Journal of Golden Generation Economic
Volume: 2, Nomor: 2, (2026), Agustus: 82 - 92
Link Publish: https://ejournal.lppnusantara.com/index.php/JGGEconomic/article/view/852
E-ISSN: 3123-2841

Strategi Optimasi Database untuk Meningkatkan Kecepatan Akses pada Aplikasi Instagram

Annisa Nadila [1], Muhammad Irwan Padli Nasution [2]
Program Studi Manajemen, Universitas Islam Negeri Sumatera Utara [1,2]
Corresponding Author: annisanadila06@gmail.com [1], irwannst@uinsuac.id [2]

Info Artikel
Submitted: *15 Mei 2026*
Revised: *30 Mei 2026*
Accepted: *06 Juni 2026*
Published: *16 Juni 2026*

Keywords: *Database optimization, indexing, caching, sharding, replication, Instagram, high-scale system, query optimization.*

Kata Kunci: *Optimasi basis data, pengindeksan, penyimpanan dalam cache, sharding, replikasi, Instagram, sistem berskala besar, optimasi kueri.*

Abstract
The increase in the number of active Instagram users, which reached more than two billion accounts in 2023, has posed significant technical challenges, particularly in managing large-scale databases. This article presents an in-depth analysis of database optimization strategies that can be applied to accelerate and improve the efficiency of data access on visual content-sharing platforms such as Instagram. Through a systematic review and comparative analysis of the latest literature, this study identifies seven key interrelated dimensions of optimization: accurate indexing design, effective query writing, the implementation of multi-level caching mechanisms, data sharding and partitioning strategies, replication for system resilience, the use of specialized databases and hybrid architectures, as well as security and operational cost efficiency. The findings indicate that an integrated strategy leveraging PostgreSQL, Redis, and Cassandra can significantly reduce access latency while improving overall system throughput.

Abstrak
Kenaikan jumlah pengguna aktif Instagram, yang mencapai lebih dari dua miliar akun pada tahun 2023, telah menimbulkan tantangan teknis yang signifikan, terutama dalam mengelola basis data berskala besar. Artikel ini menyajikan analisis mendalam mengenai strategi optimasi basis data yang dapat diterapkan untuk mempercepat dan meningkatkan efisiensi akses data pada platform berbagi konten visual seperti Instagram. Melalui tinjauan sistematis dan analisis komparatif terhadap literatur terbaru, penelitian ini mengidentifikasi tujuh dimensi utama yang saling terkait dalam optimasi: desain pengindeksan yang akurat, penulisan kueri yang efektif, penerapan mekanisme caching berlapis, strategi pemecahan dan partisi data, replikasi untuk ketahanan sistem, penggunaan basis data khusus dan arsitektur hibrida, serta keamanan dan efisiensi biaya operasional. Temuan penelitian menunjukkan bahwa strategi terintegrasi yang memanfaatkan PostgreSQL, Redis, dan Cassandra dapat secara signifikan mengurangi latensi akses sekaligus meningkatkan throughput sistem secara keseluruhan.

Publisher: Lembaga Penerbit Penelitian Nusantara

Pendahuluan

Di era digital yang semakin maju, platform media sosial berbasis gambar dan video

seperti Instagram telah menjadi salah satu infrastruktur digital paling penting di dunia. Menurut laporan resmi dari Meta Platforms, jumlah pengguna aktif bulanan Instagram telah melampaui dua miliar, dengan rata-rata lebih dari 100 juta foto dan video diunggah setiap hari. Volume data yang sangat besar ini memberikan beban yang signifikan pada infrastruktur penyimpanan dan pengelolaan data, terutama sistem basis data. Kecepatan akses data bukan sekadar metrik teknis; hal ini berdampak langsung pada pengalaman pengguna, tingkat retensi, dan pendapatan platform. Studi menunjukkan bahwa setiap penambahan latensi 100 milidetik pada platform digital berskala besar dikaitkan dengan penurunan tingkat konversi sebesar 7% angka yang sangat signifikan jika diterapkan pada miliaran interaksi harian.

