MySQL Connector/J 8.0 Release Notes

Abstract

This document contains release notes for the changes in each release of MySQL Connector/J.

For additional Connector/J documentation, see MySQL Connector/J 8.0 Developer Guide.

For legal information, see the Legal Notices.

For help with using MySQL, please visit either the MySQL Forums or MySQL Mailing Lists, where you can discuss your issues with other MySQL users.

Document generated on: 2019-01-15 (revision: 16889)

Table of Contents

Preface and Legal Notices

This document contains release notes for the changes in each release of MySQL Connector/J.

Legal Notices

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Changes in MySQL Connector/J 8.0.14 (Not yet released, General Availability)

Version 8.0.14 has no changelog entries, or they have not been published because the product version has not been released.

Changes in MySQL Connector/J 8.0.13 (2018-10-22, General Availability)

Version 8.0.13 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, 5.6, and 5.5. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

Functionality Added or Changed

- **Important Change:** Connector/J now requires Protocol Buffers 3.6.1 as an external library for using X DevAPI and for building Connector/J from source.

  See Connector/J Installation on installation requirements for Connector/J. (Bug #28499094)

- **X DevAPI:** X DevAPI now provides a connection pooling feature, which can reduce overhead for applications by allowing idle connections to be reused. Connection pools are managed by the new `Client` objects, from which sessions can be obtained. See Connecting to a Single MySQL Server Using Connection Pooling in the X DevAPI User Guide for details.

- **X DevAPI:** A new connection property, `xdevapi.connect-timeout`, now defines the timeout (in milliseconds) for establishing an X-Protocol connection to the server. Default value is 10000 (10s), and a value of 0 disables timeout, which makes Connector/J wait for the underlying socket to time out instead. See Configuration Properties for details.

  Note that if `xdevapi.connect-timeout` is not set explicitly and `connectTimeout` is, `xdevapi.connect-timeout` takes up the value of `connectTimeout`.

- The connection property `useOldUTF8Behavior` is no longer supported. The connection property never had any meaning for the MySQL Server versions supported by Connector/J 8.0, but actually corrupted the data when it was used with them. (Bug #28444461)

- Connector/J now translates the legacy value of `convertToNull` for the connection property `zeroDateTimeBehavior` to `CONVERT_TO_NULL`. This allows applications or frameworks that use the legacy value (for example, NetBeans) to work with Connector/J 8.0. (Bug #28246270, Bug #91421)

- A new connection property, `sslMode`, has been introduced to replace the connection properties `useSSL`, `requireSSL`, and `verifyServerCertificate`, which are now deprecated. Also, when not explicitly set, the connection properties `xdevapi.ssl-mode`, `xdevapi.ssl-truststore`, `xdevapi.ssl-truststore-password`, and `xdevapi.ssl-truststore-type` now take up the values of `sslMode`, `trustCertificateKeyStoreUrl`, `trustCertificateKeyStorePassword`, and `trustCertificateKeyStoreType`, respectively. See Connecting Securely Using SSL and Configuration Properties for details.

  Note that for ALL server versions, the default setting of `sslMode` is `PREFERRED`, and it is equivalent to the legacy settings of `useSSL=true`, `requireSSL=false`, and `verifyServerCertificate=false`, which are different from their default settings for Connector/J 8.0.12 and earlier in some situations. Applications that continue to use the deprecated properties and rely on their old default settings should be reviewed. (Bug #27102307)

- The value `UTF-8` for the connection property `characterEncoding` now maps to the `utf8mb4` character set on the server and, for MySQL Server 5.5.2 and later, `characterEncoding=UTF-8` can now be used to set the connection character set to `utf8mb4` even if `character_set_server` has been set to something else on the server. (Before this change, the server must have `character_set_server=utf8mb4` for Connector/J to use that character set.)

Also, if the connection property `connectionCollation` is also set and is incompatible with the value of `characterEncoding`, `characterEncoding` will be overridden with the encoding corresponding to `connectionCollation`.

