The University of Western Ontario
Department of Computer Science

A Universal Sensitive Data Store in MariaDB

Group Members:
Jawaad Ahmar, 251237757
Mirna Qusai Alber Aziz, 251227790
Daniel Esemezie, 251236885
Ryan Kennedy Hecht, 251220567
Seyed Hirbod Hosseini, 251234419

Databases II (Advanced Databases)

Project Report

December 6, 2024

Contents

1 Introduction

With the rapid development of digital threats against data security, creating robust, scalable database management systems (DBMSs) to handle sensitive information becomes increasingly important. Our project aims to develop a MariaDB-based database that is an open-source relational DBMS, and we aim to investigate its architecture to assess its effectiveness at sensitive data protection, with a focus on hashing and scalability. Being based on advanced features of MariaDB, our project aims to design a secure database solution in a distributed way that can respond to modern standards of security to demonstrate a highly available and scalable solution.

MariaDB was released in 2009 as a backward-compatible fork of MySQL. Since then, it has evolved into a competitive database management system. Its development is powered by an active open-source community led by Michael Widenius, the original founder of MySQL. It provides improved performance over MySQL, advanced replication capabilities, and robust security features such as Role-Based Access Control, data encryption, and comprehensive audit logging, making it an excellent choice for managing sensitive information at scale. When integrated with proxy solutions like MaxScale, MariaDB deployments can leverage enterprise-ready features like automated failover and intelligent load-balancing. . This not only secures sensitive data but also solves major issues that modern distributed systems face.

The importance of this project surpasses that of an academic exploration only. This art of securing sensitive information is increasingly important in the face of increasing data breaches, especially with the shift to remote work that has resulted from the COVID-19 pandemic. There was an incredible 4,379% increase in stolen records between 2015 and 2020 (Deloitte, 2021). Cybercriminals attack vulnerable networks and devices, such as IoT systems, to gather sensitive information often associated with high levels of privacy and security risks.

In this regard, password and sensitive data security using a secure database system becomes imperative at both enterprise and individual levels. With this project, we expand our understanding of the application of secure, scalable database management systems in contexts where data sensitivity and high availability are crucial.

Beyond being an academic project/assignment, our project exposes us to practical insights into real-life applications of MariaDB for sensitive data storage in real-world scenarios.

2 Body

2.1 Architectural Analysis of MariaDB

MariaDB is an open-source relational database management system (RDBMS) that evolved from MySQL. Known for its pluggable storage engine architecture, MariaDB allows modular customization to meet diverse use cases. This architectural analysis explores MariaDB's functionality, security, scalability, and ACID compliance in detail. More importantly, design principles, data flow mechanisms, and scalability are needed to meet the criteria of the project report.

Maria DB's Design Principles

MariaDB's architecture is a multi-layered system designed to provide modularity, scalability, performance, and security for diverse database applications. By and large, it can be divided into the Storage engine (the backbone of data handling), the SQL layer for operations, data flow, replication and clustering at scale, modular API and extensibility.

The SQL Layer: Command Center for Data Operations

At the top of MariaDB's architecture lies the SQL layer, which acts as the interface between users and the database. This layer is responsible for:

Query Parsing and Validation: Incoming SQL commands are parsed, validated for syntax, and checked against schema constraints. This component breaks down the SQL query into smaller parts (tokens) and checks the syntax and validity. It also ensures the query complies with the database schema and user permissions.

Optimization: The **cost-based optimizer** evaluates multiple query execution paths, choosing the most efficient one to minimize resource usage. It chooses the strategy with the lowest estimated "cost" in terms of system resources (e.g., CPU, memory, disk I/O).

Execution Coordinator: The execution coordinator in MariaDB (and other RDBMS systems) is the component of the SQL layer responsible for transforming the optimized query plan into actionable tasks and directing these tasks to the appropriate subsystems (primarily the storage engines). It acts as a central dispatcher, ensuring that every part of the query is executed correctly and efficiently. Once the execution plan is determined, this component sends specific instructions to the storage engine responsible for handling the data.

