Notes on Real-World DBMS Application

Shuvam Banerji Seal

February-March, 2025

Contents

1 Introduction

This document provides detailed notes on installing, configuring, and running MariaDB on Arch Linux. It covers the initial setup, starting the service, securing the installation, creating a port, and connecting to it. Additionally, it includes information on common errors (and solutions), local vs. remote connections, default databases, user management, and usage of the `mariadb` command-line client.

2 Why MariaDB Instead of MySQL?

MariaDB is an open-source fork of MySQL, developed by the original creators of MySQL after concerns arose over Oracle's acquisition of MySQL. Here are key reasons for choosing MariaDB:

- **Open Source Commitment**: MariaDB is fully open-source under the GPL, ensuring transparency and community-driven development, whereas MySQL has a dual-licensing model with proprietary elements.
- **Performance**: MariaDB offers improved performance with optimized query execution and better thread pooling compared to MySQL.
- **Drop-in Replacement**: MariaDB is designed to be compatible with MySQL, allowing seamless transitions for existing MySQL users.
- **Features**: MariaDB includes additional storage engines (e.g., Aria, ColumnStore) and features not found in MySQL, enhancing flexibility.
- **Arch Linux Support**: MariaDB is well-supported in Arch Linux's repositories, aligning with the distribution's minimalist and up-to-date philosophy.

Given these advantages, MariaDB is a robust alternative for database management on Arch Linux.

3 Installation

Arch Linux uses the `pacman` package manager. To install MariaDB, execute the following command in the terminal:

```
sudo pacman -S mariadb
```

This installs the latest MariaDB server and client packages from the Arch repositories.

4 Initializing the Database

Before starting MariaDB, initialize the database storage directory:

```
sudo mariadb-install-db --user=mysql --basedir=/usr --datadir=/var/lib/mysql
```

Explanation of options:

- `--user=mysql`: Runs the database as the `mysql` user.
- `--basedir=/usr`: Specifies the MariaDB installation directory (often ignored by some tools).
- `--datadir=/var/lib/mysql`: Sets the data directory (default on Arch Linux).

5 Starting the MariaDB Service

Arch Linux uses `systemd` for service management. Start and enable the MariaDB service:

```
sudo systemctl start mariadb
sudo systemctl enable mariadb
```

- `start`: Launches the service immediately.
- `enable`: Ensures the service starts on boot.

Verify the service is running:

```
systemctl status mariadb
```

6 Securing the Installation

MariaDB includes a script to secure the initial setup. Run:

```
sudo mariadb-secure-installation
```

Follow the prompts to:

- Set a root password.
- Remove anonymous users.
- Disallow remote root login (recommended for local setups).
- Remove the test database.
- Reload privilege tables.

7 Configuring the Port

By default, MariaDB listens on port 3306. To confirm or change this, edit the configuration file located at `/etc/my.cnf.d/server.cnf`. Open it with a text editor:

```
sudo nano /etc/my.cnf.d/server.cnf
```

Under the `[mysqld]` section, ensure or set the port:

```
[mysqld]
port=3306
```

To use a custom port (e.g., 3307), modify it:

```
[mysqld]
port=3307
```

Save the file and restart the service:

```
sudo systemctl restart mariadb
```

Verify the port is in use:

```
sudo netstat -tuln | grep 3307
```

8 Connecting to the Database (Local and Remote)

8.1 Who Is `localhost` and Why Do We Use `'username'@'host'`?

Localhost Explained In most configurations, `localhost` refers to the local machine (the loopback address `127.0.0.1` for IPv4 or `::1` for IPv6). When you connect to `localhost`, it usually does not use the external network interface; instead, it can use a local Unix socket or named pipe (depending on the operating system). This means:

- **Connections via `localhost`** are not accessible externally, ensuring a measure of security by restricting traffic to the local system only.
- If you want to allow remote connections, you must bind MariaDB to an external IP (e.g., `0.0.0.0` for all interfaces) and configure your firewall to allow inbound traffic on the MariaDB port (default `3306`).

Why `'username'@'host'`? In MySQL/MariaDB, user accounts are defined by *both* the username and the host from which they can connect. This is why the account name appears in the form `'user'@'host'`. Here's how it works:

- **Distinct Accounts per Host**: `'alice'@'localhost'` is considered a completely different account from `'alice'@'192.168.1.10'` or `'alice'@'%'`. This allows fine-grained control. For instance, you can grant certain privileges to a user when they connect locally but different (or no) privileges when connecting from elsewhere.

