Home
Open Questions
MariaDB Server
MariaDB MaxScale
MariaDB ColumnStore
Connectors

History
Source
Flag as Spam / Inappropriate
Translate

Created
12 years, 4 months ago
Modified
1 month, 1 week ago
Type
article
Status
active
License
CC BY-SA / Gnu FDL

History
Comments

Attachments Edit
No attachments exist

Localized Versions
- Acerca De MariaDB Connector/J [es]
- MariaDB Connector/Jについて [ja]

About MariaDB Connector/J

The most recent ***Stable (GA)*** release of MariaDB Connector/J is:
MariaDB Connector/J 3.5.3

Download MariaDB Connector/J

MariaDB Connector/J is used to connect applications developed in Java to MariaDB and MySQL databases using the standard JDBC API. The library is LGPL licensed.

Date	Release	Status	Min. Java Compat.	Release Notes	Changelog
27 Mar 2025	3.5.3	Stable (GA)	Java 8	Release Notes	Changelog
11 Feb 2025	3.5.2	Stable (GA)	Java 8	Release Notes	Changelog
27 Mar 2025	3.4.2	Stable (GA)	Java 8	Release Notes	Changelog
17 Jul 2024	3.4.1	Stable (GA)	Java 8	Release Notes	Changelog
27 Mar 2025	3.3.4	Stable (GA)	Java 8	Release Notes	Changelog
20 Feb 2024	3.3.3	Stable (GA)	Java 8	Release Notes	Changelog
25 Aug 2023	3.2.0	Stable (GA)	Java 8	Release Notes	Changelog

see all releases

About MariaDB Connector/J

MariaDB Connector/J is a Type 4 JDBC driver. It was developed specifically as a lightweight JDBC connector for use with MariaDB and MySQL database servers. It was originally based on the Drizzle JDBC code with numerous additions and bug fixes.

Server Compatibility

MariaDB Connector/J is compatible with all MariaDB and MySQL server versions.

MariaDB Connector/J releases older than 1.2.0 may be compatible with server versions older than MySQL 5.5, but those MariaDB Connector/J releases aren't supported anymore.

Java Compatibility

To determine which MariaDB Connector/J release series would be best to use for each Java version, please see the following table:

Java Version(s)	Recommended MariaDB Connector/J Release Series	JDBC Version
Java 21, Java 17, Java 11, Java 8	MariaDB Connector/J 3.5, 3.4, 3.3[1]	JDBC 4.2
Java 17, Java 11, Java 8	MariaDB Connector/J 2.7	JDBC 4.2

1. ↑ see parsec authentication restriction

Installing MariaDB Connector/J

MariaDB Connector/J can be installed using Maven, Gradle, or by manually putting the `.jar` file in your `CLASSPATH`. See Installing MariaDB Connector/J for more information.

MariaDB Connector/J `.jar` files and source code tarballs can be downloaded from the following URL:

- https://downloads.mariadb.org/connector-java/

MariaDB Connector/J `.jar` files can also be downloaded from the following URL:

- https://mariadb.com/downloads/#connectors

Installing Dependencies

JNA (net.java.dev.jna:jna) and JNA-PLATFORM (net.java.dev.jna:jna-platform) 4.2.1 or greater are also needed when you would like to connect to the server with Unix sockets or windows pipes.

Using the Driver

The following subsections show the formatting of JDBC connection strings for MariaDB and MySQL database servers. Additionally, sample code is provided that demonstrates how to connect to one of these servers and create a table.

Getting a New Connection

There are two standard ways to get a connection:

Using DriverManager

The preferred way to get a connection with MariaDB Connector/J is to use the `DriverManager` class. When the `DriverManager` class is used to locate and load MariaDB Connector/J, the application needs no further configuration. The `DriverManager` class will automatically load MariaDB Connector/J and allow it to be used in the same way as any other JDBC driver.

For example:

```
Connection connection = DriverManager.getConnection("jdb
```

> The legacy way of loading a JDBC driver also still works for MariaDB Connector/J. e.g.:
> `Class.forName("org.mariadb.jdbc.Driver")`

jdbc:mysql scheme compatibility

MariaDB Connector/J 3.0 only accepts `jdbc:mariadb:` as the protocol in connection strings by default. When both MariaDB Connector/J and the MySQL drivers are found in the class-path, using `jdbc:mariadb:` as the protocol helps to ensure that Java chooses MariaDB Connector/J.

Connector/J still allows `jdbc:mysql:` as the protocol in connection strings when the `permitMysqlScheme` option is set. For example:

```
jdbc:mysql://HOST/DATABASE?permitMysqlScheme
```

(2.x version did permit connection URLs beginning with both `jdbc:mariadb` and `jdbc:mysql`)

Using a Pool

Another way to get a connection with MariaDB Connector/J is to use a connection pool.

MariaDB Connector/J provides 2 different Datasource pool implementations:

- `MariaDbDataSource`: The basic implementation. It creates a new connection each time the `getConnection()` method is called.
- `MariaDbPoolDataSource`: A connection pool implementation. It maintains a pool of connections, and when a new connection is requested, one is borrowed from the pool.

Internal Pool

The driver's internal pool configuration provides a very fast pool implementation and deals with the issues most of the java pool have:

- 2 different connection states cleaning after release
- deals with non-activity (connections in the pool will be released if not used after some time, avoiding the issue created when the server closes the connection after @wait_timeout is reached).

See the pool documentation for more information.

External pool

When using an external connection pool, the MariaDB Driver class `org.mariadb.jdbc.Driver` must be configured.

Example using hikariCP JDBC connection pool :

```
        final HikariDataSource ds = new HikariDataSource
        ds.setMaximumPoolSize(20);
        ds.setDriverClassName("org.mariadb.jdbc.Driver")
        ds.setJdbcUrl("jdbc:mariadb://localhost:3306/db"
        ds.addDataSourceProperty("user", "root");
        ds.addDataSourceProperty("password", "myPassword
        ds.setAutoCommit(false);
```

Please note that the driver class provided by MariaDB Connector/J **is not** `com.mysql.jdbc.Driver` **but** `org.mariadb.jdbc.Driver`!

The `org.mariadb.jdbc.MariaDbDataSource` class can be used when the pool datasource configuration only permits the java.sql.Datasource implementation.

Connection Strings

The format of the JDBC connection string is:

```
jdbc:mariadb:[replication:|loadbalance:|sequential:|load-balance-read:]//<hostDescription>[,<ho
```

HostDescription:

```
<host>[:<portnumber>]  or address=(host=<host>|localSocket=<socket>|pipe=<namedpipe>)[(port=<po
```

Some notes about this:

- The host must be a DNS name or IP address.
- If the host is an IPv6 address, then it must be inside square brackets.
- The default port is `3306`.
- The default type is `master`.
- If the failover and load-balancing mode is set to `replication`, then the connector assumes that the first host is master, and the others are replicas by default, if their types are not explicitly mentioned.
- *aurora* failover prefix is available on 2.x version.
- A detailed host description option supersedes a global option description
- sslMode, pipe and localSocket are available since 3.4.1 version

Examples:

- `localhost:3306`
- `[2001:0660:7401:0200:0000:0000:0edf:bdd7]:3306`
- `somehost.com:3306`
- `address=(host=localhost)(port=3306)(type=master)`

The `jdbc:mariadb:sequential:address=(localSocket=/socket)(sslMode=disable),10.0.0.1:3306/DB?sslMode=verify-full` connection string will permit to connect to local unix socket if available, or to host 10.0.0.1 using SSL if not.

Failover and Load-Balancing Modes

Failover and Load-Balancing Modes were introduced in MariaDB Connector/J 1.2.0.

`sequential`

- **Description:** This mode supports connection failover in a multi-master environment, such as MariaDB Galera Cluster. This mode does **not** support load-balancing reads on replicas. The connector will try to connect to hosts in the order in which they were declared in the connection URL, so the first available host is used for all queries.
  For example, let's say that the connection URL is the following:
  `jdbc:mariadb:sequential:host1,host2,host3/testdb`
  When the connector tries to connect, it will always try host1 first. If that host is not available, then it will try host2. etc. When a host fails, the connector will try to reconnect to hosts in the same order.
- **Introduced:** 1.3.0

`loadbalance`

- **Description:** This mode supports connection load-balancing in a multi-master environment, such as MariaDB Galera Cluster. This mode does **not** support load-balancing reads on replicas. The connector performs load-balancing for all queries by randomly picking a host from the connection URL for each connection, so queries will be load-balanced as a result of the connections getting randomly distributed across all hosts.
  Before 2.4.2, this option was named `failover` - alias still exist for compatibility -
- **Introduced:** 1.2.0

`replication`

- **Description:** This mode supports connection failover in a primary-replica environment, such as a MariaDB Replication cluster. The mode supports environments with one or more masters. This mode does support load-balancing reads on replicas if the connection is set to read-only before executing the read. The connector performs load-balancing by randomly picking a replica from the connection URL to execute read queries for a connection
- **Introduced:** 1.2.0

`load-balance-read`

- **Description:** When running a multi-master cluster (i.e. Galera), writing to more than one node can lead to optimistic locking errors ("deadlocks"). Writing concurrently to multiple nodes also doesn't bring a whole lot of performance, due to having to (synchronously) replicate to all nodes anyway.

