloud-native,* khususnya terkait skalabilitas, elastisitas dan isolasi kinerja ketika menangani	Menyajikan tinjauan komprehensif mengenai arsitektur, komponen utama, serta tantangan dan arah pengembangan database *cloud*	faktor utama yang dikaji meliputi arsitektur *cloud-native,* pemisahan *compute* dan *storage, resource sharing, scalability, fault*	Metode survei sistematis dengan menganalisis dan mengklasifikasikan berbagai penelitian serta sistem database *cloud-native* berdasarkan arsitektur dan	Tidak melakukan pengujian eksperimental, tidak mengukur metrik kinerja seperti *throughput* atau *latency,* serta tidak secara spesifik membahas	walaupun artikel ini membahas beban kerja campuran pada database *cloud-native.* Tapi belum membahas secara spesifik mengenai dampak beban	Database *cloud-native* menawarkan skalabilitas dan fleksibilitas yang lebih baik untuk beban kerja modern, namun masih menghadapi tantangan dalam isolasi kinerja	Peningkatan isolasi kinerja, optimasi *resource management,* serta evaluasi database *cloud-native* terhadap beban kerja campuran

no	Authors	State of the art	Objective	Main Critical Factors	Methodology	Deficiencies	Research Gap	Conclusion	Future Research
		beban kerja yang beragam dalam satu sistem basis data.	*native* pada sistem modern.	*tolerance,* serta dukungan terhadap beban kerja transaksi dan analitik.	karakteristik teknisnya.	pengaruh beban kerja OLAP terhadap OLTP.	kerja OLAP terhadap OLTP.	dan pengelolaan sumber daya pada beban kerja campuran OLTP dan OLAP.	yang semakin kompleks.
7	[9]	Arsitektur tradisional CRM yang memisahkan OLTP dan OLAP menimbulkan latensi akibat replikasi data, sehingga menghambat analisis *real-time* dan pengambilan Keputusan yang cepat.	Mengeksplorasi penerapan arsitektur HTAP pada sistem CRM untuk mendukung analitik *Real-time* sekaligus mempertahankan kinerja transaksi.	Integrasi OLTP dan OLAP, *real-time data synchronization, in-memory computing, distributed storage,* serta workload *interference* pada sistem HTAP.	Studi literatur dan analisis konseptual arsitektur HTAP, dilengkapi dengan pembahasan hasil benchmark dari penelitian sebelumnya.	Tidak melakukan pengujian eksperimental langsung, dan tidak menganalisis pengaruh beban kerja OLAP terhadap OLTP dalam satu basis data secara terkontrol.	Belum terdapat analisis empiris secara khusus untuk mengukur dampak beban kerja OLAP terhadap OLTP dalam satu lingkungan basis data.	HTAP mampu meningkatkan analitik *real-time* dan efisiensi sistem CRM, namun menghadapi tantangan *interference* beban kerja dan kompleksitas implementasi.	Optimasi isolasi kinerja OLTP dan OLAP, pengelolaan sumber daya HTAP, serta penerapan HTAP pada arsitektur *cloud-native* dan *hybrid cloud.*
8	[10]	*Mammoth transaction* pada *graph database* yang menyebabkan *blocking,* peningkatan *tail*	Merancang protokol *concurrency* deterministic yang mampu menangani *mammoth*	Pendekatan tradisional seperti 2PL, menyebabkan *blocking* total, sementara	Pengembangan TuskFlow dengan memperluas protokol deterministic Aria	Graph database (Neo4j-*based)* dan skenario *mammoth transaction* pada skala industri	Belum ada kajian mendalam mengenai mekanisme *concurrency control* pada	TuskFlow mampu meningkatkan *concurrency* dan menurunkan *tail latency* hingga 45 kali	Protokol dapat diperluas ke sistem relational dengan memetakan graph

