mariaDB

WHITE PAPER

MySQL to MariaDB migration compatibility

September 2020



CONTENTS

Table of contents

01	**Introduction**	3
02	**Feature parity**	4
03	**Service life**	5
04	**Testing approach**	6
05	**Behavioral observations**	8
	Connecting	8
	SQL modes	9
	Data import	10
	Schema display	10
	Character sets and collation	11
	Data types	12
	Operators	16
06	**Conclusion**	18

01

Introduction

MariaDB and MySQL are two of the most popular open source relational databases in the world. MariaDB was created as a fork of MySQL by the founder of MySQL itself, Michael “Monty” Widenius, in order to ensure open and transparent development after MySQL was acquired by Oracle. However, since then, MariaDB and MySQL have become separate databases with different features, and with similar features having different implementations.

As major versions of MySQL reach end of life, it is natural for customers on those versions to consider available upgrade paths. MariaDB is a viable alternative, and in many cases will present a lower barrier for migration.

While evaluation and testing are key steps in confirming the viability of a migration path, an understanding of available features, compatibility, and behavior provide a good starting point for considering migration options.

02

Feature parity

When evaluating migration compatibility, it is often helpful to start with an understanding of the functionality present in your application today, followed by a consideration of how the application could be enhanced to take advantage of new capabilities.

A detailed analysis of MariaDB and MySQL features is available, "MariaDB Enterprise vs.MySQL Enterprise Edition 8" (September 2020). This detailed analysis highlights both points of feature parity (shared functionality) and product-specific differentiators.

03

Service life

To address the broadest possible audience, throughout this paper we will evaluate the viability of migration for MySQL Community Edition to MariaDB Enterprise Server. As migration decisions are often time-boxed based on software lifespan, let's first consider end-of-life (EOL) dates.

EOL DATE	PRODUCT	VENDOR
2018-12	MySQL Database 5.5	Oracle Corporation
2021-02	MySQL Database 5.6	Oracle Corporation
2022-05	MariaDB Enterprise Server 10.2	MariaDB Corporation
2023-05	MariaDB Enterprise Server 10.3	MariaDB Corporation
2023-10	MySQL Database 5.7	Oracle Corporation
2024-07	MariaDB Enterprise Server 10.4	MariaDB Corporation
2025-07	MariaDB Enterprise Server 10.5	MariaDB Corporation
2026-05	MySQL Database 8.0	Oracle Corporation

SOURCES

MariaDB Corporation Engineering Policies, https://mariadb.com/engineering-policies/
Oracle Lifetime Support Policy, http://www.oracle.com/us/support/library/lifetime-support-technology-069183.pdf

04

Testing approach

Evaluation and testing are key steps in confirming the viability of a migration path. While this testing must generally be performed with a customer application and customer data, we have performed testing to understand a basic level of common behavior and functionality between current versions of MySQL Community Edition and current versions of MariaDB Enterprise Server to support further exploration of MariaDB as a viable migration path.

Specific versions tested:

- MySQL Community Edition 5.6.49
- MySQL Community Edition 5.7.31
- MySQL Community Edition 8.0.21
- MariaDB Enterprise Server 10.2.33-8
- MariaDB Enterprise Server 10.3.24-8
- MariaDB Enterprise Server 10.4.14-8
- MariaDB Enterprise Server 10.5.5-3

MariaDB Enterprise Server is available from MariaDB Corporation, https://mariadb.com/.

MySQL Community Edition is available from Oracle Corporation, https://www.mysql.com/products/community/.

To evaluate functionality and behavior, we conducted several tests.

Data import

When evaluating a migration path, a customer will generally import their data and perform validation that data import was complete and accurate. Validation is generally performed on multiple servers when migrating to a multi-node replicated or clustered deployment.

In our testing, as a simple alternative to customer data, we leveraged various vendor-developed and community-developed test data sets to observe the import of data to a server, and the replication of data between servers.

Available data sets include:

- Sakila Sample Database, from Oracle Corporation
- Employees Sample Database, from Oracle Corporation
- Classicmodels Sample Database, from mysqltutorial.org
- Northwind Database extended for additional database services, available on GitHub

Several summaries of these test data sets are available, including:

- Chua Hock-Chuan's overview at ntu.edu.sg (2012)
- Cristian Scutaru's extended list of example databases (2019)

SQL queries

When evaluating a migration path, a customer will generally perform several types of testing to confirm proper application function. This testing may include system-level testing of the application, function-level input/output testing, performance testing, and query-level input/output testing to confirm parity of operation between old and new platforms.