  This mode supports connection failover in a multi-master environment, such as MariaDB Galera Cluster. This mode does support load-balancing reads on replicas. The connector will try to connect to primary hosts in the order in which they were declared in the connection URL, so the first available host is used for all queries.
  For example, let's say that the connection URL is the following: `jdbc:mariadb:load-balance-read:primary1,primary2,address=(host=replica1)(type=replica),address=(host=replica2)(type=replica)/DB`
  When the connector tries to connect, it will always try primary1 first. If that host is not available, then it will try primary2. etc. When a primary host fails, the connector will try to reconnect to hosts in the same order.

  For replica hosts, the connector performs load-balancing for all queries by randomly picking a replica host from the connection URL for each connection, so queries will be load-balanced as a result of the connections getting randomly distributed across all replica hosts.
- **Introduced:** 3.5.1

`aurora`

- **Description:** This mode supports connection failover in an Amazon Aurora cluster. This mode does support load-balancing reads on replica instances if the connection is set to read-only before executing the read. The connector performs load-balancing by randomly picking a replica instance to execute read queries for a connection
- **Introduced:** 1.2.0 and **not supported anymore since 3.0 version**

---

driver 3.0 is a complete rewrite of the connector. Specific support for aurora has not been implemented in 3.0, since it relies on pipelining. Aurora is not compatible with pipelining. Issues for Aurora were piling up without the community proposing any PR for them and without access for us to test those modifications. (2.x version has a 5 years support).

See failover description for more information.

Optional URL Parameters

General remark: Unknown options are accepted and silently ignored.

The following options are currently supported.

Essential Parameters

`user`

- **Description:** User name.
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 1.0.0

`password`

- **Description:** password
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 1.0.0

`connectTimeout`

- **Description:** The connect timeout value, in milliseconds, or zero for no timeout
- **Data Type:** `integer`
- **Default Value:** `30 000`
- **Introduced:** 1.1.8

`useServerPrepStmts`

- **Description:** Text (default) is a globaly a safe default behavior, always working without issue. Binary protocol (useServerPrepStmts=true) has usually good benefits, but that depends:
  if missing cache, it will have an overhead of preparing before execution.
  If hitting cache, this perform better, but difference usually isn't huge, because most of the queries have simple execution plan.
  This is totally depending on queries, but to have some order of difference, here is some realistic differences : missing cache : 50% performance loss / hitting cache: 5-10% performance gain (because simple execution plan).

Another thing to consider : Since MariaDB 10.6 Server with MDEV-19237, server now permits to avoid resending metadata when they haven't changed when enabling useServerPrepStmts option (This concerns SQL commands that return a result-set). This avoids useless information transiting on the network and parsing those metadata, and that permit huge gain (around 10-30% depending on query, metadata can be huge compare to resultset data). So, if you use a MariaDB server version 10.6, and application doesn't execute completly differents queries, binary protocol (option 'useServerPrepStmts') is recommended.

- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 1.3.0

`allowLocalInfile`

- **Description:** Permit loading data from file. see LOAD DATA LOCAL INFILE. Having this option enable can impact batch performance. Disabling it can permit some batch improvement
- **Data Type:** `boolean`
- **Default Value:** `true`
- **Introduced:** 1.2.1

TLS Parameters

more information on Using TLS/SSL with MariaDB java connector

`sslMode`

- **Description:** Enables SSL/TLS in a specific mode.
  this option replaces the deprecated options: disableSslHostnameVerification, trustServerCertificate, useSsl
- **Data Type:** `string`
- **Default Value:** `disable`
- **Valid Values:**
  - disable: Do not use SSL/TLS (alias 'false', '0')
  - trust: Only use SSL/TLS for encryption. Do not perform certificate or hostname verification. (alias 'required')
  - verify-ca: Use SSL/TLS for encryption and perform certificates verification, but do not perform hostname verification. (alias 'verify_ca')
  - verify-full: Use SSL/TLS for encryption, certificate verification, and hostname verification (alias 'verify_identity', 'true', '1')
- **Introduced:** 3.0.0

`serverSslCert`

- **Description:** |Permits providing server's certificate in DER form, or server's CA certificate. The server will be added to trustStore. This permits a self-signed certificate to be trusted.
  Can be used in one of 3 forms :
  * serverSslCert=/path/to/cert.pem (full path to certificate)
  * serverSslCert=classpath:relative/cert.pem (relative to current classpath)
  * or as verbatim DER-encoded certificate string "------BEGIN CERTIFICATE-----"
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 1.1.0

`keyStore`

- **Description:** File path of the keyStore file that contain client private key store and associate certificates (similar to java System property "javax.net.ssl.keyStore", but ensure that only the private key's entries are used).
- **Data Type:** `string`
- **Default Value:** `null`
- **Alias:** clientCertificateKeyStoreUrl
- **Introduced:** 1.1.1

`keyStorePassword`

- **Description:** Password for the client certificate keyStore (similar to java System property "javax.net.ssl.keyStorePassword")
- **Data Type:** `string`
- **Default Value:** `null`
- **Alias:** clientCertificateKeyStorePassword
- **Introduced:** 1.3.4

`enabledSslCipherSuites`

- **Description:** Force TLS/SSL cipher (comma separated list).
  Example : "TLS_DHE_RSA_WITH_AES_256_GCM_SHA384, TLS_DHE_DSS_WITH_AES_256_GCM_SHA384"
- **Data Type:** `string`
- **Default Value:** `use JRE ciphers`
- **Introduced:** 1.5.0

`enabledSslProtocolSuites`

- **Description:** Force TLS/SSL protocol to a specific set of TLS versions (comma separated list).
  Example : "TLSv1,TLSv1.1,TLSv1.2
- **Data Type:** `string`
- **Default Value:** `use JRE default`
- **Alias:** enabledSSLProtocolSuites
- **Introduced:** 1.5.0

`disableSslHostnameVerification`

- **Description:** *deprecated, use sslMode instead*
  When using ssl, the driver checks the hostname against the server's identity as presented in the server's certificate (checking alternative names or the certificate CN) to prevent man-in-the-middle attacks. This option permits deactivating this validation. Hostname verification is disabled when the trustServerCertificate option is set
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 2.1.0
- **Deprecated:** 3.0.0

`useSsl`

- **Description:** *deprecated, use sslMode instead*
  Force SSL/TLS on connection(useSSL can be used as alias).
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 1.1.0
- **Deprecated:** 3.0.0

`trustServerCertificate`

- **Description:** *deprecated, use sslMode instead*
  When using SSL/TLS, do not check server's certificate
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 1.1.1
- **Deprecated:** 3.0.0

Pool Parameters

See the pool documentation for pool configuration.

`pool`

- **Description:** Use pool. This option is useful only if not using a DataSource object, but only a connection object
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 2.2.0

`poolName`

- **Description:** Pool name that permits identifying threads.
  default: auto-generated as MariaDb-pool-<pool-index>
- **Data Type:** `string`
- **Default Value:** `MariaDB-pool`
- **Introduced:** 2.2.0

`maxPoolSize`

- **Description:** The maximum number of physical connections that the pool should contain
- **Data Type:** `integer`
- **Default Value:** `8`
- **Introduced:** 2.2.0

`minPoolSize`

- **Description:** When connections are removed due to not being used for longer than than "maxIdleTime", connections are closed and removed from the pool. "minPoolSize" indicates the number of physical connections the pool should keep available at all times. Should be less or equal to maxPoolSize.
- **Data Type:** `integer`
- **Default Value:** `maxPoolSize value`
- **Introduced:** 2.2.0

`poolValidMinDelay`

- **Description:** When asking a connection to pool, the pool will validate the connection state. "poolValidMinDelay" permits disabling this validation if the connection has been borrowed recently avoiding useless verifications in case of frequent reuse of connections. 0 means validation is done each time the connection is asked. In milleseconds.
- **Data Type:** `integer`
- **Default Value:** `1000`
- **Introduced:** 2.2.0

`maxIdleTime`

- **Description:** The maximum amount of time in seconds that a connection can stay in the pool when not used. This value must always be below @wait_timeout value - 45s
- **Data Type:** `integer`
- **Default Value:** `600` minimum value is 60 seconds
- **Introduced:** 2.2.0

`useResetConnection`

- **Description:** When a connection is closed() (given back to pool), the pool resets the connection state. Setting this option, the prepare command will be deleted, session variables changed will be reset, and user variables will be destroyed when the server permits it (>= MySQL 5.7.3), permitting saving memory on the server if the application make extensive use of variables
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 2.2.0

