Jurnal Publikasi Sistem Informasi dan Manajemen Bisnis
Volume.4, Nomor. 2 Mei 2025
e-ISSN: 2808-8980; p-ISSN: 2808-9383, Hal 487-497
DOI: https://doi.org/10.55606/jupsim.v4i2.5031
Available online at : https://journalcenter.org/index.php/jupsim

Optimalisasi Sistem Database sebagai Pilar Penguatan Sistem Informasi Manajemen di Era Transformasi Digital Modern

Putri Newa Glenda Azra[1*], Maerani Ikwanda[2], Noviana Poernama Sari[3], Tiffany Wahidah Prameswari[4], Fachmi Tamzil[5]
[1-5] Universitas Esa Unggul, Jakarta

Korespondensi Penulis: putrinewag@student.esaunggul.ac.id[*]

Abstract: *As the complexity of modern web applications continues to increase—particularly with the rise of Single-Page Applications (SPAs)—the need for effective monitoring mechanisms becomes more critical to support fast and accurate debugging processes. One promising approach is the implementation of a monitoring system based on Database Management Systems (DBMS), which allows for systematic logging and tracking of execution data within the application's runtime environment. This study aims to explore the role of DBMS in supporting bug reproduction, a process that involves recreating a previously encountered bug by collecting real-time information such as activity logs, event traces, and environmental variables. The research method includes the development of a monitoring prototype integrated into a web application, where all runtime data is stored in a centralized database. Experimental evaluations were conducted on large-scale web applications to assess the system's effectiveness in detecting and reproducing bugs. The results indicate that DBMS-based monitoring significantly improves debugging efficiency by providing contextual execution data that would otherwise be undocumented. Case studies involving the implementation of database systems like MySQL and SQLite in mobile applications demonstrated improvements in both reliability and data access speed for log management. These findings show that integrating monitoring with DBMS contributes meaningfully to maintaining software quality. However, this integration also introduces technical challenges, such as heavy logging loads, potential latency, and data security concerns. To address these issues, this study recommends optimization strategies including caching mechanisms, asynchronous I/O operations, and the adoption of cloud-native architectures to enhance system scalability and performance. By applying these approaches, bug reproduction can be performed more quickly, accurately, and efficiently—ultimately supporting the development of robust and reliable web applications in today's digital era. This research highlights the importance of runtime observability and structured monitoring as key elements in modern software maintenance and debugging workflows.*

Keywords: *Cloud, Database, Information System, Management, Monitoring*

Abstrak: Seiring dengan meningkatnya kompleksitas aplikasi web modern, terutama pada jenis Single-Page Applications (SPAs), kebutuhan akan mekanisme monitoring yang efektif semakin penting untuk mendukung proses debugging yang cepat dan akurat. Salah satu pendekatan yang menjanjikan adalah penerapan sistem monitoring berbasis Database Management System (DBMS), yang memungkinkan pencatatan dan pelacakan data eksekusi secara sistematis dalam lingkungan runtime aplikasi. Penelitian ini bertujuan untuk mengeksplorasi peran DBMS dalam mendukung reproduction bug, yaitu proses mereproduksi kembali bug yang telah terjadi, melalui pengumpulan informasi secara real-time seperti log aktivitas, pelacakan peristiwa (event tracing), dan variabel lingkungan. Metode yang digunakan dalam penelitian ini melibatkan pengembangan prototipe sistem monitoring yang terintegrasi dengan aplikasi web, di mana seluruh data runtime disimpan dalam basis data. Evaluasi eksperimental dilakukan terhadap aplikasi web berskala besar guna mengukur efektivitas sistem ini dalam mendeteksi dan mereproduksi bug. Hasil dari eksperimen menunjukkan bahwa sistem monitoring berbasis DBMS ini mampu meningkatkan efisiensi dalam proses debugging dengan menyediakan data kontekstual yang sebelumnya tidak terdokumentasikan. Studi kasus yang dilakukan pada implementasi sistem basis data seperti MySQL dan SQLite dalam aplikasi mobile menunjukkan adanya peningkatan keandalan serta kecepatan akses data dalam proses manajemen log. Keberhasilan ini menunjukkan bahwa integrasi monitoring dengan DBMS memberikan kontribusi signifikan terhadap pemeliharaan kualitas perangkat lunak. Namun demikian, integrasi ini juga menghadirkan tantangan teknis, seperti beban sistem akibat logging yang besar, potensi latensi, serta isu keamanan data. Untuk mengatasi hambatan tersebut, penelitian ini merekomendasikan strategi optimalisasi seperti penggunaan caching, penerapan I/O asinkron, serta adopsi arsitektur cloud-native untuk meningkatkan skalabilitas dan efisiensi sistem monitoring. Dengan pendekatan ini, reproduction bug dapat dilakukan secara lebih cepat, akurat, dan efisien, sekaligus mendukung pengembangan aplikasi web yang andal dan tangguh di era digital saat ini.

