MySQL Connector/J 5.1 Release Notes

Abstract

This document contains release notes for the changes in each release of MySQL Connector/J 5.1. It also contains release notes for earlier series of Connector/J.

For additional Connector/J documentation, see MySQL Connector/J 5.1 Developer Guide.

Updates to these notes occur as new product features are added, so that everybody can follow the development process. If a recent version is listed here that you cannot find on the download page (http://dev.mysql.com/downloads/), the version has not yet been released.

The documentation included in source and binary distributions may not be fully up to date with respect to release note entries because integration of the documentation occurs at release build time. For the most up-to-date release notes, please refer to the online documentation instead.

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.

For additional documentation on MySQL products, including translations of the documentation into other languages, and downloadable versions in variety of formats, including HTML and PDF formats, see the MySQL Documentation Library.

Document generated on: 2017-05-22 (revision: 11711)

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 5.1

Changes in MySQL Connector/J 5.1.42 (2017-04-28)

Version 5.1.42 is a maintenance release of the production 5.1 branch. It is suitable for use with MySQL Server versions 5.5, 5.6, and 5.7. It supports the Java Database Connectivity (JDBC) 4.2 API.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- A `null` value can now be extracted from a result set to a class belonging to the `java.time` package. Before, such an extraction resulted in a `NullPointerException` being thrown. Thanks to Martin Desharnais for contributing to the fix. (Bug #25250938, Bug #84189)
- Connector/J now checks that a MySQL server's SSL certificate and the Certificate Authority (CA) that issued it are not expired before establishing an SSL connection with the server, even if the connection property `verifyServerCertificate` is set to `false`. (Bug #20515688)
- Name-value pairs contained in the connection property `sessionVariables` can now be separated by either commas or a semicolons. (Bug #17757070)
- Added Japanese collation for the `utf8mb4` character set.

Bugs Fixed

- An invalid timezone identifier was used in `StatementRegressionTest.java` of the Connector/J testsuite. (Bug #25687718, Bug #85351)
- A `mysql` client failed to establish an SSL connection to the server using the SSL certificates provided in the Connector/J source package. It was because the certificates have been generated with the same Common Name. This fix corrects the Common Names and regenerates the SSL certificates. (Bug #25636947)
- The unit test `testsuite.simple.ResultSetTest.testPadding` failed with the error `Unknown character set: 'gb18030'` after the collation map updates in release 5.1.40. (Bug #25556597)
- In a multi-host connection, query timeouts did not occur as configured. It was because the `CancelTask` thread, when trying to access the top level, virtual connection object, ran into a race condition with the connection monitor and then hung. With this fix, the `CancelTask` thread is passed a direct reference to the underlying physical connection, with which it can execute the cancellation. (Bug #25490163, Bug #84783)
- `CallableStatement.extractProcedureName()` did not return the correct result when the procedure name contained a dash. This was due to an error in the `stripComments()` method of the `StringUtils` class, which has now been corrected. (Bug #25321524, Bug #84324)
- The `ConnectionImpl.isReadOnly()` method returned a confusing error message when it could not retrieve the read-only status of the server. The message has now been changed to “Could not retrieve transaction read-only status from server.” (Bug #25101890, Bug #83834)
- A `NullPointerException` was thrown when a null boolean value was being read from the database. (Bug #25048406, Bug #83662)
- After a `BIT` value had been retrieved from a result set, the `wasNull()` method of the result set returned value for the last `wasNull()` query instead of the value for the last retrieved column. (Bug #24841670, Bug #83368)
- Using a partially-quoted identifier (with only the database or the procedure name quoted) or a non-existent parameter to register an output parameter in a `CallableStatement` caused a

`NullPointerException`. With this fix, a partially-quoted identifier is accepted, and a non-existent parameter causes a `SQLError` to be thrown. (Bug #22333996, Bug #79561)

- `DatabaseMetaData.getProcedureColumns()` and `DatabaseMetaData.getFunctionColumns()` did not return expected results. This was due to the errors with the matching algorithm for the column names, which have now been fixed. Notice that, however, the effects of the connection parameter `getProceduresReturnsFunctions` on the two methods when JDBC 4 is used remain unchanged. (Bug #19531384, Bug #73775)
- When an `UpdatableResultSet` was used, trying to close the result set and its prepared statement simultaneously by different threads might result in a deadlock. This fix updates the synchronization mechanism for `UpdatableResultSet` to avoid the issue. (Bug #17653733, Bug #70704)
- After a connection had already switched catalog with `setCatalog()`, cached data from the old catalog was returned for a reused server-side prepared statement. With this fix, the cache of a server-side prepared statement cache now includes the catalog in its key to avoid wrong cache hits when the statement is reused on another catalog. (Bug #16714868, Bug #66430)

Changes in MySQL Connector/J 5.1.41 (2017-02-28)

Version 5.1.41 is a maintenance release of the production 5.1 branch. It is suitable for use with MySQL Server versions 5.5, 5.6, and 5.7. It supports the Java Database Connectivity (JDBC) 4.2 API.

Bugs Fixed

- Connections failed with `MySQLSyntaxErrorException: Unknown character set` when `connectionCollation=ISO-8859-13`. This was due to an error with Connector/J's internal charset mapping, which has now been fixed. (Bug #25504578)
- When loading classes through some external class loaders, `com.mysql.jdbc.Util` threw an `NoClassDefFoundError`. This was caused by `Class.getPackage()` returning null when some external class loaders were used. This fix replaces those calls of `Class.getPackage()` with calls of the new method `Class.getName()`, which return package names that are extracted from the fully-qualified class names. (Bug #25048543, Bug #83052)
- In the manifest for the Connector/J `JAR` file, the `Import-Package` directive specified version numbers for the `javax.net.ssl` package. The specification was unnecessary, and it caused the configuration of an SSL connection to a MySQL server to fail in an OSGi environment. The version requirement has now been removed. (Bug #24942672, Bug #82826)
- In a Fabric setup, when multiple threads required to have hashes computed, an `ArrayIndexOutOfBoundsException` might be thrown from inside `HashShardMapping`. This fix prevents the issue by having `HashShardMapping.getShardIndexForKey()` synchronized. (Bug #24289730, Bug #82203)
- When the configuration property `cacheResultSetMetadata` was set to `true`, a ping query using a `PreparedStatement` failed with a `NullPointerException`. This fix moves the ping query to an earlier stage of the statement execution, which prevents the exception. (Bug #23535001, Bug #81706)
- The `setFabricShardTable()` method failed to parse qualified table names (in the format of *`database_name.table_name`*), which causes `SQLExceptions` to be thrown. (Bug #23264511, Bug #81108)
- A race condition occurred when a call of `Connection.setNetworkTimeout()` was followed closely by a call of `Connection.close()`, and a `NullPointerException` might result if the connection was closed before the executor supplied to `setNetworkTimeout()` was able to set the timeout, as the executor would run into a null `mysqlConnection` object. This fix removed the race condition. (Bug #21181249, Bug #75615)
- With the connection properties `cacheServerConfiguration=true` and `elideSetAutoCommits=true`, any new connection to the server obtained after the first connection

was established had the variable `autoCommit` equaled `false`, even if the value of the variable was `true` on the server. That was because the value of `autoCommit` was not properly initialized when a new connection was established, and this fix corrects that.

Also, since release 5.1.41, the functionality of the property `elideSetAutoCommits` has been disabled due to Bug# 66884. Any value given for the property is now ignored by Connector/J. (Bug #17756825, Bug #70785)

- When using Tomcat and a web application that utilized Connector/J was down, Tomcat was unable to stop the `AbandonedConnectionCleanupThread` started internally by Connector/J, leading to multiple instances of the thread when the web application was restarted; or, Tomcat was able to stop the thread but unable to restart it on reload of the web application. Different combinations of Tomcat's default settings, usage of Tomcat's `ServletContextListener` feature, and locations of the Connector/J jar could result in the undesired behaviors, as well as warning messages in the Tomcat error log saying it was unable to stop the thread and a memory leak was likely.

  The implementation of `AbandonedConnectionCleanupThread` has now been improved, so that there are now four ways for developers to deal with the situation:

  - When the default Tomcat configuration is used and the Connector/J jar is put into a local library directory, the new built-in application detector in Connector/J now detects the stopping of the web application within 5 seconds and kills `AbandonedConnectionCleanupThread`. Any unnecessary warnings about the thread being unstoppable are also avoided. If the Connector/J jar is put into a global library directory, the thread is left running until the JVM is unloaded.

  - When Tomcat's context is configured with the attribute `clearReferencesStopThreads="true"`, Tomcat is going to stop all spawned threads when the application stops unless Connector/J is being shared with other web applications, in which case Connector/J is now protected against an inappropriate stop by Tomcat; the warning about the non-stoppable thread is still issued into Tomcat's error log.

  - When a `ServletContextListener` is implemented within each web application that calls `AbandonedConnectionCleanupThread.checkedShutdown()` on context destruction, Connector/J now, again, skips this operation if the driver is potentially shared with other applications. No warning about the thread being unstoppable is issued to Tomcat's error log in this case.

  - When `AbandonedConnectionCleanupThread.uncheckedShutdown()` is called, the `AbandonedConnectionCleanupThread` is closed even if Connector/J is shared with other applications. However, it may not be possible to restart the thread afterwards.

  (Bug #17035755, Bug #69526)

  References: See also: Bug #14570236, Bug #16443387.

Changes in MySQL Connector/J 5.1.40 (2016-10-03)

Version 5.1.40 is a maintenance release of the production 5.1 branch. It is suitable for use with MySQL Server versions 5.5, 5.6, and 5.7. It supports the Java Database Connectivity (JDBC) 4.2 API.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- A very old workaround for Bug#36326 (fixed in release 5.1.32) has been removed, so Connector/J can now use local transaction states when the MySQL server has query cache enabled. (Bug #19974685)

- Added support for the error codes of two MySQL server errors, ER_XA_RBTIMEOUT and ER_XA_RBDEADLOCK. (Bug #13702433, Bug #64188)

Bugs Fixed

- In certain cases, the exception interceptor was being triggered twice in the internal `SQLException` factory method. Also, if the exception interceptor returned an exception with the cause already initialized, the same factory method would fail to initialize the cause again, and the real cause for the exception remained uncaptured. (Bug #23743956)

- Continuing to use a Fabric connection after one of the slaves was removed might result in a `ConcurrentModificationException`. This happened because the list of slave hosts managed by Connector/J was being traversed and modified at the same time as the replication topology was changed. This patch fixes the issue by having Connector/J use a slave host list that is safe to be modified. (Bug #23738636)

- When the connection property `useCursorFetch` was "true" and updatable result sets were used, `executeQuery()` failed for JSON data with an `SQLException: Unknown type`. It was because the data decoding routine for updatable JSON data was missing from Connector/J, and it has now been implemented. (Bug #23197238)

- The `FabricMySQLConnectionProxy` method `nativeSQL()` always returned `null`. The method is now properly implemented. (Bug #23103408, Bug #81072)

- In a Fabric setup, `getMetaData().supportsMixedCaseIdentifiers()` always returned "true," even if `lower_case_table_names` was set to "1" on the servers. This was because `FabricMySQLConnectionProxy.lowerCaseTableNames()` was not implemented properly. With this fix, the function now returns the proper value for the current active connection, so that `getMetaData().supportsMixedCaseIdentifiers()` also returns the correct value.

  Users are reminded that in a Fabric setup, `lower_case_table_names`, as well as all other settings, should be configured the same way for all servers in the Fabric server group. (Bug #23103406, Bug #81056)

- A memory leakage occurred when the connection properties `cachePrepStmts` and `useServerPrepStmts` were both set to be `true` and server-side prepared statements were set as non-poolable, which resulted in the prepared statement being not closable by the client, and the number of prepared statements then kept on increasing.

  When the memory leakage described above occurred, it did not make Connector/J fail, as Connector/J switched to using client-side prepared statements when the maximum number of prepared statements was reached. However, when `rewriteBatchedStatements` was also set to true, the switch to client-side prepared statements did not occur, and Connector/J threw the MySQLSyntaxErrorException ("Can't create more than max_prepared_stmt_count statements") when the client wanted to create more prepared statements than allowed.

  This fix corrected the way prepared statements are handled in order to avoid both of the problems described above. (Bug #22954007, Bug #80615)

- `ResultSet.getString()` sometimes returned garbled data for columns of the JSON data type. This was because JSON data was binary encoded by MySQL using the utf8mb4 character set, but decoded by Connector/J using the ISO-8859-1 character set. This patch fixes the decoding for JSON data. Thanks to Dong Song Ling for contributing to the fix. (Bug #22891845, Bug #80631)

- When working with MySQL Fabric, Connector/J hung when the Fabric node was down. With this fix, when the Fabric node is down, all active connections continue to work based on the cached information on the server group and the sharding setup, although no new connections can be established. (Bug #22750465)

- When Connector/J retrieved the value of a `BIT` column as an integer using, for example, `getInt()` or `getLong()`, it returned a wrong value if the BIT value happened to be equivalent to the decimal

value of some ASCII digit. This was because Connector/J attempted to parse the value as a string-encoded integer; thus, for example, the BIT value “110001” (decimal 49 in binary) was interpreted as the string “1” (whose ASCII value in decimal is 49), so the numerical value of “1” was returned. This fix corrected the parsing behavior of Connector/J on `BIT` values, so they are always interpreted as binary coded. (Bug #21938551, Bug #78685)

- Connector/J could not parse the host name and port number from connection URLS staring with the word “address,” resulting in an `UnknownHostException`. This was because the URLs were being interpreted incorrectly as using the alternate URL format, which starts with “address=....” This patch fixes the parser for proper interpretation of the URL in the situation. (Bug #21389278, Bug #77649)

- When the connection property `useLocalTransactionState` was set to “true” and `autocommit` was set to “false” on the server, if any exception was thrown, any further calls for `rollback()` or `commit()` were not sent to the server. It was because when there was an exception while executing a query, Connector/J lost the information regarding the server's transaction state. This patch fixes this issue by preserving the previous transaction state for the current connection when any exception is thrown. (Bug #20212882, Bug #75209)

- An invalid connection URL caused Connector/J to throw a `NullPointerException`. With this fix, an `SQLException` is thrown instead in the situation. (Bug #18759269, Bug #72632)

- When a very large amount of compressed data is transmitted from the server to the client and under very special circumstances, a `CommunicationsException` might occur. It happened when a single compressed packet from the server was not big enough to provide an entire uncompressed packet to the client side. With this fix, Connector/J reads as many compressed packets as necessary to fill the size of the uncompressed packet that was being requested. Thanks to Ryosuke Yamazaki for contributing to the fix. (Bug #11756431, Bug #48346)

Changes in MySQL Connector/J 5.1.39 (2016-05-09)

Version 5.1.39 is a maintenance release of the production 5.1 branch. It is suitable for use with MySQL Server versions 5.5, 5.6, and 5.7. It supports the Java Database Connectivity (JDBC) 4.2 API.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- For MySQL server 5.7.5 and later, the EOF packet in the MySQL server/client protocol has been deprecated and replaced by the OK packet. The change is now supported by Connector/J. (Bug #23212347)

Bugs Fixed

- Connector/J hung, returned a **`NullPointerException`**, or returned an incorrect exception when using result sets with the properties `TYPE_FORWARD_ONLY` and `CONCUR_UPDATABLE`. It was due to an inaccurate check for the cursor for the result set. This fix makes sure Connector/J checks accurately on whether a cursor has been requested, both when executing a statement and fetching its results. (Bug #23201930)

- When using server-side prepared statements with `profileSQL=true` and `useInformationSchema=true`, an *`SQLException`* (“ `ResultSet is from UPDATE. No Data`”) occurred when the client tried to advance to the next row in the result set. This was due to a failure of an internal query for metadata, which is now prevented by this fix. (Bug #23188498)

- `LoadBalanceConnectionGroupManager.removeHost()` was not removing hosts as expected. This fix tries to ensure that host removals will be successful under different situations. (Bug #22848249)

  References: See also: Bug #22678872.

- For connections with `useCursorFetch=true` and fetch size set with `defaultFetchSize` or `setFetchSize`, if data from a `TIME` and a `BLOB` data column was selected together, corrupted value for the TIME data was returned. (Bug #22833410, Bug #80522)

- For a load-balanced connection, an *`ArrayIndexOutOfBoundsException`* was thrown when `ConnectionGroupManager.removeHost()` was called. It was due to an error in `LoadBalancedConnectionProxy.removeHost()`, which has now been fixed. (Bug #22730682)

- A Fabric connection threw a *`NullPointerException`* when all hosts have been removed from its host list, or when the internal load-balanced connection became null due to inconsistency of the replication connection. This fix adds to Connector/J the abilities to handle those situations. Also, a new connection property, `loadBalanceHostRemovalGracePeriod`, has been introduced, which sets the grace period for waiting to remove the currently active host from a load-balanced connection. See the entry for the new property in Driver/Datasource Class Names, URL Syntax and Configuration Properties for Connector/J for details. (Bug #22678872)

  References: See also: Bug #22848249.

- `FabricMySQLDataSource.getConnection()` threw a null pointer exception when a master failover took place. (Bug #22598938)

- The OSGi manifest file in the Connector/J JAR file did not expose the MySQL Fabric packages, so the Fabric-related classes could not be resolved even though they were present in the JAR file. (Bug #22385172)

- With some Tomcat web applications, when Connector/J connects to the server with `useLegacyDatetimeCode=false` without setting `serverTimeZone`, a NullPointerException was returned. This was because the timezone property file for Connector/J was loaded by the bootstrap class loader, which did not know the location of the property file and thus failed to load it. This fix avoids the problem by making Connector/J use the same class loader for both the property file and the Connector/J classes. (Bug #22353759, Bug #79343)

- When using JDBC 4.2 and with the connection property `cachePrepStmts` set to "`true`," after a prepared statement had been cached, rerunning the SQL statement resulted in a pre-JDBC 4.2 `PreparedStatement` object being instantiated. This fix prevents the problem by having the `PreparedStatement` instantiated by a factory instead of a constructor method. (Bug #22352812, Bug #79598)

- At every connection, Connector/J got the `sql_mode` variable from the server and tried to parse it as a number; because sql_mode is not a number (except for very old versions of MySQL), an *`NumberFormatException`* was always thrown and then caught by the code. This fix refactored the code to avoid the unnecessary throwing and catching of the error. (Bug #21181466, Bug #77171)

- When inserting multiple timestamp values into a table with `useLegacyDatetimeCode=false` and `useCursorFetch=true`, after a null value had been inserted, further inserts could not change a timestamp's value. This fix makes sure the binding of the value is reset before a new insert takes place. (Bug #20685487, Bug #75956)

- The exception message in `CallableStatement()` for incorrect output parameter registration gave little detail and the wrong error code. (Bug #18068303, Bug #71131)

- Calling `getTimestamp()` on a timestamp column resulted in a *`java.sql.SQLException`* ("`Cannot convert value ... to TIMESTAMP`"). That was due to the missing metadata for each row in the ResultSet. This fix ensures that the metadata is no longer missing. (Bug #16738378, Bug #56479)

- On very fast servers with other third-party components accessing the data, a `ConcurrentModificationException` was sometimes thrown. This fix prevents the exception by adding a synchronization to `ConnectionImpl.closeAllOpenStatements()`, and by refactoring part of the code inside the method. (Bug #16736619, Bug #59462)

Changes in MySQL Connector/J 5.1.38 (2015-12-07)

Version 5.1.38 is a maintenance release of the production 5.1 branch. It is suitable for use with MySQL Server versions 5.5, 5.6, and 5.7. It supports the Java Database Connectivity (JDBC) 4.2 API.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- When connecting to a MySQL server 5.7 instance that supports TLS, Connector/J now prefers a TLS over a plain TCP connection. (Bug #21947042)
- Two new connection properties, `allowSlaveDownConnections` and `readFromMasterWhenNoSlaves`, have been introduced for configuring replication-aware connections. See Configuring Master/Slave Replication with Connector/J and the entries for the new properties in Driver/Datasource Class Names, URL Syntax and Configuration Properties for Connector/J for details. (Bug #21286268)

Bugs Fixed

- In a Fabric environment, a `ClassCastException` resulted when a stored procedure was called with an `INOUT` parameter. This was because in this situation, what was being cast was a proxied version of the object. This fix extracts the underlying object from its proxy, allowing it to be cast to the implementing class. (Bug #22096981)
- `getTypeInfo()` erroneously returned a `PRECISION` value of “255” instead of “65535” for the data type `VARBINARY`. (Bug #21978216)
- A deadlock was observed when in the same server group, two concurrent threads were using different Fabric connections, and one executed a failover procedure while the other simultaneously called a method that acquired a lock on the underlying replication connection instance monitor. This fix revised the locking mechanism of replication connections, in order to prevent the observed deadlocks. (Bug #21966391, Bug #21934573, Bug #78710)
- State information of a Fabric group was not updated by Connector/J after a local cache's Time to Live (TTL) expired, which resulted in the client not recognizing any topology changes happening to the group like a master failover, a server rejoining, and so on. (Bug #21296840, Bug #17910835)
- Connector/J threw an `AbstractMethodError` when a JDBC 4 functionality (for example, `createBlob()`) was used on a replication-aware connection. This has been fixed by putting replication connections under the multi-host connection proxy structure that Connector/J has been using for load-balanced and fail-over connections. (Bug #11763419)

  References: See also: Bug #11763401.
- After the initial call of `Connection.setReadOnly()` following the creation of a replication-aware connection, subsequent calls of `Connection.setReadOnly()` could not change the nature of the connection. This has been fixed by putting replication connections under the multi-host connection proxy structure that Connector/J has been using for load-balanced and fail-over connections. (Bug #11763401)

  References: See also: Bug #11763419.

Changes in MySQL Connector/J 5.1.37 (2015-10-15)

Version 5.1.37 is a maintenance release of the production 5.1 branch. It is suitable for use with MySQL Server versions 5.5, 5.6, and 5.7. It is also the first release of MySQL Connector/J to support the Java Database Connectivity (JDBC) 4.2 API.

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- `methodCompressedInputStream.getNextPacketFromServer()` has been refactored to reduce memory use and garbage collection efforts caused by the use of the inflater. (Bug #21648826, Bug #78106)
- The code for executing a `REPLACE` statement when `rewriteBatchedStatements=true` has been refactored by putting multiple batched statements into a single query, making it work more like an `INSERT` statement. This increases the efficiency for running `REPLACE` statements. Thanks to Jie Han for contributing the code. (Bug #21429909, Bug #77681)
- A new connection property, `sendFractionalSeconds=true|false`, has been introduced. It controls whether fractional seconds in timestamps are to be truncated on the client side or to be sent to the server side for truncation there. See the entry for the new property in Driver/Datasource Class Names, URL Syntax and Configuration Properties for Connector/J for details. Thanks to Kwon Nam Son for contributing the code. (Bug #21304726, Bug #77449)
- A new connection property, `enableEscapeProcessing`, has been introduced for supporting JDBC 4.2. It sets the default escape processing behavior for Statement objects. See the entry for the new property in Driver/Datasource Class Names, URL Syntax and Configuration Properties for Connector/J for details.
- Connector/J now supports the JSON data type, which has been supported by the MySQL server since release 5.7.8.