`registerJmxPool`

- **Description:** Register JMX monitoring pools
- **Data Type:** `boolean`
- **Default Value:** `true`
- **Introduced:** 2.2.0

Infrequently Used Parameters

`trustStore`

- **Description:** File path of the trustStore file (similar to java System property "javax.net.ssl.trustStore"). (legacy alias trustCertificateKeyStoreUrl)
  Use the specified file for trusted root certificates.
  When set, overrides serverSslCert. (see trustStorePassword in case if a jks truststore with a password)
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 3.5.0 (or 1.3.4 in 1.x, 2.0.0 in 2.x)

`trustStorePassword`

- **Description:** Password for the trusted root certificate file (similar to java System property "javax.net.ssl.trustStorePassword").
  (legacy alias trustCertificateKeyStorePassword).
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 3.5.0 (or 1.3.4 in 1.x, 2.0.0 in 2.x)

`trustStoreType`

- **Description:** Indicate trust store type (JKS/PKCS12). default is null, then using java default type. (legacy alias trustCertificateKeystoreType).
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 3.5.0 (or 2.4.0 in 2.x)

`useMysqlMetadata`

- **Description:** databaseMetaData.getDatabaseProductName() return "MariaDB" or "MySQL" according to server type
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 2.4.0

`restrictedAuth`

- **Description:** permits to restrict authentication plugins (comma separated). For example, the following connection string only allows the mysql_native_password and client_ed25519 client authentication plugins:`jdbc:mariadb:*HOST/DATABASE?restrictedAuth=mysql_native_password,client_ed25519*`. *If not set, permit all authentication plugins.*
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 3.0.0

`maxQuerySizeToLog`

- **Description:** Only the first characters corresponding to this options size will be displayed in logs
- **Data Type:** `integer`
- **Default Value:** `1024`
- **Introduced:** 1.5.0

`allowMultiQueries`

- **Description:** permit multi-queries like `insert into ab (i) values (1); insert into ab (i) values (2).`
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 1.0.0

`dumpQueriesOnException`

- **Description:** If set to 'true', an exception is thrown during query execution containing a query string. This is useful in development, but can lead to security issue if logs are available.
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 1.1.0

`useCompression`

- **Description:** Compresses the exchange with the database through gzip. This permits better performance when the database is not in the same location.
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 1.0.0

`socketFactory`

- **Description:** to use a custom socket factory, set it to the full name of the class that implements javax.net.SocketFactory
- **Introduced:** 1.1.0

`tcpKeepAlive`

- **Description:** Sets corresponding option on the connection socket. Default to true since 3.0.0 (was false before)
- **Data Type:** `boolean`
- **Default Value:** `true`
- **Introduced:** 1.0.0

`tcpAbortiveClose`

- **Description:** This option can be used in environments where connections are created and closed in rapid succession. Often, it is not possible to create a socket in such an environment after a while, since all local “ephemeral” ports are used up by TCP connections in TCP_WAIT state. Using tcpAbortiveClose works around this problem by resetting TCP connections (abortive or hard close) rather than doing an orderly close. It is accomplished by using socket.setSoLinger(true,0) for abortive close.
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 1.1.1

`pipe`

- **Description:** On Windows, specify named pipe name to connect (windows equivalent of unix socket)
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 1.1.3

`tinyInt1isBit`

- **Description:** Datatype mapping flag, handle MySQL Tiny as BIT(boolean).
- **Data Type:** `boolean`
- **Default Value:** `true`
- **Introduced:** 1.0.0

`yearIsDateType`

- **Description:** returns Year as date type, rather than numerical.
- **Data Type:** `boolean`
- **Default Value:** `true`
- **Introduced:** 1.0.0

`sessionVariables`

- **Description:** <var>=<value> pairs separated by comma, mysql session variables, set upon establishing successful connection.
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 1.1.4

`localSocket`

- **Description:** Permits connecting to the database via Unix domain socket, if the server allows it.
  The value is the path of Unix domain socket (i.e "socket" database parameter : select @@socket) .
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 1.1.4

`localSocketAddress`

- **Description:** Hostname or IP address to bind the connection socket to a local (UNIX domain) socket.
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 1.1.7

`socketTimeout`

- **Description:** Defined the network socket timeout (SO_TIMEOUT) in milliseconds. Value of 0 disables this timeout. If the goal is to set a timeout for all queries, the server has permitted a solution to limit the query time by setting a system variable, max_statement_time. The advantage is that the connection then is still usable.
- **Data Type:** `integer`
- **Default Value:** `0`
- **Introduced:** 1.1.7

`socketTimeout`

- **Description:** Defined the network socket timeout (SO_TIMEOUT) in milliseconds. Value of 0 disables this timeout. If the goal is to set a timeout for all queries, the server has permitted a solution to limit the query time by setting a system variable, max_statement_time. The advantage is that the connection then is still usable.
- **Data Type:** `integer`
- **Default Value:** `0`
- **Introduced:** 1.1.7

`createDatabaseIfNotExist`

- **Description:** the specified database in the url will be created if nonexistent.
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 1.1.7

`cacheCallableStmts`

- **Description:**enable/disable callable Statement cache
- **Data Type:** `boolean`
- **Default Value:** `true`
- **Introduced:** 1.4.0

`connectionAttributes`

- **Description:** When performance_schema is active, permit to send server some client information in a key;value pair format (example: connectionAttributes=key1:value1,key2,value2).
  Those informations can be retrieved on server within tables performance_schema.session_connect_attrs and performance_schema.session_account_connect_attrs.
  This can permit from server an identification of client/application
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 1.4.0

`usePipelineAuth`

- **Description:** Not compatible with *aurora**
  During connection, different queries are executed. When option is active those queries are send using pipeline (all queries are send, then only all results are reads), permitting faster connection creation.
- **Data Type:** `boolean`
- **Default Value:** `true`
- **Introduced:** 1.6.0

`autocommit`

- **Description:** Set default autocommit value on connection initialization.
- **Data Type:** `boolean`
- **Default Value:** `true`
- **Introduced:** 2.2.0

`galeraAllowedState`

- **Description:** Usually, Connection.isValid just send an empty packet to server, and server send a small response to ensure connectivity. When this option is set, connector will ensure Galera server state "wsrep_local_state" correspond to allowed values (separated by comma). example "4,5", recommended is "4". see galera state to know more
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 2.2.5

`includeInnodbStatusInDeadlockExceptions`

- **Description:** add "SHOW ENGINE INNODB STATUS" result to exception trace when having a deadlock exception.
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 2.3.0

`includeThreadDumpInDeadlockExceptions`

- **Description:** add thread dump to exception trace when having a deadlock exception.
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 2.3.0

`useReadAheadInput`

- **Description:** Use a buffered inputSteam that read socket available data
- **Data Type:** `boolean`
- **Default Value:** `true`
- **Introduced:** 2.4.0

`servicePrincipalName`

- **Description:** When using GSSAPI authentication, use this value as the Service Principal Name (SPN) instead of the one defined for the user account on the database server.
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 2.4.0

`useMysqlMetadata`

- **Description:** force DatabaseMetadata.getDatabaseProductName() to return "MySQL" as database, not real database type.
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 2.4.1

`defaultFetchSize`

- **Description:** The driver will call setFetchSize(n) with this value on all newly-created Statements
- **Data Type:** `integer`
- **Default Value:** `0`
- **Introduced:** 2.4.2

`blankTableNameMeta`

- **Description:** Resultset metadata getTableName always return blank. This option is mainly for ORACLE db compatibility.
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 2.4.3

`serverRsaPublicKeyFile`

- **Description:** Indicate path to RSA server public key file for sha256_password and caching_sha2_password authentication password
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 2.5.0

`allowPublicKeyRetrieval`

- **Description:** Authorize client to retrieve RSA server public key when serverRsaPublicKeyFile is not set (for sha256_password and caching_sha2_password authentication password)
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 2.5.0

`tlsSocketType`

- **Description:** Indicate the TLS org.mariadb.jdbc.tls.TlsSocketPlugin plugin type to use. Plugin must be present in classpath
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 2.5.0

`credentialType`

- **Description:** Indicate the credential plugin type to use. Plugin must be present in classpath
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 2.5.0

`tcpKeepCount`

- **Description:** Permit to set socket option TCP_KEEPCOUNT (only if java 11+)
- **Data Type:** `integer`
- **Default Value:** `0`
- **Introduced:** 3.0.0

`tcpKeepIdle`

- **Description:** Permit to set socket option TCP_KEEPIDLE (only if java 11+)
- **Data Type:** `integer`
- **Default Value:** `0`
- **Introduced:** 3.0.0

`tcpKeepInterval`

- **Description:** Permit to set socket option TCP_KEEPINTERVAL (only if java 11+)
- **Data Type:** `integer`
- **Default Value:** `0`
- **Introduced:** 3.0.0

`permitMysqlScheme`

- **Description:** when added to connection string, permit `jdbc:mysql:` prefix in connection string
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 3.0.0

