3 papers: detailed description of DBMS

1 paper: dataset describe

1 paper: why we have chosen transactional

2 papers: 4) ERD

1-2 papers: 5) CRUD

**A detailed description of the DBMS (MariaDB DBMS):**

**MariaDB DBMS:**

**Overview**

**Definition:**

MariaDB is a free, open-source database system that's very similar to MySQL. It was made by the same folks who created MySQL, and it's designed in a way that you can switch from MySQL to MariaDB without causing big problems for your applications or databases. In simple terms, you can use MariaDB instead of MySQL with minimal fuss.

**Historical Context:**

Michael "Monty" Widenius, the original creator of MySQL, launched MariaDB in 2009 as a result of concerns about the future of MySQL under Oracle's management. The name of the MariaDB is represented after Monty's youngest daughter’s name, "Maria".

**Key Features:**

MariaDB offers several notable features, including:

* Improved Performance and Scalability: MariaDB is optimized for better efficiency and scalability compared to MySQL.
* Aria Storage Engine: This is MariaDB's default storage system, and it's renowned for its quick speed and crash recovery capabilities.
* Upgrades to InnoDB: MariaDB has made enhancements to InnoDB that lead to superior concurrency and overall performance.
* Thread Pool: This functionality makes sure that several connections and queries are managed effectively at the same time.
* Diverse Storage Options:
* TokuDB: Well-known for its proficiency in handling large datasets and data compression skills.
* ColumnStore: Its columnar format makes it appropriate for scenarios involving massive amounts of data, and it is designed for real-time analytics and OLAP operations.
* MyRocks: An LSM-based storage system praised for its operational efficiency.
* Robust Security Measures:
* Data Encryption: Enables encryption at several levels - be it entire tablespaces, individual tables, or specific columns.
* Access Control Based on Roles: Provides a convenient system to assign and withdraw privileges.
* Audit Feature: Allows for in-depth tracking of user interactions and SQL actions to meet compliance standards.
* Reliability & Duplication:
* Galera Cluster: Supports synchronous multi-master replication, addressing the requirements of high availability and expandability.
* GTID Replication: Streamlines the duplication process and enhances failover management.
* Enhanced SQL & Schema Features:
* Window Functions: Offers advanced SQL tasks related to processing within a group of rows.
* CTE (Common Table Expressions): Improves the query framework and supports recursive query execution.
* JSON Handling: It offers a comprehensive range of tools for working with JSON data, from the creation, interpreting to querying.
* Backup Tools: Features solutions such as MariaBackup for uninterrupted backups.
* Expandability: MariaDB offers the flexibility in installing plugins, custom storage engines and user-defined functions.

**Compatibility:**

MariaDB is a database system that can handle a wide range of SQL commands, functions, and other database-related tasks. It comes with various handy tools like mysqladmin and mysqldump, which are command-line programs. Additionally, there are useful add-ons like the audit plugin available. These features make managing databases easier when using MariaDB.

**Licensing and Community:**

* Open Source: MariaDB is licensed under the GPL, version 2, which ensures both its community commitment and continuous availability at no cost.
* MariaDB Foundation: This body is responsible for monitoring and guiding the development of MariaDB and guaranteeing that it remains transparent and actively engages the community.

**Adoption & Use Cases:**

* MariaDB is adopted by numerous businesses from small startup companies to big organizations across various industries. Owing to its open-source nature, performance enhancements, and robust community support which ultimately makes it a favored choice for web databases, data warehousing, and more.

**Core Functions of Maria DBMS**

**Overview**:

1. Data Retrieval:

Index Structures: MariaDB uses indexes to quickly find data. Think of indexes like a table of contents in a book that helps you locate information without reading the entire book. This speeds up most queries.

Query Optimizer: This is like a smart assistant for queries. It looks at the query and figures out the best way to execute it, considering factors like data size, available indexes, and system resources. MariaDB keeps improving this assistant to make queries faster and more efficient.