no	Authors	State of the art	Objective	Main Critical Factors	Methodology	Deficiencies	Research Gap	Conclusion	Future Research
		latency, dan penurunan *throughput* pada protokol konvensional seperti 2PL dan MVCC	*transaction* tanpa mengorbankan *isolation,* serta tetap menjaga *throughput* dan *latency* transaksi pendek.	MVCC tetap mengalami konflik dan *abort* tinggi. Sistem deterministic sebelumnya tidak dirancang khusus untuk *mammoth transaction* sehingga menimbulkan *latency.*	menggunakan mekanisme *epoch-based execution,* pemecahan *mammoth* menjadi task kecil, *paralel execution, graph tagging,* serta *query aware partitioning* untuk mengurangi konflik.		relational DBMS. Pendekatan deterministik seperti TuskFlow belum diimplementasikan ke relational database.	dibandingkan 2PL atau MVCC, dengan tetap mempertahankan *conflict serializability* melalui protokol deterministic dan optimasi berbasis struktur graph	traversal menjadi join tabel, serta berpotensi dikembangkan ke arsitektur terdistribusi untuk meningkatkan skalabilitas.
9	[11]	Scheduling pada *mixed workload* di mana *query* analitik jangka Panjang (OLAP) dapat mendominasi CPU dan menyebabkan transaksi prioritas tinggi mengalami latensi tinggi dalam sistem DBMS *non-preemptive*	Mengembangkan dan mendemonstrasikan mekanisme *preemptive scheduling* melalui sistem PreemptDB untuk mengurangi *latency* transaksi prioritas tinggi tanpa mengorbankan *throughput* pada beban kerja	*Scheduling policy, end-to-end latency* transaksi prioritas tinggi, *throughput* sistem secara keseluruhan, *overhead context switching* dan *deadlock handling.*	Eksperimental dengan membangun PreemptDB yang mendukung *preemptive scheduling* berbasis *user-level interrupt.* Evaluasi dilakukan menggunakan *mixed workload* TPC-C (short OLTP) dan	Tidak melakukan pengukuran empiris dampak alami beban kerja OLAP terhadap kinerja OLTP dalam satu basis data PostgreSQL standar atau dalam konteks sistem ERP.	Belum terdapat kajian eksperimental yang secara khusus mengukur dampak langsung beban kerja OLAP terhadap kinerja OLTP dalam sistem ERP berbasis PostgreSQL tanpa penerapan mekanisme	*Preemptive scheduling* melalui PreemptDB mampu menurunkan *end-to-end* latency transaksi prioritas tinggi secara signifikan dibandingkan kebijakan *wait* dan *cooperative,* tanpa mengorbankan *throughput* dan	Pengembangan kebijakan *scheduling* yang lebih adaptif, evaluasi *preemption* pada berbagai arsitektur DBMS, integrasi *preemptive scheduling* dalam sistem HTAP yang lebih luas.

no	Authors	State of the art	Objective	Main Critical Factors	Methodology	Deficiencies	Research Gap	Conclusion	Future Research
			campuran OLTP dan OLAP		TPC-H query 2 (long OLAP) untuk membandingkan kebijakan *wait, cooperative, dan preemptive scheduling.*		*preemption* atau optimasi arsitektural khusus.	skalabilitas sistem pada beban kerja campuran.	
10	[1]	Evaluasi performa Oracle, PostgreSQL, dan MySQL pada skenario OLTP menggunakan JMeter dan benchmark TPC-C dengan metrik response time dan throughput.	Membandingkan performa tiga DBMS berdasarkan response time dan throughput untuk mendukung pemilihan DBMS yang sesuai	Jenis DBMS, jumlah record, skenario query OLTP, response time, throughput, JMeter, dan TPC-C benchmark.	Metode kuantitatif eksperimen dengan pengujian Oracle, PostgreSQL, dan MySQL menggunakan variasi 50 sampai 100.000 record. Pengujian dilakukan lima kali dan dianalisis berdasarkan response time dan throughput.	Belum membahas mixed workload OLTP dan OLAP, belum menguji pengaruh query OLAP terhadap OLTP, dan belum spesifik pada sistem ERP berbasis PostgreSQL dalam satu basis data.	Diperlukan penelitian yang menguji dampak beban kerja OLAP terhadap kinerja OLTP pada PostgreSQL dalam konteks sistem ERP tanpa pemisahan arsitektur.	MySQL menunjukkan response time dan throughput terbaik pada konfigurasi pengujian tertentu, sedangkan PostgreSQL lebih lambat pada volume data besar.	Perlu pengujian dengan dataset lebih besar, query lebih kompleks, tools lebih beragam, benchmark lebih ketat, serta evaluasi PostgreSQL pada konteks operasional spesifik.