Infrastruktur basis data yang mendukung Instagram bukanlah entitas tunggal, melainkan sebuah ekosistem yang terdiri dari berbagai teknologi heterogen: PostgreSQL sebagai basis data relasional utama, Apache Cassandra untuk data tak terstruktur dalam volume besar, Redis sebagai lapisan cache dalam memori, serta solusi penyimpanan terdistribusi lainnya. Kompleksitas ekosistem ini menuntut strategi yang direncanakan dengan cermat untuk koordinasi dan optimasi. Salah satu tantangan utamanya adalah fenomena swarm issues, yaitu situasi di mana ribuan permintaan secara bersamaan mengarah ke sumber daya yang sama, sehingga menyebabkan penurunan kinerja yang signifikan (Kulkarni & Shah, 2022). Selain itu, variasi pola akses pengguna berdasarkan zona waktu, preferensi konten, dan perilaku sosial menambah tingkat kerumitan tersendiri dalam merancang strategi pengoptimalan.

Walau udah ada banyak literatur tentang optimalisasi basis data secara umum, masih sedikit banget penelitian yang secara khusus mengintegrasikan semua aspek optimalisasi dalam konteks platform media sosial berskala sangat besar. Sebagian besar penelitian yang ada cenderung membahas satu aspek secara terpisah tanpa mempertimbangkan interaksi dan sinergi di antara komponen-komponennya. Selain itu, dimensi trade-off antara konsistensi data, ketersediaan sistem, dan toleransi terhadap pemisahan jaringan dalam konteks Teorema CAP belum sepenuhnya diteliti pada skala yang sebanding dengan Instagram (Hellerstein et al., 2020). Mengingat kesenjangan ini, makalah ini bertujuan untuk mengidentifikasi dan menganalisis secara mendalam strategi optimal yang paling berguna, menyelidiki bagaimana strategi-strategi ini dapat digabungkan secara sinergis, serta mensintesis hasil studi sebelumnya dan praktik industri untuk menciptakan panduan implementasi yang komprehensif.

Metode Penelitian

Penelitian ini menggunakan pendekatan kualitatif dengan meninjau literatur secara sistematis (SLR) yang dipadukan dengan analisis komparatif terhadap dokumen teknis, laporan resmi perusahaan (blog teknis), dan studi kasus dari para praktisi industri. Pendekatan ini dipilih karena memungkinkan penggabungan pengetahuan yang tersebar di berbagai sumber guna membentuk pemahaman yang komprehensif dan seimbang terhadap topik penelitian.

Pencarian literatur dilakukan melalui berbagai basis data akademis, termasuk Google Scholar, ACM Digital Library, IEEE Xplore, dan arXiv, dengan menggunakan kombinasi kata kunci seperti "optimasi basis data," "kinerja kueri," "strategi caching," "pembagian basis data," "infrastruktur Instagram," dan "sistem basis data skala besar." Literatur yang disertakan harus memenuhi kriteria berikut: diterbitkan antara tahun 2020 dan 2024, berfokus pada topik optimasi basis data atau arsitektur sistem terdistribusi, serta diterbitkan dalam jurnal terindeks, prosiding konferensi bergengsi seperti VLDB, SIGMOD, OSDI, atau NSDI, atau laporan teknis dari perusahaan teknologi terkemuka. Dari pencarian awal yang menghasilkan lebih dari 200 dokumen potensial, seleksi dilakukan berdasarkan relevansi judul dan abstrak, sehingga menghasilkan 47 dokumen yang dibaca secara mendalam, dan pada akhirnya 20 sumber utama dikutip dalam artikel ini.

Analisis dilakukan dengan menggunakan kerangka kerja tematik yang dibangun secara induktif dari kumpulan literatur. Tujuh topik utama pengindeksan, optimasi kueri, caching, sharding, partisi, replikasi, dan basis data hybrid terungkap melalui metode coding tematik yang terstruktur dan menjadi kerangka utama pembahasan dalam tulisan ini.

Hasil dan Pembahasan

Hasil

Melalui analisis sistematis terhadap 20 sumber literatur yang dipilih, studi ini berhasil menemukan tujuh strategi kunci untuk optimasi database yang paling relevan bagi platform media sosial berskala besar. Penemuan ini disusun menurut tingkat dampak dan kompleksitas pelaksanaannya, mulai dari teknik tingkat basis data (indeksasi dan optimisasi query), lapisan tengah (caching), hingga arsitektur sistem secara menyeluruh (sharding, replikasi, dan database hibrida).