Kata Kunci: Basis Data, Komputasi, Manajemen, Pemantauan, Sistem Informasi

Received: April 16, 2025; Revised: April 30, 2025; Accepted: Mei 17, 2025; Published: Mei 31, 2025

1. PENDAHULUAN

Seiring dengan semakin meningkatnya pengembangan sistem informasi berbasis web, kebutuhan akan sistem monitoring dan *debugging* yang efektif semakin mendesak. Aplikasi web modern tidak hanya melayani ratusan hingga ribuan permintaan pengguna secara bersamaan, tetapi juga harus mampu menjaga kestabilan dan ketepatan respons di tengah lingkungan yang terus berubah. Kegagalan sistem atau kesalahan (*bug*) yang tidak segera terdeteksi dan ditangani dapat berdampak besar terhadap kepuasan pengguna dan reputasi layanan. Oleh karena itu, diperlukan pendekatan sistematis guna melakukan pemantauan (*monitoring*) serta reproduksi kesalahan (*bug reproduction*) secara efisien.

Dalam konteks ini, peran *Database Management System* (DBMS) menjadi sangat penting sebagai komponen utama dalam penyimpanan data *log, event*, dan jejak kesalahan sistem. DBMS tidak hanya berfungsi sebagai media penyimpanan data pengguna, tetapi juga memainkan peran strategis dalam mendukung mekanisme *monitoring* dan reproduksi kesalahan secara *real-time*. Kemampuan untuk merekam dan mengelola informasi diagnostik secara terstruktur melalui DBMS memungkinkan pengembang untuk melakukan analisis mendalam, mengidentifikasi sumber permasalahan secara cepat, serta meningkatkan keandalan dan performa aplikasi web berskala besar.

Implementasi DBMS dalam berbagai sektor informasi telah terbukti mampu meningkatkan efisiensi dan keandalan sistem informasi. Dalam pengembangan aplikasi *mobile*, misalnya, penggunaan DBMS seperti SQLite dan MySQL terbukti mendukung proses sinkronisasi data serta efisiensi operasional. Di sektor pendidikan, penerapan sistem informasi berbasis MySQL juga menghasilkan peningkatan signifikan dalam kecepatan akses data, dengan efisiensi pencarian mencapai hingga 40%. Fakta ini menegaskan bahwa optimalisasi struktur dan pemanfaatan *database* tidak hanya penting, tetapi juga menjadi fondasi utama dalam membangun sistem informasi yang responsif, terukur, dan andal.

Menurut (Hansyah et al., 2024), database sangat penting untuk transformasi digital, terutama dalam konteks system informasi manajemen kontemporer.

Pentingnya optimalisasi kinerja DBMS berbasis *cloud* turut disoroti dalam berbagai studi, yang mengusulkan strategi seperti integrasi lapisan *cache*, pengelolaan I/O secara asinkron, serta penerapan arsitektur *multi-threading* guna meningkatkan *throughput* dan menurunkan tingkat latensi. Pendekatan-pendekatan ini menjadi sangat relevan dalam konteks *monitoring* aplikasi web, yang dituntut untuk mampu memproses dan merespons *volume* data besar secara *real-time*, tanpa mengorbankan kecepatan maupun kestabilan sistem.

Database diposisikan sebagai komponen inti yang menghubungkan antara *input* dan *output* dalam sebuah sistem informasi. Basis data berfungsi sebagai pusat pengelolaan penyimpanan dan distribusi informasi lintas unit, yang menjadikannya elemen kunci dalam keseluruhan arsitektur sistem. Peran strategis ini menegaskan pentingnya DBMS tidak hanya dalam menyimpan data operasional, tetapi juga sebagai fondasi utama dalam sistem *monitoring* dan pelacakan kesalahan. Melalui pencatatan data historis dalam bentuk *log*, DBMS memungkinkan proses reproduksi *bug* dilakukan secara sistematis dan terarah.