Bugs Fixed

- A `ClassCastException` resulted for a Fabric connection when `rewriteBatchedStatements` was “true” and a batch update was rewritten. (Bug #21876798)
- A number of regression tests in the testsuite failed when they were run against MySQL 5.7, because the `NO_AUTO_CREATE_USER` SQL mode had become the default behavior of the server since MySQL 5.7.7, making some `GRANT` statements in the test fail when a new user had to be created. With this fix, `CREATE USER` statements have been added to the regressions tests when needed, to prevent the test failures. (Bug #21697684, Bug #78225)
- Connecting to MySQL 5.0 and 5.1 using Connector/J 5.1.36 resulted in an `SQLException`, with a complaint for an “Unknown system variable 'language'”. (Bug #21415165, Bug #77665)
- The `IS_GENERATEDCOLUMN` field was empty in the result returned by `DatabaseMetaData.getColumns()`. This fix corrects the field so that it contains a `YES` or `NO` according to whether the column is generated or not. (Bug #20969312, Bug #76859)
- `getTypeInfo()` returned an incorrect `PRECISION` value of “255” for the data type of `VARCHAR`. The return value has been corrected to “65535.” (Bug #20675539, Bug #76187)
- When a connection is forcefully closed with `abortInternal()` in the `ConnectionImpl` class, a null point exception sometimes resulted. This is now avoided by putting the associated `this.io.releaseResources()` call inside a try block, so that the exception, unavoidable due to a race condition, can be properly caught and ignored. (Bug #20536592, Bug #75849)
- If the MySQL server's default authentication method was SHA256 but neither one of the Connector/J connection properties `allowPublicKeyRetrieval` and `serverRSAPublicKeyFile` was set, the authentication failed with a `TransientConnectionException`, complaining that the public key could not be retrieved. With this fix, authentication continues in the situation, allowing other enabled authentication methods to be tried. (Bug #20433047, Bug #75670)
- When a lock wait timeout occurred, an `SQLException` was thrown while an `SQLTransientException` should be thrown instead. It was due to a wrong SQLState number used in the code, which has now been fixed. (Bug #16634180)

- When the time zone on the MySQL server was configured to “GMT” but the client was in a different time zone, Connector/J would make wrong adjustments for event timestamps when working with the server. (Bug #11758179, Bug #50348)

Changes in MySQL Connector/J 5.1.36 (2015-07-02)

Version 5.1.36 is a maintenance release of the production 5.1 branch. It is suitable for use with MySQL Server versions 5.5, and 5.6.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- The files in the Maven Central Repository for each Connector/J release have been reorganized and a source artifact (`*-sources.jar`) has been added, for release 5.1.21 and after. (Bug #20283655, Bug #75335)

Bugs Fixed

- Results returned by `DatabaseMetaData.getCatalogs()` were not properly sorted by catalog name. (Bug #21215151)
- A class cast exception occurred when Connector/J was executing a streaming prepared statement in a Fabric environment. (Bug #21184949, Bug #77217)
- Connector/J did not use the character set specified with the connection property `passwordCharacterEncoding` for authentication. With this fix, the property is now honored. If it is not set, Connector/J uses the value of `characterEncoding` instead; if even that is not set, Conenctor/J defaults to UTF-8 for the password's encoding. (Bug #20825727)
- A failover did not occur for a MySQL Fabric connection during the failure of a master server. It was because the state change of the Fabric connection group was not handled properly, which has been corrected by this fix. (Bug #20821888, Bug #75113)
- When the `getDate()` and `getTime()` methods of `ResultSet` were called to retrieve values with fractional seconds, a “bad format” error occurred. With this fix, the fractional seconds are dropped before the retrieved values are converted into a `Date` or `Time` object. (Bug #20804635)
- A `StringIndexOutOfBoundsException` occurred when `getProcedureColumns()` were trying to return ENUM or SET type procedures or functions that involved reserved words. That was caused by a problem in the parser for the functions' or procedures' CREATE statements, which this fix corrects. (Bug #20727196)
- At every connection, Connector/J executed a `SHOW VARIABLES WHERE` statement over a multitude of variables, which consumed a lot of time and memory. To improve the efficiency of the code, this fix replaces the statement with the more efficient `SELECT @@variable` query and also implements some other related changes. (Bug #20408891, Bug #75592)
- `getProcedures()`, `getFunctions()`, `getProcedureColumns()`, and `getFunctionColumns()` returned duplicate results when the connection parameter `nullCatalogMeansCurrent` was set to “false.” (Bug #19803348)
- An `UnsupportedEncodingException` during handshake gave rise to a `NullPointerException`. With this fix, the `NullPointerException` is no longer thrown, and a proper error message is provided by Connector/J. (Bug #18758686, Bug #72630)
- When using Connector/J with MySQL Fabric, the `createGroup()` method failed with a `ClassCastException`. (Bug #18719760, Bug #72546)

- Using Connector/J to connect with non-null user name and password to a MySQL Fabric server that had authentication disabled resulted in a `NullPointerException`. (Bug #18425861, Bug #72077)
- The change user functionality in Connector/J was dependent on Bug# 70865 of the MySQL server (for releases 5.5 and later, the server unnecessarily sends a `plugin_request_packet` for every `COM_CHANGE_USER` packet). That dependence made the `com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource` interface fail for server versions or forks that did not have the same bug. This fix removes that dependence and makes the interface function properly, whether the server suffers Bug# 70865 or not. (Bug #17810800, Bug #70927)
- `JDBC4MySQLPooledConnection` keeps a list of `statementEventListener` instances named `statementEventListeners`, which is used as monitor lock whenever the `fireStatementEvent()` method is called. However, because the pooled connection's `close()` method set `statementEventListener` to be null, if a prepared statement was closed after its holding pooled connection had already been closed, the subsequent `fireStatementEvent()` call would run into a `NullPointerException`. This fix prevents the problem by having `JDBC4MySQLPooledConnection` initializing `statementEventListeners` properly and never setting it to null, thus allowing it to be used all the time as a monitor lock. (Bug #16444069, Bug #62452)

Changes in MySQL Connector/J 5.1.35 (2015-03-23)

Version 5.1.35 is a maintenance release of the production 5.1 branch. It is suitable for use with MySQL Server versions 5.5, and 5.6.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- The failover support by Connector/J has been refactored to be implemented independently of Connector/J's load-balancing feature. This resolves a number of issues including, for example, (1) a failover being triggered inadvertently, (2) a failover process kept trying to connect to an unavailable server (while there was an available one), and (3) the number of active connections kept growing until Connector/J threw an exception when there were successive failovers within the same connection. (The issues are regressions of Bug #75168.) (Bug #20533907, Bug #75885)
- A number of new configuration properties have been introduced:
  - `enabledSSLCipherSuites`: For setting the enabled cipher suites used by JSSE. This configuration property needs to be set when using Java 7 or lower or MySQL Server Community edition version 5.7.6 or higher.
  - `readOnlyPropagatesToServer`: For controlling the implicit propagation of the read-only transaction access mode to the server, which affects optimization for InnoDB read-only transactions.
  - `noTimezoneConversionForDateType` and `cacheDefaultTimezone`: For improving Connector/J's time zone support; see the changelog entry for Bug #18028319/Bug #71084 below for details.

  See the descriptions for these properties in Driver/Datasource Class Names, URL Syntax and Configuration Properties for Connector/J for more details.
- Connector/J now exposes the `PreparedStatment.ParseInfo` class for external usage, with no capture of the connection, which allows global, highly-concurrent parse caches to be implemented.

Bugs Fixed

- Issues with a number of the regression tests in the Connector/J test suite caused them to fail when they were run against MySQL 5.7.6. Those tests have now been fixed. (Bug #20606107)
- Calling the function `Clob.setString()` with valid input resulted in a string index out of bounds exception. (Bug #20453712)
- Calling the function `Clob.position()` resulted in an exception, because the function passed the wrong index value to the `getSubString()` function. (Bug #20453712)
- When in streaming mode, if Connector/J encountered an exception while reading from the stream, a subsequent call to the `close()` method of the result set would put the thread into a blocking state, from which it would not exit. With this fix, the result set can now be properly closed when, due to an error, there is no more data to be loaded from the stream. (Bug #20272931, Bug #75309)
- The `setTimestamp` method failed in a Fabric connection with a null pointer exception. It was because the implementation for the `getCalendarInstanceForSessionOrNew` method was missing, which has been added by this fix to Connector/J. (Bug #20217686, Bug #75080)
- When using JDBC 4 features of Connector/J 5.1 with JDK 6 and above, custom implementations of the interface `LoadBalanceExceptionChecker` failed to work. This was because most of the JDBC 4-specific classes of Connector/J do not implement a JDBC interface directly, but extend other JDBC 3 classes instead, so that the `LoadBalancingConnectionProxy.isInterfaceJdbc()` check did not work. This fix makes the checking work by extending its search to include also the super classes. (Bug #20204783, Bug #75168)
- The `readRemainingMultiPackets` method in the `MysqlIO` class returned incorrect results when a row was larger than 16MB in size. This fix corrects the wrong type conversion occurred during the calculation, which caused the problem. (Bug #20112694, Bug #74998)
- Calling `getString()` after `rs.relative()` had been called with an argument smaller than “-1” resulted in a null pointer exception, because `rs.relative()` could not handle the argument properly. (Bug #19536760)
- All occurrences of the `StringBuffer` class in the Connector/J code has been replaced with the `StringBuilder` class, in order to improve code performance. (Bug #19465516, Bug #73595)
- Quoted identifiers in some SQL statements were not properly escaped. (Bug #18925727)
- A `java.sql.date` value was stored incorrectly on the server and also returned incorrectly if the client and the server were in different time zones when `useLegacyDatetimeCode=false` or `useTimezone=true`. This was due to the time-zone conversion performed by Connector/J on the SQL DATE type. To avoid the issue, a new property `noTimezoneConversionForDateType` has been created for Connector/J, which is set to “true” by default, preventing Connector/J to perform the kind of time-zone conversion that caused this bug.

  In addition, another new property `cacheDefaultTimezone` has been created: when it is set to “true” (by default), Connector/J caches the time zone first obtained from the client and uses it throughout the time the application is running. When it is set to “false,” Connector/J becomes aware of time zone changes in the client at runtime that are initiated by calling `java.util.TimeZone.setDefault(zone)`. (Bug #18028319, Bug #71084)
- A deadlock occurred when concurrent prepared statements making use of timestamp objects were executed in a single connection. To avoid this, the locking mechanism involved has been changed, so that the calendar object for the session is only locked when it is actually being used. (Bug #15936413, Bug #67760)
- There was an unnecessary call of `targetCalendar.setTime()` in the `newSetTimestampInternal()` method of the `PreparedStatement.java` class, which modified the user-supplied `Calendar` object and might cause side effects for the client application. The unnecessary call has now been eliminated by the fix put in for Bug #18028319/Bug #71084. (Bug #11761585, Bug #54095)

Changes in MySQL Connector/J 5.1.34 (2014-11-04)

Version 5.1.34 is a maintenance release of the production 5.1 branch. It is suitable for use with MySQL Server versions 5.5, and 5.6.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Connector/J now supports connections to MySQL servers through a SOCKS proxy. Two new configuration properties, `socksProxyHost` and `socksProxyPort`, have been introduced to support this new feature; see the entries for them in Driver/Datasource Class Names, URL Syntax and Configuration Properties for Connector/J for details. Note that authentication is not directly supported in the current implementation for SOCKS proxy support. (Bug #13940005, Bug #64862)
- The process for building Connector/J from source has been revamped to fix incompatibilities with Eclipse 4.4, remove dependency on Ant Contrib, improve on the documentation for the code, and to improve on the structure of the output. See Installing from Source for the updated instructions for building Connector/J 5.1.34.

Bugs Fixed

- When connecting to MySQL server 5.7.4 or earlier, the connection failed when the password was blank for any users created on the server using the `mysql_old_password` plugin. This fix refactored the `testOldPasswordPlugin` of Connector/J so that it works well with the server under the situation. (Bug #19383371)
- An exception was thrown when users tried to run the `testsuite.fabric` test cases without specifying the property `com.mysql.fabric.testsuite.port`. With this fix, the Fabric test cases will not be run unless the test environment has been configured. (Bug #18474141)

Changes in MySQL Connector/J 5.1.33 (2014-09-25)

Version 5.1.33 is a maintenance release of the production 5.1 branch. It is suitable for use with MySQL Server versions 5.5, and 5.6.

Bugs Fixed

- Many parameters specified with options when making an Ant build of Connector/J from source (for example, `com.mysql.jdbc.testsuite.url.sha256default`) were not set as system properties and thus were not available for use in the JUnit tests in the testsuite. With this fix, all com.mysql.jdbc.test.* and com.mysql.jdbc.testsuite.* parameters are forwarded to the JUnit tests. (Bug #19505524)
- The 4-byte UTF8 (`utfbmb4`) character encoding could not be used with Connector/J when the server collation was set to anything other than the default value (`utf8mb4_general_ci`). This fix makes Connector/J detect and set the proper character mapping for any `utfmb4` collation. (Bug #19479242, Bug #73663)
- XA connections failed with a class cast exception for a load-balanced configuration with `pinGlobalTxToPhysicalConnection=true`. This was because some XA-related classes used `com.mysql.jdbc.ConnectionImpl` in method parameters during calls. This fix makes the classes use `com.mysql.jdbc.Connection` instead in those cases. (Bug #19450418, Bug #73594)
- The `MANIFEST.MF` file for the JAR package of Connector/J included an empty line between the attributes `Import-Package` and `Name`, which was interpreted as a mark for the end of the main section of the manifest file. This fix removes the empty line. (Bug #19365473, Bug #73474)

- `changeUser()` calls might result in a connection exception (“Got packets out of order”) or a freeze if the `useCompression` option was set to true. (Bug #19354014, Bug #19443777, Bug #73577)

- The tests `testByteStreamInsert` and `testUseCompress` in the Connector/J testsuite failed with the message “Row size too large (> 8126)...” when they were run against the MySQL server version 5.6.20. This is due to a known limitation with the server version, for which `innodb_log_file_size` has to be set to a value greater than 10 times the largest BLOB data size found in the rows of the tables plus the length of other variable length fields (that is, VARCHAR, VARBINARY, and TEXT type fields) (see the MySQL 5.6.20 changelog for details). This fix adds, for MySQL 5.6.20, a check for `innodb_log_file_size` to assert that its value is 335544320 or larger and throws an exception if it is not so, asking the user to increase the size of `innodb_log_file_size`. (Bug #19172037)

- Using Java 6 or later, running `LOAD DATA INFILE` with a prepared statement resulted in an `IndexOutOfBoundsException`. This was a regression introduced by an issue in the patch for Bug #72008, which has now been fixed. (Bug #19171665, Bug #73163)

  References: This issue is a regression of: Bug #72008.

- Connector/J failed the test for Bug# 64205 in the testsuite, thus might return a garbled error message for an invalid query when connected to MySQL server 5.5 or later. The failure was a regression introduced by the fix for Bug #18836319, which made the issuing of `SET NAMES` dependent on the character set difference between the server and the connection. This fix corrects the way by which the error messages are interpreted, so that they will come out right at any stage of a connection. (Bug #19145408)

  References: This issue is a regression of: Bug #18836319.

- Errors occur when Connector/J mapped Windows time zones to Olson time zones for “Caucasus Standard Time” and “Georgian Standard Time”. This fix corrected and updated all the time-zone mappings in Connector/J using the data from the IANA Time Database v.2014g and the Unicode Common Locale Data Repository (CLDR) v.25. (Bug #17527948, Bug #70436)

- The test case `testsuite.simple.XATest.testRecover` failed when run against MySQL servers 5.7.0 to 5.7.4, because Connector/J did not understand the new output format for `XA RECVOER` that was introduced since MySQL 5.7.0 for fixing Bug #19505524. That fix has recently been reverted in MySQL 5.7.5, so that XA RECOVER output is now, by default, in the old format again. The test case, which runs well with MySQL 5.7.5, is now skipped for server versions 5.7.0-5.7.4. (Bug #17441747)

  References: This issue is a regression of: Bug #19505524.

Changes in MySQL Connector/J 5.1.32 (2014-08-11)

Version 5.1.32 is a maintenance release of the production 5.1 branch. It is suitable for use with MySQL Server versions 5.5, and 5.6.

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- Connector/J used to always check every SQL statement in a server-side prepared statement to see whether it contained the `ON DUPLICATE KEY UPDATE` clause; but because the clause is only used with `INSERT` statements, the checks are unnecessary for other SQL statements while they reduce the performance of Connector/J. A new, boolean connection property `avoidCheckOnDuplicateKeyUpdateInSQL` has been added, by which the checks for the `ON DUPLICATE KEY UPDATE` clause can be disabled. (Bug #18232840, Bug #71672)

- Connector/J now supports Fabric 1.5. Older versions of Fabric are no longer supported.

Bugs Fixed

- Preparing a call to a stored procedure with Fabric caused a null pointer exception to be thrown. (Bug #19034681, Bug #73070)
- A bug in the Linux kernel version 3.6 and earlier caused the `MysqlIO.clearInputStream()` method to enter an endless loop. This fix changes the way the looping condition is evaluated, in order to avoid the problem. (Bug #19022745, Bug #73053)
- Connector/J returned the incorrect return code “0” for a thrown exception when the failure happened in the context of a global XA transaction. With this fix, Connector/J now wraps any unexpected exception in an `XAException` in that case and returns the error code `XAER_RMFAIL`. (Bug #18970520, Bug #72890)
- Calling the `changeUser` method to switch to a user created using the `sha256_password` plugin would result in a null pointer exception. This was due to the fact that the `fromServer` buffer was unavailable when `changeUser` called the `sha256_password` plugin, and this fix makes the plugin accommodate to that. (Bug #18869381)
- The test `testSha256PasswordPlugin` failed when executed against a commercial version of the MySQL server. (Bug #18852682)
- Connecting to a user on the server created using the `sha256_password` plugin failed when the password specified by the client was an empty string (or when no password was specified). This fix makes Connector/J allow an empty password in that case, sending it to the server without applying RSA encryption on it. (Bug #18852587)
- Trying to use any character sets other than UTF-8 for communications between client programs and the MySQL server caused Connector/J to perform extra queries after the initial connection, resulting in higher latency and overhead for the connection. To prevent extra queries, this fix eliminates the mechanism of setting the character set to values other than “utf-8” by issuing a `SET NAMES` statement to the server, and allows the use of the connection property `characterEncoding` to set the character set value in Connector/J's response packet during handshake. (Bug #18836319, Bug #72712)
- Connector/J failed the test `ConnectionRegressionTest.testBug7607()` when Java 8 was used, due to the new static character set mappings in Java 8. The bug no longer exits after a refactoring of the character set code for Connector/J 5.1.32. (Bug #18809129)
- A null pointer exception was thrown in `isInterfaceJdbc()` sometimes when load balancing was used and the application involved runtime instrumentations. (Bug #18691866, Bug #72502)
- The keys generated by `INSERT` statements using the `ON DUPLICATE KEY UPDATE` clause were incorrect when the clause “`ON DUPLICATE KEY UPDATE`” was not written exactly as so (for example, when spaces or comments were inserted in between the words). (Bug #18344403, Bug #71923)
- XA connections failed with a `ClassCastException` for a load-balanced configuration with multiple hosts. This was because some XA-related classes used `com.mysql.jdbc.ConnectionImpl` in method parameters during calls. This fix makes the classes use `com.mysql.jdbc.Connection` instead in those cases. (Bug #16722757, Bug #62577)
- The Ant script for building Connector/J from source failed to check the availability of `javac` and `rt.jar` from JDK 1.6 before compilation. That caused compilation to fail when the two files weren't available. This fix corrects the step in the build script that checks for those files before compilation. (Bug #11748301, Bug #35829)

Changes in MySQL Connector/J 5.1.31 (2014-06-07)

Version 5.1.31 is a maintenance release of the production 5.1 branch. It is suitable for use with many MySQL Server versions, including 4.1, 5.0, 5.1, 5.4, 5.5, and 5.6.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Added support for `sha256_password` authentication with RSA encryption.

Bugs Fixed

- The option name `gatherPerfMetrics` in the `fullDebug.properties` file was misspelled as `gatherPerMetrics`. That resulted in the performance metrics not being logged. The name has been corrected in both the file and the Connector/J manual. (Bug #18598665, Bug #72326)
- The MySQL Fabric driver did not properly propagate exceptions thrown during connection creation using JDBC 4. (Bug #18549472, Bug #72301)
- In the method `MysqlIO.unpackBinaryResultSetRow`, an array was created unnecessarily to hold the null bit mask. With this fix, Connector/J avoids the creation of the array by making use of the data that is still variable in the original buffer object. (Bug #18403456, Bug #72023)
- In the method `PreparedStatement.ParseInfo`, a character array was created unnecessarily out of a string object. This fix removed the unnecessary instantiation of the array. (Bug #18403199, Bug #72006)
- When trying to get an integer from a column value that was not a number, instead of a `NumberFormatException` , an `ArrayIndexOutOfBoundsException` was thrown instead due to an off-by-one error. This fix corrects the stop condition for trimming spaces in the beginning of byte buffers in the `getShort()`, `getInt()`, and `getLong()` methods in the `StringUtils` class. (Bug #18402873, Bug #72000)
- The `StringUtils.getBytes` methods have been refactored to avoid unnecessary creations of string objects for encoding character arrays into bytes. Also, unneeded `StringBuffers` are replaced by `StringBuiders`. (Bug #18389973, Bug #72008)
- When exception interceptors were used, the `init()` method was called twice for each exception interceptor, once by `Util.loadExtensions()` and again by `ConnectionImpl.ExceptionInterceptorChain.init()`. This fix eliminates the calls by `ExceptionInterceptorChain.init()`. (Bug #18318197, Bug #71850)
- After a non-SSL socket had been transformed into an SSL socket, `Connection` was still keeping its reference to the wrapped, non-SSL socket, failing to recognize that the type of connection had been changed. This fix creates a new `StandardSSLSocketFactory` class, which implements `SocketFactory` and wraps the initial `SocketFactory.afterHandshake()` method. `MysqlIO.socketFactory` was replaced after the socket transformation, so that `afterHandshake()` is performed on the old factory but returns the new socket. (Bug #18260918, Bug #71753)
- When `rewriteBatchedStatements=true` and `useAffectedRows=true` were set, `executeBatch()` returned the number of affected rows as given by the server, which was not a reliable value unless it was a “0”. That behavior was due to the fix for Bug#68562, which was implemented in Connector/J 5.1.27 (see the changelog for the release), and has been breaking a number of applications that use Connector/J. In this new fix, Connector/J returns `Statement.SUCCESS_NO_INFO` as a result for each statement in the batch as long as there are more than one batched statement and the returned value for the affected rows from the server is greater than “0”. Connector/J returns a “0” otherwise, which indicates that no rows have been affected. (Bug #18009254, Bug #61213)
- If `profileSQL` was enabled, a memory leak would occur after a connection was lost and continuous attempts were made to reconnect. It was because `ProfilerEventHandlerFactory` kept a map in which dead connections kept on accumulating. This fix refactors the `ProfilerEventHandlerFactory` and eliminates that map. (Bug #16737192, Bug #55680)

- When `useCursorFetch` was set to `true`, Connector/J would attempt to send XA commands as server prepared statements, which were unsupported, and the commands would have to be resent as plain statements. This fix stops Connector/J from sending XA commands as server prepared statements. (Bug #16708231, Bug #67803)
- When closing a server-prepared statement twice and `cachePrepStmts=true`, the second call closed the statement as expected. However, if a call of `Connection.prepareStatement()` was made again with exactly same SQL string, Connector/J obtained the closed statement from the cache and failed by throwing an exception. With this fix, `ServerPreparedStatement.close()` detects and ignores subsequent `close()` calls on cached statements that have already been closed. (Bug #16004987, Bug #66947)

Changes in MySQL Connector/J 5.1.30 (2014-03-31)

Version 5.1.30 is a maintenance release of the production 5.1 branch. It is suitable for use with many MySQL Server versions, including 4.1, 5.0, 5.1, 5.4, 5.5, and 5.6.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Connector/J now supports MySQL Fabric. See Using Connector/J with MySQL Fabric for details.

Bugs Fixed

- Replaced a for loop with the `System.arraycopy()` method for copying array values in the `MysqlIO.unpackNativeEncodedColumn()` and `LoadBalancingConnectionProxy.addHost()` methods, in order to improve the two methods' performance. (Bug #18327245, Bug #71861)
- Avoided the use of an iterator over the list of statement interceptors in the methods `MysqlIO.invokeStatementInterceptorsPost()` and `MysqlIO.invokeStatementInterceptorsPre()`, so that Connector/J does not increase the stack size unnecessarily. (Bug #18236388, Bug #71679)
- The `Field.getStringFromBytes()` method created a useless byte array when using JVM's converter and the encoding defined by the connection. This fix makes the method call `StringUtils.toString()` using the original buffer instead of creating a temporary byte array for the call. (Bug #18228668, Bug #71623)
- Improved on the code for integer-to-hex conversion when building XA commands by avoiding the creation of temporary character arrays, thus enhancing performance. (Bug #18228302, Bug #71621)
- It was intended that if a previous query on a connection had used the `setMaxRows()` method, in the next query, Connector/J would not cancel that by setting `SQL_SELECT_LIMIT=DEFAULT` if the query contained a `LIMIT` clause. However, in the actual implementation, the maximum row setting was reused in the subsequent query in various situations beyond expectation (for example, when a table name contains the string “limit” in it). This fix removes the `LIMIT`-clause parsing and replaces it by a better way of controlling the maximum rows per session. (Bug #18110320, Bug #71396)
- There were sporadic cases of the key store file being open hundreds of times and causing some "Too many files open" errors. This fix makes sure that in com.mysql.jdbc.ExportControlled and in `MysqlIO.sendFileToServer()`, the input stream for the key store file is explicitly closed after use. (Bug #18107621, Bug #71432)
- When working with MySQL 5.6, calling `PreparedStatement.setTimestamp()` resulted in a `java.lang.StringIndexOutOfBoundsException` being thrown if the `Timestamp` contained a fractional second. This fix corrects the digit truncation performed in the `formatNanos()` method, which was the cause of the problem. (Bug #18091639)

- Calling `ResultSet.close()` on an already closed ResultSet caused an SQLException. While the exception was silently discarded, it did result in performance issues. This fix makes Connector/J comply with the Java specification that when a ResultSet object is already closed, application of the `close` method on it should be a no-op. (Bug #16722637, Bug #67318)

- Fixed the problem of the wrong source being provided when the build property `com.mysql.jdbc.noCryptoBuild` was set.

Changes in MySQL Connector/J 5.1.29 (2014-02-03)

Version 5.1.29 is a maintenance release of the production 5.1 branch. It is suitable for use with many MySQL Server versions, including 4.1, 5.0, 5.1, 5.4, 5.5, and 5.6.