API Interfacing (programming language interface): This is the language through which the user interacts with the database. MariaDB supports diverse client interfaces, such as JDBC, ODBC, and, more commonly, native MySQL protocols, to accommodate various programming languages and tools.

Result Formatter: After the storage engine processes the query, the SQL layer formats the results into a structured output (e.g., rows of data) to send back to the client application.

Execution Coordinator: Once the execution plan is determined, this component sends specific instructions to the **storage engine** responsible for handling the data. For example, if the query is transactional, it is routed to InnoDB; if it is analytical, it might be routed to ColumnStore.

Result Formatter: After the storage engine processes the query, the SQL layer formats the results into a structured output (e.g., rows of data) to send back to the client application.

The SQL layer in MariaDB orchestrates query processing by parsing, validating, and optimizing SQL commands. However, it does not directly handle the physical storage or retrieval of data. Instead, it relies on MariaDB's pluggable storage engine architecture to execute these tasks. Storage engines, such as InnoDB, take over at this stage, performing the essential operations of managing data on disk, enforcing constraints, and ensuring durability.

Maria's Pluggable Storage Engine Architecture

A **storage engine** in MariaDB is the component responsible for the physical storage, retrieval, and management of data. Each engine operates independently, providing specific functionalities tailored to different use cases. The default engine, **InnoDB**, is optimized for transactional data and is fully ACID-compliant. InnoDB, as the default storage engine, plays a pivotal role in ensuring MariaDB's robustness and reliability. It manages data storage in a transactional context, adhering to ACID principles to guarantee data integrity even in complex, high-concurrency environments. MariaDB knows very little about creating or populating a table, reading from it, and building proper indexes and caches. It assigns these operations to be handled by a plugin known as a storage engine. The default storage engine MariaDB uses is InnoDB, which is a plugin used for transactional data.

MariaDB's pluggable storage engine architecture enhances scalability by allowing different tables to use storage engines optimized for their specific workloads. This flexibility ensures that MariaDB can scale effectively across diverse applications.

InnoDB is ideal for transactional workloads that require ACID compliance and high concurrency, making it suitable for scaling OLTP (Online Transaction Processing) systems.

ColumnStore, on the other hand, excels in scaling OLAP (Online Analytical Processing) workloads by enabling distributed query processing across multiple nodes. ColumnStore's columnar format allows MariaDB to handle large-scale aggregations and analytics efficiently.

This adaptability ensures that MariaDB can scale to meet the demands of both transactional and analytical systems, often within the same deployment.

InnoDB's design includes advanced features like row-level locking for efficient multi-user access and a buffer pool that caches frequently accessed data, significantly enhancing performance by

reducing reliance on slower disk I/O. Furthermore, mechanisms like the double write buffer and adaptive flushing ensure data consistency and durability, even during unexpected crashes or power failures.

In the broader system, InnoDB integrates seamlessly with MariaDB's pluggable architecture, enabling it to handle a wide range of transactional workloads while allowing other engines, such as ColumnStore or Aria, to optimize for analytics or non-transactional needs. This modular approach differentiates MariaDB from monolithic database systems, offering unparalleled flexibility and efficiency across diverse applications.

The modular design enables critical security components, such as authentication and data encryption, to function independently from the core database logic, reducing attack vectors. Storage engines like InnoDB provide tailored solutions for specific workloads, supporting advanced features such as **Role-Based Access Control (RBAC)** and data encryption, ensuring both operational continuity and robust security. MariaDB also integrates high-availability features like replication and clustering, offering protection against data loss or downtime.

MariaDB also enhances its security through query optimization mechanisms (e.g., LIMIT ROWS EXAMINE and Big DELETEs) and monitoring of query execution to prevent denial-of-service (DoS) attacks. Features like InnoDB’s buffer pool caching and file-level encryption safeguard data at rest and during processing. Transport Layer Security (TLS) is used to secure replication and ensure encrypted communication between master and replica servers.