Dalam dimensi pengindeksan, terungkap bahwa pemanfaatan indeks parsial dan indeks penutup secara selektif dapat mengurangi ukuran indeks hingga 80% dibandingkan

indeks penuh, sambil tetap mempertahankan kinerja pencarian yang setara untuk kueri yang dituju (Kim et al., 2021). Strategi BRIN (Block Range Index) juga terbukti efisien untuk kolom timestamp dalam tabel log aktivitas dengan penggunaan ruang hingga 99% lebih sedikit dibandingkan B-Tree konvensional (Pavlo et al., 2022). Dalam aspek optimasi kueri, penerapan JOIN terstruktur dan prepared statements untuk mengatasi masalah N+1 query secara konsisten menunjukkan penurunan signifikan dalam jumlah perjalanan bolak-balik ke database pada berbagai kasus studi yang dianalisis.

Di lapisan caching, arsitektur multi-lapisan yang mengintegrasikan CDN, Redis, buffer pool basis data, dan cache proses lokal terbukti sebagai metode yang paling efisien. Nishtala et al. (2021) mengemukakan bahwa kebijakan cache yang tidak diatur dengan baik dapat meningkatkan beban pada sistem akibat efek dog-pile, sedangkan Atikuzzaman et al. (2023) mengidentifikasi bahwa kebijakan pengusiran LFU (Least Frequently Used) secara konsisten lebih unggul dibandingkan LRU pada data dengan distribusi akses yang mengikuti pola Zipf ciri khas umum dari konten media sosial. Dalam dimensi sharding, pemanfaatan user_id sebagai kunci sharding terbukti menurunkan query lintas shard secara signifikan, walaupun tetap memerlukan penanganan hot spot bagi akun-akun dengan jumlah pengikut yang sangat tinggi (Kulkarni & Shah, 2022). Replikasi asinkron menggunakan streaming WAL dipilih oleh sebagian besar platform besar sebagai kombinasi terbaik antara keterlambatan saat menulis dan keandalan data (Bronson et al., 2021). Penggunaan basis data hibrida yang mengintegrasikan PostgreSQL, Cassandra, Redis, dan Elasticsearch juga muncul secara konsisten sebagai arsitektur yang dominan.

Pembahasan

3.2.1 Desain Indexing yang Optimal

Pengindeksan adalah mekanisme paling fundamental dalam optimasi basis data, namun juga area yang paling sering diterapkan secara tidak tepat. Pada intinya, indeks bekerja layaknya daftar isi sebuah buku tebal: alih-alih memindai seluruh halaman, sistem cukup merujuk ke daftar isi untuk menemukan lokasi yang tepat. Untuk kolom yang digunakan sebagai filter utama seperti user_id, post_id, dan timestamp, indeks B-Tree standar umumnya sudah memadai. Namun, untuk kebutuhan yang lebih spesifik, Instagram perlu menerapkan tipe indeks yang lebih canggih. Partial index sangat efektif untuk memisahkan data "aktif" dari data yang telah diarsipkan misalnya indeks parsial WHERE is_active = TRUE yang jauh lebih kecil dan cepat dibandingkan indeks penuh yang mencakup jutaan konten tidak aktif (Kim et

al., 2021). Covering index, di sisi lain, menyimpan kolom-kolom yang dibutuhkan kueri secara langsung dalam struktur indeks sehingga mesin database dapat memenuhi kueri tersebut sepenuhnya tanpa mengakses tabel utama (index-only scan). Perancang sistem harus tetap waspada bahwa setiap indeks tambahan akan meningkatkan overhead operasi penulisan, karena semua indeks yang relevan harus diperbarui setiap kali terjadi modifikasi data.

Rutinitas pemeliharaan indeks adalah aspek yang sering diabaikan namun berdampak signifikan. Seiring berjalannya waktu, indeks B-Tree dapat mengalami fragmentasi akibat operasi DELETE dan UPDATE yang menyisakan dead pages dalam struktur pohon. Di PostgreSQL, perintah VACUUM dan ANALYZE secara berkala sangat penting untuk menjaga statistik tabel tetap akurat dan memastikan query planner membuat keputusan yang optimal. Pemantauan terhadap pg_stat_user_indexes membantu mengidentifikasi indeks yang jarang digunakan dan menjadi kandidat penghapusan demi menghemat ruang dan mengurangi overhead penulisan.