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- Added a new connection property `detectCustomCollations=[true|false]`. When its value is `false` (which is the default value), the `ConnectionImpl.buildCollationMapping()` method, for detecting non-standard character sets or collations, will NOT be called during the instantiation of a connection, thus shortening the time for establishing a connection.

  The introduction of this new property with the default value of `false` alters the old default behaviour of calling `ConnectionImpl.buildCollationMapping()` at each connection. If non-standard character sets or collations might be used, users should set `detectCustomCollations=true`. (Bug #17874902, Bug #71038)

Bugs Fixed

- `DatabaseMetaData.getKeywords()` did not return the latest reserved words as found in MySQL Server's documentation (for example, in Keywords and Reserved Words). This fix makes the function generate, as per the JDBC API specification, a list of the latest MySQL and SQL standard keywords that are not also SQL92 (or, depending on the JDBC version in use, SQL2003) keywords. (Bug #17647584, Bug #70701)

Changes in MySQL Connector/J 5.1.28 (2013-12-23)

Version 5.1.28 is a maintenance release of the production 5.1 branch. It is suitable for use with many MySQL Server versions, including 4.1, 5.0, 5.1, 5.4, 5.5 and 5.6.

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- Added support for live management for replication hosts. This enables users to change replication topographies (for example, adding or removing a host, or promoting a salve to master) without having to restart an application. See Live Reconfiguration of Replication Topography for details. (Bug #70842)

Bugs Fixed

- A shadow declaration of `com.mysql.jdbc.OperationNotSupportedException` is removed from `com.mysql.jdbc.RowDataDynamic`. (Bug #17833137, Bug #70969)

- An incorrect `SQLException` subclass was thrown during a query interruption. This fix creates a new, JDBC4-specific version of `MySQLQueryInterruptedException`, which subclasses `MySQLNonTransientException`. (Bug #17750877, Bug #70835)

- A validation is added to check if `blobSendChunkSize` is negative (which is the case when `maxAllowedPacket` is set with a value less than or equal to “8203”) when `useServerPrepStmts=true`. It throws an exception when the validation fails. (Bug #17184082, Bug #69777)
- When the connection property `dontTrackOpenResources=true` was used, the result set was closed after a `Statement.close()` was issued. (Bug #17164058, Bug #69746)
- The timeout limit set by `DriverManager.setLoginTimeout()` was not honored during a handshake attempt. This fix adds the `DriverManager.setLoginTimeout()` control to the function `NonRegisteringDriver.connect()`: if the defined timeout is reached, the connection creation is cancelled and an exception is thrown. (Bug #17015317, Bug #69579)
- The method `Statement.closeOnCompletion()` was not working. (Bug #16691047, Bug #68916)
- With `cacheResultSetMetadata=true`, `cacheCallableStmts=true`, and `cachePrepStmts=true`, if a stored procedure that returns a result set was called, calling the stored procedure a second time resulted in a null pointer exception being thrown for `initializeResultsMetadataFromCache`. (Bug #11762713, Bug #55340)
- Connector/J's SQL escape sequence processor was confused by multiple backslashes. The `EscapeTokenizer` has been re-factored to process multiple backslashes properly. (Bug #11759035, Bug #51313)

Changes in MySQL Connector/J 5.1.27 (2013-11-04)

Version 5.1.27 is a maintenance release of the MySQL Connector/J production 5.1 branch. It is suitable for use with many MySQL Server versions, including 4.1, 5.0, 5.1, 5.4, 5.5 and 5.6.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Added support for multi-master replication topographies, which can now be specified using the following host definition syntax: `address=(host=hostname)(port=3306)(type=[master|slave])`.
- Added the flag `CLIENT_PLUGIN_AUTH_LENENC_CLIENT_DATA` and support for authentication data up to a length of 2^{64}-1 bytes.
- Added support for `EXPLAIN` for `UPDATE` and `DELETE`. Extended slow query warning with query execution plan for `INSERT`, `REPLACE`, `UPDATE`, and `DELETE`.

Bugs Fixed

- When attempting to connect to a server using long, multibyte user name and database name, an Array Index Out Of Bounds Exception was thrown. This fix prevents the problem by increasing the space reserved for those strings. (Bug #17251955)
- When setting a connection property with a value for `blobSendChuckSize`, `largeRowSizeThreshold`, or `locatorFetchBufferSize`, the suffix “G” was not recognized and caused an error to be thrown. With this fix, Connector/J now supports using “G” or “GB” for gigabyte for the purpose, as well as “KB” for kilobyte and “MB” for megabyte. (Bug #17015673, Bug #69452)
- When there was an attempt to create a second branch with a duplicate ID on the same MySQL server instance, the `XAException.errorCode` returned by Connector/J was not equal to `XAException.XAER_DUPID`. (Bug #16979862, Bug #69506)
- The `SYSTEM TABLE` type was handled inconsistently in different methods: `DatabaseMetaData.getTableTypes()` did not return system tables, and `DatabaseMetaData.getTables()` did not return all system tables. With this fix, the table types

`SYSTEM TABLE` and `SYSTEM VIEW` are returned by `getTableTypes()`, and `getTables()` now returns tables from the internal schemas `mysql` and `performance_schema` as instances of `SYSTEM TABLE` and tables from `information_schema` as instances of `SYSTEM VIEW`. (Bug #16879239, Bug #69290)

- JDBC authentication failed when there was a null byte in the scramble, because `com.mysql.jdbc.MysqlIO.doHandshake` only read up to the first null byte when reading in the first eight bytes of the scramble. (Bug #16723048, Bug #62469)
- When `rewriteBatchedStatements=true` and `useAffectedRows=true` were set, `executeBatch()` did not return the number of affected rows as expected, but always returned a "1," even if no rows were affected (for example, when an `ON DUPLICATE KEY UPDATE` clause was applicable but the data needed no changes). The "1" was actually a hardwired value, a workaround for a known issue on the server side, which does not return the number of affected rows in such cases. This fix makes Connector/J just return the value given by the server anyway, which means the returned number of affected rows is reliable now if Connector/J returns a "0" [no rows affected]; non-zero values returned should not be trusted though. (Bug #16442951, Bug #68562)
- In a replication-aware deployment, when the master was not reachable, new connections could not be made, or the replication driver just hanged. As part of the new multiple-master support feature, users can now set the property `allowMasterDownConnections=true` to allow connections to be established even when no master hosts are available. (Bug #11757979, Bug #50105, Bug #16443992, Bug #63354)
- Calling ResultSet.absolute(0) caused Connector/J to throw an error. This fix makes ResultSet.absolute(0) behave like ResultSet.beforeFirst(), as required by the JDBC specifications. (Bug #11749136, Bug #38252)
- After setting the connection property `yearIsDateType=false`, the method `ResultSet.getMetaData().getColumnClassName()` still returned the Java type of a `YEAR` column as as `java.sql.Date` although internally it was being treated as `java.sql.Short`. This fix corrects the metadata for a `YEAR` column under this situation, so that the correct Java type of `java.sql.Short` is returned. Please note that the method `ResultSet.getMetaData().getColumnType()`, however, returns `java.sql.Types.DATE` irrespective of the `yearIsDateType` setting. (Bug #11748097, Bug #35115)
- The string returned by `DatabaseMetaData.getDriverVersion()` contains the unexpanded expression "${svn.Revision}" or "${bzr.revision-id}" instead of the actual revision number of Connector/J. (Bug #50538, Bug #11758345)

Changes in MySQL Connector/J 5.1.26 (2013-08-01)

Functionality Added or Changed

- Added a `toString()` method to the `PreparedStatementWrapper` class to help debug prepared statements. (Bug #42267, Bug #11751418)

Bugs Fixed

- An out of memory error occured when the compression protocol was enabled with the connection option `useCompression=true`. This fix stops the memory leak that caused the error by making sure that `CompressedInputStream` releases its reference to the connection object when the input stream closes. (Bug #68400, Bug #16478043)
- The results returned by the method `DatabaseMetaData.getIndexInfo()` were not sorted in the order described in the JDBC specification (NON_UNIQUE, TYPE, INDEX_NAME, and ORDINAL_POSITION). (Bug #68098, Bug #16224299)

- `DatabaseMetaData.getColumns()` threw an `MySQLSyntaxErrorException` if the schema contains tables with ANSI quoted names with leading and trailing back quotes (`). When those names were passed as parameters in unquoted form, Connector/J treated them as quoted because of the back quotes, and thus the error. This fix adds the behavior that when the connection property `pedantic` was set to `true`, methods like `DatabaseMetaData.getColumns()` treat all parameters as unquoted. (Bug #65871, Bug #14598704)

- Connector/J silently ignored calls to `ResultSet.updateRow` when the cursor was on the insert row. This fix ensures that an `SQLException` is thrown with those calls, as described in the JDBC specification. (Bug #45757, Bug #11754192)

Changes in MySQL Connector/J 5.1.25 (2013-05-01)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- When connecting to MySQL server 5.6 or later, Connector/J now sends to the server a number of connection attributes that will be exposed in the relevant `PERFORMANCE_SCHEMA` tables. By default, the following attributes are sent:

  - `client_version`: version number of MySQL Connector/J
  - `client_name`: “MySQL Connector Java”
  - `client_license`: “commercial”" or “GPL,” depending on the build used
  - `runtime_vendor`: JVM vendor name
  - `runtime_version`: JVM version

  The list of attributes can be augmented, and the sending of the attributes can be suppressed. See the description for `connectionAttributes` in Driver/Datasource Class Names, URL Syntax and Configuration Properties for Connector/J for details.

Bugs Fixed

- The “`mysql-connector-java-5.1.24.jar`” prepared for maven.org contained the artifact, META-INF/INDEX.LIST, which incorrectly pointed to “`mysql-connector-java-5.1.24-bin.jar`”. For Connector/J 5.1.25, `META-INF/INDEX.LIST` has been corrected and now points to “`mysql-connector-java-5.1.25.jar`”.

  The work around for “`mysql-connector-java-5.1.24.jar`” prepared for maven.org is to rename the JAR file to “`mysql-connector-java-5.1.24-bin.jar`” or to download the MySQL Connector/J 5.1.24 driver from http://dev.mysql.com/downloads/connector/j/. (Bug #16574419)

- The `getDriverName()` function would return “MySQL-AB JDBC Driver” as the driver name. This fix changes the driver name string to “MySQL Connector Java”. (Bug #16436511)

- With the connection option `rewriteBatchedStatements=true`, after a batch insert, the `batchStatement` was closed twice. This caused some unnecessary throwing and catching of errors, impacting code performance. This fix nullifies the `batchStatement` after the first calling of the `close()` method in order to avoid a second closing. (Bug #69308, Bug #16879267)

- The `DatabaseMetaData` methods of `getFunctions` and `getFunctionColumns` returned information for both stored functions and procedures when value of the connection option `useInformationSchema` was `false`. On the other hand, `getProcedures` and `getProcedureColumns` always return information for both stored procedures and functions. The behaviours of the four functions are inconsistent with the idea introduced since JDBC4 to separate the “get” functions for stored functions and procedures. The functions are modified as follows:

    - `getFunctions` and `getFunctionColumns` now only return information for stored functions (not for stored procedures), irrespective of the value of `useInformationSchema`.
    - For `getProcedures` and `getProcedureColumns`, in order to maintain backward compatibility with pre-JDBC4 implementations, a new connection option, `getProceduresReturnsFunctions`, is created, whose default value of `true` makes `getProcedures` and `getProcedureColumns` return information for both stored procedures and functions. Setting the option to `false` makes the functions return only information for stored procedures.

  (Bug #69298, Bug #16845965)

- `ReplicationConnection.isMasterConnection()`, which is intended to check if the current connection is the master connection, would always return “false”. This fix implements logic that returns “true” if the current connection is the master connection. (Bug #68763, Bug #16545334)
- When using Connector/J in a replication deployment, all of the slave connections would not be pinged. This fix ensures that all active physical connections to slaves are pinged, and that communication exceptions are thrown if:
    - The master connection ping fails, and the connection is currently set to use the master connection via `Connection.setReadOnly(false)`.
    - The Connection object is set to use the slave via `Connection.setReadOnly(true)`, and the currently-selected slave connection ping fails.

  (Bug #68733, Bug #16526938)
- Producing Connector/J JAR packages in Eclipse using Ant tasks would result in a JAR file without compiled class files. Compile-related tasks in the Ant script would direct compilation targets to the `bin` directory (for Eclipse compatibility), while JAR related tasks were directed to `{buildDir}/${fullProdName}`. This fix changes the JAR task reference to `${compiler.output}` instead of `{buildDir}/${fullProdName}`. (Bug #68664, Bug #16486957)
- Tomcat failed to stop abandoned connection cleanup threads. The fix for Bug#65909 introduced the ability to stop daemon threads started by the Connector/J driver but it also cleared references from daemon threads to the parent classloader. When the `clearReferencesStopThreads` property is set to “true” in `context.xml`, Tomcat analyzes classloaders to detect and stop lost threads. This fix ensures that abandoned connection cleanup threads retain a reference to the parent classloader. (Bug #68556, Bug #16443387)
- The `DatabaseMetaData.getProcedureColumns()` method returned wrong `COLUMN_TYPE` values. (Bug #68307, Bug #16707803)
- An exception occurred if `NULL` was passed to a stored procedure `INOUT` parameter. (Bug #60816, Bug #12800874)
- The result set returned by `getTables` did not contain all the columns as specified in the JDBC specification. Even though the missing columns are not meaningful to MySQL, the `getTables` method is modified to add null fields to the result set in order to make it compliant with the JDBC specification. Other methods including `getColumns`, `getFunctionColumns`, `getFunctions`, `getProcedureColumns`, and `getUDTs` are also modified to address the same issue pertaining to them. (Bug #44451, Bug #11753081)

Changes in MySQL Connector/J 5.1.24 (2013-03-12)

Bugs Fixed

- The `Statement.cancel()` was made less susceptible to deadlocking, to allow application servers to immediately cancel operations that fail due to transaction timeouts or network issues. The `abortInternal()` method that was previously part of the `MySQLConnection` interface is now

available through the `Connection` interface, so it can be tested for and called if available. (Bug #16224249)

- When a statement was accessing a table in streaming mode, terminating the statement by issuing `KILL QUERY` from another session could cause a `SQLException` "Query execution was interrupted" and a stall when the original session issued a `ResultSet.close()` or `Statement.close()` method call. This issue resulted from a race condition and so could happen intermittently. (Bug #16198772, Bug #64204)
- When the connection options `noDatetimeStringSync` and `useTimezone` were specified together, which is not allowed, the error message referred to an incorrect option name `noDatetimeSync`. (Bug #16061665, Bug #68011)
- The `connection.getQueryTimeoutKillsConnection()` method could be called with a null parameter, causing `NullPointerException` errors and stack traces. The error depended on a race condition and so occurred intermittently. (Bug #13943631, Bug #64805)

Changes in MySQL Connector/J 5.1.23 (2013-02-05)

Fixes bugs found since release 5.1.22. Also adds support and test cases for several new features from MySQL Server 5.6.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- New test cases cover new features from MySQL Server 5.6:
  - Explicit partition selection syntax.
  - `EXCHANGE PARTITION` syntax.
  - Transportable tablespaces syntax: `LOCK TABLES ... FOR EXPORT`.
  - `CREATE TABLE ... DATA DIRECTORY = '/absolute/path/to/directory/'` syntax for InnoDB tables.
  - New `ALTER TABLE` syntax: `ALGORITHM` and `LOCK` keywords.
- Added support of password expiration protocol. This introduces a new Boolean connection property `disconnectOnExpiredPasswords`:
  - If `disconnectOnExpiredPasswords=true` and the password has expired, the connection will be rejected by server with `ErrorCode == 1820` (`ER_MUST_CHANGE_PASSWORD`).
  - If `disconnectOnExpiredPasswords=false`, the connection enters "sandbox" mode, where all commands except `SET PASSWORD = ...` and `SET PASSWORD FOR CURRENT_USER() = ...` cause an error to be thrown. The user needs to reconnect though, after setting the password, because Connector/J requires additional information from the server that is not available in the "sandbox" mode.
- Static charset/collation maps were updated, particularly for the `ucs2_unicode_ci` and `utf8_unicode_ci` collations.
- `Connection.setReadOnly()` will take advantage of server-side support for read-only transactions present in MySQL 5.6 and newer. Calling `.isReadOnly()` will incur a round-trip if `useLocalSessionState` is not enabled.
- The driver now allows the mechanism for caching MySQL server configuration values replaceable at runtime, via the `serverConfigCacheFactory` property. The default is an implementation

that is a per-VM concurrent map, keyed by URL. The driver will invalidate cache entries when `SQLException` exceptions that indicate communications errors are thrown (on the assumption that the server has been or is restarting). The driver also invalidates cache entries if the server version that is being connected to differs from the one that was present when the cached values were populated.

To replace the default implementation, implement `CacheAdapterFactory<String, Map<String, String>>`, and specify the fully-qualified class name of this implementation for the `serverConfigCacheFactory` connection option.

Bugs Fixed

- Stack trace used for point-of-origin in log and exception messages caused a Permgen leak when an application was redeployed, because the `WebappClassloader` could not be garbage collected. We no longer store the entire stack trace, only the calling class and method, and only when using the usage advisor or when profiling. (Bug #16097851, Bug #67954)
- With the connection setting `useCompression=true`, the SQL statement `LOAD DATA LOCAL INFILE` could cause a `java.net.SocketException` error due to a broken pipe. (Bug #15895369, Bug #11237)
- The `nativeSQL()` method would always truncate fractional seconds rather than preserving the fractional part in the output string. Now Connector/J checks the server version: it preserves the fractional part for MySQL 5.6.4 and greater, and truncates the fractional part for older versions. (Bug #14748459, Bug #60598)
- With the new MySQL server password hashing feature enabled, different results could be returned from `ResultSet` and `CachedRowSet`. The test suite was modified to not perform comparison of `PASSWORD()` results when the setting `old_passwords=2`, because with SHA-256 password hashing enabled, the function return results are nondeterministic. (Bug #14665141)
- The cleanup thread for abandoned connections in the `NonRegisteringDriver` class was refactored to have a static shutdown method. Memory was allocated but never released. If you encountered this leak problem, implement the context listener in your application with the `AbandonedConnectionCleanupThread.shutdown()` call in the `contextDestroyed` method. This issue was found in applications running under the Tomcat application server, but it might have also applied to other application servers.

  For example:

```
@WebListener
public class YourThreadsListener implements ServletContextListener {
   public void contextDestroyed(ServletContextEvent arg0) {
      try {
          AbandonedConnectionCleanupThread.shutdown();
      } catch (InterruptedException e) {
      }
   }
   ...
}
```

  Note that if container does not support annotations, you add the description to `web.xml`:

```
<listener>
   <listener-class>user.package.YourThreadsListener</listener-class>
</listener>
```

  (Bug #14570236, Bug #65909)

- With the connection parameter `rewriteBatchedStatements=true`, `ResultSetRow.getTimeFast` could give an incorrect value for `TIME` columns containing a fractional part. (Bug #14260352)
- If a `timestamp` value was passed through prepared statement parameters, fractional-second precision was stripped off, even if the underlying field (such as `VARCHAR(255)`) could store the full value. A workaround was to convert the timestamp value to a string when specifying the prepared statement argument, for example `prepped_stmt.setString(1,time_stamp.toString()`. This was partly fixed in 5.1.19, but that fix did not cover the case with the setting `useLegacyDatetimeCode=true`. (Bug #11750017, Bug #40279, Bug #60584)
- `executeQuery()` in `Statement.java` let `TRUNCATE TABLE` queries be executed, although that method is supposed to block any request that modifies the database. `TRUNCATE TABLE` and `RENAME TABLE` are now filtered for `executeQuery()`. (Bug #11748257, Bug #35653)
- Compiling Connector/J from source failed when using a non-Sun/Oracle JDK, because the Ant target `compile-testsuite` contained a `AppletRegressionTest` that used a Sun-specific class (`sun.applet.AppletSecurity`). This fix removes the `AppletRegressionTest` and thus the dependence on the Sun class from the build process. (Bug #11745319, Bug #13509)

Changes in MySQL Connector/J 5.1.22 (2012-09-11)

Fixes bugs found since release 5.1.21.

Bugs Fixed

- **Performance:** The `com.mysql.jdbc.getCharsetNameForIndex()` method was made more efficient, resulting in better performance for queries against tables containing many columns with string data types. (Bug #14236302, Bug #65508)
- The `LoadBalancingConnectionProxy.pickNewConnection()` method could incorrectly flag the current connection as invalid after testing another connection. The selection process could choose another (potentially offline) host, leading to a “connection is closed” error when trying to use a connection after re-balancing:

```
No operations allowed after connection closed.
Connection closed after inability to pick valid new connection
during fail-over.
```

  (Bug #14563127)
- With `profileSQL=true` and `useNanosForElapsedTime=true` specified in the connection URL, query and fetch duration were not reported correctly. The `com.mysql.jdbc.MysqlIO.sqlQueryDirect()` method always measured times in milliseconds rather than switching between milliseconds and nanoseconds. (Bug #14273046, Bug #57662)
- `ResultSet` objects created by the `getGeneratedKeys()` method were not being automatically closed, leading to potential memory leaks if the application did not explicitly close the `ResultSet` objects. (Bug #14192873, Bug #65503)
- Implemented the `getVersionColumns()` method, which formerly always returned an empty result set. Now this method returns the timestamp columns that are updated every time a row is changed. (Bug #13636546, Bug #63800)
- Connecting to a server that used a `UCS2` character set would throw an exception:

```
com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: You have an error
in your SQL syntax; check the manual that corresponds to your MySQL server
version for the right syntax to use near '??' at line 1
```

Now, Connector/J sets `characterEncoding` implicitly to `UTF-8` when the server character set is `UCS2`. Because the Connector/J UTF-8 implementation does not cover all UCS-2 characters, note that truncation might occur in some cases. (Bug #11751035, Bug #41752)

- Connector/J now avoids synchronization issues in terms of locking order and prepared statements.

Changes in MySQL Connector/J 5.1.21 (2012-07-03)

Fixes bugs found since release 5.1.20.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Connector/J applications can now connect to MySQL servers that use the PAM authentication system. See Connecting Using PAM Authentication for details about the Connector/J support, and PAM Pluggable Authentication for information about this authentication feature of the MySQL server. (Bug #13608088)

Bugs Fixed

- When a new instance of authentication plugin is obtained (either the default one, or one named in Auth Challenge Packet or in Auth Method Switch Request Packet), it must be checked if plugin's confidentiality requirements are satisfied. If the plugin requested in the Auth Method Switch Request Packet required a secure connection, an insecure connection could be established instead. (Bug #13980303)
- With a connection encoding of `gbk` and a server encoding of `latin1`, a call such as `PreparedStatement.setString(1, "0f0f0702")` could throw an exception `java.lang.StringIndexOutOfBoundsException`. (Bug #13943893, Bug #64731)
- When Connector/J was connected to a MySQL 5.5 server, the error message for an invalid query could be returned in the wrong character set and appear garbled. (Bug #13702427, Bug #64205)
- By default, if an I/O error occurred while executing a query a deadlock could occur as Connector/J attempted to close all open statements (by calling method `ConnectionImpl#closeAllOpenStatements`) The deadlock would occur if any `execute*(*)` method of another instance of the `StatementImpl` class was called by another thread. A workaround was to set the property `dontTrackOpenResources` to `true`. (Bug #12590053, Bug #61247)
- Specifying a non-existent character set in the `characterEncoding` or `characterSetResults` connection options could produce a `NullPointerException` error. The error could occur with a typographical mistake, such as using a hyphen instead of an underscore in the name. (Bug #11749010, Bug #37931)
- The timezone entry was missing for “MEST - Middle European Summer Time”. (Bug #11748548, Bug #36662)

Changes in MySQL Connector/J 5.1.20 (2012-05-01)

Fixes bugs found since release 5.1.19.

Bugs Fixed

- **Important Change:** This fix corrects an issue introduced in Connector/J 5.1.19 that caused connection errors with MySQL 4.1 and earlier servers. A `java.lang.ClassCastException` exception occurred during connection initialization when `com.mysql.jdbc.ConnectionImpl.buildCollationMapping()` interpreted the output of the `SHOW COLLATION` statement. (Bug #13958793)

- Using Connector/J 5.1.19 in combination with JBoss could result in an error while establishing a connection: `MySQLNonTransientConnectionException: Bad handshake`. This issue occurred when using the old-style password hash, which requires the `mysql_old_password` plugin during handshake. A workaround was to replace the 16-byte hash with a 41-byte one, as explained in Password Hashing in MySQL. (Bug #13990612, Bug #64983)

- A `java.lang.StringIndexOutOfBoundsException` exception could occur when manipulating date/time values with fractional seconds. (Bug #13960556)

- A `MySQLSyntaxErrorException` could occur when calling certain methods while connected to a MySQL 5.6.5 or higher server. Affected methods included `StatementImpl.execute()` and `PreparedStatement.execute()`. The cause was the removal of the `SET OPTION` syntax in the MySQL Server. The methods were modified to use the newer `SET` syntax internally. (Bug #13955027)

- The savepoint identifier generated by the `java.sql.Connection#.setSavepoint()` function could be misinterpreted as a floating-point number, for example values such as `123e10` or `123e10foo`. Such values could cause replication errors on slave servers because the values are not quoted in the binary log. The fix ensures that the savepoint identifiers do not begin with digits. (Bug #11763271, Bug #55962)

- If the string `limit` was used in a column name, prepared statements incorrectly treated the statement as if it used a `LIMIT` clause. For example, a prepared statement with `maxrows` set to 0 could incorrectly reuse the value from a previous call to `setMaxRows()`. This issue applied to both quoted and unquoted column names, and server-side and client-side prepared statements. (Bug #11748492, Bug #36478)

Changes in MySQL Connector/J 5.1.19 (April 2012)

Fixes bugs found since release 5.1.18.