Buffer Pool: MariaDB's InnoDB storage engine has a special cache in memory called the buffer pool. It stores frequently accessed data, so the database doesn't have to keep reading from the hard disk. This makes data retrieval much faster.

Read-Ahead Techniques: Imagine it's predicting what pages of a book you'll need to read next and loads them in advance. This way, when you actually need that data (make a query), it's already in memory, reducing the need to read from the disk. This speeds up data retrieval by reducing the time it takes to get the data you want.

2.Data Manipulation:

In MariaDB, Data Manipulation Language (DML) is essential for working with data. Here's a closer look at these operations:

INSERT: This lets you add new records to a table. You can use variations like "INSERT IGNORE" to ignore errors and "INSERT ON DUPLICATE KEY UPDATE" to update a record if a duplicate primary key is found.

UPDATE: It's used to change existing records in a table based on specific conditions. You can also use JOINs to modify data based on related information in another table.

DELETE: This command removes records from a table. Be cautious, especially without a WHERE clause, as it can delete all records in the table. MariaDB also has "DELETE... RETURNING," which returns the deleted rows.

MERGE: MariaDB doesn't have a direct "MERGE" command, but it offers similar functionality with "INSERT... ON DUPLICATE KEY UPDATE" to insert new records or update existing ones based on duplicate keys.

When dealing with large amounts of data, MariaDB offers some efficient options:

INSERT: You can insert multiple records using a single "INSERT" command with multiple sets of values, which is faster than individual inserts.

LOAD DATA INFILE: This is a powerful tool for quickly importing data from text files, especially for large datasets. You can specify formats, delimiters, and handle data transformations during the import.

DELAYED INSERTS: MariaDB has "INSERT DELAYED," which queues insert requests for later, useful when immediate confirmation isn't needed, and it can boost application performance by deferring data insertion.

3.Data Integrity and Accuracy:

Constraints are like rules that help MariaDB keep its data in good shape. Let's dive into these rules:

PRIMARY KEY: Think of this as a special tag that makes sure every item in a table is unique. It's like giving every item a distinct barcode. This helps prevent duplicates and makes it easy to find things quickly.

FOREIGN KEY: This is like connecting two tables in a special way. It links one table to another through a code. This ensures that the relationship between these tables stays consistent and accurate.

UNIQUE: It ensures that everything in a list is different. For example, it makes sure no two people have the same email address. Unique things stay unique.

CHECK: It's like setting a rule for a particular column, such as making sure that the year you enter as a birth year is reasonable. It stops you from entering weird or incorrect data.

NOT NULL: This says that certain information can't be left blank. For example, it makes sure you always enter an address in a contact form.

Transactions are like safeguarding mechanisms for database operations. Here's what they do:

Atomicity: It ensures that when you do several things at once, either all of them happen, or none of them do. It prevents things from being half-done, which could mess up the data.

Consistency: After you do a bunch of things, the database should end up in a sensible state. Even if lots of people are making changes at the same time, everything should still make sense in the end.

Isolation: If two people are changing the same thing at the same time, one of them has to wait their turn. This prevents them from interfering with each other and causing a mess.

Durability: Once you say you've done something, it's written in stone. Even if the computer crashes, your changes are safe and sound. MariaDB uses special techniques to make sure nothing is lost.

4.Concurrency Control:

Multi-version Concurrency Control (MVCC) is a smart system that MariaDB uses to handle many things happening in the database at the same time. Here's what it does:

Versioned Records: When someone makes changes to a piece of information, MVCC keeps a copy of the old version. This way, other people who are looking at that information can still see the original data. It's like having different books for each person, so they each see what they expect.

Read Consistency: MVCC makes sure that when someone is reading something, it doesn't block someone else who's trying to change it. Even if you're in the middle of updating something, others can still see what it looked like before you started, so everything keeps running smoothly.

Phantom Reads Prevention: Sometimes, when one person is reading, they might see things that weren't there before or miss things that were there. MVCC helps prevent this confusion by showing everyone a clear picture of the data, just like taking a snapshot of a moment.