- **Security and Access Control**: By specifying the host portion in the username definition, you can restrict which machines or IP addresses are allowed to connect under that username. For example:
  - `'bob'@'localhost'`: Bob can only connect from the same machine running MariaDB.
  - `'bob'@'%'`: Bob can connect from any IP address (assuming the server is bound to allow remote connections).
  - `'bob'@'192.168.1.%'`: Bob can connect from any IP in the `192.168.1.x` subnet.
- **Practical Example**:

```
CREATE USER 'alice'@'localhost' IDENTIFIED BY 'password1';
CREATE USER 'alice'@'%'         IDENTIFIED BY 'password2';
```

  Here, `'alice'@'localhost'` and `'alice'@'%'` are two separate accounts, each potentially with different privileges and passwords.

8.2 Local Connection

To connect locally (default port 3306):

```
mariadb -u root -p
```

You will be prompted for a password.

8.3 Remote Connection

If you changed the bind address and are connecting remotely (e.g., IP `192.168.1.50` and port 3307):

```
mariadb -u root -p -h 192.168.1.50 --port=3307
```

8.4 Common Flags for `mariadb`

- `-u <user>`: Specify the username, e.g., `root`.
- `-p`: Prompt for the user's password.
- `-h <host>`: Connect to the specified host (default is `localhost`).
- `--port=<port>`: Use a custom port number.
- `--socket=<path>`: Connect using a Unix socket (local).
- `--ssl`: Force use of SSL encryption if configured.
- `--execute=<statement>`: Execute the statement and exit (non-interactive).

8.5 Handling the `mariadb` Command Interface

When you connect successfully, you will see a prompt like:

```
MariaDB [(none)]>
```

Within this interface:

- End commands with a semicolon (`;`).
- Use `SHOW DATABASES;` or `SHOW TABLES;` to list databases/tables.
- Use `HELP;` (or `HELP <command>`) for basic usage info.
- Press `Ctrl+D` or type `EXIT;` to leave the interface.

9 Default Databases in MariaDB

After a fresh initialization, MariaDB ships with several default databases:

- `mysql`: Stores user accounts, privileges, and internal metadata.
- `information_schema`: Provides read-only views of database metadata (tables, columns, etc.).
- `performance_schema`: Tracks performance-related metrics and server events.
- `sys`: Contains helper views and functions that simplify querying performance and diagnostic data.

10 Creating and Managing Users & Privileges

10.1 Creating a User

You can create a user with:

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'user_password';
```

10.2 Granting Privileges

Basic syntax to grant privileges:

```
GRANT <privileges> ON <database>.<table> TO 'username'@'localhost';
```

Common privilege sets include:

- `ALL PRIVILEGES`: Grant everything on a database or table.
- `SELECT, INSERT, UPDATE, DELETE`: Typical DML privileges.
- `CREATE, ALTER, DROP`: DDL privileges to modify schema.
- `GRANT OPTION`: Allows the user to grant privileges to others.

For example, granting all privileges on a specific database:

```
GRANT ALL PRIVILEGES ON mydatabase.* TO 'username'@'localhost';
FLUSH PRIVILEGES;
```

Remember to `FLUSH PRIVILEGES;` so changes take effect immediately.

10.3 Removing Privileges or Users

You can revoke privileges:

```
REVOKE ALL PRIVILEGES ON mydatabase.* FROM 'username'@'localhost';
```

Or drop the user entirely:

```
DROP USER 'username'@'localhost';
```

11 Using the `mariadb` Terminal

The `mariadb` command-line interface (CLI) is the main way to interact directly with MariaDB. You can perform tasks such as creating databases, managing tables, and running queries all from the `mariadb` shell.

11.1 Starting the `mariadb` Shell

Use the following command to open the MariaDB shell as a specific user (e.g., `root`):

```
mariadb -u root -p
```

When prompted, enter the password for the specified user. Once logged in, you will see a prompt like:

```
MariaDB [(none)]>
```

You can also specify a host, port, or socket if needed:

```
mariadb -u user -p -h <host> --port=<port>
```

11.2 Essential Commands Within the Shell

Below are common commands you may use:

- **Show Databases**:

  ```
      SHOW DATABASES;
  ```

  Lists all databases accessible to the current user.

- **Select a Database**:

  ```
      USE mydatabase;
  ```

  Switches the session context to the specified database.

- **Show Tables**:

  ```
      SHOW TABLES;
  ```

  Lists all tables in the current database.

- **Describe a Table**:

  ```
      DESCRIBE tablename;
  ```

  Displays column information (name, type, etc.) for `tablename`.

- **Creating Databases and Tables**:

  ```
      CREATE DATABASE mydatabase;
      CREATE TABLE mytable (
         id INT AUTO_INCREMENT PRIMARY KEY,
         name VARCHAR(255),
         created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
      );
  ```

- **Data Manipulation (CRUD)**:

  ```
      INSERT INTO mytable (name) VALUES ('Alice');
      SELECT * FROM mytable;
      UPDATE mytable SET name='Bob' WHERE id=1;
      DELETE FROM mytable WHERE id=1;
  ```

  Use these commands to insert, query, update, or delete data within tables.

- **Managing Users and Privileges** (if you are a privileged user):

  ```
      CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'secure_password';
      GRANT ALL PRIVILEGES ON mydatabase.* TO 'newuser'@'localhost';
      FLUSH PRIVILEGES;
  ```

- **Exiting the Shell**:

  ```
      EXIT;
  ```

  or press `Ctrl+D`.

11.3 Getting Help from Within the Shell

You can get brief help on many topics directly in the shell:

```
HELP;
HELP <command>;
```

For instance, `HELP CREATE TABLE;` will display syntax details for creating tables.

11.4 Tips for Using the `mariadb` Shell

- End all SQL statements with a semicolon (`;`).
- Use the `UP/DOWN` arrow keys to navigate command history.
- Use tab-completion for partially typed commands, table names, etc.
- Complex multi-line queries can be typed and edited freely until you end with `;`.
- If you make a mistake while typing, use `Ctrl+C` to cancel the current query line and return to the prompt.

12 Creating a Test Database and User (Example)

To test connectivity, create a database and user:

```
CREATE DATABASE test_db;
CREATE USER 'test_user'@'localhost' IDENTIFIED BY 'secure_password';
GRANT ALL PRIVILEGES ON test_db.* TO 'test_user'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

Connect as the new user:

```
mariadb -u test_user -p
```

(or add `--port` if you changed it).

13 Connecting Remotely (Optional)

To allow remote connections, edit `/etc/my.cnf.d/server.cnf` and set:

```
[mysqld]
bind-address=0.0.0.0
```

Restart the service and update the user privileges:

```
GRANT ALL PRIVILEGES ON test_db.* TO 'test_user'@'%' IDENTIFIED BY 'secure_password';
FLUSH PRIVILEGES;
```

Ensure the firewall allows the port (e.g., using `ufw` or `iptables`):

```
sudo ufw allow 3307
```

14 Possible Errors and Their Solutions

This section covers common errors you might encounter and how to fix them.

14.1 Missing System Tables (e.g., `Table 'mysql.plugin' doesn't exist`)

- **Cause**: The system database has not been initialized properly, or the data directory is empty/corrupted.
- **Solution**:
  1. Confirm your `datadir` setting (usually `/var/lib/mysql`) and ensure ownership is `mysql:mysql`.
  2. Re-run the initialization command (e.g., `mariadb-install-db`) to create system tables.
  3. Restart `mariadb`.

14.2 `Access denied for user 'mysql'@'localhost'`

- **Cause**: Attempting to log in with the `mysql` system account, which typically lacks administrative privileges. Or root access is restricted by password.
- **Solution**:
  1. Use `root` (or another privileged user) instead of `mysql`.
  2. If you lost the root password, reset it by starting `mariadbd` with `--skip-grant-tables` and updating the password manually.

14.3 `Can't open and lock privilege tables`

- **Cause**: Permission or ownership problems in the data directory, or the system tables are missing.
- **Solution**:
  1. Run `sudo chown -R mysql:mysql /var/lib/mysql`.
  2. Ensure SELinux/AppArmor isn't blocking access (if applicable).
  3. Re-initialize and restart if needed.