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- For a UTF-8 table using a collation other than the default (`utf8_general_ci`), the precision of the `ResultSetMetaData` could be different from the precision specified in the `CREATE TABLE` statement. The fix corrects the return value from `getMaxBytesPerChar()`.

  This fix changes the behavior of some connection string parameters. `useDynamicCharsetInfo` no longer has any effect. With the setting `cacheServerConfiguration=true`, the cached settings also include the results of the `SHOW CHARACTER SET` statement. (Bug #13495590, Bug #63456)

- Added support for pluggable authentication. via the `com.mysql.jdbc.AuthenticationPlugin` interface (which extends the standard “extension” interface). Examples are in `com/mysql/jdbc/authentication` and in `testsuite.regression.ConnectionRegressionTest.` This feature introduces three new connection properties:

  - `authenticationPlugins` defines a comma-delimited list of classes that implement `com.mysql.jdbc.AuthenticationPlugin` and are used for authentication unless disabled by the `disabledAuthenticationPlugins` property.

  - `disabledAuthenticationPlugins` defines a comma-delimited list of classes implementing `com.mysql.jdbc.AuthenticationPlugin` or mechanisms, i.e. `mysql_native_password`. The authentication plugins or mechanisms cannot be used for authentication. Authentication will fail if it requires one of these classes. It is an error to disable the default authentication plugin, either the one named by `defaultAuthenticationPlugin` property or the hardcoded one if `defaultAuthenticationPlugin` property is not set.

- `defaultAuthenticationPlugin` defines the name of a class implementing `com.mysql.jdbc.AuthenticationPlugin`, which is used as the default authentication plugin. It is an error to use a class that is not listed in `authenticationPlugins` and is not one of the built-in plugins. It is an error to set as default a plugin that is disabled by being listed in the `disabledAuthenticationPlugins` property. It is an error to set this value to null or the empty string; there must be at least one valid default authentication plugin specified for the connection, meeting all the constraints listed above.

Bugs Fixed

- **Performance:** An unnecessary call to `bind()` during socket operations could limit scalability on some platforms. (Bug #13875070, Bug #63811)
- `setMaxRows` was not correctly processed during metadata collection for client-side prepared statements, causing the entire result set to be fetched and possibly leading to an out-of-memory error. (Bug #13839395, Bug #64621)
- A problem with processing escape sequences (in `com.mysql.jdbc.EscapeProcessor#escapeSQL`) could cause certain statements to fail. For example, a `CREATE TABLE` statement with a clause such as `CONSTRAINT `fk_`` was not parsed correctly. (Bug #13612316, Bug #63526)
- The `sjis` character set was incorrectly mapped to `MS392`, which was in turn mapped to `cp932`, resulting in garbled characters in some cases. In Connector/J 5.1.19, the mappings of character sets and collations were refactored to avoid such multi-step mappings. (Bug #13589875)
- Underprivileged execution of stored procedures fixed. (Bug #13508993, Bug #61203)
- A combination of failover connections, proxied or prepared statements, and database connection pool could cause a memory leak due to improper implementation of `equals()`. (Bug #13441718, Bug #63284)
- `com.mysql.jdbc.ResultSetRow.getTimestampFast()` did not account for all valid `TIME` lengths. (Bug #60582, Bug #16592635)
- The `Connection.changeUser()` method did not check for closed connections, leading to `NullPointerException` errors when this method was called on a closed connection.
- Reduced the memory overhead for server-side prepared statements. Each prepared statement allocated a 4K buffer for converting streams. Now this allocation is skipped when no `set*Stream()` methods have been used.

Changes in MySQL Connector/J 5.1.18 (2011-10-04)

Fixes bugs found since release 5.1.17.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Added the function `MYSQL_INDEX_TO_MYSQL_CHARSET` to retrieve the server charset name, using an index instead of parsing variables to `CharsetMapping.java`

Bugs Fixed

- The `LRUCache` implementation removed the eldest entry, rather than the least-recently accessed. (Bug #13036537)
- Changed `cacheCallableStatements` to `cacheCallableStmts` in `maxPerformance.properties`, to allow proper caching. (Bug #13036309)

- Added a new `ant` flag, `com.mysql.jdbc.junit.fork`, which controls whether `JUnit` will fork new processes for testing. Valid values for the flag are:
  - `on`: fork new process; the default and legacy behavior.
  - `off`: do not fork new process; required for Windows, or `process fork failure` errors will result while running the test suite via `ant` on Windows.

  (Bug #12784170)
- Not putting a space between `VALUES()` and `ON DUPLICATE KEY UPDATE` causes connector/J to both (A) Rewrite the query, although it includes an `ON UPDATE` statement and (B) To generate the wrong query with multiple `ON DUPLICATE KEY` statements. (Bug #12565726)
- The `loadBalanceBlacklistTimeout` option was not functioning properly. Working connections were not being removed from the blacklist. (Bug #63135)
- The "old" warnings were returned when `Statement.getWarnings()` was called after `Statement.clearWarnings()`. (Bug #61866, Bug #12791594)
- Connector/J now guards against the condition where a call to `KILL QUERY` will kill the next query issued by the server, if no query is in process. (Bug #61501)
- Calling `Statement.cancel()` on a statement that isn't currently executing, will cause a later-executed query on the same connection to be unexpectedly canceled. The driver now guards against this condition, but it is an underlying server issue. The MySQL statement `KILL QUERY` (which is what the driver uses to implement `Statement.cancel()`) is rather non-deterministic, and thus the use of `Statement.cancel()` should be avoided if possible. (Bug #61501)
- A connection could not be established when the URL contained both `sessionVariables` and `characterEncoding`. (Bug #61201, Bug #12649557)
- Reverting changes made to `ConnectionImpl.java`, the `private boolean characterSetNamesMatches` function.

Changes in MySQL Connector/J 5.1.17 (2011-07-07)

Fixes bugs found since release 5.1.16.

Bugs Fixed

- `LIKE` was not optimized in then server when run against `INFORMATION_SCHEMA` tables and no wildcards were used. Databases/tables with `'_'` or `'%'` in their names (escaped or not) are handled by this code path, although slower, since it is rare to find these characters in table names in SQL. If there is a `'_'` or `'%'` in the string, `LIKE` takes care of that; otherwise, `'='` is now used instead. The only exception is the `information_schema` database, which is handled separately. The patch covers both `getTables()` and `getColumns()`. (Bug #61332)
- The first call to a stored procedure failed with “No Database Selected”. The workaround introduced in `DatabaseMetaData.getCallStmtParameterTypes` to fix the server bug where `SHOW CREATE PROCEDURE` was not respecting lowercase table names was misbehaving when the connection was not attached to a database and on case-insensitive operating systems. (Bug #61150)
- There was a concurrency bottleneck in Java's character set encoding/decoding when converting bytes to/from `String` values.

  

  Important

  No longer use `String.getBytes(...)`, or `new String(byte[]...)`. Use the `StringUtils` method instead.

  (Bug #61105)

- Connector/J now avoids a concurrent bottleneck in Java's character set encoding/decoding when converting bytes to and from instances of `String`. (Bug #61105, Bug #12622056)

Changes in MySQL Connector/J 5.1.16 (Not released)

Fixes bugs found since release 5.1.15.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- The `Connection.isServerLocal()` method can now determine if a connection is against a server on the same host.

Bugs Fixed

- When auto-reconnect was used with `cacheServerConfiguration`, errors could occur when the host changed (in an HA setup, for example). (Bug #12325877)
- `DBMD.getTables` and `getColumns` fail with table names that contain a dot (like "`junk_[Sp:e,c/C-h+a=.r]`"). The workaround is to use `useInformationSchema=True`. (Bug #11782297)
- `ResultSetRow.getTimestampFast` and `getTime` had invalid offsets. (Bug #60313)
- Fixed a timestamp offset bug in `com.mysql.jdbc.ResultSetRow.getTimestampFast()`. (Bug #60313, Bug #11890729)
- `wasNull` was not set for a `DATE` field with the value of `0000-00-00` in `getDate()`, although `zeroDateTimeBehavior` is defined as `convertToNull`. (Bug #57808)
- A bypassing of the server protocol bug, where the DB should be null-terminated whether it exists or not. This affects `COM_CHANGE_USER`. (Bug #54425)
- Fixed a bug where Connector/J would kill the matching `ConnectionID`, but on the wrong server. (Bug #54135)

Changes in MySQL Connector/J 5.1.15 (2011-02-09)

Fixes bugs found since release 5.1.14.

Bugs Fixed

- Optional logging class `com.mysql.jdbc.log.Log4JLogger` was not included in the source/binary package for 5.1.14.

  5.1.15 will ship with an SLF4J logger (which can then be plugged into Log4J). Unfortunately, it is not possible to ship a direct Log4J integration because the GPL is not compatible with Log4J's license. (Bug #59511, Bug #11766408)
- The hard-coded list of reserved words in Connector/J was not updated to reflect the list of reserved words in MySQL Server 5.5. (Bug #59224)
- MySqlProcedure accepted null arguments as parameters, however the JDBC meta data did not indicate that. (Bug #38367, Bug #11749186)
- Using Connector/J to connect from a z/OS machine to a MySQL Server failed when the database name to connect to was included in the connection URL. This was because the name was sent in z/OS default platform encoding, but the MySQL Server expected Latin1.

  It should be noted that when connecting from systems that do not use Latin1 as the default platform encoding, the following connection string options can be useful:

`passwordCharacterEncoding=ASCII` and `characterEncoding=ASCII`. (Bug #18086, Bug #11745647)

Changes in MySQL Connector/J 5.1.14 (2010-12-06)

Fixes bugs found since release 5.1.13.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Connector/J's dependence on the `org.apache.commons.logging` package has been removed. (Bug #11756133, Bug #48013)
- Connector/J's load-balancing functionality only allowed the following events to trigger failover:
  - Transaction commit/rollback
  - CommunicationExceptions
  - Matches to user-defined Exceptions using the loadBalanceSQLStateFailover, loadBalanceSQLExceptionSubclassFailover or loadBalanceExceptionChecker property.

  This meant that connections where auto-commit was enabled were not balanced, except for Exceptions, and this was problematic in the case of distribution of read-only work across slaves in a replication deployment.

  The ability to load-balance while auto-commit is enabled has now been added to Connector/J. This introduces two new properties:

  1. loadBalanceAutoCommitStatementThreshold - defines the number of matching statements which will trigger the driver to (potentially) swap physical server connections. The default value (0) retains the previously established behavior that connections with auto-commit enabled are never balanced.
  2. loadBalanceAutoCommitStatementRegex - the regular expression against which statements must match. The default value (blank) matches all statements.

  Load-balancing will be done after the statement is executed, before control is returned to the application. If rebalancing fails, the driver will silently swallow the resulting Exception (as the statement itself completed successfully). (Bug #55723)

Bugs Fixed

- Connection failover left slave/secondary in read-only mode. Failover attempts between two read-write masters did not properly set `this.currentConn.setReadOnly(false)`. (Bug #58706)
- Connector/J mapped both 3-byte and 4-byte UTF8 encodings to the same Java UTF8 encoding.

  To use 3-byte UTF8 with Connector/J set `characterEncoding=utf8` and set `useUnicode=true` in the connection string.

  To use 4-byte UTF8 with Connector/J configure the MySQL server with `character_set_server=utf8mb4`. Connector/J will then use that setting as long as `characterEncoding` has not been set in the connection string. This is equivalent to autodetection of the character set. (Bug #58232)
- The `CallableStatementRegression` test suite failed with a Null Pointer Exception because the `OUT` parameter in the `I__S.PARAMETERS` table had no name, that is `COLUMN_NAME` had the value `NULL`. (Bug #58232)

- `DatabaseMetaData.supportsMultipleResultSets()` was hard-coded to return `false`, even though Connector/J supports multiple result sets. (Bug #57380)
- Using the `useOldUTF8Behavior` parameter failed to set the connection character set to `latin1` as required.

  In versions prior to 5.1.3, the handshake was done using `latin1`, and while there was logic in place to explicitly set the character set after the handshake was complete, this was bypassed when `useOldUTF8Behavior` was true. This was not a problem until 5.1.3, when the handshake was modified to use `utf8`, but the logic continued to allow the character set configured during that handshake process to be retained for later use. As a result, `useOldUTF8Behavior` effectively failed. (Bug #57262)
- Invoking a stored procedure containing output parameters by its full name, where the procedure was located in another database, generated the following exception:

  ```
  Parameter index of 1 is out of range (1, 0)
  ```

  (Bug #57022)
- When a JDBC client disconnected from a remote server using `Connection.close()`, the TCP connection remained in the `TIME_WAIT` state on the server side, rather than on the client side. (Bug #56979)
- Leaving Trust/ClientCertStoreType properties unset caused an exception to be thrown when connecting with `useSSL=true`, as no default was used. (Bug #56955)
- When load-balanced connections swap servers, certain session state was copied from the previously active connection to the newly selected connection. State synchronized included:
  - Auto-commit state
  - Transaction isolation state
  - Current schema/catalog

  However, the read-only state was not synchronized, which caused problems if a write was attempted on a read-only connection. (Bug #56706)
- When using Connector/J configured for failover (jdbc:mysql://host1,host2,... URLs), the non-primary servers re-balanced when the transactions on the master were committed or rolled-back. (Bug #56429)
- An unhandled Null Pointer Exception (NPE) was generated in `DatabaseMetaData.java` when calling an incorrectly cased function name where no permission to access `mysql.proc` was available.

  In addition to catching potential NPEs, a guard against calling JDBC functions with `db_name.proc_name` notation was also added. (Bug #56305)
- Added diagnostic information to `SQLException` messages that are caused by usage of a closed load-balanced connection, to help clarify the root cause of a connection closure. (Bug #56200)
- Attempting to use JDBC4 functions on `Connection` objects resulted in errors being generated:

  ```
  Exception in thread "main" java.lang.AbstractMethodError:
  com.mysql.jdbc.LoadBalancedMySQLConnection.createBlob()Ljava/sql/Blob;
     at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
     at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
     at
  sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
     at java.lang.reflect.Method.invoke(Method.java:597)
  ```

```
   at
com.mysql.jdbc.LoadBalancingConnectionProxy.invoke(LoadBalancingConnectionProxy.java:476)
   at $Proxy0.createBlob(Unknown Source)
```

(Bug #56099)

- Connector/J could exit with a “parameter index out of range” error when executing stored procedures. (Bug #55328, Bug #11762701)

Changes in MySQL Connector/J 5.1.13 (2010-06-24)

Fixes bugs found since release 5.1.12.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Connector/J did not support `utf8mb4` for servers 5.5.2 and newer.

  Connector/J now auto-detects servers configured with `character_set_server=utf8mb4` or treats the Java encoding `utf-8` passed using `characterEncoding=...` as `utf8mb4` in the `SET NAMES=` calls it makes when establishing the connection. (Bug #54175)

Bugs Fixed

- The method `unSafeStatementInterceptors()` contained an erroneous line of code, which resulted in the interceptor being called, but the result being thrown away. (Bug #53041)
- There was a performance regression of roughly 25% between r906 and r907, which appeared to be caused by pushing the Proxy down to the I/O layer. (Bug #52534)
- Logic in implementations of `LoadBalancingConnectionProxy` and `LoadBalanceStrategy` behaved differently as to which `SQLException`s trigger failover to a new host. The former looked at the first two characters of the SQLState:

  ```
  if (sqlState.startsWith("08"))
  ...
  ```

  The latter used a different test:

  ```
  if (sqlEx instanceof CommunicationsException
    || "08S01".equals(sqlEx.getSQLState())) {
  ...
  ```

  This meant it was possible for a new `Connection` object to throw an `Exception` when the first selected host was unavailable. This happened because `MySqlIO.createNewIO()` could throw an `SQLException` with a `SQLState` of “08001”, which did not trigger the “try another host” logic in the `LoadBalanceStrategy` implementations, so an `Exception` was thrown after having only attempted connecting to a single host. (Bug #52231)
- In the file `DatabaseMetadata.java`, the function `private void getCallStmtParameterTypes` failed if the parameter was defined over more than one line by using the '\n' character. (Bug #52167)
- The catalog parameter, `PARAM_CAT`, was not correctly processed when calling for metadata with `getMetaData()` on stored procedures. This was because `PARAM_CAT` was hardcoded in the code to `NULL`. In the case where `nullcatalogmeanscurrent` was `true`, which is its default value, a crash did not occur, but the metadata returned was for the stored procedures from the catalog currently attached to. If, however, `nullcatalogmeanscurrent` was set to `false` then a crash resulted.

Connector/J has been changed so that when `NULL` is passed as `PARAM_CAT` it will not crash when `nullcatalogmeanscurrent` is `false`, but rather iterate all catalogs in search of stored procedures. This means that `PARAM_CAT` is no longer hardcoded to `NULL` (see Bug #51904). (Bug #51912)

- A load balanced `Connection` object with multiple open underlying physical connections rebalanced on `commit()`, `rollback()`, or on a communication exception, without validating the existing connection. This caused a problem when there was no pinging of the physical connections, using queries starting with "/* ping */", to ensure they remained active. This meant that calls to `Connection.commit()` could throw a `SQLException`. This did not occur when the transaction was actually committed; it occurred when the new connection was chosen and the driver attempted to set the auto-commit or transaction isolation state on the newly chosen physical connection. (Bug #51783)
- The `rollback()` method could fail to rethrow a `SQLException` if the server became unavailable during a rollback. The errant code only rethrew when `ignoreNonTxTables` was true and the exception did not have the error code 1196, `SQLError.ER_WARNING_NOT_COMPLETE_ROLLBACK`. (Bug #51776)
- When the `allowMultiQueries` connection string option was set to `true`, a call to `Statement.executeBatch()` scanned the query for escape codes, even though `setEscapeProcessing(false)` had been called previously. (Bug #51704)
- When a `StatementInterceptor` was used and an alternate `ResultSet` was returned from `preProcess()`, the original statement was still executed. (Bug #51666)
- Objects created by `ConnectionImpl`, such as prepared statements, hold a reference to the `ConnectionImpl` that created them. However, when the load balancer picked a new connection, it did not update the reference contained in, for example, the `PreparedStatement`. This resulted in inserts and updates being directed to invalid connections, while commits were directed to the new connection. This resulted in silent data loss. (Bug #51643)
- `jdbc:mysql:loadbalance://` would connect to the same host, even though `loadBalanceStrategy` was set to a value of `random`, and multiple hosts were specified. (Bug #51266)
- An unexpected exception when trying to register `OUT` parameters in `CallableStatement`.

  Sometimes Connector/J was not able to register `OUT` parameters for `CallableStatements`. (Bug #43576)

Changes in MySQL Connector/J 5.1.12 (2010-02-18)

Fixes bugs found since release 5.1.11.

Bugs Fixed

- The catalog parameter was ignored in the `DatabaseMetaData.getProcedure()` method. It returned all procedures in all databases. (Bug #51022)
- A call to `DatabaseMetaData.getDriverVersion()` returned the revision as `mysql-connector-java-5.1.11 ( Revision: ${svn.Revision} )`. The variable `${svn.Revision}` was not replaced by the SVN revision number. (Bug #50288)

Changes in MySQL Connector/J 5.1.11 (2010-01-21)

Fixes bugs found since release 5.1.10.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Replication connections, those with URLs that start with jdbc:mysql:replication, now use a jdbc:mysql:loadbalance connection for the slave pool. This means that it is possible to set load balancing properties such as `loadBalanceBlacklistTimeout` and `loadBalanceStrategy` to choose a mechanism for balancing the load, and failover or fault tolerance strategy for the slave pool. (Bug #49537)

Bugs Fixed

- `NullPointerException` sometimes occurred in `invalidateCurrentConnection()` for load-balanced connections. (Bug #50288)
- The `deleteRow` method caused a full table scan, when using an updatable cursor and a multibyte character set. (Bug #49745)
- For pooled connections, Connector/J did not process the session variable `time_zone` when set using the URL, resulting in incorrect timestamp values being stored. (Bug #49700)
- The `ExceptionInterceptor` class did not provide a `Connection` context. (Bug #49607)
- Ping left closed connections in the liveConnections map, causing subsequent Exceptions when that connection was used. (Bug #48605)
- Using `MysqlConnectionPoolDataSource` with a load-balanced URL generated exceptions of type `ClassCastException`:

```
ClassCastException in MysqlConnectionPoolDataSource
Caused by: java.lang.ClassCastException: $Proxy0
   at
com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource.getPooledConnection(MysqlConne
ctionPoolDataSource.java:80)
```

```
java.lang.ClassCastException: $Proxy2
 at com.mysql.jdbc.jdbc2.optional.StatementWrapper.executeQuery(StatementWrapper.java:744)
```

  (Bug #48486)
- The implementation for load-balanced `Connection` used a proxy, which delegated method calls, including `equals()` and `hashCode()`, to underlying `Connection` objects. This meant that successive calls to `hashCode()` on the same object potentially returned different values, if the proxy state had changed such that it was utilizing a different underlying connection. (Bug #48442)
- The batch rewrite functionality attempted to identify the start of the `VALUES` list by looking for “VALUES ” (with trailing space). However, valid MySQL syntax permits `VALUES` to be followed by whitespace or an opening parenthesis:

```
INSERT INTO tbl VALUES
(1);

INSERT INTO tbl VALUES(1);
```

  Queries written with the above formats did not therefore gain the performance benefits of the batch rewrite. (Bug #48172)
- A PermGen memory leaked was caused by the Connector/J statement cancellation timer (`java.util.Timer`). When the application was unloaded the cancellation timer did not terminate, preventing the ClassLoader from being garbage collected. (Bug #36565)
- With the connection string option `noDatetimeStringSync` set to `true`, and server-side prepared statements enabled, the following exception was generated if an attempt was made to obtain, using `ResultSet.getString()`, a datetime value containing all zero components:

```
java.sql.SQLException: Value '0000-00-00' can not be represented as java.sql.Date
```

(Bug #32525)

Changes in MySQL Connector/J 5.1.10 (2009-09-23)

Fixes bugs found since release 5.1.9.

Bugs Fixed

- The `DriverManager.getConnection()` method ignored a non-standard port if it was specified in the JDBC connection string. Connector/J always used the standard port 3306 for connection creation. For example, if the string was `jdbc:mysql://localhost:6777`, Connector/J would attempt to connect to port 3306, rather than 6777. (Bug #47494)

Changes in MySQL Connector/J 5.1.9 (2009-09-21)

Bugs Fixed

- In the class `com.mysql.jdbc.jdbc2.optional.SuspendableXAConnection`, which is used when `pinGlobalTxToPhysicalConnection=true`, there is a static map (XIDS_TO_PHYSICAL_CONNECTIONS) that tracks the Xid with the XAConnection, however this map was not populated. The effect was that the `SuspendableXAConnection` was never pinned to the real XA connection. Instead it created new connections on calls to `start`, `end`, `resume`, and `prepare`. (Bug #46925)

- When using the ON DUPLICATE KEY UPDATE functionality together with the rewriteBatchedStatements option set to true, an exception was generated when trying to execute the prepared statement:

```
INSERT INTO config_table (modified,id_) VALUES (?,?) ON DUPLICATE KEY UPDATE modified=?
```

The exception generated was:

```
java.sql.SQLException: Parameter index out of range (3 > number of parameters, which is
2).
  at com.sag.etl.job.processors.JdbcInsertProcessor.flush(JdbcInsertProcessor.java:135)
......
Caused by: java.sql.SQLException: Parameter index out of range (3 > number of parameters,
which is 2).
  at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:1055)
  at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:956)
  at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:926)
  at com.mysql.jdbc.PreparedStatement.checkBounds(PreparedStatement.java:3657)
  at com.mysql.jdbc.PreparedStatement.setInternal(PreparedStatement.java:3641)
  at
com.mysql.jdbc.PreparedStatement.setBytesNoEscapeNoQuotes(PreparedStatement.java:3391)
  at
com.mysql.jdbc.PreparedStatement.setOneBatchedParameterSet(PreparedStatement.java:4203)
  at com.mysql.jdbc.PreparedStatement.executeBatchedInserts(PreparedStatement.java:1759)
  at com.mysql.jdbc.PreparedStatement.executeBatch(PreparedStatement.java:1441)
  at com.sag.etl.job.processors.JdbcInsertProcessor.flush(JdbcInsertProcessor.java:131)
  ... 16 more
```

(Bug #46788)

- When Connector/J encountered an error condition that caused it to create a `CommunicationsException`, it tried to build a friendly error message that helped diagnose what was wrong. However, if there had been no network packets received from the server, the error message contained the following incorrect text:

```
The last packet successfully received from the server was 1,249,932,468,916 milliseconds
ago.  The last packet sent successfully to the server was 0 milliseconds ago.
```

(Bug #46637)

- The `getSuperTypes` method returned a result set with incorrect names for the first two columns. The name of the first column in the result set was expected to be `TYPE_CAT` and that of the second column `TYPE_SCHEM`. The method however returned the names as `TABLE_CAT` and `TABLE_SCHEM` for first and second column respectively. (Bug #44508)
- SQLException for data truncation error gave the error code as 0 instead of 1265. (Bug #44324)
- Calling `ResultSet.deleteRow()` on a table with a primary key of type `BINARY(8)` silently failed to delete the row, but only in some repeatable cases. The generated `DELETE` statement generated corrupted part of the primary key data. Specifically, one of the bytes was changed from 0x90 to 0x9D, although the corruption appeared to be different depending on whether the application was run on Windows or Linux. (Bug #43759)
- Accessing result set columns by name after the result set had been closed resulted in a NullPointerException instead of a SQLException. (Bug #41484)
- `QueryTimeout` did not work for batch statements waiting on a locked table.