See Using Character Sets and Unicode for details, including how to use the `utf8mb3` character set now for connection. (Bug #23227334, Bug #81196)

Bugs Fixed

- **X DevAPI:** Connector/J threw a `WrongArgumentException` when it encountered a JSON number with more than ten digits. This was due to an error in the JSON parser, which has now been fixed. (Bug #28594434, Bug #92264)
- **X DevAPI:** `Session.getUri()` returned a `NullPointerException` when the default value is null for any of the connection properties contained in the connection URL; and when `Session.getUri()` returned a URL, the URL contained a comma (“`,`”) before its first connection property. (Bug #23045604)
- **X DevAPI:** When handling an invalid JSON document, Connector/J threw a `NullPointerException`. With this fix, a `WrongArgumentException` is thrown instead in the situation. (Bug #21914769)
- Setting the connection property `characterEncoding` to an encoding that maps to the MySQL character set `latin1` or `utf8mb4` did not result in the corresponding default connection collation (`latin1_swedish_ci` or `utf8mb4_0900_ai_ci`, respectively) to be used on the server. With this fix, the server default is used in the situation. (Bug #28207422)
- Calling `UpdatableResultSet.updateClob()` resulted in an `SQLFeatureNotSupportedException`. It was because the implementation of the method was missing from Connector/J, and it has been added with this fix. (Bug #28207088)
- When a connection property's value contained an equal sign (“=”) in itself, an exception (“`WrongArgumentException: Malformed database URL`”) was thrown. This was due to an error in the parser for the connection URL, which has been corrected by this fix. (Bug #28150662)
- Connector/J threw a `SQLSyntaxErrorException` when the parameter `tableName` for `DatabaseMetaDataUsingInfoSchema.getTables()` had a null argument. (Bug #28034570, Bug #90887)
- Setting `rewriteBatchedStatements=true` and `useLocalTransactionState=true` caused transactions to be uncommitted for batched `UPDATE` and `DELETE` statements. It was due to the intermediate queries for enabling multiquery support on the server resetting the local transaction state as a side effect. With this fix, the local transaction state is preserved when the intermediate queries are executed. (Bug #27658489, Bug #89948)
- Rewriting prepared `INSERT` statements in a multiquery batch failed with a `BatchUpdateException` when the statements did not contain place holders. This was due a faulty mechanism for query rewriting, which has been corrected by this fix. (Bug #25501750, Bug #84813)
- When using batched prepared statements with multiple queries per statement, queries rewriting was incorrect, resulting in the wrong queries being sent to the server. (Bug #23098159, Bug #81063)
- Record updates failed for a scrollable and updatable `PreparedStatement` when the `WHERE` clause for the updater or refresher contained fractional timestamp values and the connection property `sendFractionalSeconds` was set to `false`. It was because in the situation, Connector/J did not perform the proper adjustments of the fractional parts in the `WHERE` clause values according to the length of the field’s fractional part as defined in the database. This fix makes Connector/J perform the proper adjustment to the fractional part, so that the `WHERE` clause value can be properly compared to the value fetched from the database. (Bug #22305979)

- Some tests in the testsuite failed as they could not recognize system time zone values like `CEST` or `WEST`, even with the connection property `serverTimezone` set. This was because the value of `serverTimezone` in the testsuite URLs, after being processed by the testsuite, was not actually propagated as a connection property to Connector/J. This fix makes sure the property is in the actual URLs passed to Connector/J. (Bug #21774249)

- When a Java `Date` value was bound to a `PreparedStatement` parameter, attempts to format the value by a proleptic `GregorianCalendar` failed to make the dates proleptic, so that dates before the Julian-Gregorian cutover (October 15, 1582) were stored wrongly. With this fix, a proleptic calendar is properly used if supplied to the `setDate()` method.