`prepStmtCacheSize`

- **Description:** When useServerPrepStmts is enabled, any positive value indicates that a prepared statement cache of the specified size will be used. If the value is less than or equal to zero, the cache will not be enabled. Before 3.0, an option cachePrepStmts was indicatin if cache has to be enable
- **Data Type:** `integer`
- **Default Value:** `250`
- **Introduced:** 1.3.0

`transactionReplay`

- **Description:** Enables transaction caching. If a failover occurs before a transaction is committed or rolled back, the transaction's cached statements are re-executed on the new primary server. Connector/J requires that applications only use idempotent queries. If the number of statements in the transaction cache exceeds transactionReplaySize, caching will be disabled until the transaction is committed or rolled back.
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 3.0.0

`transactionReplaySize`

- **Description:** Sets the number of statements that should be saved in the transaction cache when transactionReplay is enabled.
- **Data Type:** `integer`
- **Default Value:** `64`
- **Introduced:** 3.0.0

`useBulkStmts`

- **Description:** Use dedicated COM_STMT_BULK_EXECUTE protocol for batch insert when possible. (batch without Statement.RETURN_GENERATED_KEYS and streams) to have faster batch.
- **Data Type:** `boolean`
- **Default Value:** `true`
- **Introduced:** 3.0.0 (was false since version >= 2.3.0)

`useCatalogTerm`

- **Description:** |"schema" and "database" are server synonymous. Connector historically get/set database using Connection.setCatalog()/getCatalog(), setSchema()/getSchema() being no-op. Setting option useCatalogTerm to "schema" will change that behavior to use Schema in place of Catalog. Affected changes : database change will be done with either Connection.setCatalog()/getCatalog() or Connection.setSchema()/getSchema(), 2: DatabaseMetadata methods that use catalog or schema filtering, 3: ResultsetMetadata getCatalogName/getSchemaName
- **Data Type:** `string`
- **Default Value:** `CATALOG`
- **Introduced:** 3.2.0

`returnMultiValuesGeneratedIds`

- **Description:** for connector 2.x compatibility only, getGeneratedKeys() will then returns all ids of multi-value inserts. This is not compatible with galera servers
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 3.3.2

`pinGlobalTxToPhysicalConnection`

- **Description:** When set, commands with a specific XID will reuse previous connection used for this XID.
- **Data Type:** `boolean`
- **Default Value:** `false`
- **Introduced:** 3.4.1

`connectionCollation`

- **Description:** Connector force utf8mb4 charset at connection. Indicate what utf8mb4 collation to use if set. if not set, server default collation for utf8mb4 will be used.Useful only for server before MariaDB 11.4, because then a better solution would be to set character_set_collations
- **Data Type:** `string`
- **Default Value:** `null`
- **Introduced:** 3.5.0

`disconnectOnExpiredPasswords`

- **Description:** On connection creation, indicate behavior when password is expired. When true (default) throw an expired password error. When false, connection succeed in "sandbox" mode, only queries related to password change are allowed.
- **Data Type:** `boolean`
- **Default Value:** `true`
- **Introduced:** 3.5.2

`permitNoResults`

- **Description:** Indicate if Statement/PreparedStatement.executeQuery for command that produce no result will return an exception or just an empty result-set. When enabled, command not returning no data will end returning an empty result-set, when disabled, command not returning no data will end throwing an exception
- **Data Type:** `boolean`
- **Default Value:** `true`
- **Introduced:** 3.5.2

`oldModeNoPrecisionTimestamp`

- **Description:** When enable, Timestamps string representation will be compatible with 2.7's behavior (fractional part will only be displayed if required, not according to timestamp precision) .
- **Data Type:** `boolean`
- **Default Value:** `false`

- **Introduced:** 3.5.3

removed option

Parameter	Description
`allowMasterDownConnection`	When the `replication` Failover and Load Balancing Mode is in use, allow the creation of connections when the master is down. If no masters are available, then the default connection will be a replica, and Connection.isReadOnly() will return true. *Default: false. Since 2.2.0, removed in 3.0.0*
`interactiveClient`	Session timeout is defined by the wait_timeout server variable. Setting interactiveClient to true will tell the server to use the interactive_timeout server variable. *Default: false. Since 1.1.7*
`assureReadOnly`	When this parameter enabled when a Failover and Load Balancing Mode is in use, and a read-only connection is made to a host, assure that this connection is in read-only mode by setting the session to read-only. Default to false. Since 1.3.0, removed in 3.0.0
`autoReconnect`	If this parameter is enabled and Failover and Load Balancing Mode is **not** in use, the connector will simply try to reconnect to its host after a failure. This is referred to as **Basic Failover**. If this parameter is enabled and Failover and Load Balancing Mode is in use, the connector will blacklist the failed host and try to connect to a different host of the same type. This is referred to as **Standard Failover**. Default is false. since 1.1.7, removed in 3.0.0
`cachePrepStmts`	if useServerPrepStmts = true, cache the prepared informations in a LRU cache to avoid re-preparation of command. Next use of that command, only prepared identifier and parameters (if any) will be sent to server. This mainly permit for server to avoid reparsing query. *Default: true. Since 1.3.0, removed in 3.0.0*
`callableStmtCacheSize`	This sets the number of callable statements that the driver will cache per VM if "cacheCallableStmts" is enabled. *Default: true. Since 1.4.0, removed in 3.0.0*
`enablePacketDebug`	Driver will save the last 16 MySQL packet exchanges (limited to first 1000 bytes). Hexadecimal value of those packets will be added to stacktrace when an IOException occur. This option has no impact on performance but driver will then take 16kb more memory. *Default: false. Since 1.6.0, 2.0.1, removed in 3.0.0*
`failoverLoopRetries`	When the connector is searching silently for a valid host, this parameter defines the maximum number of connection attempts the connector will make before throwing an exception. This parameter differs from the "retriesAllDown" parameter because this silent search is used in situations where the connector can temporarily workaround the problem, such as by using the master connection to execute reads when the replica connection fails. Default: 120. since 1.2.0, removed in 3.0.0
`jdbcCompliantTruncation`	Truncation error ("Data truncated for column '%' at row %", "Out of range value for column '%' at row %") will be thrown as an error, and not as a warning. *Default: true. Since 1.4.0*
`keyPassword`	Password for the private key in client certificate keyStore. (only needed if private key password differ from keyStore password). *Since 1.5.3, removed in 3.0.0*
`loadBalanceBlacklistTimeout`	When a connection fails, this host will be blacklisted for the amount of time defined by this parameter. When connecting to a host, the driver will try to connect to a host in the list of non-blacklisted hosts and, only if none are found, attempt blacklisted ones. This blacklist is shared inside the classloader. Default: 50 seconds. since 1.2.0, removed in 3.0.0
`log`	Enable log information. *require Slf4j version > 1.4 dependency.* Log level correspond to Slf4j logging implementation *Default: false. Since 1.5.0, removed in 3.0.0*
`passwordCharacterEncoding`	Indicate password encoding charset. Charset value must be a Java charset. Example : "UTF-8" *Default: null (= platform's default charset) . Since 1.5.9, removed in 3.0.0*
`prepStmtCacheSqlLimit`	if useServerPrepStmts = true, defined queries larger than this size will not be cached. *Default: 2048. Since 1.3.0*
`profileSql`	log query execution time. *Default: false. Since 1.5.0, removed in 3.0.0*
`slowQueryThresholdNanos`	Will log query with execution time superior to this value (if defined) *Default: 1024. Since 1.5.0, removed in 3.0.0*
`retriesAllDown`	When the connector is performing a failover and all hosts are down, this parameter defines the maximum number of connection attempts the connector will make before throwing an exception. Default: 120 seconds. since 1.2.0, removed in 3.0.0

`rewriteBatchedStatements`	For insert queries, rewrite batchedStatement to execute in a single executeQuery. example:
	`insert into ab (i) values (?)` with first batch values = 1, second = 2 will be rewritten
	`insert into ab (i) values (1), (2).`
	
	If query cannot be rewriten in "multi-values", rewrite will use multi-queries : `INSERT INTO TABLE(col1) VALUES (?) ON DUPLICATE KEY UPDATE col2=?` with values [1,2] and [2,3]" will be rewritten
	`INSERT INTO TABLE(col1) VALUES (1) ON DUPLICATE KEY UPDATE col2=2; INSERT INTO TABLE(col1) VALUES (3) ON DUPLICATE KEY UPDATE col2=4`
	
	when active, the useServerPrepStmts option is set to false
	Default: false. Since 1.1.8, removed in 3.0.0

`serverTimezone`	Defines the server time zone. to use only if the jre server has a different time implementation of the server. (best to have the same server time zone when possible). *since 1.1.7, removed in 3.0.0*
	Permits connecting to the database via shared memory, if the server allows it.