14.4 Port Conflicts (e.g., `port already in use`)

- **Cause**: Another process is listening on the same port (e.g., 3306).
- **Solution**:
  1. Use `sudo netstat -tuln | grep 3306` to find the conflicting process.
  2. Stop or reconfigure the conflicting service, or change MariaDB's port in `/etc/my.cnf.d/server.cnf`.

14.5 Authentication Plugin Issues

- **Cause**: Some distributions use `unix_socket` authentication for root, leading to unexpected login methods.
- **Solution**:
  1. `ALTER USER 'root'@'localhost' IDENTIFIED VIA mysql_native_password` if you want a password-based method.
  2. Consult distribution documentation for default authentication plugins.

15 Entering the MariaDB Shell with `mariadb`

When you type the command

```
mariadb
```

(without any additional flags like `-u` or `-p`), MariaDB will attempt to authenticate you based on your system user or through an anonymous account (if available). Below is what you can do to understand how you are logged in and what privileges you have.

15.1 Checking Your Current User and Privileges

1. **Identify the Current User:**

```
SELECT USER(), CURRENT_USER();
```

   - `USER()` displays who you *attempted* to authenticate as.
   - `CURRENT_USER()` shows how the server *actually* recognized you, which might differ if you're using an anonymous account or a socket-based authentication plugin.

2. **Check Your Privileges:**

```
SHOW GRANTS;
```

   This command lists all privileges granted to the current user. If you see `"GRANT USAGE ON *.* TO ''@'localhost'"` or limited statements, you may be operating under an anonymous user with restricted privileges.

15.2 What Information Can You Access?

Depending on your privileges, you can run various `SHOW` commands to see different parts of the database server's metadata:

- **List All Databases:**

```
SHOW DATABASES;
```

  If you have only partial or no privileges, you may see limited or even zero databases listed.

- **Select a Database:**

```
USE your_database_name;
```

  Switches the active database, though you may be denied if you lack access.

- **List Tables in the Current Database:**

```
SHOW TABLES;
```

  Again, if your account does not have `SELECT` or other privileges on any table, you might see an empty list or receive permission errors.

- **Describe a Table:**

```
DESCRIBE your_table_name;
```

  Shows the columns, types, and other schema details of `your_table_name`, if allowed.

- **View Server Status or Variables (Restricted for Low-Privilege Users):**

```
SHOW STATUS;
SHOW VARIABLES;
```

  These commands show server runtime statistics and configuration variables, but you might need higher privileges (`SUPER` or `PROCESS`) to see full information.

15.3 Limitations of an Anonymous or Unprivileged Session

If you see something like:

```
+----------------+
| current_user() |
+----------------+
| @localhost     |
+----------------+
```

it means you are logged in as an anonymous user (`''@'localhost'`). Anonymous sessions often have minimal or no privileges, so you may be unable to create, modify, or drop databases and tables. You will likely encounter permission errors when attempting administrative tasks.

15.4 Moving to a Privileged Account

To gain more control, exit the session and specify a proper username (`root` or another admin user) and password if required:

```
EXIT;
mariadb -u root -p
```

If you do not have credentials for a privileged account, ask your database administrator to grant you additional privileges or create a user for you with the appropriate access rights.

15.5 Summary

When entering the MariaDB shell using only `mariadb`, the user may be authenticated in one of several ways:

1. As an anonymous (*empty*) user, if such an account exists.
2. As a user mapped via socket authentication (common for `root` on some systems).
3. As your system user, if configured.

Once inside, you can run `SHOW GRANTS`, `SELECT USER()`, and `SELECT CURRENT_USER()` to discover your effective user and privileges. If you need higher-level operations, exit and log in under a more privileged username.

16 Creating an Anime-Themed Database

Let's create a database called `anime_world` with related tables: `anime`, `characters`, `studios`, and `episodes`.