  Note that when trying to set or retrieve dates before the Julian-Gregorian cutover with `PreparedSatement` methods, a proleptic `GregorianCalendar` should always be explicitly supplied to the `setDate()` and `getDate()` method. For details, see Known Issues and Limitations. (Bug #18749544, Bug #72609)

Changes in MySQL Connector/J 8.0.12 (2018-07-27, General Availability)

Version 8.0.12 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, 5.6, and 5.5. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **X DevAPI:** The following changes have been made to the API:

  - Removed `ModifyStatement.arrayDelete()` and `ModifyStatement.merge()`.
  - Renamed `Colletion.find().limit().skip()` to `Colletion.find().limit().offset()`.
  - To simplify the class hierarchy and to have the class names reflect better the classes' functions, the following changes have been made:
    - The `FindParams` class has been renamed to `FilterParams`
    - The `AbstractFindParams` class has been renamed to `AbstractFilterParams`
    - The `DocFindParams` class has been renamed to `DocFilterParams`
    - The `TableFindParams` class has been renamed to `TableFilterParams`

    Notice that the methods in the original `FilterParams` class have been moved under the new `AbstractFilterParams` class.

  (Bug #28027459)

- **X DevAPI:** Connector/J now uses synchronous client sockets (`java.net.Socket`) by default to communicate with MySQL servers for X Protocol connections. While asynchronous sockets can still be used by setting the connection property `xdevapi.useAsyncProtocol=true`, this is not recommended, as it might result in performance degradation for Connector/J. (Bug #27522054)

- **X DevAPI:** Connector/J now gives provision for the use of a custom socket factory for X Protocol connections to MySQL Servers using Unix domain sockets. See Section 6.8, "Connecting Using Unix Domain Sockets" for details.

- Connector/J now retrieves the MySQL keyword list from the INFORMATION_SCHEMA.KEYWORDS table on the MySQL server when a connection session is established. The list can then be accessed by calling `DatabaseMetaData.getSQLKeywords()`.
- To simplify the code, the `ReadableProperty` and `ModifiableProperty` classes have been consolidated into the `RuntimeProperty` class.

Bugs Fixed

- **X DevAPI:** When creating an X DevAPI session using a `Properties` map instead of a connection string, referring to property keys like `host`, `port`, and `protocol` in lowercase caused a `NullPointerException`. With the fix, both upper and lower cases can now be used. (Bug #27652379)
- **X DevAPI:** When creating an X DevAPI session with an SSL connection using a `Properties` map instead of a connection string, a `NullPointerException` was returned when no connection password was provided. (Bug #27629553)
- **X DevAPI:** When using the `getConnection()` method with the `mysqlx:` scheme in the connection URL, Connector/J returned an ordinary JDBC connection instead of an X-Protocol connection. (Bug #26089880)
- If `wait_timeout` was set on the server and the Connector/J had the connection property `interactiveClient=false`, or if `interactive_timeout` was set on the server and Connector/J had the connection property `interactiveClient=true`, a connection is invalidated when it has idled for a longer time than the set timeout. When such a timeout occurred, Connector/J threw a `CJCommunicationsException`, without indicating it was a timeout. With this fix, the error message returned explains the issue and suggests how to avoid it. (Bug #27977617, Bug #90753)
- When an application tried to connect to a non-MySQL database through some JDBC driver and Connector/J happened to be on the class path also, Connector/J threw a `SQLNonTransientConnectionException`, which prevented the application from connecting to its database. With this fix, Connector/J returns null whenever a connection string does not start with `jdbc:mysql:` or `mysqlx:`, so connections to non-MySQL databases are not blocked. (Bug #26724154, Bug #87600)
- A `wasNull()` call on a `ResultSet` did not return the proper value unless `AbstractResultsetRow.getNull()` or `AbstractResultsetRow.getValueFromByte()` was called before. This caused data loss when Connector/J was used with frameworks like Hibernate, which rely on `wasNull()` calls to properly retrieve data. With this fix, `wasNull()` returns a correct value as long as some getter method has been called before on the `ResultSet`. (Bug #25924324, Bug #85941)

Changes in MySQL Connector/J 8.0.11 (2018-04-19, General Availability)

Version 8.0.11 is the first General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, 5.6, and 5.5. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **X DevAPI:** The locking options `lockShared()` and `lockExclusive()`, available when retrieving data from `collection.find()` and `table.select()`, now also accept an optional locking contention value, which is exposed though the enumeration `Statement.LockContention`.