Selain di bidang pendidikan dan pengembangan aplikasi, sistem *database* juga berperan fundamental dalam mendukung pengambilan keputusan berbasis *big data*. Dalam konteks monitoring aplikasi web, sistem database yang robust tidak hanya memungkinkan penyimpanan data *log* dalam skala besar, tetapi juga mendukung proses analitik yang diperlukan untuk mereproduksi kesalahan secara presisi serta memfasilitasi perbaikan sistem secara efisien dan terarah.

Oleh karena itu, penelitian ini difokuskan pada penerapan sistem *monitoring* berbasis *database* untuk mendukung proses reproduksi *bug* dalam aplikasi *web*. Dengan pendekatan ini, diharapkan dapat diperoleh sistem monitoring yang tidak hanya mencatat kesalahan secara *real-time,* tetapi juga mampu memberikan informasi yang cukup untuk menelusuri penyebab kesalahan dan mereproduksinya secara sistematis.

Rumusan masalah

Berdasarkan latar belakang di atas, maka rumusan masalah dalam penelitian ini adalah:

1. Bagaimana peran DBMS dalam mendukung proses *monitoring* dan *debugging* aplikasi berbasis web?
2. Bagaimana strategi penyimpanan serta analisis data *log* dapat dimanfaatkan untuk mereproduksi *bug* secara efisien dan terarah?
3. Teknologi dan arsitektur sistem seperti apa yang paling efektif untuk implementasi monitoring aplikasi web dalam skala besar?
4. Apa saja tantangan teknis maupun fungsional dalam mengintegrasikan sistem *monitoring* dengan DBMS, dan bagaimana solusi yang dapat diterapkan untuk mengatasinya?
5. Sejauh mana efektivitas sistem monitoring berbasis database dalam meningkatkan keandalan dan kecepatan penanganan kesalahan pada aplikasi web?

Tujuan penelitian

Penelitian ini bertujuan untuk:

1. Menganalisis peran sistem *database* (DBMS) dalam mendukung proses *monitoring* dan reproduksi *bug* pada aplikasi berbasis *web*, khususnya melalui mekanisme pencatatan (*logging*) dan pelacakan (*tracing*) kesalahan sistem.
2. Mengembangkan pendekatan monitoring yang efektif berbasis *log tracing* dan *database event tracking*, guna mengidentifikasi dan mereproduksi kesalahan secara akurat serta mengurangi waktu *downtime* sistem.
3. Mengusulkan arsitektur sistem *monitoring* yang terintegrasi dengan DBMS berbasis *cloud* dan mendukung pemrosesan data secara *real-time*, dengan mempertimbangkan efisiensi, skalabilitas, dan kecepatan akses data.
4. Mengevaluasi kinerja sistem monitoring melalui studi kasus atau simulasi, untuk mengetahui sejauh mana integrasi DBMS dapat meningkatkan kemampuan deteksi dan reproduksi *bug* dalam aplikasi web skala besar.
5. Mengidentifikasi tantangan teknis dalam implementasi sistem *monitoring* dan *debugging* berbasis *database*, serta memberikan rekomendasi perbaikan untuk pengembangan sistem informasi ke depannya.

2. METODE PENELITIAN

Penelitian ini menggunakan pendekatan kualitatif deskriptif dengan metode studi pustaka (literature review) untuk menganalisis peran dan optimalisasi sistem database dalam mendukung sistem informasi manajemen di era transformasi digital. Pendekatan ini dipilih untuk memperoleh pemahaman mendalam mengenai konsep, praktik, serta tantangan yang dihadapi dalam pengelolaan sistem basis data modern dalam konteks transformasi digital.

Sumber Data

Data yang digunakan dalam penelitian ini berasal dari sumber sekunder berupa artikel jurnal ilmiah, buku akademik, laporan teknis, dokumentasi industri, serta situs resmi teknologi seperti Microsoft Learn dan DataSunrise. Kriteria inklusi dalam pemilihan literatur adalah:Terbit antara tahun 2020 hingga 2025,Relevan dengan topik sistem database, sistem informasi manajemen, dan transformasi digital,Telah melalui proses peer-review (untuk jurnal),Mempunyai DOI atau tautan yang valid untuk keperluan verifikasi.