Berdasarkan Tabel 1.1 , kajian yang relevan dapat dikelompokkan ke dalam beberapa tema utama. Pertama, penelitian mengenai HTAP dan mixed workload menunjukkan bahwa eksekusi OLTP dan OLAP secara bersamaan dapat menimbulkan interferensi kinerja, terutama karena adanya persaingan sumber daya, kebutuhan isolasi workload, dan perbedaan karakteristik transaksi serta analitik [2], [3], [4], [8], [9], [11]. Kedua, penelitian yang berfokus pada optimasi OLTP dan performa DBMS menunjukkan bahwa response time dan throughput dipengaruhi oleh jenis DBMS, mekanisme eksekusi transaksi, concurrency control, serta pendekatan akselerasi sistem, tetapi sebagian besar masih berfokus pada OLTP murni dan belum membahas dampak OLAP terhadap OLTP dalam satu sistem basis data [1], [5], [10]. Ketiga, penelitian mengenai data warehouse dan OLAP menegaskan bahwa beban analitik lebih efektif dijalankan melalui desain data warehouse, pre-aggregated table, indexing, dan optimasi query, namun pendekatan tersebut lebih banyak membandingkan performa OLTP dan data warehouse secara terpisah, bukan menguji interferensi ketika OLAP dan OLTP berjalan bersamaan [7].

2.1.1 Sintesis Penelitian Terdahulu

Berdasarkan penelitian terdahulu yang telah dikaji, kajian mengenai kinerja basis data dapat dikelompokkan ke dalam beberapa tema utama. Kelompok pertama berfokus pada hybrid transactional and analytical processing atau HTAP, mixed workload, dan benchmark hybrid, yang menunjukkan bahwa eksekusi OLTP dan OLAP secara bersamaan dapat menimbulkan interferensi kinerja, terutama dalam bentuk penurunan throughput, peningkatan latency, serta persaingan penggunaan sumber daya sistem [2], [3], [4], [8], [9], [11]. Kelompok kedua berfokus pada optimasi performa OLTP dan DBMS, seperti akselerasi perangkat keras, mekanisme concurrency control, serta evaluasi response time dan throughput pada beberapa DBMS [1], [5], [10]. Sementara itu, kelompok ketiga membahas optimasi data warehouse dan OLAP melalui desain star schema, indexing, pre-aggregated table, dan analisis EXPLAIN plan untuk meningkatkan kinerja query analitik [7].

Meskipun penelitian-penelitian tersebut telah memberikan dasar penting mengenai karakteristik OLTP, OLAP, HTAP, dan optimasi performa basis data, sebagian besar belum sepenuhnya menjawab konteks yang dikaji dalam penelitian ini. Penelitian terdahulu umumnya masih berfokus pada rancangan arsitektur HTAP, benchmark lintas sistem, optimasi data warehouse, akselerasi perangkat keras, atau evaluasi performa DBMS secara umum [1], [3], [4], [5], [8]. Selain itu, beberapa penelitian memang membahas mixed workload, tetapi belum secara spesifik mengukur pengaruh beban kerja OLAP ringan dan OLAP berat terhadap kinerja OLTP pada sistem ERP berbasis PostgreSQL dalam satu basis data yang sama, tanpa pemisahan arsitektur seperti data warehouse atau mekanisme HTAP khusus.

Berdasarkan celah tersebut, penelitian ini diposisikan sebagai pengujian eksperimental yang lebih spesifik pada sistem ERP berbasis PostgreSQL. Penelitian ini tidak berfokus pada pengembangan

arsitektur baru, melainkan pada pengukuran empiris dampak beban kerja OLAP terhadap kinerja OLTP dalam lingkungan pengujian yang terkontrol. Pengaruh tersebut dianalisis melalui skenario baseline OLTP-only, mixed workload OLAP ringan, dan mixed workload OLAP berat, dengan metrik response time, throughput, persentase perubahan kinerja, standar deviasi, serta statistik level database melalui pg_stat_statements. Dengan demikian, penelitian ini mengisi gap pada kajian mengenai dampak mixed workload terhadap kinerja transaksi OLTP pada konteks ERP berbasis PostgreSQL.

2.2 Landasan Teori

2.2.1 Sistem Enterprise Resource Planning (ERP)

ERP merupakan sistem perangkat lunak yang mengintegrasikan pengelolaan data dan informasi dari berbagai fungsi perusahaan, seperti keuangan, akuntansi, produksi, penjualan, pembelian, sumber daya manusia, dan fungsi bisnis lainnya. Sistem ERP terdiri dari beberapa modul yang saling terhubung dalam satu pusat data terintegrasi, seperti financial accounting, logistics execution, sales distribution, materials management, production planning, plant maintenance, quality management, dan project system [12]. Sistem ERP berperan sebagai sarana untuk mengintegrasikan dan menyinkronkan data dari berbagai proses bisnis, sehingga menghasilkan satu sumber informasi yang terpadu, konsisten, dan andal.