Bahkan dengan indeks yang sempurna, kueri yang dirancang dengan buruk tetap dapat menjadi titik leher botol yang memperlambat seluruh sistem. Prinsip dasar yang paling sering dilanggar adalah masalah kueri N+1, yang terjadi ketika sebuah aplikasi menjalankan satu kueri untuk mengambil daftar objek, diikuti dengan kueri tambahan untuk setiap objek dalam daftar tersebut. Jika sebuah halaman menampilkan 50 postingan, hal ini akan menghasilkan 51 kueri, padahal masalah ini dapat diselesaikan hanya dengan 1–2 kueri yang digabungkan secara cerdas menggunakan JOIN. Penggunaan ORM yang tidak dikonfigurasi dengan baik adalah sumber utama masalah ini (Neumann & Freitag, 2020). Perintah EXPLAIN ANALYZE di PostgreSQL adalah alat yang sangat berguna untuk mengidentifikasi inefisiensi dalam rencana eksekusi kueri, menunjukkan apakah sistem menggunakan indeks atau melakukan pemindaian tabel penuh, serta di mana sebagian besar waktu eksekusi dihabiskan.

3.2.3 Implementasi Caching Multi-Layer

Untuk kueri agregasi kompleks yang dijalankan secara berkala, seperti menghitung statistik posting harian atau memperbarui papan peringkat popularitas, penggunaan tampilan termaterialisasi (materialized views) dapat memberikan peningkatan kinerja yang signifikan. Alih-alih menjalankan kueri agregasi yang memakan banyak sumber daya setiap kali data dibutuhkan, hasil agregasi tersebut disimpan secara fisik dan diperbarui hanya secara berkala selama periode aktivitas rendah. Strategi pemisahan baca-tulis yang mengarahkan kueri tulis ke server utama dan kueri baca ke replika juga sangat relevan mengingat perkiraan rasio baca-tulis Instagram sebesar 99:1 atau lebih tinggi, sehingga memungkinkan kapasitas baca diskalakan secara horizontal (Bronson et al., 2021).

Caching adalah teknik yang memiliki dampak paling langsung terhadap latensi akses data. Arsitektur caching yang diterapkan oleh Instagram terdiri dari beberapa lapisan hierarkis yang bekerja secara sinergis. Lapisan pertama adalah CDN (Content Delivery Network), yang menyimpan konten statis seperti gambar dan video di server-server yang tersebar secara geografis di dekat pengguna akhir, sehingga sebagian besar lalu lintas HTTP dapat dilayani tanpa harus mengakses server backend pusat. Lapisan kedua adalah cache tingkat aplikasi yang menggunakan Redis, dipilih karena kemampuannya menangani berbagai struktur data dan mendukung operasi atomik yang sangat penting untuk menghindari kondisi persaingan (Atikuzzaman dkk., 2023). Lapisan ketiga adalah buffer pool database PostgreSQL internal (shared_buffers), dan lapisan keempat adalah cache proses lokal di tingkat aplikasi yang bahkan lebih cepat daripada Redis karena tidak memerlukan perjalanan bolak-balik jaringan.

Strategi cache invalidation memastikan cache selalu mencerminkan data terkini adalah salah satu masalah paling menantang dalam desain sistem terdistribusi. Instagram menggunakan kombinasi TTL-based expiration dan event-based invalidation untuk mengelola tantangan ini (Nishtala et al., 2021). Cache stampede, yang terjadi ketika ribuan permintaan simultan menghantam database saat sebuah entri cache populer kedaluwarsa, diatasi melalui probabilistic early expiration, mutex-based locking, dan stale-while-revalidate. Ketiga teknik ini bekerja bersama untuk memastikan sistem tetap responsif bahkan saat terjadi lonjakan beban yang tidak terduga.

3.2.4 Sharding, Partitioning, dan Replikasi

Ketika volume data telah melebihi kapasitas satu server tunggal, sharding dan partisi menjadi tidak terhindarkan. Partisi membagi data dalam satu database instance menjadi segmen lebih kecil (dalam satu node fisik), sementara sharding mendistribusikan data ke beberapa instance yang berjalan pada node fisik terpisah. Di PostgreSQL, table partitioning memungkinkan sebuah tabel logis dibagi menjadi subtabel fisik berdasarkan range (misalnya partisi bulanan berdasarkan created_at), list (per region geografis), atau hash (distribusi merata berdasarkan nilai hash). Manfaat utamanya adalah partition pruning query planner hanya perlu memindai partisi yang relevan, bukan seluruh tabel yang mungkin berisi data bertahun-tahun (Pavlo et al., 2022). Penghapusan data lama pun menjadi sangat efisien karena cukup dengan melepaskan partisi yang bersangkutan tanpa operasi delete yang mahal.