16.1 Database and Table Creation

Log in and create the database:

```
CREATE DATABASE anime_world;
USE anime_world;
```

Create the `anime` table:

```
CREATE TABLE anime (
    anime_id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(100) NOT NULL,
    genre VARCHAR(50),
    release_year INT,
    studio_id INT
);
```

Create the `studios` table:

```sql
CREATE TABLE studios (
    studio_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    founded_year INT
);
```

Create the `characters` table:

```sql
CREATE TABLE characters (
    char_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    role VARCHAR(20),
    anime_id INT,
    FOREIGN KEY (anime_id) REFERENCES anime(anime_id)
);
```

Create the `episodes` table:

```sql
CREATE TABLE episodes (
    ep_id INT PRIMARY KEY AUTO_INCREMENT,
    anime_id INT,
    episode_number INT NOT NULL,
    title VARCHAR(100),
    air_date DATE,
    FOREIGN KEY (anime_id) REFERENCES anime(anime_id)
);
```

Add foreign key to `anime` for `studios`:

```sql
ALTER TABLE anime
ADD FOREIGN KEY (studio_id) REFERENCES studios(studio_id);
```

16.2 Sample Data

Insert data into `studios`:

```sql
INSERT INTO studios (name, founded_year) VALUES
('Studio Ghibli', 1985),
('MAPPA', 2011);
```

Insert data into `anime`:

```sql
INSERT INTO anime (title, genre, release_year, studio_id) VALUES
('Spirited Away', 'Fantasy', 2001, 1),
('Jujutsu Kaisen', 'Action', 2020, 2);
```

Insert data into `characters`:

```sql
INSERT INTO characters (name, role, anime_id) VALUES
('Chihiro', 'Protagonist', 1),
('Gojo Satoru', 'Supporting', 2);
```

Insert data into `episodes`:

```sql
INSERT INTO episodes (anime_id, episode_number, title, air_date) VALUES
(1, 1, 'The Tunnel', '2001-07-20'),
(2, 1, 'Ryomen Sukuna', '2020-10-03');
```

17 Database Functionalities and Operations

MariaDB supports a wide range of functionalities and operations.

17.1 Basic CRUD Operations

- **Create**: Insert data (as shown above).
- **Read**: Query data:

```sql
SELECT * FROM anime WHERE release_year > 2010;
```

- **Update**: Modify records:

```sql
UPDATE characters SET role = 'Main' WHERE name = 'Chihiro';
```

- **Delete**: Remove records:

```sql
DELETE FROM episodes WHERE episode_number = 1;
```

17.2 Table Management

- Add a column:

```sql
ALTER TABLE anime ADD COLUMN rating DECIMAL(2,1);
```

- Drop a table:

```sql
DROP TABLE episodes;
```

17.3 Indexes

Improve query performance:

```sql
CREATE INDEX idx_title ON anime(title);
```

17.4 Views

Create a virtual table:

```sql
CREATE VIEW anime_recent AS
SELECT title, release_year FROM anime WHERE release_year >= 2000;
```

Query the view:

```
SELECT * FROM anime_recent;
```

18 Set Operations

MariaDB supports set operations to combine query results.

18.1 UNION

Combine distinct anime and character names:

```
SELECT title AS name FROM anime
UNION
SELECT name FROM characters;
```

18.2 UNION ALL

Include duplicates:

```
SELECT title AS name FROM anime
UNION ALL
SELECT name FROM characters;
```

18.3 INTERSECT (Emulated)

MariaDB lacks native `INTERSECT`, but it can be emulated with `INNER JOIN`:

```
SELECT DISTINCT a.title
FROM anime a
INNER JOIN characters c ON a.title = c.name;
```

18.4 EXCEPT (Emulated)

Emulate `EXCEPT` with `LEFT JOIN`:

```
SELECT a.title
FROM anime a
LEFT JOIN characters c ON a.title = c.name
WHERE c.name IS NULL;
```

19 Inner Joins

Inner joins combine related data from multiple tables.

19.1 Anime and Studios

Join `anime` and `studios`:

```
SELECT a.title, s.name AS studio_name
FROM anime a
INNER JOIN studios s ON a.studio_id = s.studio_id;
```

19.2 Anime and Characters

Join `anime` and `characters`:

```
SELECT a.title, c.name AS character_name, c.role
FROM anime a
INNER JOIN characters c ON a.anime_id = c.anime_id;
```

19.3 Multi-Table Join

Join `anime`, `characters`, and `episodes`:

```
SELECT a.title, c.name AS character_name, e.title AS episode_title
FROM anime a
INNER JOIN characters c ON a.anime_id = c.anime_id
INNER JOIN episodes e ON a.anime_id = e.anime_id
WHERE e.episode_number = 1;
```