The pluggable storage engine architecture in MariaDB ensures that data storage and retrieval are optimized for specific workloads, with engines like InnoDB providing transactional support and ColumnStore enabling efficient analytics. However, efficient storage alone is not sufficient for handling the demands of modern applications, especially those requiring high availability and scalability. To address these needs, MariaDB incorporates advanced scalability mechanisms and replication models, including Galera Cluster, which ensures consistency and fault tolerance across distributed systems. These features work in tandem with load-balancing tools like MaxScale, allowing MariaDB to distribute workloads intelligently and maintain optimal performance in high-demand environments.

Scalability

Scalability is a cornerstone of MariaDB’s architecture, enabling the system to efficiently manage increasing workloads by leveraging its modular design, diverse storage engines, and advanced replication systems. MariaDB achieves scalability through two key approaches: **vertical scalability**, which focuses on maximizing the capacity of a single server, and **horizontal scalability**, which involves distributing workloads across multiple servers. Together, these methods ensure MariaDB can adapt to a wide range of application demands.

Vertical Scalability: Maximizing Single-Node Efficiency

Vertical scalability involves enhancing the performance of a single database node by optimizing its use of resources such as CPU, memory, and disk I/O. MariaDB achieves this through several mechanisms integrated within its architecture:

- **SQL Layer Optimizations**:
    - The SQL layer ensures efficient query execution by leveraging a **cost-based optimizer**, which evaluates multiple execution strategies and selects the one with the lowest resource cost. For example, indexed queries avoid full table scans, reducing disk I/O and improving response times.
    - Features like **query caching**, which stores the results of frequently executed queries, prevent unnecessary recomputation and accelerate subsequent query execution.
- **InnoDB's Role**:
    - The **InnoDB storage engine** is central to vertical scalability. Its **buffer pool** caches frequently accessed data and indexes in memory, minimizing the need for disk operations. This is particularly beneficial for read-intensive workloads where latency is critical.
    - InnoDB also supports **row-level locking**, enabling multiple concurrent transactions to update different parts of a table without contention, thereby increasing throughput under high-concurrency workloads.
    - Advanced features like **adaptive flushing** and the **doublewrite buffer** optimize disk writes, ensuring consistent and reliable performance.

By combining these optimizations, MariaDB ensures that individual nodes can handle larger datasets and more simultaneous connections, making vertical scalability a key strength for transactional workloads.

Horizontal Scalability: Distributing Workloads Across Nodes

Horizontal scalability, in contrast, involves adding more nodes to a database system to distribute workloads and increase capacity. This approach is essential for applications with demands that exceed the capabilities of a single node, particularly when both read and write operations must scale efficiently.

- **Asynchronous Replication**:
    - In a traditional **master-slave replication setup**, a single master node handles all write operations, while slave nodes replicate the data asynchronously. This setup is effective for **read scalability**, as read queries can be directed to the slave nodes, reducing the load on the master.

    - However, the asynchronous nature of this model introduces **replication lag**, making it less suitable for applications requiring real-time consistency.
- **Synchronous Replication with Galera Cluster**:
    - For applications needing both read and write scalability with strong consistency guarantees, MariaDB's **Galera Cluster** provides a robust solution. Galera enables **multi-master synchronous replication**, where all nodes in the cluster can handle both reads and writes simultaneously. This eliminates the single point of contention inherent in master-slave setups.
    - Galera's **write-set replication mechanism** ensures data consistency by broadcasting each write operation to all nodes, which validate and apply the changes in the same order. This makes Galera an excellent choice for mission-critical workloads such as financial systems and real-time analytics.

While synchronous replication introduces potential latency due to network communication between nodes, Galera mitigates this with **quorum-based decision-making**, allowing a majority of nodes to confirm transactions without requiring input from every node. This balance of consistency and performance makes Galera highly effective for geographically distributed clusters.