ERP memiliki kemampuan untuk mengintegrasikan berbagai fungsi bisnis ke dalam satu sistem terpadu, termasuk sumber daya manusia, keuangan, manufaktur, rantai pasok, operasional, pelaporan, serta aktivitas perdagangan. Sebagai sistem perangkat lunak yang canggih, ERP mencakup beragam aplikasi pendukung seperti manajemen kualitas, manajemen sumber daya manusia, *supply chain management*, keuangan dan akuntansi, manajemen proyek, penjualan, dan lain lain.

Konsep utama dalam ERP adalah sentralisasi informasi melalui penggunaan satu basis data terpusat yang menjadi fondasi seluruh

proses bisnis. Dengan basis data Tunggal ini, ERP mampu menyediakan data yang akurat dan *real time* kepada seluruh aea fungsional, sehingga meminimalkan redundansi data dan inkonsistensi informasi. Secara spesifik, ERP dikembangkan untuk meningkatkan efisiensi organisasi dengan menyediakan informasi yang tepat waktu dan akurat di sepanjang rantai pasokan dan proses bisnis Perusahaan [4].

2.2.2 Arsitektur PostgreSQL

PostgreSQL merupakan sistem manajemen basis data relasional berbasis objek (*object relational database management system)* yang dikenal memiliki tingkat reliabilitas, konsistensi, dan performa tinggi [13]. Sebagai sistem basis data relasional, PostgreSQL mengorganisasi data dalam bentuk tabel yang saling berhubungan melalui relasi logis, di mana struktur metadata memungkinkan terjadinya independensi antara level logis dan fisik penyimpanan data. Dalam arsitektur relasional, unit kerja utama adalah transaksi, yaitu sekumpulan operasi yang diperlukan sebagai satu kesatuan yang konsisten dan andal. PostgreSQL menjamin integritas transaksi melalui penerapan prinsip ACID (*Atomicity, Consistency, Isolation, Durability)* [14], di mana *atomicity* memastikan operasi dilakukan secara keseluruhan atau tidak sama sekali, *consistency* menjaga validitas data sebelum dan sesudah transaksi, *isolation* mempertahankan indepedensi antar transaksi, serta *durability* menjamin bahwa perubahan yang telah dikomit akan tetap tersimpan meskipun terjadi kegagalan sistem.

Akses terhdap data pada PostgreSQL dilakukan melalui *structured query language* (SQL), yang memungkinkan manipulasi, pengambilan, serta pengelolaan data secara terstruktur [15]. Dalam implementasinya, PostgreSQL mendukung definisi *constraint,* pengelolaan indeks, serta aturan bisnis untuk menghindari redundansi dan menjaga integritas data [16]. Karakteristik ini menjadikan PostgreSQL cocok digunakan pada sistem yang membutuhkan konsistensi transaksi yang kuat dan

integritas data yang tinggi, seperti sistem operasional dan aplikasi berbasis transaksi.

2.2.3 Karakteristik Beban Kerja OLTP dan OLAP

Beban kerja pada sistem basis data dapat memiliki karakteristik yang berbeda sesuai dengan tujuan penggunaannya. Dalam konteks sistem ERP, dua jenis beban kerja yang umum ditemukan adalah Online Transactional Processing (OLTP) dan Online Analytical Processing (OLAP). Berikut penjelasan detailnya.

2.2.3.1 *Online Transactional Processing* (OLTP)

Online Transactional Processing (OLTP) merupakan sistem pemrosesan basis data yang dirancang untuk menangani transaksi operasional secara *real-time* dengan jumlah pengguna yang tinggi dan frekuensi transaksi yang besar, dan transaksinya dilakukan secara paralel (*concurrent)* [17]. OLTP umumnya digunakan dalam sistem operasional harian seperti sistem e-commerce dan sistem transaksi yang sensitif terhadap latensi [11]. Karakteristik utama OLTP adalah banyaknya transaksi berskala kecil (*short transaction)* yang melibatkan operasi dasar seperti *insert, update, delete,* dan *select* terhadap sejumlah kecil data dalam satu waktu [18].

Sistem OLTP menekankan pada konsistensi dan integritas data, sehingga umumnya diimplementasikan pada sistem manajemen basis data relasional yang mendukung properti ACID (*Atomicity, Consistency, Isolation, Durability).* Dari sisi performa, OLTP berorientasi pada *throughput* tinggi dan *latency* rendah, sehingga desain basis data biasanya menggunakan struktur tabel ternormalisasi untuk menghindari redundansi data dan mempercepat proses pembaruan [19]. query pada OLTP cenderung sederhana dan terindeks dengan baik, karena fokus utamanya adalah efisiensi transaksi, bukan analisis data berskala besar. dalam konteks sistem yang menjalankan beban kerja campuran, seperti ketika OLTP berjalan bersamaan dengan OLAP

dalam satu basis data, sistem OLTP berpotensi mengalami penurunan kinerja akibat kontensi sumber daya seperti CPU, memori, dan I/O [2].