`sharedMemory`	The value is the base name of the shared memory. *since 1.1.4, removed in 3.0.0*
`staticGlobal`	Indicates the values of the global variables max_allowed_packet, wait_timeout, autocommit, auto_increment_increment, time_zone, system_time_zone and tx_isolation) won't be changed, permitting the pool to create new connections faster. *Default: false. Since 2.2.0, removed in 3.0.0*
`tcpNoDelay`	Sets corresponding option on the connection socket. *since 1.0.0, removed in 3.0.0*
`tcpRcvBuf`	set buffer size for TCP buffer (SO_RCVBUF). *since 1.0.0, removed in 3.0.0*
`tcpSndBuf`	set buffer size for TCP buffer (SO_SNDBUF). *since 1.0.0, removed in 3.0.0*
`trackSchema`	Permit to disabled "session_track_schema" setting when server has CLIENT_SESSION_TRACK capability *Default: True. Since 2.5.4, removed in 3.0.0*
`useBatchMultiSend`	*Not compatible with *aurora** Driver will can send queries by batch. If set to `false`, queries are sent one by one, waiting for the result before sending the next one. If set to `true`, queries will be sent by batch corresponding to the useBatchMultiSendNumber option value (default 100) or according to the max_allowed_packet server variable if the packet size does not permit sending as many queries. Results will be read later, avoiding a lot of network latency when the client and server aren't on the same host. This option is mainly effective when the client is distant from the server. More information here *Default: true (false if using* aurora *failover) . Since 1.5.0, removed in 3.0.0*
`useBatchMultiSendNumber`	When option useBatchMultiSend is active, indicate the maximum query send in a row before reading results. *Default: 100. Since 1.5.0*
`useFractionalSeconds`	Correctly handle subsecond precision in timestamps (feature available with MariaDB 5.3 and later). May confuse 3rd party components (Hibernated). *Default: true. Since 1.0.0*
`useOldAliasMetadataBehavior`	Metadata ResultSetMetaData.getTableName() returns the physical table name. "useOldAliasMetadataBehavior" permits activating the legacy code that sends the table alias if set. *Default: false. Since 1.1.9*
`validConnectionTimeout`	When multiple hosts are configured, the connector verifies that the connections haven't been lost after this much time in seconds has elapsed. When this parameter is set to 0, no verification will be done. Default:120 seconds since 1.2.0, removed in 3.0.0

JDBC API Implementation Notes

Size consideration

GSSAPI in windows isn't well supported in java, causing recurrent issues. Since 3.1, waffle-jna is marked as a dependency to provide good GSSAPI support without problems. This has the drawback to make connector and dependencies to a size of around 4Mb.

If size is important, the dependency can be removed, the connector working great, just will have some limitation using GSSAPI on windows :

this can be done like this:

- using maven

```
<dependency>
        <groupId>org.mariadb.jdbc</groupId>
        <artifactId>mariadb-java-client</artifactId>
        <version>3.1.0</version>
        <exclusions>
          <exclusion>
            <groupId>com.github.waffle</groupId>
            <artifactId>waffle-jna</artifactId>
          </exclusion>
      </exclusions>
</dependency>
```

- using graddle:

```
dependencies {
    implementation('org.mariadb.jdbc:mariadb-java-client:3.1.0') {
        exclude group: 'com.github.waffle', module: 'waffle-jna'
    }
}
```

Parsec authentication

Since 3.5.1, parsec authentication is implemented in connector. This requires java 15+ (to use java native ed25519 Algorithm implementation).

In order to use parsec authentication with previous version of java, BouncyCastle is required as dependency:

```
        <dependency>
            <groupId>org.bouncycastle</groupId>
            <artifactId>bcpkix-jdk18on</artifactId>
            <version>1.78.1</version>
        </dependency>
```

Timezone consideration

The Ideal Scenario

The simplest approach to avoid time zone headaches is for the client and server to operate in the same time zone.

Java Connector Timezone Options

There are 3 options that control timestamps behavior in the java connector:

- connectionTimeZone: (LOCAL | SERVER | <user-defined time zone>) - This option defines the connection's time zone. LOCAL retrieves the JVM's default time zone, SERVER fetches the server's global time zone upon connection creation, and <user-defined time zone> allows specifying a server time zone without requesting it during connection establishment.

- forceConnectionTimeZoneToSession: (true | false) - This setting dictates whether the connector enforces the connection time zone for the session.
- preserveInstants: (true | false) - This option controls whether the connector converts Timestamp values to the connection's time zone.

Recommendation

By default, the connector adopts the JVM's default time zone. If the client and server reside in different time zones, it's recommended to configure the connection time zone to match the JVM's default by setting forceConnectionTimeZoneToSession to true. This ensures proper operation of time functions.

(This isn't the default behavior because there is a server Requirements to set tzinfo depending on the JVM's time zones)

TIMESTAMP vs. DATETIME: Know the Difference

Just like Java's Instant and LocalDateTime, server-side TIMESTAMP and DATETIME fields serve distinct purposes. One represents a specific point in time (a moment), while the other doesn't.

- TIMESTAMP: This represents an exact moment on the timeline, expressed using the connection's time zone. When stored, it gets converted to UTC (Coordinated Universal Time) for consistency. Upon retrieval, it's converted back to the connection's time zone for display.
- DATETIME: This combines date and time-of-day information but doesn't represent a specific moment.

The Misuse of DATETIME:

Due to its wider range, DATETIME is sometimes mistakenly used to store a specific point in time. While this might work if the client and server share the same time zone, it creates problems when they differ.

A (Discouraged) Workaround:

While using DATETIME instead of TIMESTAMP is generally discouraged, a specific combination of settings ("preserveInstants=true&connectionTimeZone=SERVER") can force all Java Timestamp exchanges to be converted to the connection's time zone during storage and retrieval. However, this approach is not recommended for long-term solutions.

Compatibility with Older Connectors (Pre-3.4):

The MariaDB Connector/J versions before 3.4 offered a single "timezone" option. While this functionality remains compatible, it's now separated into two distinct settings: connectionTimeZone and forceConnectionTimeZoneToSession. Here's a breakdown of how the old option translates to the new ones: "timezone=America/Los_Angeles" is equivalent to "connectionTimeZone=America/Los_Angeles&forceConnectionTimeZone=true

To mimic the behavior of the "useLegacyDatetimeCode=false" option from MariaDB 2.x, you can set the following combination: “connectionTimeZone=SERVER&preserveInstants=true”

Note: Unlike the MySQL Connector, the MariaDB Connector/J defaults connectionTimeZone to LOCAL (JVM's default) instead of SERVER.

"LOAD DATA INFILE"

> LOAD DATA INFILE was the fastest way to load data. Now the fastest way is the standard JDBC executeBatch() when option 'useBulkStmts' is enabled

The fastest way to load lots of data is using LOAD DATA INFILE.
However, using "LOAD DATA LOCAL INFILE" (ie: loading a file from the client) may be a security problem if someone can execute a query from the client, he can have access to any file on the client (according to the rights of the user running the client process).

A specific option "allowLocalInfile" (default to true) can disable this functionality on the client side. The global variable local_infile can disable LOAD DATA LOCAL INFILE on the server side.

You can provide custom stream as well using a specific setLocalInfileInputStream

```
        Statement statement = connection.createStatement();
        org.mariadb.jdbc.Statement mariaDbStatement =
            statement.unwrap(org.mariadb.jdbc.Statement.class);
        mariaDbStatement.setLocalInfileInputStream(in);

        String sql =
            "LOAD DATA LOCAL INFILE 'notUsed'"
                + " INTO TABLE myTable "
                + " FIELDS TERMINATED BY '\\t' ENCLOSED BY ''"
                + " ESCAPED BY '\\\\' LINES TERMINATED BY '\\n'";
        statement.execute(sql);
```

Contrary to mysql connector, setLocalInfileInputStream value can only be used for next execution.

Set a Query Timeout

Driver follow the JDBC specifications, permitting Statement.setQueryTimeout() for a particular statement.

If the goal is to set a timeout for all queries, the server permits a limiting query time by setting the system variable max_statement_time.

This solution will handle query timeout better (and faster) than java solutions (JPA2, "javax.persistence.query.timeout", Pools integrated solution like tomcat jdbc-pool "queryTimeout"...).

Option "sessionVariables" permit to set this system variable easily : Example :

```
#will set a maximum query timeout of 10 seconds for this connection
jdbc:mariadb://localhost/db?user=user&sessionVariables=max_statement_time=10
```

Streaming Result Sets

By default, `Statement.executeQuery()` will read the full result set from the server. With large result sets, this will require large amounts of memory.

To avoid using too much memory, rather use Statement.setFetchSize(int numberOfRowInMemory) to indicate the number of rows that will be stored in memory
Example :
using `Statement.setFetchSize(1000)` indicates that 1000 rows will be stored in memory.
So, when the query has executed, 1000 rows will be in memory. After 1000 `ResultSet.next()`, the next 1000 rows will be stored in memory, and so on.