The combinations of `lockShared([lockCont])` or `lockExclusive([lockCont])` with `Statement.LockContention.NOWAIT` or `Statement.LockContention.SKIP_LOCKED` map directly to the SQL statement `SELECT ... FOR SHARE` or `SELECT ... FOR UPDATE` with the SQL option `NOWAIT` or `SKIP LOCKED`, for the different InnoDB locking read modes.

- **X DevAPI:** Connector/J now supports the new server-side document ID generation feature. Client-side document ID generation is no longer supported. As a result, the methods `getDocumentId()` and `getDocumentIds()` have been removed and the method `getGeneratedIds()` has been added to the `AddResult` and `AddResultImpl` classes.
- **X DevAPI:** The `SHA256_MEMORY` authentication mechanism is now supported by Connector/J for connections using the X Protocol. See the entry for the connection property `xdevapi.auth` in Configuration Properties for details.
- Connector/J now recognizes the data type `GEOMCOLLECTION`, which has been introduced in MySQL 8.0.11 as an alias and preferred name to the previously known `GEOMETRYCOLLECTION` data type. (Bug #27678308)
- The lower bound for the connection property `packetDebugBufferSize` has been changed to 1, to avoid the connection errors that occur when the value is set to 0. (Bug #26819691)
- Connector/J now supports the use of a custom `SSLSocketFactory` for returning a custom-constructed SSL socket at the time of connection establishment. (Bug #26092824, Bug #86278)
- The source directory and Java package layouts of Connector/J have been revised to make it easier to use custom protocols, APIs, value decoders, and value factories with Connector/J. See the Connector/J source code and the MySQL Connector/J X DevAPI Reference for more details.

Bugs Fixed

- When an integer value in a JSON document is modified, it becomes a a `DOUBLE` value to the MySQL server, which is returned with a decimal when fetched from the JSON document. Therefore, calling `getInteger()` upon the changed value with Connector/J resulted in an `NumberFormatException`. With this fix, `getInteger()` parses such a value correctly and returns an integer. (Bug #27226293)
- In the Ant build file `build.xml`, `com.mysql.cj.api.conf` was missing in the list of OSGi exported packages, causing missing dependencies in OSGi bundles that use Connector/J. (Bug #25765250, Bug #85566)
- Name change of the `com.mysql.jdbc.SocketFactory` interface to `com.mysql.cj.api.io.SocketFactory` caused backward incompatibility for older Connector/J applications. The old interface has now been reimplemented to avoid the incompatibility. (Bug #25223137, Bug #84099)

Changes in MySQL Connector/J 8.0.10 (Skipped version number)

There are no release notes for this skipped version number.

Changes in MySQL Connector/J 8.0.9 (2018-01-30, Release Candidate)

Version 8.0.9 Release Candidate is the first release candidate of the 8.0 branch of MySQL Connector/J, providing an insight into upcoming features. It is suitable for use with MySQL Server versions 5.5, 5.6, 5.7, and 8.0. It supports the Java Database Connectivity (JDBC) 4.2 API.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **X DevAPI:** In the process of refining the definition of the X DevAPI to cover the most relevant usage scenarios, the following API components have been removed from the X DevAPI implementation for Connector/J:

  - Components that support DDLs for views, including the `createView()`, `dropView()`, and `modifyView()` methods.
  - Components that support DDLS for tables, including the `createTable()`, `dropTable()`, and `modifyTable()` methods.
  - Components that support session configurations, including the `SessionConfig` object, the `PersistenceHandler` interface, the `PasswordHandler` interface, and the `SessionConfigManager` class.

- **X DevAPI:** Added the `setSavepoint()`, `rollbackTo()`, and `releaseSavepoint()` methods to the `Session` interface to support the SAVEPOINT, ROLLBACK TO SAVEPOINT, and RELEASE SAVEPOINT statements. See MySQL Connector/J X DevAPI Reference for more details.