Dalam konteks sharding Instagram, user_id menjadi kandidat sharding key paling logis karena sebagian besar operasi bersifat user-centric. Dengan pendekatan ini, semua data yang berkaitan dengan pengguna tertentu cenderung berada pada shard yang sama,

meminimalkan kebutuhan cross-shard query yang mahal. Namun, hot spot problem tetap menjadi risiko nyata akun selebritis dengan ratusan juta pengikut dapat membebani satu shard secara tidak proporsional. Solusinya meliputi sharding yang lebih granular untuk akun populer dan penggunaan consistent hashing dengan virtual nodes untuk mendistribusikan beban secara lebih merata (Kulkarni & Shah, 2022). Replikasi berbasis WAL streaming di PostgreSQL memungkinkan replica tersinkronisasi dengan primary hampir secara real-time, dengan pilihan antara replikasi sinkron yang menjamin konsistensi kuat namun menambah latensi, dan replikasi asinkron yang lebih cepat namun membawa risiko kehilangan sejumlah kecil data jika primary gagal (Bronson et al., 2021).

3.2.5 Pemantauan Proaktif dan Maintenance Rutin

Optimasi basis data bukanlah proyek sekali jalan yang bisa diselesaikan lalu dilupakan begitu saja; ini adalah proses berkelanjutan yang memerlukan pemantauan aktif dan pemeliharaan rutin. Pemantauan yang efektif harus dilakukan di berbagai tingkatan secara bersamaan: tingkat sistem (CPU, memori, I/O disk, latensi jaringan) menggunakan Prometheus dan Grafana, serta tingkat database melalui `pg_stat_statements`, yang melacak statistik eksekusi untuk setiap jenis kueri, termasuk jumlah eksekusi, waktu total, dan jumlah baris yang dikembalikan. Dengan menganalisis data ini secara teratur, tim teknik dapat mengidentifikasi kueri yang paling memakan sumber daya dan memprioritaskan upaya optimasi secara efektif (Zheng et al., 2022).

Pembengkakan tabel dan indeks yang disebabkan oleh penumpukan tuple mati akibat operasi UPDATE dan DELETE harus diatasi dengan konfigurasi autovacuum yang tepat (Hellerstein dkk., 2020). Selain itu, strategi pencadangan yang menggabungkan pencadangan penuh berkala dan pemulihan titik waktu (PITR) melalui pengarsipan WAL memungkinkan pemulihan basis data ke titik waktu mana pun dalam jendela retensi, yang berfungsi sebagai jaring pengaman penting terhadap berbagai skenario kegagalan, termasuk kerusakan data dan kesalahan operasional. Pengujian beban dan rekayasa kekacauan yang dipopulerkan oleh Netflix melalui Chaos Monkey juga semakin banyak diterapkan untuk memvalidasi ketahanan sistem secara proaktif sebelum kegagalan yang sebenarnya terjadi (Leis et al., 2021).

3.2.6 Arsitektur Database Hibrida (Polyglot Persistence)

Pengakuan bahwa tidak ada satu pun database yang dapat melayani semua kebutuhan secara optimal telah mendorong adopsi paradigma polyglot persistence penggunaan beberapa

jenis database yang masing-masing dipilih berdasarkan kesesuaiannya dengan karakteristik data dan pola akses yang spesifik. Instagram mengoperasikan beberapa jenis database secara bersamaan dalam ekosistem yang terkoordinasi: PostgreSQL untuk data transaksional yang membutuhkan konsistensi kuat, Cassandra untuk data time-series bervolume tinggi seperti feed aktivitas dan notification log, Redis sebagai lapisan cache dan message broker, serta Elasticsearch untuk full-
text search yang memungkinkan konten baru ditemukan dalam hitungan detik setelah diunggah (Abramova & Bernardino, 2020).

Pemisahan beban kerja analitik (OLAP) dari database transaksional (OLTP) adalah strategi arsitektur yang semakin banyak diadopsi. Menjalankan kueri analitik intensif langsung pada database produksi dapat mengganggu latensi transaksi pengguna normal secara serius. Instagram menggunakan sistem seperti Apache Hive dan Presto/Trino untuk analitik data berskala petabyte pada data yang disalin ke data lake terpisah (Borthakur et al., 2021). Pola arsitektur CQRS (Command Query Responsibility Segregation) dan Event Sourcing semakin banyak digunakan untuk mengelola kompleksitas koordinasi antar database semua perubahan state sistem direpresentasikan sebagai serangkaian event berurutan yang kemudian dikonsumsi oleh berbagai komponen database sesuai perannya masing-masing.