Database configuration, data set, and application design are key factors in testing. In forming a test to aid customers considering migration to MariaDB, we could not test every possible variation, so we chose to use several methods for testing SQL behavior which we think will be of general benefit in understanding differences in server behavior.

- Example queries were executed against all tested versions, and variations in warnings, errors, and output were noted.
- Default server behavior was tested via queries generated by sqltest, available on GitHub.
- Queries were repeated with `sql_mode=ANSI` to increase alignment to ISO standard SQL (ISO 9075). The goal of this testing was not ISO compliance validation, but to understand the difference in behavior between product versions.
  - `sql_mode=ANSI` serves a similar function in MySQL and MariaDB.
  - The MySQL 5.6 Reference Manual notes, "This mode changes syntax and behavior to conform more closely to standard SQL," and "Equivalent to `REAL_AS_FLOAT, PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE`. Additional information may be found in the MySQL 5.6 Reference Manual."
  - The MariaDB Public Knowledge Base notes, "Changes the SQL syntax to be closer to ANSI SQL. Sets: `REAL_AS_FLOAT, PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE`." Additional information may be found in the MariaDB Public Knowledge Base.

Our objective in this testing is to understand how different server platforms and versions respond differently to the same input. By understanding and documenting both points where we observe behavioral differences and points where no differences are observed, we enable the customer to more rapidly focus on other areas of compatibility evaluation.

05

Behavioral observations

Connecting

All testing has been performed through a database connection.

Both MySQL and MariaDB provide a command-line client and a range of language-specific connectors, in addition to a broader range of language-specific connectors and third-party clients available from the MySQL and MariaDB community.

Many community-sourced options, such as the database connector in the PHP language, have been community tested with both MySQL and MariaDB.

MariaDB Corporation provides vendor support for:

CONNECTOR	LANGUAGE/INTERFACE
MariaDB Connector/C	C language
MariaDB Connector/C++	C++ language
MariaDB Connector/J	Java language (7, 8, 11), JDBC 4.1/4.2
MariaDB Connector/Node.js	Node.js 6 and later
MariaDB Connector/ODBC	ODBC 3.5
MariaDB Connector/Python	Python 3.6 and later
MariaDB Connector/R2DBC	Java 8+ and R2DBC 0.8.2

Additional information on MariaDB clients and connectors may be found in the MariaDB Enterprise Documentation.

Additional observations

When moving from earlier MySQL releases to MySQL 8.0 there is a big change in the default user authorization (`caching_sha2_password`) which can cause issues with scripts and programs. This often requires upgrades in the connection libraries and changes in apps to deal with it, since it does require an SSL connection. Some people instead take steps to change either the default password authentication or each user's password authenticator to `mysql_native_password` .

Customers may wish to evaluate whether the client and connector compatibility with MariaDB presents a reduced migration burden.

SQL modes

In MariaDB and MySQL, SQL Modes are used to override default database server behavior in areas such as query processing, error handling, and data handling. The default SQL Mode differs between products and versions.

Specific customer testing should occur based on the SQL Modes they currently use, with an understanding of how these may need to change when changing database products and versions.

We observed the following `SQL_MODE` defaults:

PRODUCT/VERSION	SQL_MODE DEFAULT
MySQL 5.6	`NO_ENGINE_SUBSTITUTION`

MySQL 5.7	`ONLY_FULL_GROUP_BY`
	`STRICT_TRANS_TABLES`
	`NO_ZERO_IN_DATE`
	`NO_ZERO_DATE`
	`ERROR_FOR_DIVISION_BY_ZERO`
	`NO_AUTO_CREATE_USER`
	`NO_ENGINE_SUBSTITUTION`

MySQL 8.0	`ONLY_FULL_GROUP_BY`
	`STRICT_TRANS_TABLES`
	`NO_ZERO_IN_DATE`
	`NO_ZERO_DATE`
	`ERROR_FOR_DIVISION_BY_ZERO`
	`NO_ENGINE_SUBSTITUTION`

MariaDB ES 10.2	`STRICT_TRANS_TABLES`
	`ERROR_FOR_DIVISION_BY_ZERO`
	`NO_AUTO_CREATE_USER`
	`NO_ENGINE_SUBSTITUTION`

MariaDB ES 10.3	`STRICT_TRANS_TABLES`
	`ERROR_FOR_DIVISION_BY_ZERO`
	`NO_AUTO_CREATE_USER`
	`NO_ENGINE_SUBSTITUTION`

MariaDB ES 10.4	`STRICT_TRANS_TABLES`
	`ERROR_FOR_DIVISION_BY_ZERO`
	`NO_AUTO_CREATE_USER`
	`NO_ENGINE_SUBSTITUTION`

MariaDB ES 10.5	`STRICT_TRANS_TABLES`
	`ERROR_FOR_DIVISION_BY_ZERO`
	`NO_AUTO_CREATE_USER`
	`NO_ENGINE_SUBSTITUTION`

Additional observations

MySQL 5.7 has deprecated various `SQL_MODE` strings that are being combined, such as making strict table settings also imply strict dates and strict division-by-0. This may impact customers whose applications need to use `STRICT_TRANS_TABLES` while also using '0000-00-00' dates. In MySQL 5.7, this generates a bunch of warnings, which for MySQL 8.0 have turned into errors since the deprecated strings were removed.