If another query is run on same connection while the resultset has not been completly read, the connector will fetch all remaining rows before executing the query. This can lead to still needing lots of memory. Recommendation is then to use another connection for simultaneous operations.

Note that the server usually expects clients to read off the result set relatively quickly. The net_write_timeout server variable controls this behavior (defaults to 60s). If you don't expect results to be handled in this amount of time there is a different possibility:

- With you can use the query "SET STATEMENT net_write_timeout=10000 FOR XXX" with XXX your "normal" query. This will indicate that specifically for this query, net_write_timeout will be set to a longer time (10000 in this example).
- for older servers, a specific query will have to temporarily set net_write_timeout ("SET STATEMENT net_write_timeout=..."), and set it back afterward.
- if your application usually uses a lot of long queries with fetch size, the connection can be set using option "sessionVariables=net_write_timeout=xxx"

Even using setFetchSize, the server will send all results to the client.

If another query is executed on the same connection when a streaming resultset has not been fully read, the connector will put the whole remaining streaming resultset in memory in order to execute the next query. This can lead to OutOfMemoryError if not handled.

Before version 1.4.0, the only accepted value for fetch size was `Statement.setFetchSize(Integer.MIN_VALUE)` (equivalent to `Statement.setFetchSize(1)`). This value is still accepted for compatibility reasons but rather use `Statement.setFetchSize(1)` , since according to JDBC the value must be >= 0.

Prepared Statements

The driver uses server prepared statements as a standard to communicate with the database (since 1.3.0). If the "allowMultiQueries" options are set to true, the driver will only use text protocol. Prepared statements (parameter substitution) is handled by the driver, on the client side.

CallableStatement

Callable statement implementation won't need to access stored procedure metadata (mysql.proc) table if both of following are true

- CallableStatement.getMetadata() is not used
- Parameters are accessed by index, not by name

When possible, following the two rules above provides both better speed and eliminates concerns about SELECT privileges on the mysql.proc table.

Generated keys limitation

Java permit retrieving last generated keys,using Statement.getGeneratedKeys().

Example:

```
    Statement stmt = sharedConn.createStatement();
    stmt.execute(
            "INSERT INTO executeGenerated(t2) values (100)", Statement.RETURN_GENERATED_KEYS);
    ResultSet rs = stmt.getGeneratedKeys();
    rs.next();
    System.out.println(rs.getInt(1));
```

Only the the first generated key will be returned, meaning that for multi-insert the generated key retrieved will correspond to the first generated value of the command.

If retrieving all generated values for multiple insert is needed, please use INSERT...RETURNING command (since MariaDB 10.5).

Optional JDBC Classes

The following optional interfaces are implemented by the org.mariadb.jdbc.MariaDbDataSource class : javax.sql.DataSource, javax.sql.ConnectionPoolDataSource, javax.sql.XADataSource

careful : org.mariadb.jdbc.MySQLDataSource doesn't exist anymore and should be replaced with org.mariadb.jdbc.MariaDbDataSource since v1.3.0

Usage Examples

The following code provides a basic example of how to connect to a MariaDB or MySQL server and create a table.

Creating a Table on a MariaDB or MySQL Server

```
Connection  connection = DriverManager.getConnection("jdbc:mariadb://localhost:3306/test", "use
Statement stmt = connection.createStatement();
stmt.executeUpdate("CREATE TABLE a (id int not null primary key, value varchar(20))");
stmt.close();
connection.close();
```

Services

The driver implements 3 kinds of services:

- Credential service: permit giving credential
- Authentication service: permit adding client authentication plugins.
- SSL factory service: custom TSL implementation

Credential service

Credentials are usually set using user/password in the connection string or by using DriverManager.getConnection(String url, String user, String password).

Credential plugins permit to provide credential information from other means. Those plugins have to be activated setting option `credentialType` to designated plugin.

The driver has 3 default plugins :

AWS IAM

This permits AWS database IAM authentication. The plugin generate a token using IAM credential and region. Token is valid for 15 minutes and cached for 10 minutes.

To use this credential authentication, com.amazonaws:aws-java-sdk-rds dependency must be registred in classpath. Implementation use SDK DefaultAWSCredentialsProviderChain and DefaultAwsRegionProviderChain to get IAM credential and region. see DefaultAWSCredentialsProviderChain and DefaultAwsRegionProviderChain to check how those information can be retrieved (environment variable / system properties, files, ...)

Example: `jdbc:mariadb://host/db?credentialType=AWS-IAM&useSsl&serverSslCert=/somepath/rds-combined-ca-bundle.pem`

with AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY and AWS_REGION environment variable set.

Environment

User and Password are retrieved from environment variables. default environment variables are MARIADB_USER and MARIADB_PWD, but can be changed by setting additional option `userKey` and `pwdKey`

Example : using connection string `jdbc:mariadb://host/db?credentialType=ENV` user and password will be retrieved from environment variable MARIADB_USER and MARIADB_PWD.

Property

User and Password are retrieved from java properties. default property name are mariadb.user and mariadb.pwd, but property names can be changed by setting additional option `userKey` and `pwdKey`

Example : using connection string `jdbc:mariadb://host/db?credentialType=PROPERTY&userKey=mariadbUser&pwdKey=mariadbPwd` user and password will be retrieved from java properties `mariadbUser` and `mariadbPwd`

Authentication service

Client authentication plugins are now defined as services. This permits to easily add new client authentication plugins.

Client authentication plugins are now defined as services. This permits to easily add new client authentication plugins.

List of authentication plugins in java connector :

- mysql_clear_password
- auth_gssapi_client
- client_ed25519
- mysql_native_password
- mysql_old_password
- dialog (PAM)
- sha256_password
- caching_sha2_password

New authentication plugins can be created implementing interface org.mariadb.jdbc.authentication.AuthenticationPlugin, and listing new plugin in a META-INF/services/org.mariadb.jdbc.authentication.AuthenticationPlugin file.

SSL factory service

Custom SSL implementation can be used implementing A connection to a server initially creates a socket. When set, SSL socket is layered over this existing socket. Implementing org.mariadb.jdbc.tls.TlsSocketPlugin permit to provide custom SSL implementation for example create a new HostnameVerifier implementation.

Custom implementation need to implement org.mariadb.jdbc.tls.TlsSocketPlugin and register service META-INF/services/org.mariadb.jdbc.tls.TlsSocketPlugin

Custom implementation are activated using option `tlsSocketType`

Easy to use logging

In MariaDB Connector/J 3.0, logging can now be enabled at runtime. Connector/J uses the slf4j API if it is installed. Otherwise, Connector/J uses the JDK logger / console.

logger name is "org.mariadb.jdbc".

Connector/J supports the following Java logging levels:

Log Levels	Description
INFO	Logs connection errors
DEBUG/FINE	Logs SQL statements
TRACE/FINEST	Logs network exchanges

Be careful with "trace" level, purpose is to log all exchanges with server. This means huge amount of data. Bad configuration can lead to problems, like quickly filling the disk.

Example of configuring "trace" level on driver for logback: file logback.xml in src/main/resources/

```
<?xml version="1.0" encoding="UTF-8"?>

<configuration>

    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
            <pattern>%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} - %msg%n</pattern>
        </encoder>
    </appender>

    <logger name="org.mariadb.jdbc" level="trace" additivity="false">
        <appender-ref ref="STDOUT"/>
    </logger>

    <root level="error">
        <appender-ref ref="STDOUT"/>
    </root>

</configuration>
```

Exemple of generated logs :

```
11:47:04.613 [main] TRACE o.m.j.c.socket.impl.PacketWriter - send: conn=17532 (M)
+-------------------------------------------------+
|  0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f |
+-------------------------------------------------+------------------+
| 09 00 00 00 03 53 45 4C  45 43 54 20 31          | .....SELECT 1    |
+-------------------------------------------------+------------------+

11:47:04.613 [main] TRACE o.m.j.c.socket.impl.PacketReader - read: conn=17532 (M)
+-------------------------------------------------+
|  0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f |
+-------------------------------------------------+------------------+
| 01 00 00 01 01                                   | .....            |
+-------------------------------------------------+------------------+

11:47:04.613 [main] TRACE o.m.j.c.socket.impl.PacketReader - read: conn=17532 (M)
+-------------------------------------------------+
|  0  1  2  3  4  5  6  7   8  9  a  b  c  d  e  f |
+-------------------------------------------------+------------------+
| 18 00 00 02 03 64 65 66  00 00 00 01 31 00 00 0C | .....def....1... |
| 3F 00 01 00 00 00 03 81  00 00 00 00             | ?...........     |
+-------------------------------------------------+------------------+
```

Continuous Integration and Automated Tests

For MariaDB Connector/J's continuous integration and automated test results, please see MariaDB Connector/J's Travis CI.

Reporting Bugs

If you find a bug, please report it via the CONJ project on MariaDB's Jira bug tracker.

Source Code

The source code is available at the mariadb-connector-j repository on GitHub.