2.2.3.2 Online Analytical Processing (OLAP)

Online Analytical processing (OLAP) merupakan sistem pemrosesan basis data yang dirancang untuk mendukung analisis data dalam skala besar guna keperluan pengambilan keputusan. OLAP berorientasi pada eksplorasi dan analisis data historis melalui query kompleks yang melibatkan agregasi, join antar tabel dalam jumlah besar, serta pemrosesan data multidimensi. OLAP umumnya digunakan dalam *business intelligent*, *data warehouse,* pelaporan manajerial, dan analisis tren yang membutuhkan pemrosesan data dalam volume besar.

karakteristik utama OLAP adalah eksekusi query yang bersifat *read-intensive* dengan waktu eksekusi relatif lebih panjang dibandingkan OLTP. Query OLAP sering kali melibatkan operasi seperti *GROUP BY, SUM, AVG, COUNT,* serta join pada banyak tabel untuk menghasilkan ringkasan informasi dari jutaan hingga miliaran baris data. oleh karena itu, desain basis data untuk OLAP cenderung menggunakan struktur denormalisasi atau skema khusus seperti *star schema* dan *snowflake schema* guna mengoptimalkan performa analisis. OLAP lebih menkankan pada kemampuan memproses data dalam jumlah besar secara efisien meskipun dengan waktu respons yang lebih lama.

2.2.3.3 Perbandingan OLTP dan OLAP

Tabel 2. 2 Perbandingan OLTP dan OLAP

Aspek Perbandingan	OLTP	OLAP
Tujuan Utama	Mendukung transaksi operasional harian secara *real-time*	Mendukung analisis data untuk pengambilan keputusan
Jenis Query	Query sederhana (INSERT, UPDATE, DELETE, SELECT)	Query kompleks (JOIN banyak tabel, agregasi, GROUP BY)
Karakteristik Transaksi	*Short Transaction,* jumlah banyak, *concurrent*	*Long running query,* jumlah lebih sedikit

Aspek Perbandingan	OLTP	OLAP
Orientasi Kinerja	*Throughput* tinggi dan *latency* rendah	Efisiensi pemrosesan data besar
Struktur Data	Tabel ternormalisasi	Denormalisasi, *star schema, snowflake schema*
Beban Sistem	Sensitif terhadap *blocking* dan *contention*	Konsumsi CPU, memori, dan I/O tinggi
Penggunaan Umum	Sistem ERP, *e-commerce,* sistem perbankan	*Business intelligence, data warehouse, reporting*

Berdasarkan Tabel 2.2, terdapat perbedaan mendasar antara OLTP dan OLAP dari segi tujuan, karakteristik query, hingga pola penggunaan sumber daya sistem. OLTP dirancang untuk mendukung transaksi operasional harian secara *real-time,* seperti pencatatan penjualan, pembaruan stok, atau transaksi keuangan pada sistem ERP. oleh karena itu, query yang digunakan pada OLTP umumnya bersifat sederhana dan melibatkan operasi dasar seperti INSERT, UPDATE, DELETE, dan SELECT. Transaksi pada OLTP bersifat singkat (*short transaction),* jumlahnya banyak, dan dijalankan secara paralel (*concurrent),* sehingga sistem harus mampu menjaga konsistensi data serta mempertahankan waktu respons yang rendah dengan *throughput* yang tinggi.

Sebaliknya, OLAP berorientasi pada analisis data untuk mendukung pengambilan keputusan manajerial. Query OLAP cenderung lebih kompleks karena melibatkan operasi JOIN pada banyak tabel, agregasi data, serta penggunaan klausa GROUP BY untuk menghasilkan ringkasan informasi. Eksekusi query OLAP biasanya bersifat *long-running* dengan jumlah query yang lebih sedikit dibandingkan OLTP, namun memproses data dalam volume yang jauh lebih besar. Dari sisi orientasi kinerja, OLAP tidak terlalu menekankan latensi rendah, melainkan efisiensi dalam pemrosesan data skala besar.

Perbedaan juga terlihat pada struktur data yang digunakan. OLTP umumnya menggunakan tabel yang ternormalisasi untuk menjaga

integritas data dan meminimalkan redundansi, sedangkan OLAP sering menggunakan struktur denormalisasi sepeti *star schema* atau *snowflake schema* guna mempercepat proses analisis dan agregasi data. Dalam hal beban sistem, OLTP lebih sensitif terhadap *blocking* dan *contention* akibat tingginya tingkat konkurensi transaksi, sementara OLAP cenderung mengonsumsi sumber daya komputasi yang lebih besar, seperti CPU, memori, dan I/O *disk,* terutama menjalankan query analitik yang kompleks.