Teknik Pengumpulan Data

Teknik pengumpulan data dilakukan melalui pencarian sistematis pada database jurnal ilmiah seperti Google Scholar, IEEE Xplore, ScienceDirect, SSRN, dan DOAJ. Kata kunci yang digunakan meliputi: database optimization, management information system, digital transformation, logging systems, dan cloud-based database.

Teknik Analisis Data

Data dianalisis menggunakan metode analisis isi (content analysis) untuk mengidentifikasi pola, tren, dan hubungan antar konsep. Prosedur analisis dilakukan melalui langkah-langkah berikut:Koding isi dokumen berdasarkan tema utama (struktur database, keamanan, performa, pemantauan/logging, dan integrasi dengan sistem informasi),Sintesis literatur untuk membandingkan berbagai pandangan dan praktik yang ditemukan,Penarikan kesimpulan mengenai strategi optimalisasi sistem database dan kontribusinya terhadap penguatan sistem informasi manajemen.

Validitas Data

Validitas data dijaga dengan triangulasi sumber, yakni membandingkan temuan dari berbagai jenis literatur (akademik, industri, dan dokumentasi teknis). Selain itu, analisis dilakukan secara sistematis dan terdokumentasi, untuk memastikan objektivitas dan replikasi data.

3. KAJIAN PUSTAKA

Landasan teori

Sistem Manajemen Basis Data (DBMS): DBMS adalah perangkat lunak yang mengelola penyimpanan, pengambilan, dan manipulasi data dalam basis data. DBMS relasional (RDBMS) seperti MySQL mengorganisir data dalam tabel-tabel terstruktur. Peran DBMS sangat penting dalam sistem informasi karena basis data berfungsi sebagai pusat penyimpanan dan distribusi informasi antar-unit (Bratha, 2022). Sebagai contoh, MySQL dapat menangani skala data sangat besar. (Rawat et al., 2021) melaporkan MySQL mampu mendukung *database* dengan 5 milyar baris untuk data observasi. DBMS modern mendukung transaksi, konsistensi data, indeks untuk *query* cepat, serta replikasi untuk ketersediaan tinggi. Di sisi lain, SQLite adalah DBMS embedded yang ringan dan *serverless*, sering digunakan pada perangkat *mobile* sebagai penyimpanan data lokal karena kemudahannya dalam sinkronisasi ke server lain.

Sistem Monitoring Aplikasi: *Monitoring* aplikasi meliputi pengumpulan metrik performa (CPU, memori, waktu *respons*), *log* kesalahan, serta jejak eksekusi (*tracing*). Alat *Application Performance Monitoring* (APM) mencakup agregasi *log*, metrik, dan notifikasi. Dalam konteks *web*, *monitoring* mencakup *server-side* (misal pemantauan basis data, permintaan latensi) dan *client-side* (kinerja UI, interaksi pengguna). Data *log* aplikasi biasanya dikirimkan ke penyimpanan terpusat, sering kali juga dimasukkan ke DBMS untuk analisis historis. Hal ini memungkinkan deteksi error secara *real-time* serta analisis lebih lanjut. Misalnya, *database logging* secara komprehensif menjadi pusat pelacakan kesalahan dalam aplikasi *web* berskala besar. Dengan menyimpan histori *event* dan *exception* ke dalam DBMS, sistem *monitoring* dapat merespons anomali lebih cepat (misalnya dengan notifikasi) dan membantu proses *debug* secara terarah.

Reproduksi *Bug*: Reproduksi *bug* adalah upaya untuk menciptakan kembali kondisi kesalahan berdasarkan laporan *bug*. Proses ini penting untuk menemukan akar penyebab. Namun, terutama pada aplikasi web modern seperti SPA, reproduksi sering sulit dilakukan. (Wang et al., 2025) menyatakan bahwa informasi dalam laporan *bug* dari pengguna sering tidak memadai untuk mereproduksi kondisi lingkungan eksekusi. Misalnya, *state* atau *input* pengguna di *browser* mungkin tidak tercatat. *Monitoring* aplikasi yang baik harus menangkap semua aktivitas signifikan (misal *event* klik, permintaan API) agar *bug* dapat direproduksi kembali di lingkungan pengembang. Tanpa informasi lengkap, *debugging* bisa memakan waktu sangat lama atau gagal menemukan penyebab *bug*.

