A Report on

MariaDB

Topic
Walmart Retail Data Analysis
Using MariaDB

Submitted By

Yesha Desai, Rithika Kandimalla,
Priyanka Prabakarrao, Sai Surya Prakash
Tamminedi

Under The
Guidance Of

Prof. Clinton Daniel

1. Introduction about MariaDB

MariaDB Server, an open-source relational database, stands out as a highly favored choice. Developed by the original MySQL creators, it's committed to remaining open source. It enjoys widespread adoption in numerous cloud platforms and comes as the default option in many Linux distributions. MariaDB's core principles are centered around delivering top-notch performance, unwavering stability, and a commitment to open-source principles. The MariaDB Foundation guarantees that contributions will be accepted based on their technical merit.

MariaDB is designed to be a drop-in replacement for MySQL, which means it is compatible with MySQL's APIs and command syntax while offering some additional features and improvements. Here is a detailed description of MariaDB:

- **Open Source and Licensing**: MariaDB is released under the GNU General Public License (GPL), which means it is open source and freely available for anyone to use, modify, and distribute. This open-source nature ensures that MariaDB remains accessible and transparent.

- **History and Fork from MySQL**: MariaDB was created by Michael "Monty" Widenius, the original author of MySQL, in response to concerns about the future of MySQL's open-source status under Oracle Corporation. The first version of MariaDB was released in 2009, and it has since continued to evolve as a separate but compatible database system.

- **Compatibility with MySQL**: MariaDB strives to maintain compatibility with MySQL to ensure a seamless transition for users. It can be used as a drop-in replacement for MySQL. Most MySQL client libraries and applications can work with MariaDB without modification.

- **Features and Improvements:** MariaDB has introduced numerous features and improvements over MySQL, such as the Aria storage engine, thread-pooling for better performance, and additional SQL functions. It also includes enhancements for high availability and scalability, making it suitable for both small and large-scale applications.

- **Storage Engines:** MariaDB supports multiple storage engines, including InnoDB (the default storage engine in MySQL and MariaDB), Aria, TokuDB, and more. These storage engines offer different features and performance characteristics, allowing users to choose the one that best suits their needs.

- **Security Feature:** MariaDB provides security features like encryption at rest and in transit, user account management, and role-based access control (RBAC). It also offers support for external authentication mechanisms and LDAP integration.

- **Replication:** MariaDB offers advanced replication features, including multi-source replication and Galera Cluster integration for synchronous multi-master replication. This makes it suitable for building high-availability and distributed database

architectures.

- **Performance Optimization:** MariaDB has a range of performance optimizations, including query optimization, improved storage engine performance, and multi-threading enhancements. It supports features like the "Query Cache" and the "Optimizer Switch" plugin to fine-tune performance.

- **Community and Support:** MariaDB has an active and growing community of users and developers who contribute to its development. It is supported by various organizations and has a well-established ecosystem, including documentation, forums, and commercial support options.

- **Cross-Platform:** MariaDB is available on various operating systems, including Linux, Windows, macOS, and more. It can be used in a wide range of environments, from personal projects to enterprise-level applications.

- **Extensions and Plugins:** MariaDB supports extensions and plugins that allow users to add custom functionality and features to the database system.

- **Licensing and Business Model:** While MariaDB is open source, there are also enterprise editions and commercial support options available for organizations that require additional features and support.

2. 6 Ways To Do More With MariaDB

MariaDB and MySQL are both general-purpose databases. However, only MariaDB adheres to the open-source mission while providing a consistent set of advanced features and functionality. MariaDB can deploy anywhere: on-prem or in any private, public or hybrid cloud.

i. Oracle Database compatibility

MariaDB is the only open source database compatible with Oracle Database data types, sequences, PL/SQL stored procedures and more, making it possible to "lift and shift" without having to modify database schemas and rewrite store procedures.

ii. High availability and scalability

Maintain continuous availability and hide failures from applications using zero-interruption failover features such as transaction replay. Vertical scale-out through parallel query, read replication and multi-master clustering.

iii. Columnar storage format

MariaDB supports both row and columnar storage. It can be deployed as a data warehouse for interactive, ad hoc analytics or as a hybrid transactional/analytical processing, storing current data in row storage and historical data in columnar.