License

GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

For licensing questions, see the Licensing FAQ.

F.A.Q.

Error *"Could not read resultset: unexpected end of stream, read 0 bytes from 4"*

There is an issue communicating with the server.

Most of the time this will be caused by reading a query that has a large resultset: the server usually expects clients to

read off the result set relatively quickly. The net_write_timeout server variable controls this behavior (defaults to 60s). If the client doesn't read the whole resultset in that amount of time, the server will discard the connection. If you don't expect results to be handled in this amount of time there is another possibility:

- You can use the query "SET STATEMENT net_write_timeout=10000 FOR XXX" with XXX being your "normal" query. This will indicate that specifically for this query, net_write_timeout will be set to a longer time (10000 in this example).
- for older servers, a specific query will have to temporarily set net_write_timeout ("SET STATEMENT net_write_timeout=..."), and set it back afterward.
- if your application usually uses a lot of long queries with fetch size, the connection can be set using the "sessionVariables=net_write_timeout=xxx" option.

How to Do a Lightweight Ping / Avoid Mass "select 1"

Connection.isValid() is a good approach. Connection.isValid() is doing a ping (ping in mysql protocol, not network ping). Connection pool using JDBC4 Validation are using automatically this Connection.isValid()

↑ Java Connector ↑ Installing MariaDB Connector/J →

Comments

Include Archived ☐

MariaDB Connector/J Compatiblity — *4 years, 6 months ago* hkt infotech

Hi, I am using version of [mariadb-java-client-1.4.6.jar]

And I wonder if this maria connector is compatible to which version of mysql server and mariadb server.

Thanks.

key/trust store classpath urls like serverSslCert instead of file paths — *4 years, 12 months ago* T B

Is there a way to specify keystore and truststore resources using the classpath like the serverSslCert option allows? This would allow applications to be more portable, to use bundled resources rather than files on the filesystem for key/trust stores.

JDBC URL parameter for Isolation Level — *6 years, 9 months ago* Alan MacKenzie

Is there a driver connection string parameter to set the Isolation Level used for all connections? Something like the following jdbc:mysql:*mymariadb:3306/MyDB?IsolationLevel=READ_UNCOMMITTED;*

I've tried with combinations of IsolationLevel, isolation_level, transactionIsolation, transaction_isolation, tx_isolation, and with the parameter value as READ_COMMITTED, and READ UNCOMMITTED (with a space) but nothing sets the transaction level

Re: JDBC URL parameter for Isolation Level — *6 years, 9 months ago* Diego Dupin

If that can be set globally, you can set global variable tx_isolation There is no direct configuration option to do that, but you can set that with the option "sessionVariables". that must be something like jdbc:mariadb:*host:3306/MyDB?sessionVariables=tx_isolation=READ_UNCOMMITTED;*

MariaDb serverSslCert and javax.net.ssl.* properties — *6 years, 9 months ago* Mark

I am using the following connection string

```
jdbc:mariadb:${db.server.name}:${db.server.port}/${db.name}?serverSslCert=<path to pem file>&jdbcCompliantTruncation=false&verifyServerCertificate=true&useSSL=true&enabledSslProtocolSuites=TLSv1.1&autoReconnect=true&disableSslHostnameVerification=
```

I use this connection in an application which also acts as a client to servers requiring client authentication; the authentication information is provided via *javax.net.ssl.keyStore(Password)* properties which point to a JKS file. When defined, these properties interfere with the database connection and cause it to fail with

```
Unsupported record version Unknown-0.0
```

but when I run without these properties the connection works fine. I surmise that the driver is using these properties instead of the provided *serverSslCert* property; is there a way to make the driver ignore the *javax.net.ssl.** properties and work with what is given in its *serverSslCert* property?

Re: MariaDb serverSslCert and javax.net.ssl.* properties — *6 years, 9 months ago* Diego Dupin

I imagine that the double slash have been removed due to posting format.

The connection string seems right.

That is strange because this message seems to correspond to this error https://jira.mariadb.org/browse/MDEV-12190 : When server is compiled with yassl and connector send a TLSv1.2 version, then yassl implementation was lost and send this exact message. these has been corrected since MDEV-12190. It seems that options are not taken in account. maybe the & to replace & when using jboss for example ?

Re: MariaDb serverSslCert and javax.net.ssl.* properties — *6 years, 9 months ago* Mark

We are using a 10.1.x version of MariaDb so perhaps you have identified the problem. I will see about upgrading and hope that this problem will be resolved.

Re: MariaDb serverSslCert and javax.net.ssl.* properties — *6 years, 9 months ago* Mark

The "&" are included because I took this out of an XML document where the '&' characters have to be encoded. I will check with our admins to see what version of MariaDb we are running. Thank you for your suggestion.

Re: MariaDb serverSslCert and javax.net.ssl.* properties — *6 years, 9 months ago* Diego Dupin

MDEV-12190 has been backported recently 10.1 too, so an update to latest 10.1.x version must be sufficient.

Re: Using MariaDb with SSL and JPA — *7 years, 5 months ago* vgriffin

After doing a fair amount of web searching, and a log of trial and error, I got an SSL connection working with the connection to mydb defined in my persistence.xml file. I'm using MariaDB 10.3 (Alpha) and Hibernate 5.2.12, configured as JPA rather than Hibernate. I'm using IntelliJ and put "myCert.pem" into my resources directory so it ends up on the top level of my classpath. Note that the CN used in the certificate MUST match the domain used for connection. (So far, I've only used "localhost". Another server is tomorrow's task. You'll see localhost in the example, too.) The line in persistence.xml that makes the magic happen is:

<property name="javax.persistence.jdbc.url" value="jdbc:mariadb:*localhost:3306/mydb?useSSL=true&serverSslCert=classpath:myCert.pem*"/>

The last problem was that I didn't remember initially that the XML file needed & in the String. Note that user and password are already javax.persistence.jdbc properties.

Re: About MariaDB Connector/J — *8 years, 3 months ago* Jarod Liu

When configure multiply server, how to let connector print log which server is connected?

re: "Pre-built .jar files can be downloaded from:" - default to older version — *8 years, 7 months ago* H Json

The link above in:

"Pre-built .jar files can be downloaded from: https://mariadb.com/my_portal/download/java-client"

defaults you to the old version: 1.4.6, instead of 1.5.3.

Re: About MariaDB Connector/J — *8 years, 7 months ago* Meirav Rath

Hi, Is there a place for a session variable such as application user in the Connector/J script? I mean 'con.setClientInfo' such as what exists in SAP HANA here: https://help.sap.com/saphelp_hanaplatform/helpdata/en/e9/0fa1f0e06e4840aa3ee2278afae16b/content.htm and https://help.sap.com/hana_one/html/_m__s_e_s_s_i_o_n__c_o_n_t_e_x_t.html Thanks!

Re: About MariaDB Connector/J — *8 years, 7 months ago* Diego Dupin

Hi, If performance_schema variable is activated, in Table performance_schema.session_connect_attrs, for each connection, you will have :

- _client_name (example : MariaDB connector/J)
- _client_version (1.5.4-SNAPSHOT)
- _os (example : Windows 10)
- _pid (example : 1636)
- _thread (example : 13)
- _java_vendor (example : Oracle Corporation)
- _java_version (example : 1.8.0_91)

You can add specific information using option "connectionAttributes"

Re: About MariaDB Connector/J — *8 years, 7 months ago* Diego Dupin

Sorry, updating the answer delete your comment. New version 1.5.4, Connection.setClientInfo() permit 3 differents Session-Specific Client Information :

- ApplicationName
- ClientUser
- ClientHostname

This without having performance_schema variable activated.

Re: About MariaDB Connector/J — *8 years, 7 months ago* Meirav Rath

Hi again, When will version 1.5.4 be released?

Re: About MariaDB Connector/J — *8 years, 7 months ago* Diego Dupin

version 1.5.4 is available on maven central or will be available on site download in a few hours.

Re: About MariaDB Connector/J — *8 years, 7 months ago* Meirav Rath

Great! So, I have a MariaDB 10.1.14 and the session_connect_attrs table does not have these three new parameters, and I create a short javascript that uses the new connector to make a short connection, make a simple select and then logout, what will happen? I tried to add new variables to this table but my user didn't have the permission to do it, nor did the root user. Obviously, I switched on the performance_schema.

Re: About MariaDB Connector/J — *8 years, 7 months ago* Diego Dupin

I may have not been clear.