- **X DevAPI:** A new `patch()` function has been added to the `ModifyStatement` interface. The function accepts an JSON-like object describing document changes and applies them to documents matched by the `modify()` filter. See MySQL Connector/J X DevAPI Reference for more details.

- **X DevAPI:** The `createIndex()` method for the `Collection` interface now has a new syntax. See MySQL Connector/J X DevAPI Reference for more details.

- **X DevAPI:** Added the following methods for single-document operations in the X DevAPI:

  - `replaceOne()`
  - `addOrReplaceOne()`
  - `getOne()`
  - `removeOne()`

  See MySQL Connector/J X DevAPI Reference for more details.

- **X DevAPI:** Setters and getters methods have been added for the configuration properties with the `MysqlDataSource`, `MysqlXADataSource`, and `MysqlConnectionPoolDataSource` classes.

- **X DevAPI:** The connection property `enabledTLSProtocols` can now be used to select the allowed TLS versions for an X Protocol connection to the server.

- Connector/J now supports the new `caching_sha2_password` authentication plugin, which is the default authentication plugin for MySQL 8.0.4 and later (see Caching SHA-2 Pluggable Authentication for details).

  

  Note

  To authenticate accounts with the `caching_sha2_password` plugin, either a secure connection to the server using SSL or an unencrypted connection that supports password exchange using an RSA key pair (enabled by setting one or both of the connecting properties `allowPublicKeyRetrieval` and `serverRSAPublicKeyFile`) must be used.

  Because earlier versions of Connector/J 8.0 do not support the `caching_sha2_password` authentication plugin and therefore will not be able to connect to accounts that authenticate with the new plugin (which might include the root account created by default during a new installation of a MySQL 8.0 Server), it is highly recommended that you upgrade now to Connector/J 8.0.9, to help ensure that your applications continue to work smoothly with the latest MySQL 8.0 Server.

- Connector/J now takes advantage of the MySQL Server 8.0 data dictionary by making the connection property `useInformationSchema` true by default; this makes Connector/J, by default, access the data dictionary more efficiently by querying tables in the INFORMATION_SCHEME. See INFORMATION_SCHEMA and Data Dictionary Integration for details. Users can still set useInformationSchema to false; but for MySQL 8.0.3 and later, some data dictionary queries might then fail, due to deprecations of older data dictionary features.
- In the past, query texts were always passed as strings to `QueryInterceptor` methods, even if the texts were not actually used by them. Now, only suppliers for the texts are passed, and the texts are only extracted by `get()` calls on the suppliers.

Bugs Fixed

- The connection property `nullNamePatternMatchesAll`, when set to false (which was the default value), caused some `DatabaseMetaData` methods to throw an error when a null search string was used with them. The behavior was not compliant with the JDBC specification, which requires that a search criterion be ignored when a null search string is used for it. The connection property has now been removed from Connector/J 8.0. (Bug #26846249, Bug #87826)
- Trying to print the query in a `PreparedStatement` using the `toString()` method after it has been closed resulted in an exception (`No operations allowed after statement closed`) being thrown. (Bug #26748909)
- When working with MySQL Server 8.0, an update or delete statement for a `CONCUR_UPDATABLE ResultSet` failed when the `ResultSet`'s primary keys included a boolean column and the character set used was not `latin1`. (Bug #26266731)
- Connector/J failed to recognize a server greeting error it received during a handshake with the server and parsed the error message as a normal greeting packet, causing an *`ArrayIndexOutOfBoundsException`* to be thrown. (Bug #24924097)

Changes in MySQL Connector/J 8.0.8 (2017-09-28, Development Milestone)

Version 8.0.8 Development Milestone is the latest development release of the 8.0 branch of MySQL Connector/J, providing an insight into upcoming features. It is suitable for use with MySQL Server versions 5.5, 5.6, 5.7, and 8.0. It supports the Java Database Connectivity (JDBC) 4.2 API.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Packaging:** RPM and Debian packages for installing Connector/J are now available from the Connector/J Download page.
- **X DevAPI:** Connector/J has implemented a new interface of the X Dev API that allows the retrieving, adding, removing, and updating of persistent session continuation data. The implementation includes the following:
  - A `SessionConfig` object that holds the information for a session configuration data set.
  - A `PersistenceHandler` interface that allows custom implementations of persistence handlers.
  - A `PasswordHandler` interface that allows custom implementations of password handling code.
  - A `SessionConfigManager` class for editing and fetching `Sessionconfig` objects, and defining instances of the `PersistenceHandler` and `PasswordHandler`.