iv. Federation

MariaDB and MySQL can access tables in other MariaDB/MySQL databases, but only MariaDB can federate heterogeneous databases, including Oracle Database, Microsoft SQL Server and IBM Db2, to consolidate data access and/or simplify database migration.

v. Temporal tables

MariaDB is the only open source database to implement system-versioned, application-time period and bitemporal tables, granting developers the ability to query data based on a previous point in time and DBAs to audit and/or recover data after it was changed.

vi. JSON functionality

MariaDB goes beyond the bounds of a typical relational database with its ability to handle JavaScript Object Notation (JSON)-formatted data. This compatibility enables users to combine structured and semi-structured data.

3. How is MariaDB better than other databases?

MariaDB is often considered better than other databases in certain scenarios due to its features, performance, and open-source nature. Here are some advantages of MariaDB that make it stand out:

Oracle DBMS:
Cost-effectiveness: MariaDB is an open-source DBMS, meaning it's freely available to use and modify, while Oracle DBMS requires expensive licensing fees.
Community support: MariaDB has a large and active community of developers and users who provide support and contribute to its development, while Oracle's support is primarily vendor-driven.

MySQL:
Advanced features: MariaDB offers additional features, such as Galera multi-master clustering and support for columnar storage engines, which MySQL lacks.
Performance: MariaDB's performance can be optimized for specific workloads, while MySQL's performance is more general-purpose.

PostgreSQL:
Simplicity: MariaDB is easier to set up and manage than PostgreSQL, which has a more complex configuration process.
Compatibility: MariaDB is highly compatible with Oracle Database, making it a viable option for Oracle migrations, while PostgreSQL's compatibility with Oracle is limited.

SQLite:
Scalability: MariaDB is designed for large-scale applications with high performance and scalability, while SQLite is primarily intended for lightweight embedded applications.
Feature set: MariaDB offers a broader range of features compared to SQLite, which has a more limited feature set.

IBM Db2 on Cloud:
Cost: MariaDB can be more cost-effective for businesses seeking a high-performance, scalable DBMS without the high licensing fees of IBM Db2 on Cloud.
Ease of use: MariaDB is generally easier to manage and use than IBM Db2 on Cloud, which has a more complex administrative interface.

Microsoft Azure SQL:
Open-source: MariaDB's open-source nature provides greater flexibility and control compared to Microsoft Azure SQL, which is a proprietary DBMS.
Community support: MariaDB benefits from a large and active community of developers and users, while Microsoft Azure SQL's support is primarily vendor-driven.

Amazon Redshift:
Transactional workloads: MariaDB is designed for transactional applications, while Amazon Redshift is optimized for analytical workloads and data warehousing.
Cost-efficiency: MariaDB can be more cost-efficient for transactional workloads compared to Amazon Redshift, which is geared towards larger data analysis tasks.

Redis:
Relational data storage: MariaDB is a relational DBMS, while Redis is an in-memory data store, making them suitable for different types of data and applications.
Acidity: MariaDB guarantees ACID compliance, ensuring data consistency and integrity, while Redis doesn't provide the same level of transaction support.

Google BigQuery:
Transactional workloads: MariaDB is designed for transactional applications, while Google BigQuery is optimized for analytical workloads and data warehousing.
Cost-efficiency: MariaDB can be more cost-efficient for transactional workloads compared to Google BigQuery, which is geared towards larger data analysis tasks.

Google AlloyDB for PostgreSQL:
Cost: MariaDB can be a more cost-effective alternative to Google AlloyDB for PostgreSQL, especially for businesses that don't require the specific performance and scalability features of AlloyDB.
Oracle compatibility: MariaDB's high compatibility with Oracle Database makes it a viable option for Oracle migrations, while Google AlloyDB for PostgreSQL is primarily designed for PostgreSQL applications.

Amazon Aurora:
Cost: MariaDB can be a more cost-effective alternative to Amazon Aurora, especially for businesses that don't require the specific performance and scalability features of Aurora.
Oracle compatibility: MariaDB's high compatibility with Oracle Database makes it a viable option for Oracle migrations, while Amazon Aurora is primarily designed for MySQL applications.