  When a batch statement was issued to the server and was forced to wait because of a locked table, Connector/J only terminated the first statement in the batch when the timeout was exceeded, leaving the rest hanging. (Bug #34555)
- The `parseURL` method in class `com.mysql.jdbc.Driver` did not work as expected. When given a URL such as “jdbc:mysql://www.mysql.com:12345/my_database” to parse, the property `PORT_PROPERTY_KEY` was found to be `null` and the `HOST_PROPERTY_KEY` property was found to be “www.mysql.com:12345”.

  

  Note

  Connector/J has been fixed so that it will now always fill in the `PORT` property (using 3306 if not specified), and the `HOST` property (using `localhost` if not specified) when `parseURL()` is called. The driver also parses a list of hosts into `HOST.n` and `PORT.n` properties as well as adding a property `NUM_HOSTS` for the number of hosts it has found. If a list of hosts is passed to the driver, `HOST` and `PORT` will be set to the values given by `HOST.1` and `PORT.1` respectively. This change has centralized and cleaned up a large section of code used to generate lists of hosts, both for load-balanced and fault tolerant connections and their tests.

  (Bug #32216)
- Attempting to delete rows using `ResultSet.deleteRow()` did not delete rows correctly. (Bug #27431)
- The `setDate` method silently ignored the Calendar parameter. The code was implemented as follows:

```
public void setDate(int parameterIndex, java.sql.Date x, Calendar cal) throws SQLException {
   setDate(parameterIndex, x);
}
```

  From reviewing the code it was apparent that the Calendar parameter `cal` was ignored. (Bug #23584)

Changes in MySQL Connector/J 5.1.8 (2009-07-16)

Bugs Fixed

- The reported milliseconds since the last server packets were received/sent was incorrect by a factor of 1000. For example, the following method call:

```
SQLError.createLinkFailureMessageBasedOnHeuristics(
(ConnectionImpl) this.conn,
System.currentTimeMillis() - 1000,
System.currentTimeMillis() - 2000,
e,
false);
```

  returned the following string:

```
The last packet successfully received from the server
was 2 milliseconds ago. The last packet sent successfully to the
server was 1 milliseconds ago.
```

  (Bug #45419)

- Calling `Connection.serverPreparedStatement()` variants that do not take result set type or concurrency arguments returned statements that produced result sets with incorrect defaults, namely `TYPE_SCROLL_SENSITIVE`. (Bug #45171)

- The result set returned by `getIndexInfo()` did not have the format defined in the JDBC API specifications. The fourth column, `DATA_TYPE`, of the result set should be of type `BOOLEAN`. Connector/J however returns `CHAR`. (Bug #44869)

- The result set returned by `getTypeInfo()` did not have the format defined in the JDBC API specifications. The second column, `DATA_TYPE`, of the result set should be of type `INTEGER`. Connector/J however returns `SMALLINT`. (Bug #44868)

- The `DEFERRABILITY` column in database metadata result sets was expected to be of type `SHORT`. However, Connector/J returned it as `INTEGER`.

  This affected the following methods: `getImportedKeys()`, `getExportedKeys()`, `getCrossReference()`. (Bug #44867)

- The result set returned by `getColumns()` did not have the format defined in the JDBC API specifications. The fifth column, `DATA_TYPE`, of the result set should be of type `INTEGER`. Connector/J however returns `SMALLINT`. (Bug #44865)

- The result set returned by `getVersionColumns()` did not have the format defined in the JDBC API specifications. The third column, `DATA_TYPE`, of the result set should be of type `INTEGER`. Connector/J however returns `SMALLINT`. (Bug #44863)

- The result set returned by `getBestRowIdentifier()` did not have the format defined in the JDBC API specifications. The third column, `DATA_TYPE`, of the result set should be of type `INTEGER`. Connector/J however returns `SMALLINT`. (Bug #44862)

- Connector/J contains logic to generate a message text specifically for streaming result sets when there are `CommunicationsException` exceptions generated. However, this code was never reached.

  In the `CommunicationsException` code:

```
private boolean streamingResultSetInPlay = false;

public CommunicationsException(ConnectionImpl conn, long lastPacketSentTimeMs,
long lastPacketReceivedTimeMs, Exception underlyingException) {

this.exceptionMessage = SQLError.createLinkFailureMessageBasedOnHeuristics(conn,
      lastPacketSentTimeMs, lastPacketReceivedTimeMs, underlyingException,
      this.streamingResultSetInPlay);
```

`streamingResultSetInPlay` was always false, which in the following code in `SQLError.createLinkFailureMessageBasedOnHeuristics()` never being executed:

```
if (streamingResultSetInPlay) {
    exceptionMessageBuf.append(
    Messages.getString("CommunicationsException.ClientWasStreaming")); //$NON-NLS-1$
} else {
...
```

(Bug #44588)

- The `SQLError.createLinkFailureMessageBasedOnHeuristics()` method created a message text for communication link failures. When certain conditions were met, this message included both “last packet sent” and “last packet received” information, but when those conditions were not met, only “last packet sent” information was provided.

  Information about when the last packet was successfully received should be provided in all cases. (Bug #44587)

- `Statement.getGeneratedKeys()` retained result set instances until the statement was closed. This caused memory leaks for long-lived statements, or statements used in tight loops. (Bug #44056)

- Using `useInformationSchema` with `DatabaseMetaData.getExportedKeys()` generated the following exception:

  ```
  com.mysql.jdbc.exceptions.MySQLIntegrityConstraintViolationException: Column
  'REFERENCED_TABLE_NAME' in where clause is ambiguous
  ...
  at com.mysql.jdbc.PreparedStatement.executeInternal(PreparedStatement.java:1772)
  at com.mysql.jdbc.PreparedStatement.executeQuery(PreparedStatement.java:1923)
  at
  com.mysql.jdbc.DatabaseMetaDataUsingInfoSchema.executeMetadataQuery(
  DatabaseMetaDataUsingInfoSchema.java:50)
  at
  com.mysql.jdbc.DatabaseMetaDataUsingInfoSchema.getExportedKeys(
  DatabaseMetaDataUsingInfoSchema.java:603)
  ```

  (Bug #43714)

- `LoadBalancingConnectionProxy.doPing()` did not have blacklist awareness.

  `LoadBalancingConnectionProxy` implemented `doPing()` to ping all underlying connections, but it threw any exceptions it encountered during this process.

  With the global blacklist enabled, it catches these exceptions, adds the host to the global blacklist, and only throws an exception if all hosts are down. (Bug #43421)

- The method `Statement.getGeneratedKeys()` did not return values for `UNSIGNED BIGINTS` with values greater than `Long.MAX_VALUE`.

  Unfortunately, because the server does not tell clients what TYPE the auto increment value is, the driver cannot consistently return BigIntegers for the result set returned from `getGeneratedKeys()`, it will only return them if the value is greater than `Long.MAX_VALUE`. If your application needs this consistency, it will need to check the class of the return value from `.getObject()` on the ResultSet returned by `Statement.getGeneratedKeys()` and if it is not a BigInteger, create one based on the `java.lang.Long` that is returned. (Bug #43196)

- When the MySQL Server was upgraded from 4.0 to 5.0, the Connector/J application then failed to connect to the server. This was because authentication failed when the application ran from EBCDIC platforms such as z/OS. (Bug #43071)

- When connecting with `traceProtocol=true`, no trace data was generated for the server greeting or login request. (Bug #43070)

- Connector/J generated an unhandled `StringIndexOutOfBoundsException`:

```
java.lang.StringIndexOutOfBoundsException: String index out of range: -1
at java.lang.String.substring(String.java:1938)
at com.mysql.jdbc.EscapeProcessor.processTimeToken(EscapeProcessor.java:353)
at com.mysql.jdbc.EscapeProcessor.escapeSQL(EscapeProcessor.java:257)
at com.mysql.jdbc.StatementImpl.executeUpdate(StatementImpl.java:1546)
at com.mysql.jdbc.StatementImpl.executeUpdate(StatementImpl.java:1524)
```

  (Bug #42253)

- A `ConcurrentModificationException` was generated in `LoadBalancingConnectionProxy`:

```
java.util.ConcurrentModificationException
 at java.util.HashMap$HashIterator.nextEntry(Unknown Source)
 at java.util.HashMap$KeyIterator.next(Unknown Source)
 at
com.mysql.jdbc.LoadBalancingConnectionProxy.getGlobalBlacklist(LoadBalancingConnectionProxy.java:520)
 at com.mysql.jdbc.RandomBalanceStrategy.pickConnection(RandomBalanceStrategy.java:55)
 at
com.mysql.jdbc.LoadBalancingConnectionProxy.pickNewConnection(LoadBalancingConnectionProxy.java:414)
 at
com.mysql.jdbc.LoadBalancingConnectionProxy.invoke(LoadBalancingConnectionProxy.java:390)
```

  (Bug #42055)

- SQL injection was possible when using a string containing U+00A5 in a client-side prepared statement, and the character set being used was SJIS/Windows-31J. (Bug #41730)

- If there was an apostrophe in a comment in a statement that was being sent through Connector/J, the apostrophe was still recognized as a quote and put the state machine in `EscapeTokenizer` into the `inQuotes` state. This led to further parse errors.

  For example, consider the following statement:

```
String sql = "-- Customer's zip code will be fixed\n" +
  "update address set zip_code = 99999\n" +
  "where not regexp '^[0-9]{5}([[.-.]])?([0-9]{4})?$'";
```

  When passed through Connector/J, the `EscapeTokenizer` did not recognize that the first apostrophe was in a comment and thus set `inQuotes` to true. When that happened, the quote count was incorrect and thus the regular expression did not appear to be in quotation marks. With the parser not detecting that the regular expression was in quotation marks, the curly braces were recognized as escape sequences and were removed from the regular expression, breaking it. The server thus received SQL such as:

```
-- Customer's zip code will be fixed
update address set zip_code = '99999'
where not regexp '^[0-9]([[.-.]])?([0-9])?$'
```

  (Bug #41566)

- MySQL Connector/J 5.1.7 was slower than previous versions when the `rewriteBatchedStatements` option was set to `true`.

Note

The performance regression in `indexOfIgnoreCaseRespectMarker()`has been fixed. It has also been made possible for the driver to rewrite `INSERT` statements with `ON DUPLICATE KEY UPDATE` clauses in them, as long as the `UPDATE` clause contains no reference to `LAST_INSERT_ID()`, as that would cause the driver to return bogus values for `getGeneratedKeys()` invocations. This has resulted in improved performance over version 5.1.7.

(Bug #41532)

- When accessing a result set column by name using `ResultSetImpl.findColumn()` an exception was generated:

```
java.lang.NullPointerException
at com.mysql.jdbc.ResultSetImpl.findColumn(ResultSetImpl.java:1103)
at com.mysql.jdbc.ResultSetImpl.getShort(ResultSetImpl.java:5415)
at org.apache.commons.dbcp.DelegatingResultSet.getShort(DelegatingResultSet.java:219)
at com.zimbra.cs.db.DbVolume.constructVolume(DbVolume.java:297)
at com.zimbra.cs.db.DbVolume.get(DbVolume.java:197)
at com.zimbra.cs.db.DbVolume.create(DbVolume.java:95)
at com.zimbra.cs.store.Volume.create(Volume.java:227)
at com.zimbra.cs.store.Volume.create(Volume.java:189)
at com.zimbra.cs.service.admin.CreateVolume.handle(CreateVolume.java:48)
at com.zimbra.soap.SoapEngine.dispatchRequest(SoapEngine.java:428)
at com.zimbra.soap.SoapEngine.dispatch(SoapEngine.java:285)
```

(Bug #41484)

- The `RETURN_GENERATED_KEYS` flag was being ignored. For example, in the following code the `RETURN_GENERATED_KEYS` flag was ignored:

```
PreparedStatement ps = connection.prepareStatement("INSERT INTO table
  values(?,?)",PreparedStatement.RETURN_GENERATED_KEYS);
```

(Bug #41448)

- When using Connector/J 5.1.7 to connect to MySQL Server 4.1.18 the following error message was generated:

```
Thu Dec 11 17:38:21 PST 2008 WARN: Invalid value {1} for server variable named {0},
falling back to sane default of {2}
```

This occurred with MySQL Server version that did not support `auto_increment_increment`. The error message should not have been generated. (Bug #41416)

- When `DatabaseMetaData.getProcedureColumns()` was called, the value for `LENGTH` was always returned as 65535, regardless of the column type (fixed or variable) or the actual length of the column.

  However, if you obtained the `PRECISION` value, this was correct for both fixed and variable length columns. (Bug #41269)

- `PreparedStatement.addBatch()` did not check for all parameters being set, which led to inconsistent behavior in `executeBatch()`, especially when rewriting batched statements into multi-value `INSERT`s. (Bug #41161)

- Error message strings contained variable values that were not expanded. For example:

```
Mon Nov 17 11:43:18 JST 2008 WARN: Invalid value {1} for server variable named {0},
```

```
falling back to sane default of {2}
```

(Bug #40772)

- When using `rewriteBatchedStatements=true` with:

```
INSERT INTO table_name_values (...) VALUES (...)
```

Query rewriting failed because “values” at the end of the table name was mistaken for the reserved keyword. The error generated was as follows:

```
testBug40439(testsuite.simple.TestBug40439)java.sql.BatchUpdateException: You have an
error in your SQL syntax; check the manual that corresponds to your MySQL server version
for the right syntax to use near 'values (2,'toto',2),(id,data, ordr) values
(3,'toto',3),(id,data, ordr) values (' at line 1
at com.mysql.jdbc.PreparedStatement.executeBatchedInserts(PreparedStatement.java:1495)
at com.mysql.jdbc.PreparedStatement.executeBatch(PreparedStatement.java:1097)
at testsuite.simple.TestBug40439.testBug40439(TestBug40439.java:42)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
at testsuite.simple.TestBug40439.main(TestBug40439.java:57)
```

(Bug #40439)

- A statement interceptor received the incorrect parameters when used with a batched statement. (Bug #39426)
- Using Connector/J 5.1.6 the method `ResultSet.getObject` returned a `BYTE[]` for following:

```
SELECT TRIM(rowid) FROM tbl
```

Where `rowid` had a type of `INT(11) PRIMARY KEY AUTO_INCREMENT`.

The expected return type was one of `CHAR`, `VARCHAR`, `CLOB`, however, a `BYTE[]` was returned.

Further, adding `functionsNeverReturnBlobs=true` to the connection string did not have any effect on the return type. (Bug #38387)

Changes in MySQL Connector/J 5.1.7 (2008-10-21)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- When statements include `ON DUPLICATE UPDATE`, and `rewriteBatchedStatements` is set to true, batched statements are not rewritten into the form `INSERT INTO table VALUES (), (), ()`, instead the statements are executed sequentially.

Bugs Fixed

- When an application using Connector/J ran into an error, the `SQLException.getMessage()` method was returning null instead of an error description. This has been fixed by calling the `setRowPositionValidity()` method in the `ResultSetImpl` constructor. (Bug #11749413, Bug #38943)
- When using `trustCertificateKeyStoreUrl` or `clientCertificateKeyStoreUrl`, an `IllegalStateException` was caused by an uninitialized `TrustManagerFactoryImpl` object. (Bug #11748637, Bug #36948, Bug #38192)

- `Statement.getGeneratedKeys()` returned two keys when using `ON DUPLICATE KEY UPDATE` and the row was updated, not inserted. (Bug #42309)
- When using the replication driver with `autoReconnect=true`, Connector/J checks in `PreparedStatement.execute` (also called by `CallableStatement.execute`) to determine if the first character of the statement is an "S", in an attempt to block all statements that are not read-only-safe, for example non-`SELECT` statements. However, this also blocked `CALL`s to stored procedures, even if the stored procedures were defined as `SQL READ DATA` or `NO SQL`. (Bug #40031)
- With large result sets `ResultSet.findColumn` became a performance bottleneck. (Bug #39962)
- Connector/J ignored the value of the MySQL Server variable `auto_increment_increment`. (Bug #39956)
- Connector/J failed to parse `TIMESTAMP` strings for nanos correctly. (Bug #39911)
- When the `LoadBalancingConnectionProxy` handles a `SQLException` with SQL state starting with "08", it calls `invalidateCurrentConnection`, which in turn removes that `Connection` from `liveConnections` and the `connectionsToHostsMap`, but it did not add the host to the new global blacklist, if the global blacklist was enabled.

  There was also the possibility of a `NullPointerException` when trying to update stats, where `connectionsToHostsMap.get(this.currentConn)` was called:

  ```
  int hostIndex = ((Integer) this.hostsToListIndexMap.get(this.connectionsToHostsMap.get(this.currentConn)
  ```

  This could happen if a client tried to issue a rollback after catching a `SQLException` caused by a connection failure. (Bug #39784)
- When configuring the Java Replication Driver the last slave specified was never used. (Bug #39611)
- When an `INSERT ON DUPLICATE KEY UPDATE` was performed, and the key already existed, the `affected-rows` value was returned as 1 instead of 0. (Bug #39352)
- When using loadbalancing, Connector/J might not cycle through the whole list of configured hosts in its attempt to make a connection. This fix corrects the implementation of the configuration parameter `retriesAllDown`, making sure that Connector/J cycles through the whole list of hosts for each of its attempt to connect. (Bug #38785, Bug #11749331)
- When using the random load balancing strategy and starting with two servers that were both unavailable, an `IndexOutOfBoundsException` was generated when removing a server from the `whiteList`. (Bug #38782)
- Connector/J threw the following exception when using a read-only connection:

  ```
  java.sql.SQLException: Connection is read-only. Queries leading to data
  modification are not allowed.
  ```

  (Bug #38747)
- Connector/J was unable to connect when using a non-`latin1` password. (Bug #37570)
- The `useOldAliasMetadataBehavior` connection property was ignored. (Bug #35753)
- Incorrect result is returned from `isAfterLast()` in streaming `ResultSet` when using `setFetchSize(Integer.MIN_VALUE)`. (Bug #35170)
- When `getGeneratedKeys()` was called on a statement that had not been created with `RETURN_GENERATED_KEYS`, no exception was thrown, and batched executions then returned erroneous values. (Bug #34185)

- The `loadBalance bestResponseTime` blacklists did not have a global state. (Bug #33861)

Changes in MySQL Connector/J 5.1.6 (2008-03-07)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Multiple result sets were not supported when using streaming mode to return data. Both normal statements and the result sets from stored procedures now return multiple results sets, with the exception of result sets using registered `OUTPUT` parameters. (Bug #33678)
- XAConnections and datasources have been updated to the JDBC-4.0 standard.
- The profiler event handling has been made extensible using the `profilerEventHandler` connection property.
- Add the `verifyServerCertificate` property. If set to "false" the driver will not verify the server's certificate when `useSSL` is set to "true"

  When using this feature, the keystore parameters should be specified by the `clientCertificateKeyStore*` properties, rather than system properties, as the JSSE doesn't it straightforward to have a nonverifying trust store and the "default" key store.

Bugs Fixed

- When `nullCatalogMeansCurrent` was set to `false`, calls to `getColumns()` with null catalog and schema returned all tables in all catalogs, instead of all columns from every matching table from any database. (Bug #11747185, Bug #31187)
- Compiling in Eclipse could produce a compilation warning for line 814 in the file`com.mysql.jdbc.DatabaseMetaData.java`. (Bug #11746221, Bug #24595)
- `DatabaseMetaData.getColumns()` returns incorrect `COLUMN_SIZE` value for `SET` column. (Bug #36830)
- When trying to read `Time` values like “00:00:00” with `ResultSet.getTime(int)` an exception is thrown. (Bug #36051)
- JDBC connection URL parameters is ignored when using `MysqlConnectionPoolDataSource`. (Bug #35810)
- When `useServerPrepStmts=true` and slow query logging is enabled, the connector throws a `NullPointerException` when it encounters a slow query. (Bug #35666)
- When using the keyword “loadbalance” in the connection string and trying to perform load balancing between two databases, the driver appears to hang. (Bug #35660)
- JDBC data type getter method was changed to accept only column name, whereas previously it accepted column label. (Bug #35610)
- Prepared statements from pooled connections caused a `NullPointerException` when `closed()` under JDBC-4.0. (Bug #35489)
- In calling a stored function returning a `bigint`, an exception is encountered beginning:

  ```
  java.sql.SQLException: java.lang.NumberFormatException: For input string:
  ```

  followed by the text of the stored function starting after the argument list. (Bug #35199)
- The JDBC driver uses a different method for evaluating column names in `resultsetmetadata.getColumnName()` and when looking for a column in

`resultset.getObject(columnName)`. This causes Hibernate to fail in queries where the two methods yield different results, for example in queries that use alias names:

```
SELECT column AS aliasName from table
```

(Bug #35150)

- `MysqlConnectionPoolDataSource` does not support `ReplicationConnection`. Notice that we implemented `com.mysql.jdbc.Connection` for `ReplicationConnection`, however, only accessors from ConnectionProperties are implemented (not the mutators), and they return values from the currently active connection. All other methods from `com.mysql.jdbc.Connection` are implemented, and operate on the currently active connection, with the exception of `resetServerState()` and `changeUser()`. (Bug #34937)
- `ResultSet.getTimestamp()` returns incorrect values for month/day of `TIMESTAMP`s when using server-side prepared statements (not enabled by default). (Bug #34913)
- `RowDataStatic` doesn't always set the metadata in `ResultSetRow`, which can lead to failures when unpacking `DATE`, `TIME`, `DATETIME` and `TIMESTAMP` types when using absolute, relative, and previous result set navigation methods. (Bug #34762)
- When calling `isValid()` on an active connection, if the timeout is nonzero then the `Connection` is invalidated even if the `Connection` is valid. (Bug #34703)
- It was not possible to truncate a `BLOB` using `Blog.truncate()` when using 0 as an argument. (Bug #34677)
- When using a cursor fetch for a statement, the internal prepared statement could cause a memory leak until the connection was closed. The internal prepared statement is now deleted when the corresponding result set is closed. (Bug #34518)
- When retrieving the column type name of a geometry field, the driver would return `UNKNOWN` instead of `GEOMETRY`. (Bug #34194)
- Statements with batched values do not return correct values for `getGeneratedKeys()` when `rewriteBatchedStatements` is set to `true`, and the statement has an `ON DUPLICATE KEY UPDATE` clause. (Bug #34093)
- The internal class `ResultSetInternalMethods` referenced the nonpublic class `com.mysql.jdbc.CachedResultSetMetaData`. (Bug #33823)
- A `NullPointerException` could be raised when using client-side prepared statements and enabled the prepared statement cache using the `cachePrepStmts`. (Bug #33734)
- Using server side cursors and cursor fetch, the table metadata information would return the data type name instead of the column name. (Bug #33594)
- `ResultSet.getTimestamp()` would throw a `NullPointerException` instead of a `SQLException` when called on an empty `ResultSet`. (Bug #33162)
- Load balancing connection using best response time would incorrectly "stick" to hosts that were down when the connection was first created.

  We solve this problem with a black list that is used during the picking of new hosts. If the black list ends up including all configured hosts, the driver will retry for a configurable number of times (the `retriesAllDown` configuration property, with a default of 120 times), sleeping 250ms between attempts to pick a new connection.

  We've also went ahead and made the balancing strategy extensible. To create a new strategy, implement the interface `com.mysql.jdbc.BalanceStrategy` (which also includes our standard "extension" interface), and tell the driver to use it by passing in the class name using the `loadBalanceStrategy` configuration property. (Bug #32877)

- During a Daylight Savings Time (DST) switchover, there was no way to store two timestamp/datetime values , as the hours end up being the same when sent as the literal that MySQL requires.