There is in fact 2 differents way to achieve that : standard JDBC : using connection , you can add some specific informations:

Properties properties = new Properties();
properties.setProperty("ApplicationName", "your application");
properties.setProperty("ClientUser", "your user");
properties.setProperty("ClientHostname", "myhost");

connection.setClientInfo(properties); and so after that, you will be able to retrieve those informations : String currentHost = connection.getClientInfo("myhost");

other solution is the first explain : If performance_schema variable is activated on server, in Table performance_schema.session_connect_attrs, for each connection, you will have :

- _client_name (example : MariaDB connector/J)
- _client_version (1.5.4-SNAPSHOT)
- _os (example : Windows 10)
- _pid (example : 1636)
- _thread (example : 13)
- _java_vendor (example : Oracle Corporation)
- _java_version (example : 1.8.0_91)

You can add specific information using option "connectionAttributes" . Example connecting with url "jdbc:mariadb:localhost/db?user=xxx&connectionAttributes=key1:value1,key2,value2"

you will have then have 2 additionnals informations :

- key1 with value "value1"
- key2 with value "value2"

Re: About MariaDB Connector/J — *8 years, 7 months ago* Meirav Rath

It works great, thank you so mcuh for the help and for all the fast responses!

Re: About MariaDB Connector/J — *8 years, 7 months ago* Meirav Rath

Excellent, thanks!

Re: About MariaDB Connector/J — *8 years, 10 months ago* Jérôme Baton

Running the build of 1.4.6, errors signal a missing "testj" database. And looking at the code, thats what used in the UTs.

Either the page is to be corrected or the code changed

Re: About MariaDB Connector/J — *8 years, 10 months ago* Diego Dupin

right, this is changed to "testj" according to unit tests

The documentation fails to mention dependency on SLF4J — *9 years, 8 months ago* openiduser58

As a result, when using v. 1.2.2 driver and trusting auto-registration of JDBC driver, one gets a "No suitable driver found" error without explanation. Only when doing Class.forName("org.mariadb.jdbc.Driver") - the reason for the error shows up. So either the dependency should be moved or SLF4J should be added to the requirements in this article.

Re: About the MariaDB Java Client — *10 years, 3 months ago* Azzouz Nezar

I downloaded and installed MariaDB version 5.5.5-10.16. So far I checked mostly what an application I have needs.

While pleased, one thing I could not do: check primary key and auto-increment of a table with auto-increment. I used client mariadb-java-client-1.1.3 & 1.1.7.

When I used drivers mysql-conncor-java-3.1.14 and 5.0.8, I get the primary key but still no auto-increment. Any thoughts? Thank you.

Re: About the MariaDB Java Client — 10 years, 3 months ago Vladislav Vaintroub

Not sure what you're refering to when speaking about primary keys. For autoincrement, the things are standard, and you could use getGeneratedKeys() as described here for example http://dev.mysql.com/doc/connector-j/en/connector-j-usagenotes-last-insert-id.html .

Re: About the MariaDB Java Client — 10 years, 3 months ago Azzouz Nezar

Thank you for the link.

The situation in my case is different. Here is a simplified code for my example where normally I get the auto-increment:

ResultSet rscol = null;

String autoInc =null;

int colAutoInc=0;

Statement stmt = cnx.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);

ResultSet rs = stmt.executeQuery("Select * from " + db_table);

DatabaseMetaData dbmd =cnx.getMetaData();*cnx=Connection*

for (i = 1; i <=numberOfColumns; i++) {

rscol = dbmd.getColumns(tableCatalog, tableSchema, db_table, columnName);

while (rscol.next()) {

autoInc = rscol.getString(12);*returns "auto-increment" if column is in fact auto-increment*

if (!autoInc.equals("")) colAutoInc=i;

}

}

Re: About the MariaDB Java Client — 10 years, 3 months ago Vladislav Vaintroub

http://docs.oracle.com/javase/7/docs/api/java/sql/DatabaseMetaData.html#getColumns%28java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.String%29

It says the column #12 is REMARKS, which does not have much meaning

To retrieve primary key, there is a DatabaseMetaData.getPrimaryKeys() that does exactly that.
http://docs.oracle.com/javase/7/docs/api/java/sql/DatabaseMetaData.html#getPrimaryKeys%28java.lang.String,%20java.lang.String,%20java.lang.String%29

Re: About the MariaDB Java Client — 10 years, 3 months ago Azzouz Nezar

I built an application for productivity that interfaces MySQL and MS-SQL. Drivers from MySQL and Microsoft are used and work fine. User can switch between MySQL and MS-SQL if he chooses to. Today I am in the process of adding MariaDB to the list of databases. I downloaded 2 clients from MariaDB: "mariadb-java-client-1.1.3.jar and mariadb-java-1.1.7.jar". I could not connect with either one to mySQL database. The drivers were placed in my java JRE directory like the mySQL driver.

Database used: MySQL, Vesion:4.0.21.-nt-max-log

Currently using: MySQL-AB JDBC Driver Version: mysql-connector-java-3.1.14 ($Date:2006-10-18 17:40:15 +0200 (Wed, 18 Oct 2006)$,$Revision:5888$)

Re: About the MariaDB Java Client — 10 years, 3 months ago Vladislav Vaintroub

That driver was not tested with server version 4.0 (nor 4.1 nor 5.0) an is not likely to work with any server version for which lifecycle support ended . MySQL Sever 4.0 is out of support maybe for 7 or 8 years something like that.

Re: About the MariaDB Java Client — 10 years, 3 months ago Azzouz Nezar

Thank you for commenting. Yes I had MSQL Server 4.0 running for at least 12 yrs on an XP-machine, and few years before that on a Window 2000. I will download a newer version to continue development on my application. I have a question about the MariaDB Client. When I run my application, I see that MariaDB client that I put in my Java JRE directory already loaded. It does not happen with MySQL driver that I have been using. So the question is: does MariaDB clients have a feature that Java recognizes and loads the Client automatically?

Re: About the MariaDB Java Client — 10 years, 3 months ago Vladislav Vaintroub

Yes, autoregistration is part of JDBC 4.0, which came wiith Java 6 , in 2006 The connector/j driver you are using is probably quite old if it does not have this feature.

Re: About the MariaDB Java Client — 10 years, 3 months ago Azzouz Nezar

The exception I have is "java.nio.BufferUnderflowException"

Re: About the MariaDB Java Client — 10 years, 3 months ago Vladislav Vaintroub

Yeah, in the long past the packet server sent was on new connection was shorter according. So underflow exception can be explained.

Re: About the MariaDB Java Client — 10 years, 3 months ago Azzouz Nezar

Thank you.

Re: About the MariaDB Java Client — 10 years, 12 months ago Max Lam

Hi,

I have posted a question on stackoverflow.com about a JDBC url for load balanced setup of MariaDB (specifically referring to MariaDB Galera), you may find it at http://stackoverflow.com/questions/19629690/what-is-the-right-mariadb-galera-jdbc-url-properties-for-loadbalance.

Since the JDBC Client is already at ver 1.1.7, this is just a followup comment if there had been any updates to the JDBC client for connecting to a clustered MariaDB (Galera) using the MariaDB JDBC Client just like how it was done to connect to MySQL Cluster with more than one IP of the SQL node e.g.
jdbc:mysql:loadbalance:*192.168.1.2:3306,192.168.1.3:3306/test?loadBalanceConnectionGroup=first&loadBalanceEnableJMX=true*

Thanks a bunch for the effort.

Streaming result sets — 11 years, 1 month ago Erik Mattheis

Is the comment above still accurate? Looking at the source of `org.mariadb.jdbc.MySQLResultSet.setFetchSize(int)` , I see the following:

```
public void setFetchSize(int rows) throws SQLException {
    // ignored - we fetch 'em all!
```

```
}
```

Re: Streaming result sets — 11 years, 1 month ago Vladislav Vaintroub

"we fetch'em all" is not accurate. However that fetch size is ignored here is accurate. The one fetch size that matters for streaming is in MySQLStatement, not in MySQLResultSet

Re: Streaming result sets — 11 years, 1 month ago Erik Mattheis

Can someone update this page with the correct instructions?

Obviously calling `ResultSet.setFetchSize(Integer.MIN_VALUE)` will have no impact on the resultset.

Calling `Statement.setFetchSize(Integer.MIN_VALUE)` does the trick.

Re: Streaming result sets — 11 years, 1 month ago Vladislav Vaintroub

I updated the page with correct instructions.

Re: About the MariaDB Java Client — 11 years, 6 months ago david yu

My old java code uses:

Class.forName("com.mysql.jdbc.Driver");

but now it will cause

java.lang.ClassNotFoundException: com.mysql.jdbc.Driver

Could you tell me is this an issue in MariaDB connector or we should change our code to remove this statement after switch to MariaDB.

Thanks.

Re: About the MariaDB Java Client — 11 years, 6 months ago Vladislav Vaintroub

Yes, please remove this statement. It should work ok without it.

Re: About the MariaDB Java Client — 11 years, 6 months ago Netanel Weinberg

Is there a plan for key "zeroDateTimeBehavior" like in the original mysql client or a workaround?

Re: About the MariaDB Java Client — 10 years, 5 months ago Daniel Black

I'm not sure what zeroDateTimeBehavior is however I can't see it listed in https://mariadb.atlassian.net/browse/CONJ

Perhaps you could add it as a request.