3.1 MariaDB vs MySQL

Feature	MariaDB	MySQL
Performance	Faster	Slower
Scalability	More scalable	Less scalable
Security	More secure	Less secure
Community Support	Larger and more active	Smaller and less active
Licensing	Fully open-source	Dual licensing model
Ease of Upgrading	Backward compatible	Not backward compatible

MySQL and MariaDB are both open-source database technologies. You can use them to store data in a tabular format with rows and columns. MySQL is the most widely adopted open-source database. And it's the primary relational database for many popular websites, applications, and commercial products. MariaDB is a modified version of MySQL. MariaDB was made by MySQL's original development team due to licensing and distribution concerns after MySQL was acquired by Oracle Corporation. Since the acquisition, MySQL and MariaDB have evolved differently. However, MariaDB adopts MySQL's data and table definition files and also uses identical client protocols, client APIs, ports, and sockets. This is intended to let MySQL users switch to MariaDB hassle-free.

	MYSQL	MARIADB
MongoDB-compatible API	No	Yes
Columnar storage	No	Yes
Temporal tables	No	Yes
Oracle database compatibility	No	Yes
Non-blocking backups	No	Yes
Write-anywhere clustering	Yes	Yes
Transaction replay	No	Yes
Secure by default	Yes	Yes

MariaDB is more scalable and offers a higher query speed when compared to MySQL. This makes it good for managing large-sized data. You will also find more features in MariaDB that MySQL does not have, like sequence storage engines and virtual columns. You can also use multiple engines in one table.

HeidiSQL, like many database management tools, may be more permissive and flexible in handling data type conversions implicitly or automatically when querying the database. In contrast, MySQL can be stricter about data type compatibility and may not automatically perform implicit conversions.

3.2 MariaDB vs PostgreSQL

Here are some areas where MariaDB may be considered better than PostgreSQL:

i. **Compatibility with MySQL**: MariaDB was originally created as a fork of MySQL, and it strives to maintain compatibility with MySQL. This means that if you have an existing MySQL database and want to switch to MariaDB, you can often do so with minimal changes to your code and queries. PostgreSQL, on the other hand, has a different syntax and behavior from MySQL, which may require more extensive changes when migrating.

ii. **Performance:** MariaDB has been known for its performance improvements over MySQL, especially in terms of read-heavy workloads. It offers the Aria storage engine, which can be faster for some use cases, and also incorporates various optimization techniques for better performance.

iii. **Licensing:** MariaDB uses the GPL (GNU General Public License) as its open-source license, which is considered more permissive and business-friendly compared to PostgreSQL's PostgreSQL License. This can make it easier to integrate MariaDB into commercial applications without triggering some of the restrictions of PostgreSQL's license.

iv. **Flexibility**: MariaDB offers a **very flexible** approach to data typing. You can automatically correct the data type to match the destination. You can create additional rules, such as whether the system accepts the data instantly or triggers an alert. If you are using many different types of data input, the flexibility MariaDB offers is useful. PostgreSQL does not offer the same degree of flexibility. It has strict data integrity checks in place. If data does not match the destination type, the system shows an error and prevents you from inserting that data.

4. Choosing Transactional over Analytical

MariaDB, like its predecessor MySQL, is primarily designed for transactional (OLTP - Online Transaction Processing) workloads. It excels at handling tasks such as inserting, updating, and deleting records in a database while ensuring data integrity and ACID (Atomicity, Consistency, Isolation, Durability) compliance. It is widely used for applications that require high-speed transaction processing, such as e-commerce websites, content management systems, and various web applications.

For analytical (OLAP - Online Analytical Processing) workloads, which involve complex queries, aggregations, and reporting, MariaDB may not be the optimal choice out of the box. However, MariaDB can be used for analytical purposes to some extent. You can perform analytical queries on the data stored in MariaDB, but it may not be as efficient or feature-rich as dedicated analytical databases like Apache Hive, Apache Spark, or data warehousing solutions like Amazon Redshift, Google BigQuery, or Snowflake.

5. Project Title- **Walmart Retail Data Analysis Using MariaDB**

5.1 About the Dataset:

To assess the distribution of store sizes across different store types ('A', 'B', 'C') within Walmart and to understand the influence of markdowns on sales during holiday and non-holiday periods, enabling data-driven decisions to optimize store sizes and markdown strategies for increased profitability.