  See MySQL Connector/J X DevAPI Reference for more details.

- **X DevAPI:** A new connection property, `xdevapi.auth`, has been added for specifying the authentication mechanism for connections using the X Protocol. Allowed values are `MYSQL41`, `PLAIN`, and `EXTERNAL`. See the entry for the new property in Configuration Properties for details.

- **X DevAPI:** To support row locks for the `find()` method of the X DevAPI, the `FindStatement` and the `SelecStatement` interfaces have been extended with the following methods:

  - lockExclusive(), which works like `SELECT ... FOR UPDATE` for relational tables.

  - lockShared(), which works like the `SELECT ... LOCK IN SHARED MODE` (for MySQL 5.7) or `SELECT ... FOR SHARE` (for MySQL 8.0) for relational tables.

  See MySQL Connector/J X DevAPI Reference for more details.

- **X DevAPI:** Connector/J now supports the expanded syntax for the `IN` and `NOT IN` operator, which can check if a sub-expression is contained inside another one; for example:

```
// For documents
coll.find("$.b IN [100,101,102]").execute();
coll.find("'some text with 5432' in $.a").execute();
coll.find("1 in [1, 2, 4]").execute();
coll.find("{'a': 3} not in {'a': 1, 'b': 2}").execute();
// For relational tables
tbl.select().where("3 not in [1, 2, 4]").execute();
tbl.select().where("'qqq' not in $.a").execute();
tbl.select().where("{'a': 1} in {'a': 1, 'b': 2}").execute();
```

- **X DevAPI:** A number of changes have been implemented for the "`drop`" methods for the X DevAPI:

  - Removed `dropCollection(schemaName, collectionName)` and `dropTable(schemaName, tableName)` from `Session`.

  - Added `dropCollection(collectionName)` and `dropTable(tableName)` to `Schema`.

  - `Schema.dropView()` now executes immediately and returns `void`; also, the `ViewDrop` interface has been removed.

  - `Collection.dropIndex()` now executes immediately and returns `void`; also the `DropCollectionIndexStatement` interface has been removed.

  - The "`drop`" methods now succeed even if the objects to be dropped do not exist.

- Conversion from the MySQL `TIME` data to `java.sql.Date` is now supported. In the past, a `getDate()` retrieving data from a TIME column would throw an `SQLException`. Now, such a retrieval returns a `java.sql.Date` object containing the time value expressed in number of milliseconds from the Java epoch; also returned is the warning: "Date part does not exist in SQL TIME field, thus it is set to January 1, 1970 GMT while converting to java.sql.Date." (Bug #26750807)

- A new connection property, `enabledTLSProtocols`, can now be used to override the default restrictions on the TLS versions to be used for connections, which are determined by the version of the MySQL Server that is being connected to. By providing a comma-separated list of values to this option (for example, "TLSv1,TLSv1.1,TLSv1.2") users can, for example, prevent connections from using older TLS version, or allow connections to use TLS versions only supported by a user-compiled MySQL Server. See the entry for the new property in Configuration Properties for details. Thanks to Todd Farmer for contributing the code. (Bug #26646676)

- Updated the timezone mappings using the latest IANA and CLDR time zone databases. (Bug #25946965)

- A new option for the `loadBalancingStrategy` connection property called `serverAffinity` has been added. The servers listed in the new connection property `serverAffinityOrder` (which should be a subset of the servers in the host list of the connection URL) are contacted in the order

they are listed until a server is available or until the list of servers is exhausted, at which point a random load-balancing strategy is used with the hosts not listed by `serverAffinityOrder`. See descriptions for `loadBalancingStrategy` and `serverAffinityOrder` in Configuration Properties for details. (Bug #20182108)