Now, let's discuss about locks:

Row-level Locking: MariaDB is clever in how it locks things. Instead of locking the whole table, it can lock just one row. So, if you're changing one thing, someone else can change something else at the same time. This makes things much faster.

Shared and Exclusive Locks: Shared locks allow many people to read something at the same time, but only one person can change it (exclusive lock). It's like sharing a book but not letting others write in it.

Deadlock Detection: Sometimes, when everyone is waiting for each other, the database gets stuck. MariaDB has a way to notice this and fix it by canceling one of the actions so the others can continue.

In a nutshell, MVCC and these smart locking methods make sure that MariaDB can handle lots of things happening all at once, keeping data safe and transactions running smoothly, even when the database is super busy.

5. Redundancy Removal:

MariaDB has some clever techniques to get rid of redundant data, which is when you have the same information stored in multiple places. This can save space and make things work better.

Normalization: This is like a structured way of organizing data so that you don't store the same thing over and over. MariaDB helps by making sure that data that depends on something else is stored in a smart way, reducing repetition.

Decomposition: Instead of copying the same information in many places, you can split it up and keep it in one central spot. For instance, you can have a separate list of addresses and refer to it when needed, so you don't repeat the same address multiple times.

Referential Integrity: MariaDB keeps an eye on the connections between different pieces of data. It ensures that these connections stay consistent, which not only keeps the data reliable but also stops you from duplicating the same information all over.

Storage Engines: Different ways of storing data can also help reduce redundancy:

So, MariaDB uses these tricks to organize data efficiently, get rid of repeated stuff, and use storage space wisely. This not only saves space but also makes the database quicker and more efficient.

6. Data Independence:

Data independence is a crucial concept in modern databases, and MariaDB ensures it in two ways:

Logical Data Independence: This means that when you make changes to how data is organized logically, like adding new categories or changing data types, it won't disrupt how users or applications see and use the data. With MariaDB, you can make these changes without having to redesign existing applications. It's like rearranging your data without affecting how people view it. MariaDB also supports creating different "views" of the data for applications, so even if the underlying organization changes, the way data is presented to applications remains consistent.

Physical Data Independence: This ensures that changes in how data is physically stored and managed, like improving data storage efficiency, won't impact how users and applications access the data. In MariaDB, you can optimize how the data is stored without affecting how applications interact with it. MariaDB also supports different ways of storing data, so you can switch or optimize your data storage method without causing disruptions to how applications work with the data. It's like upgrading the storage system without affecting how people use the data.

7. Data Relationships:

In databases, maintaining connections between data is crucial. MariaDB, a database system, has some key features to help with this:

Foreign Keys:

Foreign keys are like the glue that holds different parts of your database together. They make sure your data is linked correctly and stays reliable.

Referential Actions: In MariaDB, you can set rules for what happens when you update or delete a piece of data. For example, if you delete something in one table, the related data in another table can be automatically deleted too (called 'ON DELETE CASCADE'). Or, you can set it to make the related data value become empty if the linked data is deleted ('ON DELETE SET NULL').

Consistency Maintenance: MariaDB keeps things in order by making sure that when you add, change, or delete data, it doesn't mess up the connections between different pieces of information.

Indexed Foreign Keys: MariaDB makes sure that finding related data is quick by creating an index on the foreign keys. This helps when you're searching for connected data.

Join Operations: Joining data is like combining pieces of information from different tables in your database. In MariaDB, there are a few ways to do this:

* INNER JOIN: This gets you data that matches a specific condition from both tables. It only gives you results when there's a match in both tables.
* LEFT JOIN: This gets you all the data from the left table and the matching data from the right table. If there's no match, it includes 'NULL' in the result.
* RIGHT JOIN: This works like LEFT JOIN but gets all the data from the right table and matching data from the left table. It also uses 'NULL' when there's no match.
* Advanced Joins: Besides these basic joining methods, MariaDB lets you do more complex things like joining a table with itself (SELF JOIN) or creating the combination of all possible pairs from two tables (CROSS JOIN).