Load Balancing and Query Distribution

MariaDB enhances horizontal scalability with intelligent **load-balancing** mechanisms that distribute queries across nodes to optimize performance and resource utilization. Tools like **MariaDB MaxScale** act as query routers, managing traffic distribution, failover, and query routing:

- In master-slave setups, MaxScale directs write queries to the master node and evenly distributes read queries among the slave nodes, ensuring no single node becomes a bottleneck.
- In Galera Clusters, where all nodes can handle reads and writes, MaxScale intelligently routes queries based on node load and proximity, minimizing latency and ensuring efficient utilization of cluster resources.

Additionally, MariaDB supports **query parallelization**, which splits large queries into smaller subqueries that can be executed concurrently across multiple threads or nodes. This significantly improves performance for computationally intensive operations like analytics and reporting.

Fault Tolerance and Scalability

MariaDB tightly integrates scalability with **fault tolerance** to ensure the database remains reliable even as workloads and nodes increase. The Galera Cluster's **automatic failover** mechanism ensures that if a node fails, the remaining nodes reconfigure themselves to maintain

availability without disruption. This seamless recovery process ensures that scaling out does not compromise system stability.

To further enhance resilience, Galera uses a **Group Cache (GCache)** to store recent write sets locally on each node. This allows new or recovering nodes to synchronize with the cluster without requiring a full state transfer, reducing recovery time and network overhead. Executed queries, allow MariaDB to avoid recomputation and serve cached results directly

Building on MariaDB's scalability features, the database's data flow mechanisms are designed to ensure the efficient and secure movement of data throughout its architecture, supporting both single-node and distributed environments. These mechanisms govern how queries are processed, how the SQL layer communicates with storage engines, and how replication ensures data consistency across nodes.

Data Flow Mechanisms

MariaDB's architecture ensures efficient and secure data flow through its SQL layer, storage engines, and replication systems. Key data flow mechanisms include:

1. **SQL Layer**:
    - The SQL layer handles query parsing, optimization, and execution. It ensures only valid queries are processed by applying syntax validation and constraint enforcement.
    - Features like **query optimizations (e.g., LIMIT ROWS EXAMINE and Big DELETEs)** and monitoring prevent resource-heavy queries from overloading the system.
2. **Storage Engine Interaction**:
    - Storage engines like InnoDB process table creation, indexing, and data retrieval. Data flows from the SQL layer to the storage engines, which handle disk I/O, caching, and encryption.
3. **Replication Data Flow**:
    - Replication is handled through asynchronous or synchronous models:
        - **Asynchronous Replication**: Master-slave setups replicate changes with minimal overhead but may face replication lag.
        - **Synchronous Replication**: Galera Cluster enables consistent replication across nodes, ensuring data integrity in distributed setups.
4. **Caching and Buffer Pools**:
    - InnoDB's **buffer pool** caches frequently accessed data and indexes in memory, reducing disk reads and improving query response times. The buffer pool's **state saving and restoration** further optimize data flow during restarts or failures.
5. **Disk I/O and Encryption**:

    - Disk I/O operations leverage file-level encryption to secure data at rest. Data flow during read and write operations remains encrypted, ensuring security without significant performance degradation.
6. **Network Communication**:
    - Data transfers between nodes in replication setups are encrypted using **TLS**, securing the flow of data across networks and protecting against man-in-the-middle attacks.

MariaDB's data flow mechanisms not only enhance its scalability but also ensure robust security and efficient data management across its architecture. These mechanisms are tightly integrated with features like query optimization, caching, and replication to protect data integrity and prevent vulnerabilities such as denial-of-service (DoS) attacks. By securing data at rest through encryption and safeguarding data in transit with protocols like Transport Layer Security (TLS), MariaDB establishes a strong foundation for secure operations. Furthermore, integrating replication mechanisms, including Galera Cluster, guarantees consistent and scalable replication, addressing challenges such as replication lag and single points of failure.

Complementing these mechanisms is MariaDB's commitment to ACID compliance, which underpins its reliability and transactional integrity.