  Note that to get this scenario to work with MySQL (since it doesn't support per-value timezones), you need to configure your server (or session) to be in UTC, and tell the driver not to use the legacy date/time code by setting `useLegacyDatetimeCode` to "false". This will cause the driver to always convert to/from the server and client timezone consistently.

  This bug fix also fixes Bug #15604, by adding entirely new date/time handling code that can be switched on by `useLegacyDatetimeCode` being set to "false" as a JDBC configuration property. For Connector/J 5.1.x, the default is "true", in trunk and beyond it will be "false" (that is, the old date/time handling code will be deprecated) (Bug #32577, Bug #15604)

- When unpacking rows directly, we don't hand off error message packets to the internal method which decodes them correctly, so no exception is raised, and the driver than hangs trying to read rows that aren't there. This tends to happen when calling stored procedures, as normal SELECTs won't have an error in this spot in the protocol unless an I/O error occurs. (Bug #32246)

- When using a connection from `ConnectionPoolDataSource`, some `Connection.prepareStatement()` methods would return null instead of the prepared statement. (Bug #32101)

- Using `CallableStatement.setNull()` on a stored function would throw an `ArrayIndexOutOfBounds` exception when setting the last parameter to null. (Bug #31823)

- `MysqlValidConnectionChecker` doesn't properly handle connections created using `ReplicationConnection`. (Bug #31790)

- Retrieving the server version information for an active connection could return invalid information if the default character encoding on the host was not ASCII compatible. (Bug #31192)

- Further fixes have been made to this bug in the event that a node is nonresponsive. Connector/J will now try a different random node instead of waiting for the node to recover before continuing. (Bug #31053)

- `ResultSet` returned by `Statement.getGeneratedKeys()` is not closed automatically when statement that created it is closed. (Bug #30508)

- `DatabaseMetadata.getColumns()` doesn't return the correct column names if the connection character isn't UTF-8. A bug in MySQL server compounded the issue, but was fixed within the MySQL 5.0 release cycle. The fix includes changes to all the sections of the code that access the server metadata. (Bug #20491)

- Fixed `ResultSetMetadata.getColumnName()` for result sets returned from `Statement.getGeneratedKeys()` - it was returning null instead of "GENERATED_KEY" as in 5.0.x.

Changes in MySQL Connector/J 5.1.5 (2007-10-09)

- New Features, Compared to the 5.0 Series of Connector/J
- Functionality Added or Changed
- Bugs Fixed

New Features, Compared to the 5.0 Series of Connector/J

- Support for JDBC-4.0 `NCHAR`, `NVARCHAR` and `NCLOB` types.

- JDBC-4.0 support for setting per-connection client information (which can be viewed in the comments section of a query using `SHOW PROCESSLIST` on a MySQL server, or can be extended to support custom persistence of the information using a public interface).

- Support for JDBC-4.0 XML processing using JAXP interfaces to DOM, SAX and StAX.
- JDBC-4.0 standardized unwrapping to interfaces that include vendor extensions.

Functionality Added or Changed

- Added `autoSlowLog` configuration property, overrides `slowQueryThreshold*` properties, driver determines slow queries by those that are slower than 5 * stddev of the mean query time (outside the 96% percentile).

Bugs Fixed

- When a connection is in read-only mode, queries that are wrapped in parentheses were incorrectly identified DML statements. (Bug #28256)
- When calling `setTimestamp` on a prepared statement, the timezone information stored in the calendar object was ignored. This resulted in the incorrect `DATETIME` information being stored. The following example illustrates this:

```
Timestamp t = new Timestamp( cal.getTimeInMillis() );
ps.setTimestamp( N, t, cal );
```

(Bug #15604)

Changes in MySQL Connector/J 5.1.4 (Not Released)

Only released internally.

Version 5.1.4 has no changelog entries, or they have not been published because the product version has not been released.

Changes in MySQL Connector/J 5.1.3 (2007-09-10)

- New Features, Compared to the 5.0 Series of Connector/J
- Functionality Added or Changed
- Bugs Fixed

New Features, Compared to the 5.0 Series of Connector/J

- Support for JDBC-4.0 `NCHAR`, `NVARCHAR` and `NCLOB` types.
- JDBC-4.0 support for setting per-connection client information (which can be viewed in the comments section of a query using `SHOW PROCESSLIST` on a MySQL server, or can be extended to support custom persistence of the information using a public interface).
- Support for JDBC-4.0 XML processing using JAXP interfaces to DOM, SAX and StAX.
- JDBC-4.0 standardized unwrapping to interfaces that include vendor extensions.

Functionality Added or Changed

- Connector/J now connects using an initial character set of `utf-8` solely for the purpose of authentication to permit user names or database names in any character set to be used in the JDBC connection URL. (Bug #29853)
- Added two configuration parameters:
  - `blobsAreStrings`: Should the driver always treat BLOBs as Strings. Added specifically to work around dubious metadata returned by the server for `GROUP BY` clauses. Defaults to false.

    - `functionsNeverReturnBlobs`: Should the driver always treat data from functions returning `BLOBs` as Strings. Added specifically to work around dubious metadata returned by the server for `GROUP BY` clauses. Defaults to false.

- Setting `rewriteBatchedStatements` to `true` now causes CallableStatements with batched arguments to be re-written in the form "CALL (...); CALL (...); ..." to send the batch in as few client/server round trips as possible.

- The driver now picks appropriate internal row representation (whole row in one buffer, or individual byte[]s for each column value) depending on heuristics, including whether or not the row has `BLOB` or `TEXT` types and the overall row-size. The threshold for row size that will cause the driver to use a buffer rather than individual byte[]s is configured by the configuration property `largeRowSizeThreshold`, which has a default value of 2KB.

- The data (and how it is stored) for `ResultSet` rows are now behind an interface which enables us (in some cases) to allocate less memory per row, in that for "streaming" result sets, we re-use the packet used to read rows, since only one row at a time is ever active.

- Added experimental support for statement "interceptors" through the `com.mysql.jdbc.StatementInterceptor` interface, examples are in `com/mysql/jdbc/interceptors`. Implement this interface to be placed "in between" query execution, so that it can be influenced (currently experimental).

- The driver will automatically adjust the server session variable `net_write_timeout` when it determines its been asked for a "streaming" result, and resets it to the previous value when the result set has been consumed. (The configuration property is named `netTimeoutForStreamingResults`, with a unit of seconds, the value '0' means the driver will not try and adjust this value).

- JDBC-4.0 ease-of-development features including auto-registration with the `DriverManager` through the service provider mechanism, standardized Connection validity checks and categorized `SQLExceptions` based on recoverability/retry-ability and class of the underlying error.

- `Statement.setQueryTimeout()`s now affect the entire batch for batched statements, rather than the individual statements that make up the batch.

- Errors encountered during `Statement`/`PreparedStatement`/`CallableStatement.executeBatch()` when `rewriteBatchStatements` has been set to `true` now return `BatchUpdateExceptions` according to the setting of `continueBatchOnError`.

  If `continueBatchOnError` is set to `true`, the update counts for the "chunk" that were sent as one unit will all be set to `EXECUTE_FAILED`, but the driver will attempt to process the remainder of the batch. You can determine which "chunk" failed by looking at the update counts returned in the `BatchUpdateException`.

  If `continueBatchOnError` is set to "false", the update counts returned will contain all updates up-to and including the failed "chunk", with all counts for the failed "chunk" set to `EXECUTE_FAILED`.

  Since MySQL doesn't return multiple error codes for multiple-statements, or for multi-value `INSERT`/`REPLACE`, it is the application's responsibility to handle determining which item(s) in the "chunk" actually failed.

- New methods on com.mysql.jdbc.Statement: `setLocalInfileInputStream()` and `getLocalInfileInputStream()`:

    - `setLocalInfileInputStream()` sets an `InputStream` instance that will be used to send data to the MySQL server for a `LOAD DATA LOCAL INFILE` statement rather than a `FileInputStream` or `URLInputStream` that represents the path given as an argument to the statement.

This stream will be read to completion upon execution of a `LOAD DATA LOCAL INFILE` statement, and will automatically be closed by the driver, so it needs to be reset before each call to `execute*()` that would cause the MySQL server to request data to fulfill the request for `LOAD DATA LOCAL INFILE`.

If this value is set to `NULL`, the driver will revert to using a `FileInputStream` or `URLInputStream` as required.

- `getLocalInfileInputStream()` returns the `InputStream` instance that will be used to send data in response to a `LOAD DATA LOCAL INFILE` statement.

  This method returns `NULL` if no such stream has been set using `setLocalInfileInputStream()`.

- Setting `useBlobToStoreUTF8OutsideBMP` to `true` tells the driver to treat `[MEDIUM/LONG]BLOB` columns as `[LONG]VARCHAR` columns holding text encoded in UTF-8 that has characters outside the BMP (4-byte encodings), which MySQL server can't handle natively.

  Set `utf8OutsideBmpExcludedColumnNamePattern` to a regex so that column names matching the given regex will still be treated as `BLOBs` The regex must follow the patterns used for the `java.util.regex`package. The default is to exclude no columns, and include all columns.

  Set `utf8OutsideBmpIncludedColumnNamePattern` to specify exclusion rules to utf8OutsideBmpExcludedColumnNamePattern". The regex must follow the patterns used for the `java.util.regex` package.

Bugs Fixed

- `setObject(int, Object, int, int)` delegate in PreparedStatementWrapper delegates to wrong method. (Bug #30892)
- NPE with null column values when `padCharsWithSpace` is set to true. (Bug #30851)
- Collation on `VARBINARY` column types would be misidentified. A fix has been added, but this fix only works for MySQL server versions 5.0.25 and newer, since earlier versions didn't consistently return correct metadata for functions, and thus results from subqueries and functions were indistinguishable from each other, leading to type-related bugs. (Bug #30664)
- An `ArithmeticException` or `NullPointerException` would be raised when the batch had zero members and `rewriteBatchedStatements=true` when `addBatch()` was never called, or `executeBatch()` was called immediately after `clearBatch()`. (Bug #30550)
- Closing a load-balanced connection would cause a `ClassCastException`. (Bug #29852)
- Connection checker for JBoss didn't use same method parameters using reflection, causing connections to always seem "bad". (Bug #29106)
- `DatabaseMetaData.getTypeInfo()` for the types `DECIMAL` and `NUMERIC` will return a precision of 254 for server versions older than 5.0.3, 64 for versions 5.0.3 to 5.0.5 and 65 for versions newer than 5.0.5. (Bug #28972)
- `CallableStatement.executeBatch()` doesn't work when connection property `noAccessToProcedureBodies` has been set to `true`.

  The fix involves changing the behavior of `noAccessToProcedureBodies`,in that the driver will now report all parameters as `IN` parameters but permit callers to call registerOutParameter() on them without throwing an exception. (Bug #28689)

- `DatabaseMetaData.getColumns()` doesn't contain `SCOPE_*` or `IS_AUTOINCREMENT` columns. (Bug #27915)

- Schema objects with identifiers other than the connection character aren't retrieved correctly in `ResultSetMetadata`. (Bug #27867)
- `Connection.getServerCharacterEncoding()` doesn't work for servers with version >= 4.1. (Bug #27182)
- The automated SVN revisions in `DBMD.getDriverVersion()`. The SVN revision of the directory is now inserted into the version information during the build. (Bug #21116)
- Specifying a "validation query" in your connection pool that starts with "/* ping */" _exactly_ will cause the driver to instead send a ping to the server and return a fake result set (much lighter weight), and when using a ReplicationConnection or a LoadBalancedConnection, will send the ping across all active connections.

Changes in MySQL Connector/J 5.1.2 (2007-06-29)

This is a new Beta development release, fixing recently discovered bugs.

Functionality Added or Changed

- Setting the configuration property `rewriteBatchedStatements` to `true` will now cause the driver to rewrite batched prepared statements with more than 3 parameter sets in a batch into multi-statements (separated by ";") if they are not plain (that is, without `SELECT` or `ON DUPLICATE KEY UPDATE` clauses) `INSERT` or `REPLACE` statements.

Changes in MySQL Connector/J 5.1.1 (2007-06-22)

This is a new Alpha development release, adding new features and fixing recently discovered bugs.

Functionality Added or Changed

- **Incompatible Change:** Pulled vendor-extension methods of `Connection` implementation out into an interface to support `java.sql.Wrapper` functionality from `ConnectionPoolDataSource`. The vendor extensions are javadoc'd in the `com.mysql.jdbc.Connection` interface.

  For those looking further into the driver implementation, it is not an API that is used for pluggability of implementations inside our driver (which is why there are still references to `ConnectionImpl` throughout the code).

  We've also added server and client `prepareStatement()` methods that cover all of the variants in the JDBC API.

  `Connection.serverPrepare(String)` has been re-named to `Connection.serverPrepareStatement()` for consistency with `Connection.clientPrepareStatement()`.
- Row navigation now causes any streams/readers open on the result set to be closed, as in some cases we're reading directly from a shared network packet and it will be overwritten by the "next" row.
- Made it possible to retrieve prepared statement parameter bindings (to be used in `StatementInterceptors`, primarily).
- Externalized the descriptions of connection properties.
- The data (and how it is stored) for `ResultSet` rows are now behind an interface which enables us (in some cases) to allocate less memory per row, in that for "streaming" result sets, we re-use the packet used to read rows, since only one row at a time is ever active.
- Similar to `Connection`, we pulled out vendor extensions to `Statement` into an interface named `com.mysql.Statement`, and moved the `Statement` class into `com.mysql.StatementImpl`.

The two methods (javadoc'd in `com.mysql.Statement` are `enableStreamingResults()`, which already existed, and `disableStreamingResults()` which sets the statement instance back to the fetch size and result set type it had before `enableStreamingResults()` was called.

- Driver now picks appropriate internal row representation (whole row in one buffer, or individual byte[]s for each column value) depending on heuristics, including whether or not the row has `BLOB` or `TEXT` types and the overall row-size. The threshold for row size that will cause the driver to use a buffer rather than individual byte[]s is configured by the configuration property `largeRowSizeThreshold`, which has a default value of 2KB.
- Added experimental support for statement "interceptors" through the `com.mysql.jdbc.StatementInterceptor` interface, examples are in `com/mysql/jdbc/interceptors`.

  Implement this interface to be placed "in between" query execution, so that you can influence it. (currently experimental).

  `StatementInterceptors` are "chainable" when configured by the user, the results returned by the "current" interceptor will be passed on to the next on in the chain, from left-to-right order, as specified by the user in the JDBC configuration property `statementInterceptors`.
- See the sources (fully javadoc'd) for `com.mysql.jdbc.StatementInterceptor` for more details until we iron out the API and get it documented in the manual.
- Setting `rewriteBatchedStatements` to `true` now causes `CallableStatements` with batched arguments to be re-written in the form `CALL (...); CALL (...); ...` to send the batch in as few client/server round trips as possible.

Changes in MySQL Connector/J 5.1.0 (2007-04-11)

This is the first public alpha release of the current Connector/J 5.1 development branch, providing an insight to upcoming features. Although some of these are still under development, this release includes the following new features and changes (in comparison to the current Connector/J 5.0 production release):

Important change: Due to a number of issues with the use of server-side prepared statements, Connector/J 5.0.5 has disabled their use by default. The disabling of server-side prepared statements does not affect the operation of the connector in any way.

To enable server-side prepared statements you must add the following configuration property to your connector string:

```
useServerPrepStmts=true
```

The default value of this property is `false` (that is, Connector/J does not use server-side prepared statements).

Note

The disabling of server-side prepared statements does not affect the operation of the connector. However, if you use the `useTimezone=true` connection option and use client-side prepared statements (instead of server-side prepared statements) you should also set `useSSPSCompatibleTimezoneShift=true`.

Functionality Added or Changed

- Refactored `CommunicationsException` into a JDBC-3.0 version, and a JDBC-4.0 version (which extends `SQLRecoverableException`, now that it exists).

Note

This change means that if you were catching `com.mysql.jdbc.CommunicationsException` in your applications instead of looking at the SQLState class of `08`, and are moving to Java 6 (or newer), you need to change your imports to that exception to be `com.mysql.jdbc.exceptions.jdbc4.CommunicationsException`, as the old class will not be instantiated for communications link-related errors under Java 6.

- Added support for JDBC-4.0 categorized `SQLExceptions`.
- Added support for JDBC-4.0's `NCLOB`, and `NCHAR`/`NVARCHAR` types.
- `com.mysql.jdbc.java6.javac`: Full path to your Java-6 `javac` executable
- Added support for JDBC-4.0's SQLXML interfaces.
- Re-worked Ant buildfile to build JDBC-4.0 classes separately, as well as support building under Eclipse (since Eclipse can't mix/match JDKs).

  To build, you must set `JAVA_HOME` to J2SDK-1.4.2 or Java-5, and set the following properties on your Ant command line:

  - `com.mysql.jdbc.java6.javac`: Full path to your Java-6 `javac` executable
  - `com.mysql.jdbc.java6.rtjar`: Full path to your Java-6 `rt.jar` file
- New feature—driver will automatically adjust session variable `net_write_timeout` when it determines it has been asked for a "streaming" result, and resets it to the previous value when the result set has been consumed. (configuration property is named `netTimeoutForStreamingResults` value and has a unit of seconds, the value `0` means the driver will not try and adjust this value).
- Added support for JDBC-4.0's client information. The back-end storage of information provided using `Connection.setClientInfo()` and retrieved by `Connection.getClientInfo()` is pluggable by any class that implements the `com.mysql.jdbc.JDBC4ClientInfoProvider` interface and has a no-args constructor.

  The implementation used by the driver is configured using the `clientInfoProvider` configuration property (with a default of value of `com.mysql.jdbc.JDBC4CommentClientInfoProvider`, an implementation which lists the client information as a comment prepended to every query sent to the server).

  This functionality is only available when using Java-6 or newer.
- `com.mysql.jdbc.java6.rtjar`: Full path to your Java-6 `rt.jar` file
- Added support for JDBC-4.0's `Wrapper` interface.

Changes in MySQL Connector/J 5.0

Changes in MySQL Connector/J 5.0.8 (2007-10-09)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- `blobsAreStrings`: Should the driver always treat BLOBs as Strings. Added specifically to work around dubious metadata returned by the server for `GROUP BY` clauses. Defaults to false.

- Added two configuration parameters:
  - `blobsAreStrings`: Should the driver always treat BLOBs as Strings. Added specifically to work around dubious metadata returned by the server for `GROUP BY` clauses. Defaults to false.
  - `functionsNeverReturnBlobs`: Should the driver always treat data from functions returning `BLOBs` as Strings. Added specifically to work around dubious metadata returned by the server for `GROUP BY` clauses. Defaults to false.
- `functionsNeverReturnBlobs`: Should the driver always treat data from functions returning `BLOBs` as Strings. Added specifically to work around dubious metadata returned by the server for `GROUP BY` clauses. Defaults to false.
- XAConnections now start in auto-commit mode (as per JDBC-4.0 specification clarification).
- Driver will now fall back to sane defaults for `max_allowed_packet` and `net_buffer_length` if the server reports them incorrectly (and will log this situation at `WARN` level, since it is actually an error condition).

Bugs Fixed

- Connections established using URLs of the form `jdbc:mysql:loadbalance://` weren't doing failover if they tried to connect to a MySQL server that was down. The driver now attempts connections to the next "best" (depending on the load balance strategy in use) server, and continues to attempt connecting to the next "best" server every 250 milliseconds until one is found that is up and running or 5 minutes has passed.

  If the driver gives up, it will throw the last-received `SQLException`. (Bug #31053)
- `setObject(int, Object, int, int)` delegate in PreparedStatementWrapper delegates to wrong method. (Bug #30892)
- NPE with null column values when `padCharsWithSpace` is set to true. (Bug #30851)
- Collation on `VARBINARY` column types would be misidentified. A fix has been added, but this fix only works for MySQL server versions 5.0.25 and newer, since earlier versions didn't consistently return correct metadata for functions, and thus results from subqueries and functions were indistinguishable from each other, leading to type-related bugs. (Bug #30664)
- An `ArithmeticException` or `NullPointerException` would be raised when the batch had zero members and `rewriteBatchedStatements=true` when `addBatch()` was never called, or `executeBatch()` was called immediately after `clearBatch()`. (Bug #30550)
- Closing a load-balanced connection would cause a `ClassCastException`. (Bug #29852)
- Connection checker for JBoss didn't use same method parameters using reflection, causing connections to always seem "bad". (Bug #29106)
- `DatabaseMetaData.getTypeInfo()` for the types `DECIMAL` and `NUMERIC` will return a precision of 254 for server versions older than 5.0.3, 64 for versions 5.0.3 to 5.0.5 and 65 for versions newer than 5.0.5. (Bug #28972)
- `CallableStatement.executeBatch()` doesn't work when connection property `noAccessToProcedureBodies` has been set to `true`.

  The fix involves changing the behavior of `noAccessToProcedureBodies`,in that the driver will now report all parameters as `IN` parameters but permit callers to call registerOutParameter() on them without throwing an exception. (Bug #28689)
- When a connection is in read-only mode, queries that are wrapped in parentheses were incorrectly identified DML statements. (Bug #28256)

- `UNSIGNED` types not reported using `DBMD.getTypeInfo()`, and capitalization of type names is not consistent between `DBMD.getColumns()`, `RSMD.getColumnTypeName()` and `DBMD.getTypeInfo()`.

  This fix also ensures that the precision of `UNSIGNED MEDIUMINT` and `UNSIGNED BIGINT` is reported correctly using `DBMD.getColumns()`. (Bug #27916)

- `DatabaseMetaData.getColumns()` doesn't contain `SCOPE_*` or `IS_AUTOINCREMENT` columns. (Bug #27915)

- Schema objects with identifiers other than the connection character aren't retrieved correctly in `ResultSetMetadata`. (Bug #27867)

- Cached metadata with `PreparedStatement.execute()` throws `NullPointerException`. (Bug #27412)

- `Connection.getServerCharacterEncoding()` doesn't work for servers with version >= 4.1. (Bug #27182)

- The automated SVN revisions in `DBMD.getDriverVersion()`. The SVN revision of the directory is now inserted into the version information during the build. (Bug #21116)

- Specifying a "validation query" in your connection pool that starts with "/* ping */" _exactly_ will cause the driver to instead send a ping to the server and return a fake result set (much lighter weight), and when using a ReplicationConnection or a LoadBalancedConnection, will send the ping across all active connections.

Changes in MySQL Connector/J 5.0.7 (2007-07-20)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- The driver will now automatically set `useServerPrepStmts` to `true` when `useCursorFetch` has been set to `true`, since the feature requires server-side prepared statements to function.

- `tcpKeepAlive` - Should the driver set SO_KEEPALIVE (default `true`)?

- Give more information in EOFExceptions thrown out of MysqlIO (how many bytes the driver expected to read, how many it actually read, say that communications with the server were unexpectedly lost).

- Driver detects when it is running in a ColdFusion MX server (tested with version 7), and uses the configuration bundle `coldFusion`, which sets `useDynamicCharsetInfo` to `false` (see previous entry), and sets `useLocalSessionState` and autoReconnect to `true`.

- `tcpNoDelay` - Should the driver set SO_TCP_NODELAY (disabling the Nagle Algorithm, default `true`)?

- Added configuration property `slowQueryThresholdNanos` - if `useNanosForElapsedTime` is set to `true`, and this property is set to a nonzero value the driver will use this threshold (in nanosecond units) to determine if a query was slow, instead of using millisecond units.

- `tcpRcvBuf` - Should the driver set SO_RCV_BUF to the given value? The default value of '0', means use the platform default value for this property.

- Setting `useDynamicCharsetInfo` to `false` now causes driver to use static lookups for collations as well (makes ResultSetMetadata.isCaseSensitive() much more efficient, which leads to performance increase for ColdFusion, which calls this method for every column on every table it sees, it appears).

- Added configuration properties to enable tuning of TCP/IP socket parameters:
  - `tcpNoDelay` - Should the driver set SO_TCP_NODELAY (disabling the Nagle Algorithm, default `true`)?
  - `tcpKeepAlive` - Should the driver set SO_KEEPALIVE (default `true`)?
  - `tcpRcvBuf` - Should the driver set SO_RCV_BUF to the given value? The default value of '0', means use the platform default value for this property.
  - `tcpSndBuf` - Should the driver set SO_SND_BUF to the given value? The default value of '0', means use the platform default value for this property.
  - `tcpTrafficClass` - Should the driver set traffic class or type-of-service fields? See the documentation for java.net.Socket.setTrafficClass() for more information.
- Setting the configuration parameter `useCursorFetch` to `true` for MySQL-5.0+ enables the use of cursors that enable Connector/J to save memory by fetching result set rows in chunks (where the chunk size is set by calling setFetchSize() on a Statement or ResultSet) by using fully materialized cursors on the server.
- `tcpSndBuf` - Should the driver set SO_SND_BUF to the given value? The default value of '0', means use the platform default value for this property.
- `tcpTrafficClass` - Should the driver set traffic class or type-of-service fields? See the documentation for java.net.Socket.setTrafficClass() for more information.
- Added new debugging functionality - Setting configuration property `includeInnodbStatusInDeadlockExceptions` to `true` will cause the driver to append the output of `SHOW ENGINE INNODB STATUS` to deadlock-related exceptions, which will enumerate the current locks held inside InnoDB.
- Added configuration property `useNanosForElapsedTime` - for profiling/debugging functionality that measures elapsed time, should the driver try to use nanoseconds resolution if available (requires JDK >= 1.5)?