Bugs Fixed

- **Important Change:** Following the changes in MySQL Server 8.0.3, the system variables `tx_isolation` and `tx_read_only` have been replaced with `transaction_isolation` and `transaction_read_only` in the code of Connector/J. Users should update Connector/J to this latest release in order to connect to MySQL 8.0.3. They should also make the same adjustments to their own applications if they use the old variables in their codes. (Bug #26440544)
- **X DevAPI:** Calling `schema.dropView()` with a null argument resulted in a `NullPointerException`. (Bug #26750807)
- **X DevAPI:** When `dropCollection()` was applied on a null collection, a `NullPointerException` occurred. (Bug #26393132)
- When using cached server-side prepared statements, a memory leak occurred as references to opened statements were being kept while the statements were being decached; it happened when either the `close()` method has been called twice on a statement, or when there were conflicting cache entries for a statement and the older entry had not been closed and removed from the opened statement list. This fix makes sure the statements are properly closed in both cases. Thanks to Eduard Gurskiy for contributing to the fix. (Bug #26633984, Bug #87429)
- The regression test for Bug#63800 failed because the default value of the system variable `explicit_defaults_for_timestamp` of MySQL Server has been changed since release 8.0.2. The test has been adjusted to take the change into consideration. (Bug #26501245)
- Running callable statements against MySQL Server 8.0 resulted in the `SQLException`: `ResultSet is from UPDATE. No Data`. (Bug #26259384)
- Secure JDBC connections did not fall back to the default truststore when a custom one was not provided. (Bug #26243128)
- In `com/mysql/jdbc/ServerPreparedStatement.java`, the arguments `resultSetType` and `resultSetConcurrency` for a call of `Connection.preparedStatement()` were swapped. (Bug #25874048, Bug #85885)
- Some JDBC proxied objects were missing the proper handlings of the `equals()` methods, thus even comparison of one of these proxied objects to its own self with `equals()` yielded false. This patch introduces proper handlings for the `equals()` method in all the relevant proxies. (Bug #21931572, Bug #78313)
- A server-side prepared statement was not closed when the same statement was being prepared again while the original statement was being cached. This was caused by the silent replacement of the cache entry of the old statement by the new. When this happened repeatedly, it caused eventually the complaint that `max_prepared_stmt_count` was exceeded. This fix makes sure that when a cache entry for a statement replaces an older one, the older statement is immediately closed. (Bug #20066806, Bug #74932)

Changes in MySQL Connector/J 8.0.7 (2017-07-10, Development Milestone)

MySQL Connectors and other MySQL client tools and applications now synchronize the first digit of their version number with the (highest) MySQL server version they support. This change makes it easy and intuitive to decide which client version to use for which server version.

Connector/J 8.0.7 is the first release to use the new numbering. It is the successor to Connector/J 6.0.6.

Functionality Added or Changed

- **X DevAPI:** There are changes to some methods related to the `Result` interface:
  - `getLastDocumentId()` and `getLastDocumentIds()` have been replaced with `getDocumentId()` and `getDocumentIds()`, which are put under a new `AddResult` interface that extends `Result`.
  - A new `getAutoIncrementValue()` method is added to the new `InsertResult` interface that extends `Result`.