ACID Compliance

Lastly ACID compliance is as well fundamental to MariaDB's reliability and transactional integrity. Its default storage engine, InnoDB, ensures full ACID compliance, which encompasses:

1. **Atomicity**: Transactions are executed fully or not at all. If an error occurs during a transaction, the system rolls back to its previous state, maintaining data integrity.
2. **Consistency**: Transactions take the database from one valid state to another. Constraints are enforced at the SQL parsing and optimization layer, ensuring data remains consistent.
3. **Isolation**: Concurrent transactions operate independently. InnoDB supports multiple isolation levels, such as **Read Committed** and **Serializable**, to prevent phenomena like dirty reads and phantom reads.
4. **Durability**: Once a transaction is committed, the changes are permanently recorded, even in the event of a system crash. This is achieved through mechanisms like InnoDB's **doublewrite buffer** and **adaptive flushing**.

MariaDB's architecture is a harmonious integration of modular design, scalability, secure data flow mechanisms, and a strong commitment to reliability through ACID compliance. Its multi-layered system—encompassing the SQL layer, pluggable storage engines, scalability

mechanisms, and data flow processes—ensures MariaDB can adapt to the evolving demands of modern database applications.

The **SQL layer** serves as the command center, parsing, validating, and optimizing queries before delegating tasks to the underlying storage engines. MariaDB's **pluggable storage engine architecture**, exemplified by InnoDB, provides a robust foundation for transactional workloads while allowing for specialized engines like ColumnStore to handle analytical processes. This flexibility ensures MariaDB is well-suited for both transactional and analytical systems.

MariaDB's **scalability** is achieved through a dual approach: vertically by optimizing single-node performance, and horizontally by distributing workloads across nodes using replication and clustering models such as Galera Cluster. These scalability features are seamlessly integrated with **data flow mechanisms** that manage query processing, caching, disk I/O, and secure data transfers. By incorporating tools like Transport Layer Security (TLS) for encryption and replication models tailored to different needs, MariaDB ensures secure and efficient operations across its architecture.

Underpinning all these capabilities is **ACID compliance**, which guarantees the reliability and integrity of MariaDB's transactional operations. With mechanisms to enforce atomicity, consistency, isolation, and durability, MariaDB delivers a system that is resilient to failures and able to maintain data consistency under concurrent and high-demand workloads.

Together, these components create a cohesive and adaptable database system capable of meeting diverse application requirements, from mission-critical financial systems to large-scale analytical platforms. MariaDB's blend of modularity, performance optimization, and security makes it a powerful and reliable choice for developers and enterprises alike.

Building on its robust architectural design, MariaDB's functionality and features showcase its ability to translate this architecture into practical, high-performance solutions for diverse database needs.

2.2 Functionality and Features Investigation

MariaDB encompasses many functionalities and features that are useful to a password management system, particularly when integrated with Galera Cluster. Galera Cluster is a powerful software package for Linux Operating Systems that allows for the setup of clusters within MariaDB. Leveraging the InnoDB storage engine and its fork XtraDB, Galera Cluster facilitates synchronous data replication across all nodes (Galera cluster for mariadb 2020). This is done through broadcasting the write set associated with the transaction to every node in the cluster (About Galera replication 2024). This ensures that data updates or changes are consistent throughout all primary and secondary storage units, significantly enhancing data reliability. Galera's replication is sometimes called virtually synchronous, meaning that while the

replication process is logically synchronous, the writing and committing to the tablespace happens independently (Overview of galera cluster 2024). Moreover, Galera Cluster supports automatic failover, which strengthens the system's resilience by maintaining operations even in the event of node failure. In addition to Galera Cluster, MariaDB Enterprise Server provides pluggable storage engines, which offer flexibility and functionality. Among these is InnoDB, the default storage engine, which plays a pivotal role in managing sensitive data efficiently by ensuring high performance and transaction reliability, making it an ideal choice for a secure and universal data storage system. InnoDB is successful with these approaches through its compatibility with resources such as Galera Cluster, MariaDB replication, and MariaDB Enterprise Server, alongside its own features and resources (MariaDB, 2024)