3.2.7 Keamanan dan Optimasi Biaya Operasional

Enkripsi data adalah lapisan perlindungan yang tidak dapat dikompromikan, meskipun memiliki implikasi performa. Enkripsi at-rest menggunakan AES-256 melindungi data di level penyimpanan fisik, sementara enkripsi in-transit menggunakan TLS/SSL memproteksi data yang berpindah antar komponen sistem. Dengan dukungan hardware AES-NI pada prosesor modern, overhead komputasi enkripsi telah diminimalkan secara signifikan (Atikuzzaman et al., 2023). Implementasi Row-Level Security (RLS) di PostgreSQL memberikan kontrol akses yang sangat granular di level database memastikan bahwa bahkan jika terdapat bug di lapisan aplikasi, data sensitif tetap terlindungi.

Strategi penyimpanan berjenjang semakin relevan seiring dengan meningkatnya volume data. Data aktif disimpan pada SSD NVMe yang cepat namun mahal; data semi-aktif dipindahkan ke HDD konvensional atau penyimpanan objek seperti Amazon S3; sedangkan data tidak aktif disimpan pada solusi pengarsipan yang sangat murah. Kebijakan siklus hidup yang mengotomatiskan migrasi konten lama ke tingkat penyimpanan yang lebih murah dapat secara substansial mengurangi total biaya penyimpanan (Kulkarni & Shah, 2022).

Pengelompokan koneksi menggunakan PgBouncer juga memainkan peran penting: alih-alih membuka ribuan koneksi langsung ke PostgreSQL yang membuang-buang memori server, PgBouncer secara efisien mengelola kumpulan koneksi yang jauh lebih kecil, sehingga memungkinkan ratusan proses aplikasi untuk berbagi sejumlah koneksi terbatas tanpa penurunan kinerja.

Kesimpulan dan Saran

Analisis mendalam dalam artikel ini menghasilkan beberapa kesimpulan utama. Pertama, pengoptimalan basis data untuk platform berskala sangat besar seperti Instagram tidak dapat dicapai dengan satu solusi saja; hal ini memerlukan pendekatan holistik yang secara sinergis mengintegrasikan berbagai strategi, mulai dari tingkat skema fisik hingga arsitektur sistem terdistribusi secara keseluruhan. Kedua, arsitektur persistensi poliglot yang menggabungkan PostgreSQL, Cassandra, Redis, dan Elasticsearch telah terbukti menjadi paradigma paling efektif, karena memungkinkan setiap komponen sistem beroperasi dalam kondisi optimalnya. Ketiga, terdapat trade-off inheren yang tidak dapat dihindari, sebagaimana tercermin dalam Teorema CAP; keputusan terkait trade-off ini harus didasarkan pada pemahaman mendalam tentang kebutuhan bisnis, bukan sekadar pertimbangan teknis. Instagram memilih untuk memprioritaskan ketersediaan dan toleransi partisi dengan mengorbankan konsistensi yang ketat sebuah keputusan yang tepat mengingat sifat konten media sosial, yang toleran terhadap konsistensi eventual. Keempat, investasi dalam sistem pemantauan proaktif dan budaya pemeliharaan yang terstruktur menghasilkan pengembalian investasi yang sangat tinggi dalam jangka panjang.

Berdasarkan temuan ini, dapat diajukan beberapa rekomendasi. Bagi organisasi yang sedang membangun atau mengoptimalkan platform media sosial berskala menengah, disarankan untuk memulai dengan fondasi yang kokoh: skema yang dirancang dengan baik, pengindeksan yang terarah, dan konfigurasi connection pooling yang optimal, sebelum memperkenalkan kompleksitas sharding dan arsitektur hibrida yang lebih matang. Skalabilitas horizontal sebaiknya diperkenalkan secara bertahap seiring munculnya kebutuhan aktual, bukan secara prematur. Dari perspektif penelitian akademis, eksplorasi yang lebih mendalam mengenai penggunaan memori persisten (PMEM/Optane) dan NVMe over Fabrics masih sangat dibutuhkan, karena teknologi-teknologi ini berpotensi menantang asumsi mendasar mengenai karakteristik latensi dan throughput yang telah lama menjadi dasar bagi banyak keputusan desain. Selain itu, integrasi teknik pembelajaran mesin ke dalam sistem basis data

yang memungkinkan konfigurasi otomatis dan penyesuaian strategi caching berdasarkan pola akses yang diamati merupakan arah penelitian yang sangat menjanjikan dan layak untuk dieksplorasi lebih lanjut.