2.2.4 Resource Management & Lock Mechanism

Resource management dalam sistem basis data berfokus pada bagaimana DBMS membagi dan mengendalikan pemakaian sumber daya (CPU, memori, jaringan, dan I/O) agar kinerja tetap stabil ketika banyak query dan transaksi berjalan bersamaan. Di arsitektur modern yang terdisagregasi (compute terpisah dari *storage),* manajemen sumber daya menjadi semakin penting karena ada biaya jaringan, salah satu pendekatan yang sering dipakai adalah *computation pushdown (*sebagian komputasi dipindahkan mendekati storage) untuk menekan transfer data, tetapi keputusan *pushdown vs caching* tetap harus adaptif supaya tidak memindahkan *bottleneck* dari jaringan menjadi saturasi komputasi di *storage layer* [20]. Temuan lain juga menunjukkan bahwa karakteristik hardware (NUMA, cache) dapat sangat memengaruhi kinerja, sehingga strategi *resource management* yang "cukup baik" di skala kecil belum tentu stabil di skala banyak core dan multi-socket [21].

Dalam penelitian ini, konsep resource management digunakan sebagai landasan untuk menjelaskan bahwa eksekusi OLTP dan OLAP secara bersamaan berpotensi menimbulkan perebutan sumber daya pada sistem basis data. Namun, pengukuran resource pada penelitian ini tidak dilakukan secara langsung pada tingkat sistem operasi seperti penggunaan CPU, RAM, jaringan, atau disk I/O OS. Indikator resource

yang digunakan dibatasi pada statistik level database yang tercatat melalui pg_stat_statements, yaitu total_exec_time, shared_blks_read, dan shared_blks_hit. Dengan demikian, analisis resource dalam penelitian ini dipahami sebagai pengamatan dari sisi internal PostgreSQL, bukan sebagai pengukuran menyeluruh terhadap seluruh sumber daya perangkat keras.

Sementara itu, *lock mechanism* adalah bagian dari *concurrency control* yang menjaga konsistensi data saat transaksi berjalan paralel. Mekanisme ini mencakup penguncian dan konflik agar transaksi tetap memenuhi isolasi, namun konsekuensinya bisa berupa *blocking, waiting,* atau penurunan *throughput* saat konflik tinggi. Banyak DBMS modern juga mengandalkan pendekatan multi-versi seperti MVCC (*Multi-Version Concurrency Control)* untuk mengurangi konflik baca tulis.

2.2.5 Throughput

Throughput merupakan salah satu metrik kinerja sistem yang paling penting dalam evaluasi performa basis data, terutama dalam konteks pemrosesan transaksi dan beban kerja operasional. Dalam sistem basis data, *throughput* biasanya diukur sebagai jumlah operasi atau transaksi yang dapat diproses dalam satu satuan waktu, seperti transaksi per detik (TPS) atau operasi per detik (ops/s) [22], [23]. *Throughput* mencerminkan kemampuan sistem untuk menangani volume pekerjaan dalam periode tertentu dan menunjukkan efisiensi pemrosesan secara keseluruhan di bawah beban yang diberikan [23]. Contohnya di dalam database relasional OLTP, *throughput* yang tinggi artinya sistem mampu melayani banyak permintaan transaksi secara simultan dengan efektif, yang esensial untuk aplikasi operasional seperti sistem ERP dan layanan finansial *real-time.*

Pengukuran *throughput* dalam sistem basis data sering digunakan dalam *benchmark* standar seperti TPC-C [24] atau TATP, di mana

beban kerja diarahkan hingga titik batas maksimum yang dapat ditangani oleh server [22], [23], sehingga *throughput* mencerminkan kapasitas puncak sistem dalam memproses transaksi operasional . Penggunaan throughput sebagai metrik utama juga ditemukan pada penelitian [1], yang membandingkan performa Oracle, PostgreSQL, dan MySQL pada skenario OLTP menggunakan JMeter dan pendekatan benchmark TPC-C. Dalam penelitian tersebut, throughput digunakan untuk menilai kemampuan masing-masing DBMS dalam menangani beban kerja berdasarkan jumlah request yang dapat diproses dalam satuan waktu.