Note

If `useNanosForElapsedTime` is set to `true`, and this property is set to "0" (or left default), then elapsed times will still be measured in nanoseconds (if possible), but the slow query threshold will be converted from milliseconds to nanoseconds, and thus have an upper bound of approximately 2000 milliseconds (as that threshold is represented as an integer, not a long).

Bugs Fixed

- Don't send any file data in response to LOAD DATA LOCAL INFILE if the feature is disabled at the client side. This is to prevent a malicious server or man-in-the-middle from asking the client for data that the client is not expecting. Thanks to Jan Kneschke for discovering the exploit and Andrey "Poohie" Hristov, Konstantin Osipov and Sergei Golubchik for discussions about implications and possible fixes. (Bug #29605)
- Parser in client-side prepared statements runs to end of statement, rather than end-of-line for '#' comments. Also added support for '--' single-line comments. (Bug #28956)
- Parser in client-side prepared statements eats character following '/' if it is not a multi-line comment. (Bug #28851)
- PreparedStatement.getMetaData() for statements containing leading one-line comments is not returned correctly.

As part of this fix, we also overhauled detection of DML for `executeQuery()` and `SELECT`s for `executeUpdate()` in plain and prepared statements to be aware of the same types of comments. (Bug #28469)

Changes in MySQL Connector/J 5.0.6 (2007-05-15)

Functionality Added or Changed

- Added an experimental load-balanced connection designed for use with SQL nodes in a MySQL Cluster/NDB environment (This is not for master-slave replication. For that, we suggest you look at `ReplicationConnection` or `lbpool`).

  If the JDBC URL starts with `jdbc:mysql:loadbalance://host-1,host-2,...host-n`, the driver will create an implementation of `java.sql.Connection` that load balances requests across a series of MySQL JDBC connections to the given hosts, where the balancing takes place after transaction commit.

  Therefore, for this to work (at all), you must use transactions, even if only reading data.

  Physical connections to the given hosts will not be created until needed.

  The driver will invalidate connections that it detects have had communication errors when processing a request. A new connection to the problematic host will be attempted the next time it is selected by the load balancing algorithm.

  There are two choices for load balancing algorithms, which may be specified by the `loadBalanceStrategy` JDBC URL configuration property:

  - `random`: The driver will pick a random host for each request. This tends to work better than round-robin, as the randomness will somewhat account for spreading loads where requests vary in response time, while round-robin can sometimes lead to overloaded nodes if there are variations in response times across the workload.
  - `bestResponseTime`: The driver will route the request to the host that had the best response time for the previous transaction.
- `bestResponseTime`: The driver will route the request to the host that had the best response time for the previous transaction.
- Added configuration property `padCharsWithSpace` (defaults to `false`). If set to `true`, and a result set column has the `CHAR` type and the value does not fill the amount of characters specified in the DDL for the column, the driver will pad the remaining characters with space (for ANSI compliance).
- When `useLocalSessionState` is set to `true` and connected to a MySQL-5.0 or later server, the JDBC driver will now determine whether an actual `commit` or `rollback` statement needs to be sent to the database when `Connection.commit()` or `Connection.rollback()` is called.

  This is especially helpful for high-load situations with connection pools that always call `Connection.rollback()` on connection check-in/check-out because it avoids a round-trip to the server.
- Added configuration property `useDynamicCharsetInfo`. If set to `false` (the default), the driver will use a per-connection cache of character set information queried from the server when necessary, or when set to `true`, use a built-in static mapping that is more efficient, but isn't aware of custom character sets or character sets implemented after the release of the JDBC driver.

Note

This only affects the `padCharsWithSpace` configuration property and the `ResultSetMetaData.getColumnDisplayWidth()` method.

- New configuration property, `enableQueryTimeouts` (default `true`).

  When enabled, query timeouts set with `Statement.setQueryTimeout()` use a shared `java.util.Timer` instance for scheduling. Even if the timeout doesn't expire before the query is processed, there will be memory used by the `TimerTask` for the given timeout which won't be reclaimed until the time the timeout would have expired if it hadn't been cancelled by the driver. High-load environments might want to consider disabling this functionality. (this configuration property is part of the `maxPerformance` configuration bundle).

- Give better error message when "streaming" result sets, and the connection gets clobbered because of exceeding `net_write_timeout` on the server.

- `random`: The driver will pick a random host for each request. This tends to work better than round-robin, as the randomness will somewhat account for spreading loads where requests vary in response time, while round-robin can sometimes lead to overloaded nodes if there are variations in response times across the workload.

- `com.mysql.jdbc.[NonRegistering]Driver` now understands URLs of the format `jdbc:mysql:replication://` and `jdbc:mysql:loadbalance://` which will create a ReplicationConnection (exactly like when using `[NonRegistering]ReplicationDriver`) and an experimental load-balanced connection designed for use with SQL nodes in a MySQL Cluster/NDB environment, respectively.

  In an effort to simplify things, we're working on deprecating multiple drivers, and instead specifying different core behavior based upon JDBC URL prefixes, so watch for `[NonRegistering]ReplicationDriver` to eventually disappear, to be replaced with `com.mysql.jdbc[NonRegistering]Driver` with the new URL prefix.

- Fixed issue where a failed-over connection would let an application call `setReadOnly(false)`, when that call should be ignored until the connection is reconnected to a writable master unless `failoverReadOnly` had been set to `false`.

- Driver will now use `INSERT INTO ... VALUES (DEFAULT)`form of statement for updatable result sets for `ResultSet.insertRow()`, rather than pre-populating the insert row with values from `DatabaseMetaData.getColumns()`(which results in a `SHOW FULL COLUMNS` on the server for every result set). If an application requires access to the default values before `insertRow()` has been called, the JDBC URL should be configured with `populateInsertRowWithDefaultValues` set to `true`.

  This fix specifically targets performance issues with ColdFusion and the fact that it seems to ask for updatable result sets no matter what the application does with them.

- More intelligent initial packet sizes for the "shared" packets are used (512 bytes, rather than 16K), and initial packets used during handshake are now sized appropriately as to not require reallocation.

Bugs Fixed

- More useful error messages are generated when the driver thinks a result set is not updatable. (Thanks to Ashley Martens for the patch). (Bug #28085)

- `Connection.getTransactionIsolation()` uses "`SHOW VARIABLES LIKE`" which is very inefficient on MySQL-5.0+ servers. (Bug #27655)

- Fixed issue where calling `getGeneratedKeys()` on a prepared statement after calling `execute()` didn't always return the generated keys (`executeUpdate()` worked fine however). (Bug #27655)

- `CALL /* ... */ some_proc()` doesn't work. As a side effect of this fix, you can now use `/* */` and `#` comments when preparing statements using client-side prepared statement emulation.

  If the comments happen to contain parameter markers (`?`), they will be treated as belonging to the comment (that is, not recognized) rather than being a parameter of the statement.

  

  Note

  The statement when sent to the server will contain the comments as-is, they're not stripped during the process of preparing the `PreparedStatement` or `CallableStatement`.

  (Bug #27400)

- `ResultSet.get*()` with a column index < 1 returns misleading error message. (Bug #27317)
- Using `ResultSet.get*()` with a column index less than 1 returns a misleading error message. (Bug #27317)
- Comments in DDL of stored procedures/functions confuse procedure parser, and thus metadata about them can not be created, leading to inability to retrieve said metadata, or execute procedures that have certain comments in them. (Bug #26959)
- Fast date/time parsing doesn't take into account `00:00:00` as a legal value. (Bug #26789)
- `PreparedStatement` is not closed in `BlobFromLocator.getBytes()`. (Bug #26592)
- When the configuration property `useCursorFetch` was set to `true`, sometimes server would return new, more exact metadata during the execution of the server-side prepared statement that enables this functionality, which the driver ignored (using the original metadata returned during `prepare()`), causing corrupt reading of data due to type mismatch when the actual rows were returned. (Bug #26173)
- `CallableStatements` with `OUT/INOUT` parameters that are "binary" (`BLOB`, `BIT`, `(VAR)BINARY`, `JAVA_OBJECT`) have extra 7 bytes. (Bug #25715)
- Whitespace surrounding storage/size specifiers in stored procedure parameters declaration causes `NumberFormatException` to be thrown when calling stored procedure on JDK-1.5 or newer, as the Number classes in JDK-1.5+ are whitespace intolerant. (Bug #25624)
- Client options not sent correctly when using SSL, leading to stored procedures not being able to return results. Thanks to Don Cohen for the bug report, testcase and patch. (Bug #25545)
- `Statement.setMaxRows()` is not effective on result sets materialized from cursors. (Bug #25517)
- `BIT(> 1)` is returned as `java.lang.String` from `ResultSet.getObject()` rather than `byte[]`. (Bug #25328)

Changes in MySQL Connector/J 5.0.5 (2007-03-02)

- Functionality Added or Changed
- Bugs Fixed
- Other Changes

Functionality Added or Changed

- Usage Advisor will now issue warnings for result sets with large numbers of rows. You can configure the trigger value by using the `resultSetSizeThreshold` parameter, which has a default value of 100.
- The `rewriteBatchedStatements` feature can now be used with server-side prepared statements.

- **Important change:** Due to a number of issues with the use of server-side prepared statements, Connector/J 5.0.5 has disabled their use by default. The disabling of server-side prepared statements does not affect the operation of the connector in any way.

  To enable server-side prepared statements, add the following configuration property to your connector string:

  ```
  useServerPrepStmts=true
  ```

  The default value of this property is `false` (that is, Connector/J does not use server-side prepared statements).

- Improved speed of `datetime` parsing for ResultSets that come from plain or nonserver-side prepared statements. You can enable old implementation with `useFastDateParsing=false` as a configuration parameter.

- Usage Advisor now detects empty results sets and does not report on columns not referenced in those empty sets.

- Fixed logging of XA commands sent to server, it is now configurable using `logXaCommands` property (defaults to `false`).

- Added configuration property `localSocketAddress`, which is the host name or IP address given to explicitly configure the interface that the driver will bind the client side of the TCP/IP connection to when connecting.

- We've added a new configuration option `treatUtilDateAsTimestamp`, which is `false` by default, as (1) We already had specific behavior to treat java.util.Date as a java.sql.Timestamp because it is useful to many folks, and (2) that behavior will very likely be required for drivers JDBC-post-4.0.

Bugs Fixed

- Connection property `socketFactory` wasn't exposed using correctly named mutator/accessor, causing data source implementations that use JavaBean naming conventions to set properties to fail to set the property (and in the case of SJAS, fail silently when trying to set this parameter). (Bug #26326)

- A query execution which timed out did not always throw a `MySQLTimeoutException`. (Bug #25836)

- Storing a `java.util.Date` object in a `BLOB` column would not be serialized correctly during `setObject`. (Bug #25787)

- Timer instance used for `Statement.setQueryTimeout()` created per-connection, rather than per-VM, causing memory leak. (Bug #25514)

- `EscapeProcessor` gets confused by multiple backslashes. We now push the responsibility of syntax errors back on to the server for most escape sequences. (Bug #25399)

- `INOUT` parameters in `CallableStatements` get doubly escaped. (Bug #25379)

- When using the `rewriteBatchedStatements` connection option with `PreparedState.executeBatch()` an internal memory leak would occur. (Bug #25073)

- Fixed issue where field-level for metadata from `DatabaseMetaData` when using `INFORMATION_SCHEMA` didn't have references to current connections, sometimes leading to Null Pointer Exceptions (NPEs) when introspecting them using `ResultSetMetaData`. (Bug #25073)

- `StringUtils.indexOfIgnoreCaseRespectQuotes()` isn't case-insensitive on the first character of the target. This bug also affected `rewriteBatchedStatements` functionality when prepared statements did not use uppercase for the `VALUES` clause. (Bug #25047)

- Client-side prepared statement parser gets confused by in-line comments `/*...*/` and therefore cannot rewrite batch statements or reliably detect the type of statements when they are used. (Bug #25025)
- Results sets from `UPDATE` statements that are part of multi-statement queries would cause an `SQLException` error, "Result is from UPDATE". (Bug #25009)
- Specifying `US-ASCII` as the character set in a connection to a MySQL 4.1 or newer server does not map correctly. (Bug #24840)
- Using `DatabaseMetaData.getSQLKeywords()` does not return a all of the of the reserved keywords for the current MySQL version. Current implementation returns the list of reserved words for MySQL 5.1, and does not distinguish between versions. (Bug #24794)
- Calling `Statement.cancel()` could result in a Null Pointer Exception (NPE). (Bug #24721)
- Using `setFetchSize()` breaks prepared `SHOW` and other commands. (Bug #24360)
- Calendars and timezones are now lazily instantiated when required. (Bug #24351)
- Using `DATETIME` columns would result in time shifts when `useServerPrepStmts` was true. This occurred due to different behavior when using client-side compared to server-side prepared statements and the `useJDBCCompliantTimezoneShift` option. This is now fixed if moving from server-side prepared statements to client-side prepared statements by setting `useSSPSCompatibleTimezoneShift` to `true`, as the driver can't tell if this is a new deployment that never used server-side prepared statements, or if it is an existing deployment that is switching to client-side prepared statements from server-side prepared statements. (Bug #24344)
- Connector/J now returns a better error message when server doesn't return enough information to determine stored procedure/function parameter types. (Bug #24065)
- A connection error would occur when connecting to a MySQL server with certain character sets. Some collations/character sets reported as "unknown" (specifically `cias` variants of existing character sets), and inability to override the detected server character set. (Bug #23645)
- Inconsistency between `getSchemas` and `INFORMATION_SCHEMA`. (Bug #23304)
- `DatabaseMetaData.getSchemas()` doesn't return a `TABLE_CATALOG` column. (Bug #23303)
- When using a JDBC connection URL that is malformed, the `NonRegisteringDriver.getPropertyInfo` method will throw a Null Pointer Exception (NPE). (Bug #22628)
- Some exceptions thrown out of `StandardSocketFactory` were needlessly wrapped, obscuring their true cause, especially when using socket timeouts. (Bug #21480)
- When using a server-side prepared statement the driver would send timestamps to the server using nanoseconds instead of milliseconds. (Bug #21438)
- When using server-side prepared statements and timestamp columns, value would be incorrectly populated (with nanoseconds, not microseconds). (Bug #21438)
- `ParameterMetaData` throws `NullPointerException` when prepared SQL has a syntax error. Added `generateSimpleParameterMetadata` configuration property, which when set to `true` will generate metadata reflecting `VARCHAR` for every parameter (the default is `false`, which will cause an exception to be thrown if no parameter metadata for the statement is actually available). (Bug #21267)
- The manifest file in the Connector/J JAR package was incorrect. (Bug #15641, Bug #11745464)
- Fixed an issue where `XADataSources` couldn't be bound into JNDI, as the `DataSourceFactory` didn't know how to create instances of them.

Other Changes

- Avoid static synchronized code in JVM class libraries for dealing with default timezones.
- Performance enhancement of initial character set configuration, driver will only send commands required to configure connection character set session variables if the current values on the server do not match what is required.
- Re-worked stored procedure parameter parser to be more robust. Driver no longer requires `BEGIN` in stored procedure definition, but does have requirement that if a stored function begins with a label directly after the "returns" clause, that the label is not a quoted identifier.
- Throw exceptions encountered during timeout to thread calling `Statement.execute*()`, rather than `RuntimeException`.
- Changed cached result set metadata (when using `cacheResultSetMetadata=true`) to be cached per-connection rather than per-statement as previously implemented.
- Reverted back to internal character conversion routines for single-byte character sets, as the ones internal to the JVM are using much more CPU time than our internal implementation.
- When extracting foreign key information from `SHOW CREATE TABLE` in `DatabaseMetaData`, ignore exceptions relating to tables being missing (which could happen for cross-reference or imported-key requests, as the list of tables is generated first, then iterated).
- Fixed some Null Pointer Exceptions (NPEs) when cached metadata was used with `UpdatableResultSets`.
- Take `localSocketAddress` property into account when creating instances of `CommunicationsException` when the underlying exception is a `java.net.BindException`, so that a friendlier error message is given with a little internal diagnostics.
- Fixed cases where `ServerPreparedStatements` weren't using cached metadata when `cacheResultSetMetadata=true` was used.
- Use a `java.util.TreeMap` to map column names to ordinal indexes for `ResultSet.findColumn()` instead of a HashMap. This enables us to have case-insensitive lookups (required by the JDBC specification) without resorting to the many transient object instances needed to support this requirement with a normal `HashMap` with either case-adjusted keys, or case-insensitive keys. (In the worst case scenario for lookups of a 1000 column result set, TreeMaps are about half as fast wall-clock time as a HashMap, however in normal applications their use gives many orders of magnitude reduction in transient object instance creation which pays off later for CPU usage in garbage collection).
- When using cached metadata, skip field-level metadata packets coming from the server, rather than reading them and discarding them without creating `com.mysql.jdbc.Field` instances.

Changes in MySQL Connector/J 5.0.4 (2006-10-20)

Bugs Fixed

- DBMD.getColumns() does not return expected COLUMN_SIZE for the SET type, now returns length of largest possible set disregarding whitespace or the "," delimiters to be consistent with the ODBC driver. (Bug #22613)
- Added new _ci collations to CharsetMapping - utf8_unicode_ci not working. (Bug #22456)
- Driver was using milliseconds for Statement.setQueryTimeout() when specification says argument is to be in seconds. (Bug #22359)
- Workaround for server crash when calling stored procedures using a server-side prepared statement (driver now detects prepare(stored procedure) and substitutes client-side prepared statement). (Bug #22297)

- Driver issues truncation on write exception when it shouldn't (due to sending big decimal incorrectly to server with server-side prepared statement). (Bug #22290)
- Newlines causing whitespace to span confuse procedure parser when getting parameter metadata for stored procedures. (Bug #22024)
- When using information_schema for metadata, COLUMN_SIZE for getColumns() is not clamped to range of java.lang.Integer as is the case when not using information_schema, thus leading to a truncation exception that isn't present when not using information_schema. (Bug #21544)
- Column names don't match metadata in cases where server doesn't return original column names (column functions) thus breaking compatibility with applications that expect 1-to-1 mappings between `findColumn()` and `rsmd.getColumnName()`, usually manifests itself as "Can't find column (")" exceptions. (Bug #21379)
- Driver now sends numeric 1 or 0 for client-prepared statement `setBoolean()` calls instead of '1' or '0'.
- Fixed configuration property `jdbcCompliantTruncation` was not being used for reads of result set values.
- DatabaseMetaData correctly reports `true` for `supportsCatalog*()` methods.
- Driver now supports `{call sp}` (without "()" if procedure has no arguments).

Changes in MySQL Connector/J 5.0.3 (2006-07-26, Beta)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Added configuration option `noAccessToProcedureBodies` which will cause the driver to create basic parameter metadata for `CallableStatements` when the user does not have access to procedure bodies using `SHOW CREATE PROCEDURE` or selecting from `mysql.proc` instead of throwing an exception. The default value for this option is `false`

Bugs Fixed

- Fixed `Statement.cancel()` causes `NullPointerException` if underlying connection has been closed due to server failure. (Bug #20650)
- If the connection to the server has been closed due to a server failure, then the cleanup process will call `Statement.cancel()`, triggering a `NullPointerException`, even though there is no active connection. (Bug #20650)

Changes in MySQL Connector/J 5.0.2 (2006-07-11)

Bugs Fixed

- `MysqlXaConnection.recover(int flags)` now permits combinations of `XAResource.TMSTARTRSCAN` and `TMENDRSCAN`. To simulate the “scanning” nature of the interface, we return all prepared XIDs for `TMSTARTRSCAN`, and no new XIDs for calls with `TMNOFLAGS`, or `TMENDRSCAN` when not in combination with `TMSTARTRSCAN`. This change was made for API compliance, as well as integration with IBM WebSphere's transaction manager. (Bug #20242)
- Fixed `MysqlValidConnectionChecker` for JBoss doesn't work with `MySQLXADataSources`. (Bug #20242)
- Added connection/datasource property `pinGlobalTxToPhysicalConnection` (defaults to `false`). When set to `true`, when using `XAConnections`, the driver ensures that operations on a given XID are always routed to the same physical connection. This enables the `XAConnection` to support `XA START ... JOIN` after `XA END` has been called, and is also a workaround for

transaction managers that don't maintain thread affinity for a global transaction (most either always maintain thread affinity, or have it as a configuration option). (Bug #20242)

- Better caching of character set converters (per-connection) to remove a bottleneck for multibyte character sets. (Bug #20242)
- Fixed `ConnectionProperties` (and thus some subclasses) are not serializable, even though some J2EE containers expect them to be. (Bug #19169)
- Fixed driver fails on non-ASCII platforms. The driver was assuming that the platform character set would be a superset of the MySQL `latin1` character set when doing the handshake for authentication, and when reading error messages. We now use Cp1252 for all strings sent to the server during the handshake phase, and a hard-coded mapping of the `language` system variable to the character set that is used for error messages. (Bug #18086)
- Fixed can't use `XAConnection` for local transactions when no global transaction is in progress. (Bug #17401)

Changes in MySQL Connector/J 5.0.1 (Not Released)

Not released due to a packaging error

Version 5.0.1 has no changelog entries, or they have not been published because the product version has not been released.

Changes in MySQL Connector/J 5.0.0 (2005-12-22)

Bugs Fixed

- Added support for Connector/MXJ integration using url subprotocol `jdbc:mysql:mxj://...`. (Bug #14729)
- Idle timeouts cause `XAConnections` to whine about rolling themselves back. (Bug #14729)
- When fix for Bug #14562 was merged from 3.1.12, added functionality for `CallableStatement`'s parameter metadata to return correct information for `.getParameterClassName()`. (Bug #14729)
- Added service-provider entry to `META-INF/services/java.sql.Driver` for JDBC-4.0 support. (Bug #14729)
- Fuller synchronization of `Connection` to avoid deadlocks when using multi-threaded frameworks that multi-thread a single connection (usually not recommended, but the JDBC spec permits it anyways), part of fix to Bug #14972). (Bug #14729)
- Moved all `SQLException` constructor usage to a factory in `SQLError` (ground-work for JDBC-4.0 `SQLState`-based exception classes). (Bug #14729)
- Removed Java5-specific calls to `BigDecimal` constructor (when result set value is `''`, `(int)0` was being used as an argument indirectly using method return value. This signature doesn't exist prior to Java5.) (Bug #14729)
- Implementation of `Statement.cancel()` and `Statement.setQueryTimeout()`. Both require MySQL-5.0.0 or newer server, require a separate connection to issue the `KILL QUERY` statement, and in the case of `setQueryTimeout()` creates an additional thread to handle the timeout functionality.

  Note: Failures to cancel the statement for `setQueryTimeout()` may manifest themselves as `RuntimeExceptions` rather than failing silently, as there is currently no way to unblock the thread that is executing the query being cancelled due to timeout expiration and have it throw the exception instead. (Bug #14729)
- Return "[VAR]BINARY" for `RSMD.getColumnTypeName()` when that is actually the type, and it can be distinguished (MySQL-4.1 and newer). (Bug #14729)

- Attempt detection of the MySQL type `BINARY` (it is an alias, so this isn't always reliable), and use the `java.sql.Types.BINARY` type mapping for it.
- Added unit tests for `XADatasource`, as well as friendlier exceptions for XA failures compared to the "stock" `XAException` (which has no messages).
- If the connection `useTimezone` is set to `true`, then also respect time zone conversions in escape-processed string literals (for example, `"{ts ...}"` and `"{t ...}"`).
- Do not permit `.setAutoCommit(true)`, or `.commit()` or `.rollback()` on an XA-managed connection as per the JDBC specification.
- `XADataSource` implemented (ported from 3.2 branch which won't be released as a product). Use `com.mysql.jdbc.jdbc2.optional.MysqlXADataSource` as your datasource class name in your application server to utilize XA transactions in MySQL-5.0.10 and newer.
- Moved `-bin-g.jar` file into separate `debug` subdirectory to avoid confusion.
- Return original column name for `RSMD.getColumnName()` if the column was aliased, alias name for `.getColumnLabel()` (if aliased), and original table name for `.getTableName()`. Note this only works for MySQL-4.1 and newer, as older servers don't make this information available to clients.
- Setting `useJDBCCompliantTimezoneShift=true` (it is not the default) causes the driver to use GMT for *all* `TIMESTAMP`/`DATETIME` time zones, and the current VM time zone for any other type that refers to time zones. This feature can not be used when `useTimezone=true` to convert between server and client time zones.
- `PreparedStatement.setString()` didn't work correctly when `sql_mode` on server contained `NO_BACKSLASH_ESCAPES` and no characters that needed escaping were present in the string.
- Add one level of indirection of internal representation of `CallableStatement` parameter metadata to avoid class not found issues on JDK-1.3 for `ParameterMetadata` interface (which doesn't exist prior to JDBC-3.0).