***Debugging*:** *Debugging* adalah proses menemukan dan memperbaiki cacat perangkat lunak. *Monitoring* dan *logging* sangat membantu proses ini. Praktik umum meliputi *log debugging* (mencetak pesan kesalahan saat *runtime*) dan *tracing* (mengikuti alur eksekusi di banyak komponen). Sistem *monitoring* modern menyediakan visualisasi *log* dan *trace* untuk memudahkan analisis. Misalnya, *log* tingkat *error* atau *exception* dapat dipetakan ke *user session* tertentu. Ketersediaan *log* historis dalam DBMS memungkinkan pengembang menanyakan ulang kejadian berdasarkan waktu atau konteks, sehingga mempercepat identifikasi penyebab. Seorang *developer* sangat penting untuk memiliki keterampilan *debugging* yang mahir, dan harus mampu untuk menggunakan alat *debugging*, serta memahami stack trace untuk mengidentifikasi akar penyebab masalah yang ada. (Wang et al., 2025)

Arsitektur *Cloud*: Dalam arsitektur *cloud*, DBMS dioperasikan sebagai layanan terdistribusi (misal *Database as a Service*) (Sunkara et al., 2023). *Cloud* memungkinkan elastisitas skala *database*, distribusi beban kerja, dan redundansi geografis. Komponen DB di cloud sering mendukung replikasi otomatis, auto-scaling, dan caching layer untuk menjaga

performa. Optimasi DBMS *cloud* mencakup strategi seperti *caching query*, pengelolaan I/O asinkron, dan pemrosesan *multithread* untuk *throughput* tinggi (Sunkara et al., 2023). Arsitektur *cloud-native* juga memudahkan penyimpanan *log volume* besar (misalnya menggunakan *object storage* atau basis data terdistribusi) tanpa membebani *server* aplikasi. Hal ini penting untuk *monitoring web* skala besar, di mana sistem harus memproses banyak data dalam waktu nyata tanpa mengorbankan kecepatan respon.

4. HASIL DAN PEMBAHASAN

Monitoring aplikasi berbasis web: peran DBMS, logging, dan infrastruktur

Peran DBMS dalam pemantauan aplikasi web sangat krusial karena basis data berfungsi sebagai penyimpan terpusat bagi *log* dan *event* aplikasi (Wang et al., 2024). Dengan menggunakan DBMS, data *log* dapat disusun dalam format terstruktur (misalnya JSON) pada tabel khusus, memanfaatkan kemampuan *query* dan indeks untuk pencarian cepat. Misalnya, MySQL atau PostgreSQL dapat menyimpan *log* operasi dalam tabel relasional dengan kolom-kolom terdefinisi, memanfaatkan ACID *compliance* untuk menjaga konsistensi data. Menurut studi literatur, sistem basis data berperan sebagai inti yang sangat penting dalam sistem informasi manajemen, di mana DBMS menyediakan mekanisme integritas data, akses *multi-user*, serta prosedur *backup* dan *recovery* untuk mendukung keandalan sistem. Dengan demikian DBMS tidak hanya menyimpan data utama tetapi juga *log* aktivitas aplikasi, sehingga proses *debugging* menjadi lebih sistematis: tim pengembang dapat menelusuri "jejak" eksekusi dan kesalahan melalui *query* pada data *log* yang tersimpan di *database*.

Strategi penyimpanan dan analisis data *log*

Untuk memproduksi *bug* secara efisien, *log* aplikasi sebaiknya disimpan dan dianalisis dengan strategi yang terstruktur. Data *log* dapat disimpan di basis data relasional jika berbentuk terstruktur, atau pada NoSQL/*Spesialized Log System* untuk *volume* besar data tidak terstruktur. DBMS relasional menawarkan penyimpanan terstruktur dan kapabilitas kueri yang kuat, namun untuk aplikasi skala besar seringkali digunakan *database* NoSQL (seperti MongoDB atau Cassandra) karena skalabilitas dan *throughput* tulis tingginya. Sebagai ilustrasi, DataSunrise merekomendasikan tabel operasi *log* dengan skema terdefinisi lengkap agar pencarian dan analisis SQL dapat digunakan. Dalam praktiknya, *log* disimpan dengan penyajian metadata lengkap (*timestamp*, jenis *event*, ID pengguna, dsb) sehingga *query* fokus dapat dipakai untuk mengidentifikasi pola kesalahan atau *anomalies*.