Walmart Dataset:

- Store (int) - Store number
- Date (Date)- Week
- Temperature (decimal) - Average temperature in the region
- Fuel_Price (decimal)- Cost of fuel in the region
- MarkDown's (varchar) - Anonymized data related to promotional markdowns that Walmart is running.
- CPI (varchar) - The consumer price index
- Unemployment (varchar) - The unemployment rate
- IsHoliday(varchar) - Whether the week is a special holiday week

Store Dataset

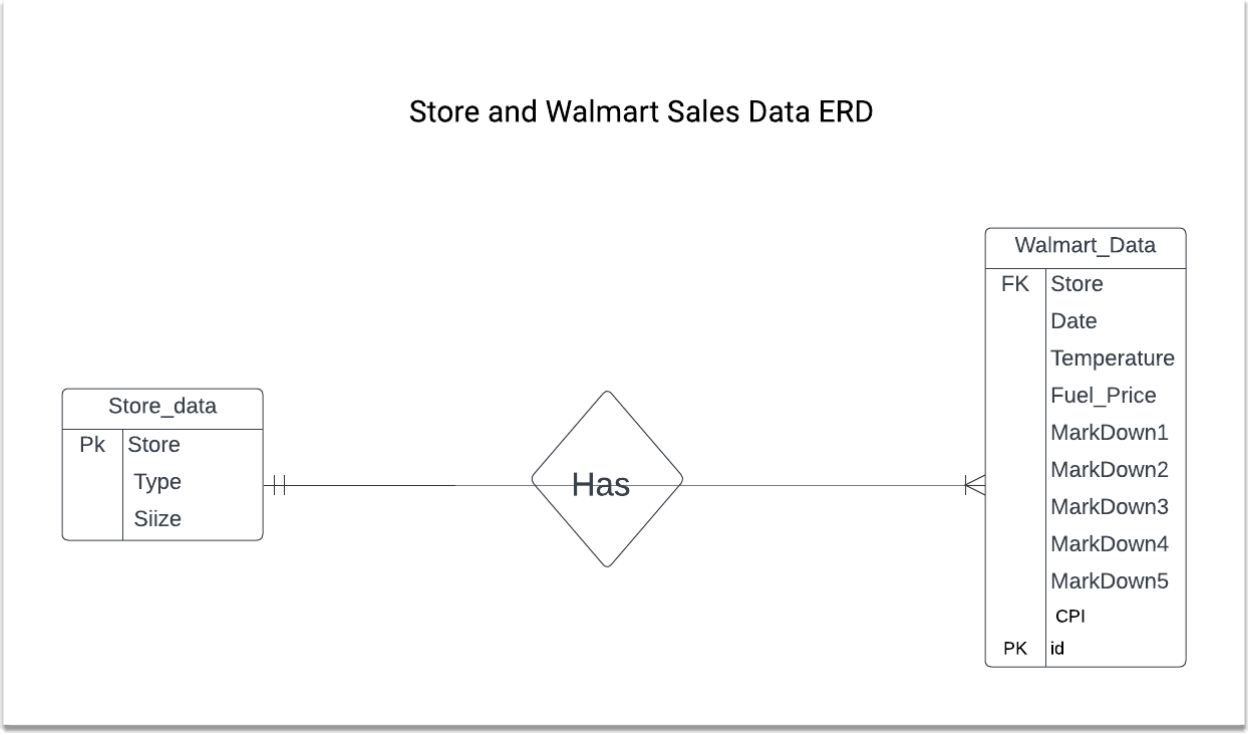
- Store(int) - Stores numbered from 1 to 45
- Type (varchar) - Store type has been provided, there are 3 types — A, B and C.
- Size (int) - Stores size has provided

5.2 Our Product is **Transactional** for the following reasons:

Transactional Database Design:

- Data Updates: This database appears to be designed for handling transactional data, such as sales, store information, and markdowns. It allows for frequent data updates, including inserts, updates, and deletes.
- Primary Key and Foreign Key Constraints: The use of primary keys and foreign key constraints (e.g., fk_walmart_store) suggests that data integrity and transactional consistency are important. These constraints help maintain the relationships between tables, which is a characteristic of transactional databases.
- Data Validation: The database includes data validation in the form of a check constraint (i.e., chk_positive_fuel_price) to ensure that certain data rules are met. Data validation is a common feature in transactional databases to maintain data accuracy and consistency.
- Insertion of New Records: You are inserting new data, such as sales and store information, into the database. Transactional databases are typically used to record day-to-day operations and are well-suited for handling insertions of new records.