Optimization: MariaDB is smart about making all these connections work quickly. It uses advanced techniques to handle complex queries with many joins efficiently. This means you get your results fast, even when your database is large and complex.

**Advantages of MariaDB**

While there are some features not supported in MariaDB, it has many advantages:

* MariaDB supports more storage engines, giving users more options.
* It can handle a larger number of connections, which can be beneficial for high-traffic scenarios.
* Replication in MariaDB can be almost twice as fast as in MySQL.
* MariaDB is generally faster and easier to work with.
* It allows the creation of versioned tables, which can be useful for managing historical data.
* MariaDB offers compatibility features with Oracle Database, making it easier for users to switch from Oracle.
* MariaDB Server supports a columnar storage format, which is useful for certain types of data.
* MariaDB also provides a distributed SQL database solution called MariaDB Xpand.

**A detailed description of the KDD Nuggets referenced data**

Bank Database Design

Objective: The main objective of the bank database that is taken from the KDD Nuggets referenced data is ensuring seamless transactions of the bank's operations, by taking the structure of the bank as a prime focus element, managing the client interactions with the system effectively, and efficient manner of handling the data.

Entities that are present in the Database:

1. Account:

This entity is more focussed on representing the banking accounts of clients: Attributes which could be included are, Account\_ID, Balance, Date\_Created, Branch\_ID (foreign key that is linking to Branches), account\_type\_id (foreign key that is linking to account\_types), and client\_id (foreign key that is linking to Clients).

2. Account\_Types:

This entity describes various kinds of accounts that are available such as: savings, checking, fixed deposit, etc. Attributes which could be included are, Account\_Type\_ID, Type\_Name, Interest\_Rate, and other specific features that are related to the account type.

3. Branches:

This entity provides the information that is in particular towards the branches of the bank. Attributes which could be included are, Branch\_ID, Branch\_Name, Branch\_Location, Branch\_Manager, etc.

4. Clients:

This entity focuses on providing the information about the clients/customers of the bank clients. Attributes which could be included are, Client\_ID, Client\_Name, Address, Contact\_Details, etc.

5. Transactions:

This entity primarily aims in keeping the records of all the transactions that are operated between accounts. Attributes which could be included are, Transaction\_ID, Source\_Account\_ID (foreign key that is linking to Account), Destination\_Account\_ID (foreign key that is linking to Account), Amount, Transaction\_Date, and Transaction\_Type\_ID (foreign key that is linking to Transaction\_Types).

6. Transaction\_Type:

This entity provides the information about the type of transaction that would occur such as deposit, transfer, withdrawal, etc. Attributes which could be included are, Transaction\_Type\_ID, Type\_Description, Applicable\_Fees, etc.

Key Relationships:

Accounts to Branches: A branch can service more than one account, but each account is linked to a single branch.

Accounts to Account Types: Each kind of account is unique, although each type might include more than one account in its classification.

Clients to Accounts: Although a client may hold many accounts, each account is held exclusively by that client.

Transactions to Accounts: Multiple transactions may be made on a single account. There is a source account and maybe a destination account for every transaction.

Transactions to Transaction\_Type: Every transaction has a unique type, although there are several transactions that can be connected to a single type.

*Note: This data set does not address the relations of branch and manager, manager and employees of the bank. Hence, we are not addressing those relations while designing the database.*

Additional Features:

- Limit on Account Creation: Here, regardless of the branch a client is in, the database contains a stored function that would make sure that they don't create more than ten accounts.

- Transaction Procedure: When a transaction takes place, this process records the information and modifies the source and destination accounts. This process also manages transaction costs, which vary according to the kind of transaction.

Optimization Strategies:

- Indexing: To expedite data retrieval processes, the database employs indexing on certain columns, which functions as a directory.

- Denormalization: When the same data is stored in multiple tables, there's a chance that the database design purposefully introduces redundancy. As a result, fewer joins are required, which lowers the complexity and processing time of some queries.

In essence, this dataset offers a thorough understanding of a banking system, safeguarding the data integrity, structuring the relational data, and managing the operations effectively.