Selain itu, penelitian kinerja basis data modern juga melihat *throughput* sebagai indikator utama dalam mempelajari dampak interferensi antar beban kerja, misalnya ketika query analitik (OLAP) dan transaksi (OLTP) berjalan bersamaan dalam satu sistem yang sama, yang dapat menyebabkan penurunan *throughput* OLTP karena perebutan sumber daya bersama [23]. Oleh karena itu, pengukuran *throughput* menjadi dasar dalam menganalisis performa sistem basis data pada berbagai skenario beban kerja.

2.2.6 Response Time (Latency)

Response time atau *latency* merupakan salah satu metrik kinerja utama dalam sistem basis data yang menggambarkan berapa lama waktu yang dibutuhkan oleh sistem untuk memproses permintaan dan mengembalikan hasil [22]. Dalam konteks sistem basis data, *response time* mencakup durasi dari saat query dikirim oleh pengguna atau aplikasi hingga saat hasil query tersebut diterima kembali. Metrik ini diukur dalam satuan waktu seperti milidetik (ms) dan sering digunakan untuk menilai responsivitas sistem terhadap permintaan-permintaan pengguna atau klien.

Dalam banyak studi benchmarking sistem basis data, response time dievaluasi bersama metrik lain seperti throughput karena keduanya

saling melengkapi dalam memberikan gambaran kinerja keseluruhan [22] Penggunaan response time sebagai metrik evaluasi juga ditemukan pada penelitian [1], yang membandingkan performa Oracle, PostgreSQL, dan MySQL pada skenario OLTP menggunakan JMeter dan pendekatan benchmark TPC-C. Dalam penelitian tersebut, response time digunakan untuk menilai seberapa cepat masing-masing DBMS memproses permintaan pada skenario pengujian yang dilakukan. Throughput menunjukkan jumlah operasi yang dapat diselesaikan dalam satu satuan waktu, sedangkan response time menunjukkan kecepatan penyelesaian masing-masing permintaan atau transaksi. Semakin rendah response time, semakin cepat sistem memberikan hasil sehingga pengalaman pengguna semakin baik.

Pada sistem OLTP, *response time* menjadi sangat penting karena banyak aplikasi transaksi operasional menuntut jawaban instan terhadap ratusan bahkan ribuan query singkat secara bersamaan. Dalam konteks ini, latensi rendah menjadi indikator utama kualitas layanan sistem transaksi [22]. Hal ini berbeda dengan sistem OLAP yang biasanya diukur berdasarkan waktu eksekusi query analitik kompleks dengan durasi lebih panjang, meskipun metrik *response time* tetap relevan dalam mengevaluasi performa query analitik [21].

Pengukuran *response time* perlu mempertimbangkan berbagai faktor termasuk kompleksitas query, ukuran dataset, teknik indexing, mekanisme manajemen sumber daya, serta beban sistem saat eksekusi. Peningkatan kontensi CPU, I/O, dan mekanisme *concurrency control* dapat menyebabkan peningkatan *latency* secara signifikan pada beban kerja transaksi . Sistem dengan desain arsitektur dan manajemen sumber daya yang efisien mampu mempertahankan *response time* yang rendah meskipun dihadapkan pada beban tinggi, sedangkan peningkatan latensi sering mengindikasikan adanya *bottleneck* pada CPU, memori, atau I/O *disk* [21].

2.3 Kerangka Konseptual Penelitian

Kerangka konseptual dalam penelitian ini digunakan untuk menjelaskan hubungan antara beban kerja OLTP, beban kerja OLAP, dan kinerja sistem ERP berbasis PostgreSQL. Penelitian ini memandang OLTP sebagai beban kerja utama yang mewakili transaksi operasional pada sistem ERP, sedangkan OLAP diposisikan sebagai beban kerja analitik yang dijalankan secara bersamaan dalam satu basis data. Ketika OLTP dan OLAP berjalan secara simultan, terdapat potensi interferensi beban kerja yang dapat memengaruhi kinerja transaksi.

Kerangka evaluasi penelitian ini disusun melalui tiga skenario pengujian, yaitu baseline OLTP-only, mixed workload OLAP ringan, dan mixed workload OLAP berat. Skenario baseline digunakan untuk mengetahui performa dasar OLTP tanpa keberadaan query analitik. Skenario mixed workload OLAP ringan dan OLAP berat digunakan untuk mengamati perubahan kinerja OLTP ketika beban kerja analitik dengan tingkat kompleksitas berbeda dijalankan secara bersamaan. Perubahan kinerja tersebut dianalisis melalui metrik response time, throughput, total_exec_time, disk_read, dan cache_hit.