Teknik analisis selanjutnya memanfaatkan alat *big data* atau *streaming*: misalnya aliran *log* dikumpulkan melalui *message queue* (Kafka) dan diproses secara *real-time* oleh Spark atau Elasticsearch untuk deteksi cepat. Penting juga memperhatikan format *log*, seperti penggunaan format JSON terstruktur agar *parsing* lebih mudah. Selain itu, data sensitif dalam *log* harus ditangani khusus – misalnya melakukan enkripsi atau pemfilteran sebelum penyimpanan. Pada akhirnya, penggabungan mekanisme *indexing* dan visualisasi (Grafana, Kibana, dsb.) memungkinkan tim secara cepat memfilter dan menganalisis *log*, sehingga reproduksi bug dapat dilakukan secara terarah berdasarkan bukti *log* yang tersimpan.

Arsitektur teknologi untuk monitoring skala besar

Arsitektur *monitoring* aplikasi *web* berskala besar kini cenderung berbasis *cloud* dan *microservices*. Sistem terdistribusi memanfaatkan arsitektur *observability* yang lengkap, menggabungkan logging, metrik, dan tracing terdistribusi. (Madupati, 2025) menekankan bahwa pada arsitektur *microservices*, tingkat observability (melalui *logging*, metrik, dan *distributed tracing*) harus tinggi agar kondisi sistem tetap stabil. Implementasi praktisnya melibatkan penggunaan *tool* seperti Prometheus dan Grafana untuk metrik/alert, serta Jaeger atau Zipkin untuk *distributed tracing*, sehingga tiap layanan mikro dapat dipantau kesehatannya secara mandiri. Data *log* dari berbagai komponen kemudian digabung melalui *pipeline event-driven* (misalnya Kafka), kemudian dianalisis secara *real time* oleh sistem analitik (Spark, Flink, dsb.) atau dimasukkan ke penyimpanan skala besar.

Untuk skalabilitas, sistem *logging* biasanya dijalankan secara terdistribusi: *log server* berklaster dan *load-balancer* mendistribusikan beban tulis *log*. DataSunrise merekomendasikan agar infrastruktur *logging* dirancang terpisah dari *database* aplikasi utama untuk menghindari hambatan performa dan masalah kapasitas (DataSunrise, 2024). Selain itu, sistem *time-series* atau *NoSQL* (seperti ElasticSearch) kerap dipakai untuk menyimpan *log* secara efisien serta mendukung pencarian *real-time*. *Cloud monitoring architecture* juga menggunakan beberapa zona/region untuk ketersediaan tinggi – misalnya Azure Log Analytics dan AWS CloudWatch – sehingga data operasional dikumpulkan ke dalam *workspace* khusus. Desain ini memungkinkan tim operasional memantau performa dan *event* aplikasi secara global serta memberi peringatan otomatis ketika terdeteksi pola kesalahan.

Studi kasus oleh (Ningsih et al., 2024) menunjukkan bahwa metode pengawasan berbasis cloud seperti AWS Lambda, DynanmoDB, dan EventBridge dapat meningkatkan efesiensi pemrosesan log dalam skala besar dan kecepatan deteksi bottleneck.

Tantangan integrasi monitoring dengan DBMS dan solusi

Integrasi sistem *monitoring* dengan DBMS menghadirkan beberapa tantangan teknis dan fungsional. Secara teknis, pencatatan *log* yang masif dapat menimbulkan *overhead* performa. Setiap operasi *logging* berarti penulisan tambahan ke *database*, yang jika tidak diatur dapat memperlambat aplikasi. Sematext menekankan bahwa tanpa protokol pengiriman yang tepat, *log* yang berlebihan dapat memperlambat program dan bahkan hilang sebelum sempat disimpan. Selain itu, *volume log* yang cepat meningkat berpotensi memenuhi kapasitas penyimpanan jika tidak ada kebijakan retensi dan rotasi. Dari sisi keamanan/fungsional, *log* sering memuat data sensitif misalnya informasi pengguna, sehingga memerlukan enkripsi atau penyaringan sebelum disimpan. Keterpaduan format *log* antar modul juga bisa menjadi masalah jika tidak distandarisasi (misalnya JSON struktur yang konsisten), sehingga analisis gabungan menjadi sulit.