MariaDB further excels in query optimization, with features designed to maximize performance for diverse workloads. InnoDB, in particular, is adept at optimizing read and write operations by utilizing the asynchronous I/O subsystem (native AIO) on Linux (MySQL 8.4 Reference Manual, 2024). This subsystem performs read-ahead and write requests for data file pages, enabled by default. By leveraging native AIO and tuning the I/O scheduler, InnoDB's use is able to reduce latency, increase throughput, and optimize performance for the specific workload. Furthermore, InnoDB is ACID-compliant, meaning it guarantees the core principles of atomicity, consistency, isolation, and durability (MariaDB, 2023). These properties ensure that all transactions are executed reliably and in a timely manner, maintaining the integrity of the system, all the while ensuring query optimization.

Beyond data storage methods, and query optimization, MariaDB further implements comprehensive security measures to protect its infrastructure and data. For instance, its end-to-end security strategy includes access control and monitoring systems. So, any team using the database management system (DBMS) must pass through an extensive multi-factor authentication process before being authorized to perform maintenance or support operations (MariaDB Trust Center, 2024). This reduces the risk of unauthorized access and strengthens the overall security framework. In addition, MariaDB's infrastructure showcases a series of safety measures available within the DBMS. MariaDB Enterprise Server includes replication and clustering for High Availability (HA), namely through the implementation of Galera Cluster (MariaDB Trust Center, 2024). They further have the MariaDB MaxScale proxy available, enhancing reliability by enabling load balancing, query routing, and failover management (MariaDB Trust Center, 2024).

To ensure the reliability of its systems in production environments, MariaDB undergoes rigorous quality assurance testing, making the DBMS a reliable choice for managing sensitive data in a password management system (MariaDB Trust Center, 2024).

2.3 Case Study Findings

Our investigation of MariaDB with Galera Cluster demonstrates its effectiveness as a solution for sensitive data storage, combining robust scalability, comprehensive security, and reliable replication capabilities. Implementation and analysis revealed both the system's strengths and the practical challenges of deploying it in modern cloud environments.

Galera Cluster's multi-master architecture fundamentally enhances system reliability by enabling read/write operations across all nodes, effectively eliminating single points of failure (Codership, 2021). While MariaDB's MaxScale offers sophisticated load-balancing capabilities, our implementation deliberately chose a simpler path: an application-level load balancer integrated directly into the backend code. This decision stemmed from practical considerations about future cloud deployment costs and complexity. MaxScale would have introduced additional infrastructure requirements and operational overhead, whereas our application-level approach achieved the necessary scalability while maintaining architectural simplicity. The security architecture implements multiple reinforcing layers of protection. Beyond the foundational use of environment-variable-based credentials for database connections, the system leverages MariaDB's native node-to-node encryption for secure cluster communication. We integrated the Zod validation library specifically to ensure comprehensive input validation and parameterized queries, providing robust protection against SQL injection attacks. These security measures work in concert with Galera Cluster's synchronous replication to maintain data integrity even during node failures.

Our deployment in a serverless environment surfaced significant technical challenges, particularly around connection management. The traditional connection pooling approach proved incompatible with MariaDB's Node.js package in Vercel's serverless functions, necessitating a dynamic connection strategy. While creating and dropping connections introduces some performance overhead, this approach successfully maintains system reliability within the constraints of a serverless architecture. The containerized deployment of Galera Cluster presented additional configuration challenges. Unlike MySQL, MariaDB lacks extensive documentation and community examples for Docker-based cluster deployments. This scarcity of resources, combined with limited options for free virtual machine hosting capable of supporting cluster requirements, significantly influenced our architectural decisions. These constraints ultimately shaped a more practical implementation.