Hasil pengujian kemudian dianalisis menggunakan nilai rata-rata, standar deviasi, persentase perubahan kinerja, One-Way ANOVA, dan Two-Way ANOVA without replication. One-Way ANOVA digunakan untuk melihat perbedaan kinerja OLTP antar skenario workload pada masing-masing perangkat, sedangkan Two-Way ANOVA without replication digunakan untuk melihat pengaruh faktor workload dan perangkat terhadap response time dan throughput. Dengan kerangka ini, penelitian dapat menjelaskan hubungan antara keberadaan beban kerja OLAP dan perubahan kinerja OLTP secara lebih sistematis.

2.4 Tools/ Software yang digunakan

Di dalam penelitian ini digunakan beberapa *software* untuk mendukung jalannya eksperimen, mengelola basis data, serta untuk pengujian kinerja sistem. Pemilihan tools dilakukan dengan mempertimbangkan kesesuaian terhadap kebutuhan penelitian, kemudahan penggunaan, dan relevansinya dengan lingkungan sistem

ERP yang diuji. Berikut merupakan penjelasan detail mengenai *tools* dan *software* yang digunakan.

2.4.1 PostgreSQL

Gambar 2. 1 Logo PostgreSQL

Gambar 2.1 merupakan logo dari PostgreSQL. PostgreSQL adalah sistem manajemen basis data relasional berbasis objek (Object-Relational Database Management System/ORDBMS) yang bersifat open-source dan mendukung standar SQL secara luas. Selain mendukung model relasional tradisional, PostgreSQL juga menyediakan fitur berbasis objek yang memungkinkan pengelolaan data yang lebih kompleks dan fleksibel [25].

Dalam penelitian ini, PostgreSQL berperan sebagai sistem manajemen basis data utama sekaligus lingkungan eksekusi eksperimen. Seluruh beban kerja OLTP dan OLAP dijalankan secara langsung pada PostgreSQL dalam satu basis data yang sama.

Pengujian kinerja, termasuk pengukuran response time dan throughput, dilakukan dengan memanfaatkan mekanisme eksekusi query serta fungsi internal PostgreSQL, seperti pencatatan waktu eksekusi melalui fungsi waktu. Seluruh eksperimen dijalankan pada PostgreSQL versi 17.2

2.4.2 DBeaver

Gambar 2. 2 Logo DBeaver

DBeaver merupakan alat bantu antarmuka basis data yang digunakan untuk berinteraksi dengan sistem manajemen basis data PostgreSQL. Dalam penelitian ini, DBeaver digunakan untuk mengeksekusi query OLTP dan OLAP, termasuk menjalankan blok PL/pgSQL untuk pengujian berulang. DBeaver juga digunakan untuk menampilkan hasil eksekusi, pesan keluaran dari blok PL/pgSQL, serta hasil query statistik yang diperoleh dari PostgreSQL.

Pengamatan kinerja dalam penelitian ini tidak hanya didasarkan pada waktu eksekusi query yang ditampilkan oleh DBeaver, tetapi juga pada metrik yang diperoleh melalui ekstensi `pg_stat_statements`. Metrik tersebut meliputi total waktu eksekusi query (`total_exec_time`), jumlah pembacaan blok dari disk (`shared_blks_read`), serta jumlah akses blok melalui cache (`shared_blks_hit`). Dengan demikian, DBeaver berperan sebagai antarmuka eksekusi dan observasi hasil pengujian, sedangkan metrik kinerja utama diperoleh dari mekanisme statistik PostgreSQL.

2.4.3 Microsoft Excel

Microsoft Excel merupakan perangkat lunak spreadsheet yang digunakan untuk mengolah, menghitung, dan menyajikan data hasil pengujian. Dalam penelitian ini, Microsoft Excel digunakan untuk melakukan rekapitulasi data mentah hasil eksperimen, menghitung nilai response time, throughput, rata-rata, standar deviasi, serta persentase perubahan kinerja antar skenario pengujian.

Selain itu, Microsoft Excel juga digunakan sebagai alat bantu analisis statistik melalui fitur Data Analysis ToolPak. Fitur tersebut digunakan untuk melakukan uji One-Way ANOVA dan Two-Way ANOVA terhadap data hasil pengujian. One-Way ANOVA digunakan untuk menguji perbedaan kinerja OLTP antar skenario workload pada masing-masing perangkat, sedangkan Two-Way ANOVA digunakan untuk menguji pengaruh skenario workload dan perangkat pengujian secara simultan terhadap response time dan throughput. Hasil pengolahan data pada Microsoft Excel kemudian digunakan sebagai dasar dalam penyusunan tabel, grafik, dan interpretasi hasil penelitian.