  See MySQL Connector/J X DevAPI Reference for more details. (Bug #25207784)
- **X DevAPI:** It is no longer permitted to pass an empty search condition, such as the NULL value or an empty string, to the `Collection.Modify()` and `Collection.Remove()` methods.
- **X DevAPI:** Connections using the X Protocol are now secure by default. Also, the `xdevapi.ssl-enable` connection option has been replaced by the `xdevapi.ssl-mode` option, which has `DISABLED`, `REQUIRED` (default), `VERIFY_CA`, and `VERIFY_IDENTITY` as its permitted values; see the description for the new option in Configuration Properties for details.
- **X DevAPI:** Consolidated the `BaseSession`, `NodeSession`, and `XSession` interfaces into a single `com.mysql.cj.api.xdevapi.Session` interface. The following related changes were also made:
  - Renamed `XSessionFactory` to `SessionFactory`.
  - Consolidated the `AbstractSession`, `NodeSessionImpl`, and `XSessionImpl` classes into the `com.mysql.cj.xdevapi.SessionImpl` class.
  - Removed the `Session.bindToDefaultShard()` method and the `VirtualNodeSession` interface.
  - The `mysqlx.getNodeSession()` method has been renamed to `mysqlx.getSession()` and it now returns a `Session` object.
  - The `DatabaseObject.getSession()` method now returns a `Session` object (instead of the old `Session` interface).

  See MySQL Connector/J X DevAPI Reference for more details.
- To avoid using JDBC statements inside core Connector/J classes, the following changes have been implemented:
  - Created a new `com.mysql.cj.api.Query` interface, which is implemented by `StatementImpl`.
  - Replaced the `com.mysql.cj.api.jdbc.interceptors.StatementInterceptor` interface with the `com.mysql.cj.api.interceptors.QueryInterceptor` interface.
  - Added a new method, `PacketPayload preProcess(PacketPayload queryPacket)`, to `QueryInterceptor`.
  - Renamed the connection property `statementInterceptors` to `queryInterceptors`. See Configuration Properties for details.
- Added Japanese collation for the `utf8mb4` character set.

Bugs Fixed

- **X DevAPI:** `createView()` failed with a `NullPointerException` when there were null inputs to it. This fix adds checks for nulls, and makes Connector/J throw the proper errors for them. (Bug #25575156)
- **X DevAPI:** `createaTable()` failed with a `NullPointerException` when there were null inputs to it. This fix adds checks for nulls, and makes Connector/J throw the proper errors for them. (Bug #25575103)
- **X DevAPI:** The connection properties `enabledSSLCipherSuites`, `clientCertificateKeyStoreUrl`, `clientCertificateKeyStoreType`, and `clientCertificateKeyStorePassword` were ignored for connections using the X Protocol. (Bug #25494338)
- **X DevAPI:** Calling `getNodeSession()` with an URL string containing SSL parameters caused a `CJCommunicationsException`. This has been fixed by creating a byte buffer to handle SSL handshake data. (Notice that `getNodeSession()` has since been consolidated into `getSession()`.) (Bug #23597281)
- **X DevAPI:** Concurrent asynchronous operations resulted in hangs, null pointer exceptions, or other unexpected exceptions. This has been fixed by correcting a number of problems with the `SerializingBufferWriter` and by limiting the number of buffers sent with a gathering write. (Bug #23510958)
- **X DevAPI:** When a thread failed to make a connection to the server using the X Protocol, the client application hung. A new connection property, `xdevapi.asyncResponseTimeout` (default value is 300s), now provides a duration beyond which the attempt to connect timeouts, and a proper error is then thrown. See description for the new option in Configuration Properties for details. (Bug #22972057)
- Connector/J failed a number of regression tests in the testsuite related to geographic information system (GIS) functions because of changes to GIS support by the MySQL server. The fix corrects the tests. (Bug #26239946, Bug #26140577)
- Attempts to connect to a server started with collation `utf8mb4_de_pb_0900_ai_ci` resulted in null pointer exceptions. (Bug #26090721)
- Configuration templates named by the connection property `useConfigs` were not recognized by Connector/J. (Bug #25757019, Bug #85555)
- A `NullPointerException` was returned when `getDate()`, `getTime()`, or `getTimestamp()` was called with a null `Calendar`. This fix makes Connector/J throw an `SQLException` in the case. (Bug #25650305)
- An `ArrayIndexOutOfBoundsException` was thrown when a server-side prepared statement was used and there was a `NULL` in a `BLOB`, `TEXT`, or `JSON` type column in the `ResultSet`. (Bug #25215008, Bug #84084)

Index

D

E

G

H

I

J

K

L

M

U

W

X

Z