It is our further understanding that `NO_AUTO_CREATE_USER` is the default behavior in MySQL 8.0, but that this `SQL_MODE` is no longer recognized.

Customers may wish to evaluate whether the fine-grained `SQL_MODE` behavior available in MariaDB presents a reduced migration burden.

For a full list of MariaDB Server SQL Modes, see the MariaDB Public Knowledge Base.

For a full list of MySQL Database SQL Modes, see the MySQL Documentation.

Data import

We observed the following differences while testing data imports in default SQL Mode:

- MySQL 8.0 has deprecated `ZEROFILL`.
- MySQL 8.0 has deprecated specifying precision in `FLOAT`, `DOUBLE`, and `DECIMAL` column specifications.
- MySQL 8.0 has a new default (compiled-in) character set of `utf8mb4` and a collation of `utf8mb4_0900_ai_ci` (instead of the old defaults of `latin1` and `latin1_swedish_ci`).
- MariaDB Enterprise Server has character-set default of `latin1` and collation default of `latin1_swedish_ci`, though some distribution-specific packages ship with overrides in the config files.
- MySQL 5.7 and above have an explicit `JSON` field type whereas MariaDB Enterprise Server translates a JSON field request into a `LONGTEXT` with extra options. MariaDB Enterprise Server 10.4 and above support check constraints on JSON fields.
- MySQL 8.0 deprecates the "`&&`", "`!`", and "`||`" (the `OR` alias) operators.
- MySQL 8.0 warns users about using `NATIONAL`, `NCHAR`, and `NVARCHAR` types because those types currently imply a `CHAR` or `VARCHAR` field with a character set of `utf8mb3` while a future release is going to change them to imply the `utf8mb4` character set.
- MySQL's `DATE_ADD()` function turns `0000-01-01` into `0000-00-00` while MariaDB turns it into `NULL`. Other invalid dates are turned into `NULL` on both.

Schema display

The same method was used for both MySQL and MariaDB to import data, and the same method was used for both platforms to view schema. Schema viewing is a common action when validating that the behavior of two platforms is the same.

We observed several differences in default SQL Mode:

- MySQL 8.0 has removed integer field digit sizes, so if you specify an `INT`, an `INT1` … `INT8`, etc. with or without a "(`NUM`)" suffix, the resulting field has no "(`NUM`)" suffix when displaying the schema.
- MySQL doesn't show the "`DEFAULT NULL`" qualifier for various `TEXT`, `BLOB`, and `JSON` column types even when it is in effect. While this appears cosmetic, it may cause confusion for someone trying to compare schema.
- MySQL puts extra parens into the display of a `MULTIPOINT()` value in a geometry field whereas MariaDB does not (e.g., "`MULTIPOINT((0 0),(1 1))`"). This differs from the output of other objects that contain multiple points, such as `POLYGON()`, which displays without the extra parentheses in both.
- MySQL renders a timestamp field with a current-time default as:
  `DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP`
  while MariaDB renders it using functions:
  `DEFAULT current_timestamp() ON UPDATE current_timestamp()`.
- In the Northwind extended example data there is a table that MySQL dumps in a different order from MariaDB: `EmployeeTerritories`. MariaDB dumps it in its primary key order (the int + varchar in the table) whereas MySQL dumps it in the order of a foreign key, `FK_EmployeeTerritories_Territories` (as revealed by `EXPLAIN`).
- MySQL quotes numbers in default values, so a schema dump of an `INT` with a default value will differ slightly with MariaDB's representation that omits the quotes.

Character sets and collation

Character sets allow users to store characters from many different languages in the database. Some character sets also support special characters like currency symbols, mathematical symbols, musical symbols and emoji.

Using names as an example, the character set defines which characters can appear in each name. The collation defines the order in which the names would be sorted, and whether two names would be equal:

- In a case-insensitive collation, "Mueller" and "mueller" would be considered the same name.
- In a case-sensitive collation, "Mueller" and "mueller" would be considered different names.
- In a German collation, "Mueller" and "Müller" would be considered the same name.
- In an English collation, "Mueller" and "Müller" would be considered different names.
- In a collation that ignores padding, "Mueller" and "Mueller " would be considered the same name.
- In a no-pad collation, "Mueller" and "Mueller " would be considered different names.