Berbagai solusi telah diusulkan. Secara arsitektural, penulisan *log* sebaiknya dilakukan asinkron (misalnya melalui antrean atau *worker thread*) agar tidak menghambat jalannya aplikasi. Teknik *batching* dan *sampling* juga dapat diterapkan untuk mengurangi frekuensi penulisan individual pada DB. Pendekatan lain adalah memisahkan infrastruktur *log*: menggunakan DB terpisah atau *cluster* NoSQL khusus untuk *log*, sehingga beban *read/write log* tidak mencampur ke DB operasional. Pada tataran aplikasi, digunakan *level logging* yang dapat dikonfigurasi (DEBUG/INFO/ERROR) untuk mengontrol detail *log* sesuai kebutuhan. Untuk *compliance* dan privasi, Azure Well-Architected menyarankan ekspor data *log* secara rutin dan pemfilteran data sensitif agar aturan regulasi terpenuhi (Microsoft, 2025). Pendekatan *observability* juga mencakup automasi analisis *log* (SIEM) yang dapat memproses *log* di luar lingkungan aplikasi utama, serta *deployment monitoring tools* untuk memastikan *pipeline logging* sehat (misalnya *health monitoring* pada data *collection endpoints*).

Efektivitas monitoring berbasis database terhadap keandalan sistem

Sistem monitoring berbasis *database* terbukti meningkatkan keandalan dan kecepatan penanganan kesalahan secara signifikan. Dengan adanya *log* terpusat dan *trace event*, tim pengembang dapat cepat mendeteksi anomali dan menentukan akar masalah. Misalnya, Sematext menyatakan bahwa *log* yang mengikutsertakan metadata waktu (“timestamp”) memungkinkan admins memonitor pola abnormal dan melakukan *root-cause analysis* terhadap isu sistem secara *real time*. Selain itu, kemampuan untuk menset alarm berdasarkan pola *log* membantu tim merespons masalah sebelum pengguna terpengaruh. Studi kasus mendukung hal ini: (Madupati, 2025) mencatat bahwa pengimplementasian *observability* (termasuk *logging*

dan *tracing*) menjaga kondisi sistem tetap stabil dan memudahkan pengambilan keputusan operasional dalam skala besar. Singkatnya, dengan *monitoring* terintegrasi, waktu henti akibat *bug* berkurang karena deteksi dini dan perbaikan lebih cepat – suatu peningkatan *uptime* dan reliabilitas layanan yang penting di lingkungan produksi. Hasil praktisnya sering terlihat pada metrik penurunan MTTR (*Mean Time To Repair*) dan peningkatan kepuasan pengguna karena gangguan dapat ditangani lebih tanggap. Kajian ini menunjukkan bahwa pendekatan *log tracing* dan *event tracking* berbasis *database* dapat secara signifikan meningkatkan keandalan dan responsivitas aplikasi *web* dalam menangani kesalahan.

5. KESIMPULAN

Penerapan sistem *monitoring* aplikasi berbasis *web* yang terintegrasi dengan Sistem Manajemen Basis Data (DBMS) memiliki peran krusial dalam menjaga keandalan, performa, dan juga responsivitas sistem. DBMS tidak hanya berfungsi untuk penyimpanan data utama, tetapi berfungsi juga untuk menjadi pusatnya tempat penyimpanan *log* aktivitas dan *event* aplikasi secara terstruktur. Hal ini memungkinkan proses *debugging*, reproduksi *bug*, dan analisis kesalahan dilakukan secara sistematis dan efisien. Dengan adanya dukungan dari teknologi *cloud* dan arsitektur yang terdistribusi, *monitoring* skala besar dapat dilakukan secara *real-time* dengan menggunakan *log*, metrik, dan *tracing* yang terintegrasi. Aplikasi seperti Kafka, Prometheus, dan Elasticsearch dapat membantu untuk mempercepat proses analitik, sementara penyimpanan *log* yang terpisah dari *database* operasional dapat menjaga performa sistem. Penelitian ini menegaskan bahwa DBMS tidak hanya berfungsi sebagai tempat penyimpanan data, tetapi bisa juga sebagai pondasi utama untuk membangun sistem *monitoring* yang andal dan efisien. Dengan pendekatan ini, kendala aplikasi meningkat, waktu perbaikan kesalahan berkurang, dan kualitas layanan kepada *user* menjadi lebih terjaga.