5.3 Entity- Relationship Diagram of "Walmart sales forecast"



In the simplified Entity-Relationship Diagram (ERD) provided earlier for the 'Store_data' and 'Walmart_data' tables, we can describe the cardinality and modality of the relationships as follows:

- ✓ Relationship between 'Store_data' and 'Walmart_data':

Cardinality: Many-to-One (M:1)

- Each record in the 'Walmart_data' table (e.g., sales data) is associated with one and only one store from the 'Store_data' table.
- Each store in the 'Store_data' table can be associated with multiple records in the 'Walmart_data' table.

Modality: Mandatory from the 'Walmart_data' side

- Every record in the 'Walmart_data' table must be associated with a store in the 'Store_data' table (due to the Foreign Key constraint). There can be no "orphaned" records in 'Walmart_data'.

This means that for each entry in the 'Walmart_data' table (sales data), there is a corresponding store in the 'Store_data' table, and each store can have multiple associated sales records.

5.4 CRUD operations performed on our project:

We have created two tables in a database, ‘store_data’ and ‘walmart_data’, and established a foreign key relationship between them. This is a common practice in relational database management to ensure data integrity.

Creating Tables:

Table 1 is named ‘store_data’ and has three columns: Store (as the primary key), Type, and Size. It will store information about different stores, including their type and size.

Table 2 is named walmart_data and has multiple columns to store data related to Walmart stores. The id column is used as an auto-incremented primary key. The Store column will be used as a foreign key to reference the Store column in the store_data table. The other columns will store data like date, temperature, fuel price, markdowns, CPI, unemployment, and whether a day is a holiday.

This SQL statement adds a foreign key constraint called ‘fk_walmart_store’ on the Store column in the ‘walmart_data’ table, which references the Store column in the ‘store_data’ table. This constraint ensures that the values in the Store column of ‘walmart_data’ must exist in the Store column of the ‘store_data’ table, maintaining data integrity and preventing the insertion of invalid data.

Inserting data in the tables:

We have inserted data into the store_data and walmart_data tables and then selecting specific records from them.
We inserted four records into the store_data table. Each record consists of three values: Store, Type, and Size, provided values for Stores 46, 47, 48, and 49 with their corresponding types and sizes.

We inserted data into the walmart_data table using the INSERT INTO statement:
We inserted two records into the walmart_data table. Each record contains values for various columns, including Store, Date, Temperature, Fuel_Price, and more. The first record corresponds to Store 46, and the second record corresponds to Store 27.

Updating the table:
We performed two UPDATE operations to modify specific records in the store_data and walmart_data tables.

Update1: This UPDATE statement modifies the store_data table by changing the TYPE value to 'B' for the record where the Store column equals 46. This means that we have updated the type of Store 46 from 'A' to 'B'.

Update2: In this UPDATE statement, we have modified the walmart_data table by changing the Temperature value to 74.5 for a specific record. To identify the record to be updated, we have specified two conditions:
Store = 1: This condition restricts the update to records where the Store column is equal to 1.
Date = '2010-02-05': This condition further narrows down the selection by looking for records

with a specific date ('2010-02-05').
The UPDATE statement will change the temperature for Store 1 on February 5, 2010, to 74.5. This allows us to update specific data points in the walmart_data table without affecting other records.

These UPDATE statements are essential for modifying existing data in a database when changes or corrections are necessary. It ensures that your database remains up-to-date and accurate.

Delete:
The DELETE statement to remove specific records from the store_data and walmart_data tables.

1) DELETE statement 1: removes a record from the store_data table where the Store column is equal to 48. It effectively deletes the information for Store 48 from the store_data table.