PRODUCT/VERSION	DEFAULT CHARACTER SET	DEFAULT COLLATION
MySQL 5.6	`latin1`	`latin1_swedish_ci`
MySQL 5.7	`latin1`	`latin1_swedish_ci`
MySQL 8.0	`utf8mb4`	`utf8mb4_0900_ai_ci`
MariaDB ES 10.2	`latin1`	`latin1_swedish_ci`
MariaDB ES 10.3	`latin1`	`latin1_swedish_ci`
MariaDB ES 10.4	`latin1`	`latin1_swedish_ci`
MariaDB ES 10.5	`latin1`	`latin1_swedish_ci`

Character set and collation may be overridden. Differences in default behavior should be a migration consideration.

Data types

MariaDB Enterprise Server and MySQL Community Edition support a wide range of data types.

Numeric data types

	MYSQL 5.6	MYSQL 5.7	MYSQL 8.0	MARIADB ES 10.2	MARIADB ES 10.3	MARIADB ES 10.4	MARIADB ES 10.5
BIGINT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
BIT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
BOOL	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
BOOLEAN	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
DEC	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
DECIMAL	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
DOUBLE	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
DOUBLE PRECISION	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
FIXED	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
FLOAT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
FLOAT4	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
FLOAT8	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
INT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
INT1	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
INT2	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
INT3	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
INT4	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
INT8	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
INTEGER	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
LONG	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
MEDIUM INT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
MIDDLE INT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
NUMERIC	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
REAL	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
SERIAL	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
SMALLINT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
TINYINT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**

Character string types

	MYSQL 5.6	MYSQL 5.7	MYSQL 8.0	MARIADB ES 10.2	MARIADB ES 10.3	MARIADB ES 10.4	MARIADB ES 10.5
BINARY	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
CHAR	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
CHARACTER	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
CHARACTER VARYING	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
CHAR BYTE	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
CHAR VARYING	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
DOUBLE	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
JSON	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
LONGTEXT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
LONG CHARACTER VARYING	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
LONG CHAR VARYING	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
LONG VARBINARY	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
LONG VARCHAR	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
LONG VARCHARACTER	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
MEDIUM TEXT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
NATIONAL CHAR	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
NATIONAL CHARACTER	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
NATIONAL CHARACTER VARYING	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
NATIONAL CHAR VARYING	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
NATIONAL VARCHAR	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
NATIONAL VARCHARACTER	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
NCHAR VARYING	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
NVARCHAR	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
TEXT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
TINYTEXT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
VARBINARY	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
VARCHAR	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
VARCHARACTER	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**

LOB data types

	MYSQL 5.6	MYSQL 5.7	MYSQL 8.0	MARIADB ES 10.2	MARIADB ES 10.3	MARIADB ES 10.4	MARIADB ES 10.5
BLOB	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
LONGBLOB	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
MEDIUM BLOB	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
TINYBLOB	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**

Data and time

	MYSQL 5.6	MYSQL 5.7	MYSQL 8.0	MARIADB ES 10.2	MARIADB ES 10.3	MARIADB ES 10.4	MARIADB ES 10.5
DATE	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
DATETIME	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
TIME	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
TIMESTAMP	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
YEAR	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**

GIS (geometric) data types

	MYSQL 5.6	MYSQL 5.7	MYSQL 8.0	MARIADB ES 10.2	MARIADB ES 10.3	MARIADB ES 10.4	MARIADB ES 10.5
GEOMETRY	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
GEOMETRYCOLLECTION	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
LINESTRING	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
MULTILINESTRING	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
MULTIPOINT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
MULTIPOLYGON	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
POINT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
POLYGON	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**

SET and ENUM

	MYSQL 5.6	MYSQL 5.7	MYSQL 8.0	MARIADB ES 10.2	MARIADB ES 10.3	MARIADB ES 10.4	MARIADB ES 10.5
SET	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
ENUM	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**

Operators

	MYSQL 5.6	MYSQL 5.7	MYSQL 8.0	MARIADB ES 10.2	MARIADB ES 10.3	MARIADB ES 10.4	MARIADB ES 10.5
ALL	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
AND	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
ANY	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
BETWEEN	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
BINARY	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
DIV	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
EXISTS	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
IN	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
IS	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
IS NOT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
LIKE	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
MOD	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
NOT	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
NOT BETWEEN	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
NOT IN	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
NOT LIKE / NOT REGEXP	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
OR	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
REGEXP	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
RLIKE	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
SOME	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
SOUNDS LIKE	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
XOR	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**