DAFTAR PUSTAKA

Bratha, W. G. E. (2022). Literature Review Komponen Sistem Informasi Manajemen: Software, Database Dan Brainware. *Jurnal Ekonomi Manajemen Sistem Informasi*, 3(3), 344–360.

Coronel, C., & Morris, S. (2019). *Database Systems: Design, Implementation, & Management* (13th ed.). Cengage Learning.

Dalle, J., & Fitri, A. (2021). Transformasi Digital Berbasis Cloud Computing: Studi Kasus Pemanfaatan Database di Instansi Pemerintah. *Jurnal Teknologi Informasi dan Ilmu Komputer*, 8(2), 157–166. https://doi.org/10.25126/jtiik.2021822020

DataSunrise. (2024). *Database Logging: Best Practices and Solutions*. https://www.datasunrise.com/knowledge-center/database-for-logging/

Hansyah, S., Zai, D., Irwan, M., & Nasution, P. (2024). Peran Basis Data Dalam Transformasi Digital di Era Industri. *Journal of Sharia Economics Scholar (JoSES)*, 20(2), 84–86. https://doi.org/10.5281/zenodo.12528183

Madupati, B. (2025). Observability in Microservices Architectures: Leveraging Logging, Metrics, and Distributed Tracing in Large-Scale Systems. *SSRN Electronic Journal*, 10(11), 24–31. https://doi.org/10.2139/ssrn.5076624

Microsoft. (2025). *Architecture best practices for Log Analytics*. Microsoft Ignite. https://learn.microsoft.com/en-us/azure/well-architected/service-guides/azure-log-analytics

Ningsih, N., Karimatun Nisa, Endryco Farel Rianrachmatullah, Bintang Desta Ramadhani, & Afifah Dwi Ramadhani. (2024). Optimalisasi monitoring tag dengan AWS Cloud: Studi kasus aplikasi tagtracker pada PT. XYZ. *INFOTECH: Jurnal Informatika & Teknologi*, 5(1), 88–98. https://doi.org/10.37373/infotech.v5i1.1163

O'Brien, J. A., & Marakas, G. M. (2017). *Management Information Systems* (11th ed.). McGraw-Hill Education.

Rawat, B., Purnama, S., & Mulyati, M. (2021). MySQL Database Management System (DBMS) On FTP Site LAPAN Bandung. *International Journal of Cyber and IT Service Management*, 1(2), 173–179. https://doi.org/10.34306/ijcitsm.v1i2.47

Rifa'i, M. A., & Nugroho, Y. (2023). Penerapan Data Warehouse dalam Penguatan Sistem Informasi Manajemen Rumah Sakit. *Jurnal Teknologi dan Sistem Informasi*, 9(1), 45–55. https://doi.org/10.21070/jtsi.v9i1.1735

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). *Database System Concepts* (7th ed.). McGraw-Hill Education.

Sunkara, J. R., Bauskar, S. R., Madhavaram, C. R., Galla, E. P., & Gollangi, H. K. (2023). Optimizing Cloud Computing Performance with Advanced DBMS Techniques: A Comparative Study. *Journal for ReAttach Therapy and Developmental Diversities*, 6(2), 2493–2502. https://doi.org/10.53555/jrtdd.v6i10s(2).3206

Wang, D., Galster, M., & Morales-Trujillo, M. (2024). Application Monitoring for bug reproduction in web-based applications. *Journal of Systems and Software*, 207(September 2023), 111834. https://doi.org/10.1016/j.jss.2023.111834

Wang, D., Galster, M., & Morales-Trujillo, M. (2025). Information needs in bug reports for web applications. *Journal of Systems and Software*, 219(September 2024), 112230. https://doi.org/10.1016/j.jss.2024.112230