2) DELETE statement 2: we removed a record from the walmart_data table by specifying two conditions:

Store = 2: This condition restricts the deletion to records where the Store column is equal to 2.
Date = '2012-08-10': This condition further narrows down the selection by specifying the date as '2012-08-10'.
The DELETE statement will delete the record for Store 2 on August 10, 2012, from the walmart_data table. This operation removes the specified data entry from the table.

5.5 Business Questions:

1. What is the total store size for each store type (e.g., 'A', 'B','C') ?
Ans.
It can provide insights into the distribution of store sizes among different types, which can be valuable for inventory management and resource allocation decisions.

```
127
128     --  total store size for each store type
129  SELECT
130      Type,
131      SUM(Size) AS TotalStoreSize
132  FROM store_data
133  GROUP BY Type;
134
```

store_data (3r × 2c)

Type	TotalStoreSize
A	3,899,450
B	1,760,242
C	243,250

2. What are the total markdown amounts for all stores during holiday vs non-holiday

periods.
Ans**.**

By grouping the data based on the IsHoliday flag, we can compare the total markdown spending during holiday periods and non-holiday periods for each store. This helps to understand how promotional discounts and markdowns vary during different time frames.

```sql
-- Total markdown amounts for all stores during holiday and non-holiday
SELECT
   Store,IsHoliday,
    SUM(
        COALESCE(CAST(MarkDown1 AS DECIMAL(10, 2)), 0) +
        COALESCE(CAST(MarkDown2 AS DECIMAL(10, 2)), 0) +
        COALESCE(CAST(MarkDown3 AS DECIMAL(10, 2)), 0) +
        COALESCE(CAST(MarkDown4 AS DECIMAL(10, 2)), 0) +
        COALESCE(CAST(MarkDown5 AS DECIMAL(10, 2)), 0)
    ) AS total_markdown
FROM walmart_data
GROUP BY IsHoliday,Store;
```

walmart_data (91r × 3c)

e	IsHoliday	total_markdown
1	FALSE	1,539,571.19
2	FALSE	1,882,503.7
3	FALSE	453,638.19
4	FALSE	2,043,528.02
5	FALSE	542,019.39

6. Conclusion

In conclusion, our report has provided a comprehensive overview of MariaDB and its application in our project Walmart Retail Data Analysis. We have covered various aspects of MariaDB and its advantages over other databases, emphasizing its open-source nature, compatibility with MySQL, performance improvements, features, and community support.

We have explored the choice of MariaDB for transactional workloads and its limitations for analytical purposes, highlighting the need for dedicated analytical databases in such scenarios.

Our project, "Walmart Retail Data Analysis Using MariaDB", involves the analysis of Walmart sales data to make data-driven decisions. We have introduced the dataset, its data types, and an entity-relationship diagram to represent the database structure. Additionally, we have discussed the CRUD operations performed in our project, including table creation, data insertion, updates, and deletion, ensuring data integrity and accuracy.

Finally, we have outlined some business questions that can be addressed using the data collected in the project. These questions encompass Markdown analysis, sales trend estimation, and the impact of holidays on sales and promotions.

In summary, MariaDB plays a crucial role in the success of our project by providing a reliable and efficient database management system for the analysis of Walmart's retail data. It enables us to derive meaningful insights and make informed business decisions based on the collected data.

7. References

- KDD Nuggets Dataset Link:
  https://www.kaggle.com/datasets/aslanahmedov/walmart-sales-forecast?select=train.csv
  https://www.kaggle.com/datasets/aslanahmedov/walmart-sales-forecast?select=stores.csv

- https://mariadb.org/
- https://mariadb.org/documentation/
- https://aws.amazon.com/compare/the-difference-between-mariadb-vs-mysql/#:~:text=MariaDB%20is%20slightly%20faster%20than%20MySQL%20in%20replication%20and%20querying.&text=MySQL%20supports%20super%20read%2Donly,columns%20and%20temporary%20table%20space.

- https://aws.amazon.com/compare/the-difference-between-mariadb-and-postgresql/#:~:text=If%20you%20are%20using%20many,you%20from%20inserting%20that%20data.
- https://mariadb.com/resources/blog/the-place-between-transactions-and-analytics-and-what-it-means-for-you/