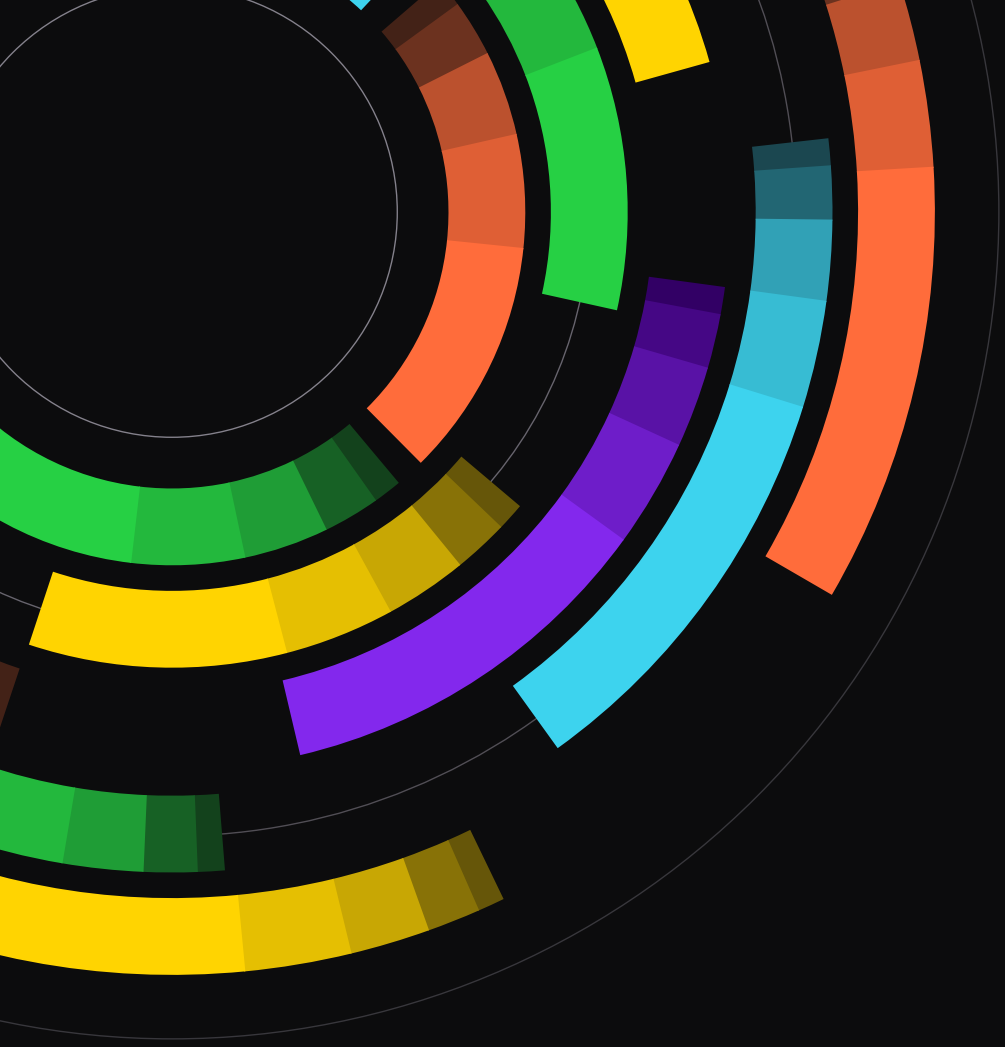
	MYSQL 5.6	MYSQL 5.7	MYSQL 8.0	MARIADB ES 10.2	MARIADB ES 10.3	MARIADB ES 10.4	MARIADB ES 10.5
&	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
&&	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
!	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
=/=	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
|	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
||	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
:=	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
-	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
=	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
>	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
⩾	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
>>	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
⩽	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
⟺	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
<>	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
<< %	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
+	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
/	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
*	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
~	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**
^	Yes	Yes	Yes	**Yes**	**Yes**	**Yes**	**Yes**

06

Conclusion

MariaDB Enterprise Server shares many features with MySQL Community Edition, including support for similar data types, operators, clients, and connectors. Since MySQL 8.0 deviates from the behavior of earlier MySQL versions, an opportunity may exist for customers to migrate with less burden to modern versions of MariaDB Enterprise Server.

While customers should always perform specific testing in support of an application migration, including both testing of data and application function, the presence of similar behavior in MariaDB Enterprise Server and MySQL Community Edition should enable a higher-level testing focus.



mariaDB

mariadb.com