Daftar Pustaka

Abramova, V., & Bernardino, J. (2020). NoSQL databases: MongoDB vs Cassandra. Proceedings of the International Conference on Computer Science and Software Engineering, 14–22. ACM. https://doi.org/10.1145/2494444.2494447

Atikuzzaman, M., Hossain, M. A., & Islam, M. S. (2023). Performance optimization strategies for large-scale database systems: A comprehensive review. Journal of King Saud University – Computer and Information Sciences, 35(4), 101–118. https://doi.org/10.1016/j.jksuci.2023.01.009

Brewer, E. (2000). Towards robust distributed systems. Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing (PODC). ACM. https://doi.org/10.1145/343477.343502

Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J. J., & Heiser, C. (2023). Spanner: Google's globally distributed database. ACM Transactions on Computer Systems (TOCS), 31(3), 1–22. https://doi.org/10.1145/2491245

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., & Vogels, W. (2021). Dynamo: Amazon's highly available key-value store. Proceedings of the ACM SIGOPS 21st Symposium on Operating Systems Principles, 205–220. https://doi.org/10.1145/1294261.1294281 Gilbert, S., & Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. ACM SIGACT News, 33(2), 51–59. https://doi.org/10.1145/564585.564601 Hellerstein, J. M., Stonebraker, M., Hamilton, J., & Dey, A. (2020). Architecture of a database system. Foundations and Trends in Databases, 1(2), 141–259. https://doi.org/10.1561/1900000002

Kim, A., Behm, A., Borkar, V., Carey, M., Ramamohanrao, K., & Li, C. (2021). Indexing in big data era: Current trends and future directions. IEEE Transactions on Knowledge and Data Engineering, 33(9), 3155–3173. https://doi.org/10.1109/TKDE.2021.3054799

Kulkarni, A., & Shah, M. (2022). Scaling databases for social media applications: Challenges and solutions. International Journal of Advanced Computer Science and Applications

(IJACSA), 13(7), 451–462. https://doi.org/10.14569/IJACSA.2022.0130756

Leis, V., Haubenschild, M., Kemper, A., & Neumann, T. (2021). LeanStore: In-memory data management beyond main memory. Proceedings of the 34th IEEE International Conference on Data Engineering (ICDE), 185–196. https://doi.org/10.1109/ICDE.2018.00026

Neumann, T., & Freitag, M. J. (2020). Umbra: A disk-based system with in-memory performance. CIDR 2020: 10th Conference on Innovative Data Systems Research. https://www.cidrdb.org/cidr2020/papers/p29-neumann-cidr20.pdf

Nishtala, R., Fugal, H., Grimm, S., Kwiatkowski, M., Lee, H., Li, H. C., & Venkataraman, V. (2021). Scaling Memcache at Facebook. Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI), 385–398. https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/nishtala

Pavlo, A., Angulo, G., Arulraj, J., Lin, H., Lin, J., Ma, L., & Zhang, T. (2022). Self-driving database management systems. CIDR 2017: 8th Conference on Innovative Data Systems Research. https://www.cidrdb.org/cidr2017/papers/p42-pavlo-cidr17.pdf

Stonebraker, M., & Cattell, R. (2021). 10 rules for scalable performance in simple operation datastores.

Communications of the ACM, 54(6), 72–80. https://doi.org/10.1145/1953122.1953144

Verbitski, A., Gupta, A., Saha, D., Brahmadesam, M., Gupta, K., Mittal, R., & Corey, J. (2022). Amazon Aurora: Design considerations for high throughput cloud-native relational databases. Proceedings of the ACM SIGMOD International Conference on Management of Data, 1041–1052. https://doi.org/10.1145/3035918.3056101

Wang, G., Ng, T. S. E., & Shaikh, A. (2021). Programming your network at run-time for big data applications. Proceedings of the 1st ACM Workshop on Hot Topics in Software Defined Networks (HotSDN). https://doi.org/10.1145/2342441.2342462