The system's extensibility, enabled by MariaDB's pluggable storage engines (MariaDB Corporation, 2022) and supported by an active open-source community (MariaDB Foundation, 2023), provides crucial flexibility for future adaptation. This architectural flexibility proved particularly valuable given our need to accommodate both serverless computing constraints and container orchestration requirements. Our findings indicate broad applicability across sectors handling sensitive data. The architecture particularly suits password management systems, where

consistent replication ensures data reliability. Healthcare and financial services benefit from the robust security compliance features, while e-commerce platforms can leverage the system's scalability for dynamic transaction processing.

The implementation process revealed crucial insights about deploying the MariaDB Galera Cluster in contemporary cloud environments. The necessity of balancing theoretical capabilities against practical constraints—particularly around serverless architecture limitations and container orchestration complexity—emerged as a central theme. While our implementation successfully achieved its objectives, the challenges encountered provide valuable guidance for similar deployments in production environments. This investigation ultimately validates MariaDB with Galera Cluster as a viable solution for sensitive data management, while highlighting the practical complexities of implementing enterprise-grade database features in modern cloud architectures. The balance between advanced capabilities and deployment simplicity proves especially crucial in resource-constrained environments, where architectural decisions must carefully weigh feature richness against operational complexity.

3 Conclusion and Future Work

In this project, we assessed MariaDB as a secure, scalable, and robust DBMS for sensitive data storage. We found MariaDB to be an excellent DBMS that satisfied the critical requirements surrounding the protection and proper management of sensitive information.. With pluggable storage engines and modularity for better optimization of its performance, this relational DBMS stands out and proves itself useful in applications regarding password management and sensitive data handling.

Some of the possible future works could be performance testing/tuning of MariaDB for really specific real-world scenarios.

References

Codership. (2021). Galera Cluster for MySQL: High Availability and Scalability. https://galeracluster.com/2024/01/galera-cluster-for-mysql-5-7-44-and-mysql-8-0-35-released/

Codership Ltd. (2024). Overview of galera cluster. Galera Cluster for MySQL. https://galeracluster.com/library/documentation/overview.html

Deloitte. (2021, May 25). The growing threat of data breaches. Deloitte Canada. https://www2.deloitte.com/ca/en/pages/risk/articles/growing-threat-of-data-breaches.html

IONOS editorial team. (2020, October 7). Galera cluster for mariadb. IONOS Digital Guide. https://www.ionos.ca/digitalguide/hosting/technical-matters/mariadb-galera-clusters/

MariaDB. (2023, June 3). What is acid database compliance? https://mariadb.com/resources/blog/acid-compliance-what-it-means-and-why-you-should-care/

MariaDB. (2024). About Galera replication. MariaDB KnowledgeBase. https://mariadb.com/kb/en/about-galera-replication/#:~:text=In%20MariaDB%20Cluster%2C%20the%20Server,native%20MariaDB%20in%20most%20cases.

MariaDB. (2024). Open-source database (RDBMS) for the Enterprise. https://mariadb.com/docs/server/storage-engines/innodb/

MariaDB Corporation. (2022). MariaDB Platform Features. https://mariadb.com/kb/en/about-mariadb-software/

MariaDB Corporation. (2024). Query optimizations. MariaDB. Retrieved December 5, 2024, from https://mariadb.com/kb/en/query-optimizations/

MariaDB Corporation. (2024). Replication with secure connections. MariaDB. Retrieved December 5, 2024, from https://mariadb.com/kb/en/replication-with-secure-connections/

MariaDB Foundation. (2023). MariaDB Community Resources. https://mariadb.com/kb/en/community/

MariaDB Trust Center. (2024). MariaDB. https://mariadb.com/trust/

MySQL 8.4 Reference Manual :: 10.5.8 optimizing innodb disk I/O. MySQL. (2024). https://dev.mysql.com/doc/refman/8.4/en/optimizing-innodb-diskio.html#:~:text=InnoDB%20uses%20the%20asynchronous%20I,influence%20on%20I%2FO%20performance.

Oracle Corporation. (2023). InnoDB Storage Engine Documentation. https://dev.mysql.com/doc/refman/8.0/en/innodb-storage-engine.html