Changes in MySQL Connector/J 3.1

Changes in MySQL Connector/J 3.1.15 (Not released)

Important change: Due to a number of issues with the use of server-side prepared statements, Connector/J 5.0.5 has disabled their use by default. The disabling of server-side prepared statements does not affect the operation of the connector in any way.

To enable server-side prepared statements you must add the following configuration property to your connector string:

```
useServerPrepStmts=true
```

The default value of this property is `false` (that is, Connector/J does not use server-side prepared statements).

Bugs Fixed

- Specifying `US-ASCII` as the character set in a connection to a MySQL 4.1 or newer server does not map correctly. (Bug #24840)
- The manifest file in the Connector/J JAR package was incorrect. (Bug #15641, Bug #11745464)

Changes in MySQL Connector/J 3.1.14 (2006-10-19)

Bugs Fixed

- Check and store value for continueBatchOnError property in constructor of Statements, rather than when executing batches, so that Connections closed out from underneath statements don't cause NullPointerExceptions when it is required to check this property. (Bug #22290)
- Driver now sends numeric 1 or 0 for client-prepared statement setBoolean() calls instead of '1' or '0'. (Bug #22290)
- Fixed bug where driver would not advance to next host if roundRobinLoadBalance=true and the last host in the list is down. (Bug #22290)
- Driver issues truncation on write exception when it shouldn't (due to sending big decimal incorrectly to server with server-side prepared statement). (Bug #22290)
- Fixed bug when calling stored functions, where parameters weren't numbered correctly (first parameter is now the return value, subsequent parameters if specified start at index "2"). (Bug #22290)
- Removed logger autodetection altogether, must now specify logger explicitly if you want to use a logger other than one that logs to STDERR. (Bug #21207)
- DDriver throws NPE when tracing prepared statements that have been closed (in asSQL()). (Bug #21207)
- ResultSet.getSomeInteger() doesn't work for BIT(>1). (Bug #21062)
- Escape of quotation marks in client-side prepared statements parsing not respected. Patch covers more than bug report, including NO_BACKSLASH_ESCAPES being set, and stacked quote characters forms of escaping (that is, '' or ""). (Bug #20888)
- Fixed can't pool server-side prepared statements, exception raised when re-using them. (Bug #20687)
- Fixed Updatable result set that contains a BIT column fails when server-side prepared statements are used. (Bug #20485)
- Fixed updatable result set throws ClassCastException when there is row data and moveToInsertRow() is called. (Bug #20479)
- Fixed ResultSet.getShort() for UNSIGNED TINYINT returns incorrect values when using server-side prepared statements. (Bug #20306)
- ReplicationDriver does not always round-robin load balance depending on URL used for slaves list. (Bug #19993)
- Fixed calling toString() on ResultSetMetaData for driver-generated (that is, from DatabaseMetaData method calls, or from getGeneratedKeys()) result sets would raise a NullPointerException. (Bug #19993)
- Connection fails to localhost when using timeout and IPv6 is configured. (Bug #19726)
- ResultSet.getFloatFromString() can't retrieve values near Float.MIN/MAX_VALUE. (Bug #18880)
- `DatabaseMetaData.getTables()` or `getColumns()` with a bad catalog parameter threw an exception rather than return an empty result set (as required by the specification). (Bug #18258)
- Fixed memory leak with profileSQL=true. (Bug #16987)
- Fixed NullPointerException in MysqlDataSourceFactory due to Reference containing RefAddrs with null content. (Bug #16791)

Changes in MySQL Connector/J 3.1.13 (2006-05-26)

Bugs Fixed

- `Fixed PreparedStatement.setObject(int, Object, int)` doesn't respect scale of BigDecimals. (Bug #19615)
- Fixed `ResultSet.wasNull()` returns incorrect value when extracting native string from server-side prepared statement generated result set. (Bug #19282)
- Fixed invalid classname returned for `ResultSetMetaData.getColumnClassName()` for `BIGINT type`. (Bug #19282)
- Fixed case where driver wasn't reading server status correctly when fetching server-side prepared statement rows, which in some cases could cause warning counts to be off, or multiple result sets to not be read off the wire. (Bug #19282)
- Fixed data truncation and `getWarnings()` only returns last warning in set. (Bug #18740)
- Fixed aliased column names where length of name > 251 are corrupted. (Bug #18554)
- Improved performance of retrieving `BigDecimal`, `Time`, `Timestamp` and `Date` values from server-side prepared statements by creating fewer short-lived instances of `Strings` when the native type is not an exact match for the requested type. (Bug #18496)
- Added performance feature, re-writing of batched executes for `Statement.executeBatch()` (for all DML statements) and `PreparedStatement.executeBatch()` (for INSERTs with VALUE clauses only). Enable by using "rewriteBatchedStatements=true" in your JDBC URL. (Bug #18041)
- Fixed issue where server-side prepared statements don't cause truncation exceptions to be thrown when truncation happens. (Bug #18041)
- Fixed `CallableStatement.registerOutParameter()` not working when some parameters pre-populated. Still waiting for feedback from JDBC experts group to determine what correct parameter count from `getMetaData()` should be, however. (Bug #17898)
- Fixed calling `clearParameters()` on a closed prepared statement causes NPE. (Bug #17587)
- Map "latin1" on MySQL server to CP1252 for MySQL > 4.1.0. (Bug #17587)
- Added additional accessor and mutator methods on ConnectionProperties so that DataSource users can use same naming as regular URL properties. (Bug #17587)
- Fixed `ResultSet.wasNull()` not always reset correctly for booleans when done using conversion for server-side prepared statements. (Bug #17450)
- Fixed `Statement.getGeneratedKeys()` throws `NullPointerException` when no query has been processed. (Bug #17099)
- Fixed updatable result set doesn't return `AUTO_INCREMENT` values for `insertRow()` when multiple column primary keys are used. (the driver was checking for the existence of single-column primary keys and an autoincrement value > 0 instead of a straightforward `isAutoIncrement()` check). (Bug #16841)
- `DBMD.getColumns()` returns wrong type for `BIT`. (Bug #15854)
- `lib-nodist` directory missing from package breaks out-of-box build. (Bug #15676)
- Fixed issue with `ReplicationConnection` incorrectly copying state, doesn't transfer connection context correctly when transitioning between the same read-only states. (Bug #15570)
- No "dos" character set in MySQL > 4.1.0. (Bug #15544)
- `INOUT` parameter does not store `IN` value. (Bug #15464)
- `PreparedStatement.setObject()` serializes `BigInteger` as object, rather than sending as numeric value (and is thus not complementary to `.getObject()` on an `UNSIGNED LONG` type). (Bug #15383)

- Fixed issue where driver was unable to initialize character set mapping tables. Removed reliance on `.properties` files to hold this information, as it turns out to be too problematic to code around class loader hierarchies that change depending on how an application is deployed. Moved information back into the `CharsetMapping` class. (Bug #14938)
- Exception thrown for new decimal type when using updatable result sets. (Bug #14609)
- Driver now aware of fix for `BIT` type metadata that went into MySQL-5.0.21 for server not reporting length consistently . (Bug #13601)
- Added support for Apache Commons logging, use "com.mysql.jdbc.log.CommonsLogger" as the value for the "logger" configuration property. (Bug #13469)
- Fixed driver trying to call methods that don't exist on older and newer versions of Log4j. The fix is not trying to auto-detect presence of log4j, too many different incompatible versions out there in the wild to do this reliably.

  If you relied on autodetection before, you will need to add "logger=com.mysql.jdbc.log.Log4JLogger" to your JDBC URL to enable Log4J usage, or alternatively use the new "CommonsLogger" class to take care of this. (Bug #13469)
- LogFactory now prepends `com.mysql.jdbc.log` to the log class name if it cannot be found as specified. This enables you to use “short names” for the built-in log factories, for example, `logger=CommonsLogger` instead of `logger=com.mysql.jdbc.log.CommonsLogger`. (Bug #13469)
- `ResultSet.getShort()` for `UNSIGNED TINYINT` returned wrong values. (Bug #11874)

Changes in MySQL Connector/J 3.1.12 (2005-11-30)

Bugs Fixed

- Process escape tokens in `Connection.prepareStatement(...)`. You can disable this behavior by setting the JDBC URL configuration property `processEscapeCodesForPrepStmts` to `false`. (Bug #15141)
- Usage advisor complains about unreferenced columns, even though they've been referenced. (Bug #15065)
- Driver incorrectly closes streams passed as arguments to `PreparedStatements`. Reverts to legacy behavior by setting the JDBC configuration property `autoClosePStmtStreams` to `true` (also included in the 3-0-Compat configuration “bundle”). (Bug #15024)
- Deadlock while closing server-side prepared statements from multiple threads sharing one connection. (Bug #14972)
- Unable to initialize character set mapping tables (due to J2EE classloader differences). (Bug #14938)
- Escape processor replaces quote character in quoted string with string delimiter. (Bug #14909)
- `DatabaseMetaData.getColumns()` doesn't return `TABLE_NAME` correctly. (Bug #14815)
- `storesMixedCaseIdentifiers()` returns `false` (Bug #14562)
- `storesLowerCaseIdentifiers()` returns `true` (Bug #14562)
- `storesMixedCaseQuotedIdentifiers()` returns `false` (Bug #14562)
- `storesMixedCaseQuotedIdentifiers()` returns `true` (Bug #14562)
- If `lower_case_table_names=0` (on server):

    - `storesLowerCaseIdentifiers()` returns `false`
    - `storesLowerCaseQuotedIdentifiers()` returns `false`
    - `storesMixedCaseIdentifiers()` returns `true`
    - `storesMixedCaseQuotedIdentifiers()` returns `true`
    - `storesUpperCaseIdentifiers()` returns `false`
    - `storesUpperCaseQuotedIdentifiers()` returns `true`

    (Bug #14562)
- `storesUpperCaseIdentifiers()` returns `false` (Bug #14562)
- `storesUpperCaseQuotedIdentifiers()` returns `true` (Bug #14562)
- If `lower_case_table_names=1` (on server):
    - `storesLowerCaseIdentifiers()` returns `true`
    - `storesLowerCaseQuotedIdentifiers()` returns `true`
    - `storesMixedCaseIdentifiers()` returns `false`
    - `storesMixedCaseQuotedIdentifiers()` returns `false`
    - `storesUpperCaseIdentifiers()` returns `false`
    - `storesUpperCaseQuotedIdentifiers()` returns `true`

    (Bug #14562)
- `storesLowerCaseQuotedIdentifiers()` returns `true` (Bug #14562)
- Fixed `DatabaseMetaData.stores*Identifiers()`:
    - If `lower_case_table_names=0` (on server):
        - `storesLowerCaseIdentifiers()` returns `false`
        - `storesLowerCaseQuotedIdentifiers()` returns `false`
        - `storesMixedCaseIdentifiers()` returns `true`
        - `storesMixedCaseQuotedIdentifiers()` returns `true`
        - `storesUpperCaseIdentifiers()` returns `false`
        - `storesUpperCaseQuotedIdentifiers()` returns `true`
    - If `lower_case_table_names=1` (on server):
        - `storesLowerCaseIdentifiers()` returns `true`
        - `storesLowerCaseQuotedIdentifiers()` returns `true`
        - `storesMixedCaseIdentifiers()` returns `false`
        - `storesMixedCaseQuotedIdentifiers()` returns `false`
        - `storesUpperCaseIdentifiers()` returns `false`

    - `storesUpperCaseQuotedIdentifiers()` returns `true`

  (Bug #14562)

- `storesMixedCaseIdentifiers()` returns `true` (Bug #14562)
- `storesLowerCaseQuotedIdentifiers()` returns `false` (Bug #14562)
- Java type conversion may be incorrect for `MEDIUMINT`. (Bug #14562)
- `storesLowerCaseIdentifiers()` returns `false` (Bug #14562)
- Added configuration property `useGmtMillisForDatetimes` which when set to `true` causes `ResultSet.getDate()`, `.getTimestamp()` to return correct millis-since GMT when `.getTime()` is called on the return value (currently default is `false` for legacy behavior). (Bug #14562)
- Extraneous sleep on `autoReconnect`. (Bug #13775)
- Reconnect during middle of `executeBatch()` should not occur if `autoReconnect` is enabled. (Bug #13255)
- `maxQuerySizeToLog` is not respected. Added logging of bound values for `execute()` phase of server-side prepared statements when `profileSQL=true` as well. (Bug #13048)
- OpenOffice expects `DBMD.supportsIntegrityEnhancementFacility()` to return `true` if foreign keys are supported by the datasource, even though this method also covers support for check constraints, which MySQL *doesn't* have. Setting the configuration property `overrideSupportsIntegrityEnhancementFacility` to `true` causes the driver to return `true` for this method. (Bug #12975)
- Added `com.mysql.jdbc.testsuite.url.default` system property to set default JDBC url for testsuite (to speed up bug resolution when I'm working in Eclipse). (Bug #12975)
- `logSlowQueries` should give better info. (Bug #12230)
- Don't increase timeout for failover/reconnect. (Bug #6577)
- Fixed client-side prepared statement bug with embedded `?` characters inside quoted identifiers (it was recognized as a placeholder, when it was not).
- Do not permit `executeBatch()` for `CallableStatements` with registered `OUT`/`INOUT` parameters (JDBC compliance).
- Fall back to platform-encoding for `URLDecoder.decode()` when parsing driver URL properties if the platform doesn't have a two-argument version of this method.

Changes in MySQL Connector/J 3.1.11 (2005-10-07)

Bugs Fixed

- The configuration property `sessionVariables` now permits you to specify variables that start with the "`@`" sign. (Bug #13453)
- URL configuration parameters do not permit "`&`" or "`=`" in their values. The JDBC driver now parses configuration parameters as if they are encoded using the application/x-www-form-urlencoded format as specified by `java.net.URLDecoder` (http://java.sun.com/j2se/1.5.0/docs/api/java/net/URLDecoder.html).

  If the "`%`" character is present in a configuration property, it must now be represented as `%25`, which is the encoded form of "`%`" when using application/x-www-form-urlencoded encoding. (Bug #13453)

- Workaround for Bug #13374: `ResultSet.getStatement()` on closed result set returns `NULL` (as per JDBC 4.0 spec, but not backward-compatible). Set the connection property `retainStatementAfterResultSetClose` to `true` to be able to retrieve a `ResultSet`'s statement after the `ResultSet` has been closed using `.getStatement()` (the default is `false`, to be JDBC-compliant and to reduce the chance that code using JDBC leaks `Statement` instances). (Bug #13277)
- `ResultSetMetaData` from `Statement.getGeneratedKeys()` caused a `NullPointerException` to be thrown whenever a method that required a connection reference was called. (Bug #13277)
- Backport of `VAR[BINARY|CHAR] [BINARY]` types detection from 5.0 branch. (Bug #13277)
- Fixed `NullPointerException` when converting `catalog` parameter in many `DatabaseMetaDataMethods` to `byte[]`s (for the result set) when the parameter is `null`. (`null` is not technically permitted by the JDBC specification, but we have historically permitted it). (Bug #13277)
- Backport of `Field` class, `ResultSetMetaData.getColumnClassName()`, and `ResultSet.getObject(int)` changes from 5.0 branch to fix behavior surrounding `VARCHAR BINARY`/`VARBINARY` and related types. (Bug #13277)
- Read response in `MysqlIO.sendFileToServer()`, even if the local file can't be opened, otherwise next query issued will fail, because it is reading the response to the empty `LOAD DATA INFILE` packet sent to the server. (Bug #13277)
- When `gatherPerfMetrics` is enabled for servers older than 4.1.0, a `NullPointerException` is thrown from the constructor of `ResultSet` if the query doesn't use any tables. (Bug #13043)
- `java.sql.Types.OTHER` returned for `BINARY` and `VARBINARY` columns when using `DatabaseMetaData.getColumns()`. (Bug #12970)
- `ServerPreparedStatement.getBinding()` now checks if the statement is closed before attempting to reference the list of parameter bindings, to avoid throwing a `NullPointerException`. (Bug #12970)
- Tokenizer for `=` in URL properties was causing `sessionVariables=...` to be parameterized incorrectly. (Bug #12753)
- `cp1251` incorrectly mapped to `win1251` for servers newer than 4.0.x. (Bug #12752)
- `getExportedKeys()` (Bug #12541)
- Specifying a catalog works as stated in the API docs. (Bug #12541)
- Specifying `NULL` means that catalog will not be used to filter the results (thus all databases will be searched), unless you've set `nullCatalogMeansCurrent=true` in your JDBC URL properties. (Bug #12541)
- `getIndexInfo()` (Bug #12541)
- `getProcedures()` (and thus indirectly `getProcedureColumns()`) (Bug #12541)
- `getImportedKeys()` (Bug #12541)
- Specifying `""` means “current” catalog, even though this isn't quite JDBC spec compliant, it is there for legacy users. (Bug #12541)
- `getCrossReference()` (Bug #12541)
- Added `Connection.isMasterConnection()` for clients to be able to determine if a multi-host master/slave connection is connected to the first host in the list. (Bug #12541)

- `getColumns()` (Bug #12541)
- Handling of catalog argument in `DatabaseMetaData.getIndexInfo()`, which also means changes to the following methods in `DatabaseMetaData`:
  - `getBestRowIdentifier()`
  - `getColumns()`
  - `getCrossReference()`
  - `getExportedKeys()`
  - `getImportedKeys()`
  - `getIndexInfo()`
  - `getPrimaryKeys()`
  - `getProcedures()` (and thus indirectly `getProcedureColumns()`)
  - `getTables()`

  The `catalog` argument in all of these methods now behaves in the following way:
  - Specifying `NULL` means that catalog will not be used to filter the results (thus all databases will be searched), unless you've set `nullCatalogMeansCurrent=true` in your JDBC URL properties.
  - Specifying `""` means “current” catalog, even though this isn't quite JDBC spec compliant, it is there for legacy users.
  - Specifying a catalog works as stated in the API docs.
  - Made `Connection.clientPrepare()` available from “wrapped” connections in the `jdbc2.optional` package (connections built by `ConnectionPoolDataSource` instances).

  (Bug #12541)
- `getBestRowIdentifier()` (Bug #12541)
- Made `Connection.clientPrepare()` available from “wrapped” connections in the `jdbc2.optional` package (connections built by `ConnectionPoolDataSource` instances). (Bug #12541)
- `getTables()` (Bug #12541)
- `getPrimaryKeys()` (Bug #12541)
- `Connection.prepareCall()` is database name case-sensitive (on Windows systems). (Bug #12417)
- `explainSlowQueries` hangs with server-side prepared statements. (Bug #12229)
- Properties shared between master and slave with replication connection. (Bug #12218)
- Geometry types not handled with server-side prepared statements. (Bug #12104)
- `maxPerformance.properties` mis-spells “elideSetAutoCommits”. (Bug #11976)
- `ReplicationConnection` won't switch to slave, throws “Catalog can't be null” exception. (Bug #11879)
- `Pstmt.setObject(...., Types.BOOLEAN)` throws exception. (Bug #11798)

- Escape tokenizer doesn't respect stacked single quotation marks for escapes. (Bug #11797)
- `GEOMETRY` type not recognized when using server-side prepared statements. (Bug #11797)
- Foreign key information that is quoted is parsed incorrectly when `DatabaseMetaData` methods use that information. (Bug #11781)
- The `sendBlobChunkSize` property is now clamped to `max_allowed_packet` with consideration of stream buffer size and packet headers to avoid `PacketTooBigExceptions` when `max_allowed_packet` is similar in size to the default `sendBlobChunkSize` which is 1M. (Bug #11781)
- `CallableStatement.clearParameters()` now clears resources associated with `INOUT`/`OUTPUT` parameters as well as `INPUT` parameters. (Bug #11781)
- Fixed regression caused by fix for Bug #11552 that caused driver to return incorrect values for unsigned integers when those integers where within the range of the positive signed type. (Bug #11663)
- Moved source code to Subversion repository. (Bug #11663)
- Incorrect generation of testcase scripts for server-side prepared statements. (Bug #11663)
- Fixed statements generated for testcases missing `;` for “plain” statements. (Bug #11629)
- Spurious `!` on console when character encoding is `utf8`. (Bug #11629)
- `StringUtils.getBytes()` doesn't work when using multibyte character encodings and a length in *characters* is specified. (Bug #11614)
- `DBMD.storesLower/Mixed/UpperIdentifiers()` reports incorrect values for servers deployed on Windows. (Bug #11575)
- Reworked `Field` class, `*Buffer`, and `MysqlIO` to be aware of field lengths > `Integer.MAX_VALUE`. (Bug #11498)
- Escape processor didn't honor strings demarcated with double quotation marks. (Bug #11498)
- Updated `DBMD.supportsCorrelatedQueries()` to return `true` for versions > 4.1, `supportsGroupByUnrelated()` to return `true` and `getResultSetHoldability()` to return `HOLD_CURSORS_OVER_COMMIT`. (Bug #11498)
- Lifted restriction of changing streaming parameters with server-side prepared statements. As long as `all` streaming parameters were set before execution, `.clearParameters()` does not have to be called. (due to limitation of client/server protocol, prepared statements can not reset *individual* stream data on the server side). (Bug #11498)
- `ResultSet.moveToCurrentRow()` fails to work when preceded by a call to `ResultSet.moveToInsertRow()`. (Bug #11190)
- `VARBINARY` data corrupted when using server-side prepared statements and `.setBytes()`. (Bug #11115)
- `Statement.getWarnings()` fails with NPE if statement has been closed. (Bug #10630)
- Only get `char[]` from SQL in `PreparedStatement.ParseInfo()` when needed. (Bug #10630)

Changes in MySQL Connector/J 3.1.10 (2005-06-23)

Bugs Fixed

- Initial implemention of `ParameterMetadata` for `PreparedStatement.getParameterMetadata()`. Only works fully for `CallableStatements`, as current server-side prepared statements return every parameter as a `VARCHAR` type.

- Fixed connecting without a database specified raised an exception in `MysqlIO.changeDatabaseTo()`.

Changes in MySQL Connector/J 3.1.9 (2005-06-22)

Bugs Fixed

- Production package doesn't include JBoss integration classes. (Bug #11411)
- Removed nonsensical “costly type conversion” warnings when using usage advisor. (Bug #11411)
- Fixed `PreparedStatement.setClob()` not accepting `null` as a parameter. (Bug #11360)
- Connector/J dumping query into `SQLException` twice. (Bug #11360)
- `autoReconnect` ping causes exception on connection startup. (Bug #11259)
- `Connection.setCatalog()` is now aware of the `useLocalSessionState` configuration property, which when set to `true` will prevent the driver from sending `USE ...` to the server if the requested catalog is the same as the current catalog. (Bug #11115)
- `3-0-Compat`: Compatibility with Connector/J 3.0.x functionality (Bug #11115)
- `maxPerformance`: Maximum performance without being reckless (Bug #11115)
- `solarisMaxPerformance`: Maximum performance for Solaris, avoids syscalls where it can (Bug #11115)
- Added `maintainTimeStats` configuration property (defaults to `true`), which tells the driver whether or not to keep track of the last query time and the last successful packet sent to the server's time. If set to `false`, removes two syscalls per query. (Bug #11115)
- `VARBINARY` data corrupted when using server-side prepared statements and `ResultSet.getBytes()`. (Bug #11115)
- Added the following configuration bundles, use one or many using the `useConfigs` configuration property:
  - `maxPerformance`: Maximum performance without being reckless
  - `solarisMaxPerformance`: Maximum performance for Solaris, avoids syscalls where it can
  - `3-0-Compat`: Compatibility with Connector/J 3.0.x functionality

  (Bug #11115)
- Try to handle `OutOfMemoryErrors` more gracefully. Although not much can be done, they will in most cases close the connection they happened on so that further operations don't run into a connection in some unknown state. When an OOM has happened, any further operations on the connection will fail with a “Connection closed” exception that will also list the OOM exception as the reason for the implicit connection close event. (Bug #10850)
- Setting `cachePrepStmts=true` now causes the `Connection` to also cache the check the driver performs to determine if a prepared statement can be server-side or not, as well as caches server-side prepared statements for the lifetime of a connection. As before, the `prepStmtCacheSize` parameter controls the size of these caches. (Bug #10850)
- Don't send `COM_RESET_STMT` for each execution of a server-side prepared statement if it isn't required. (Bug #10850)
- 0-length streams not sent to server when using server-side prepared statements. (Bug #10850)
- Driver detects if you're running MySQL-5.0.7 or later, and does not scan for `LIMIT ?[,?]` in statements being prepared, as the server supports those types of queries now. (Bug #10850)

- Reorganized directory layout. Sources now are in `src` folder. Don't pollute parent directory when building, now output goes to `./build`, distribution goes to `./dist`. (Bug #10496)
- Added support/bug hunting feature that generates `.sql` test scripts to `STDERR` when `autoGenerateTestcaseScript` is set to `true`. (Bug #10496)
- `SQLException` is thrown when using property `characterSetResults` with `cp932` or `eucjpms`. (Bug #10496)
- The data type returned for `TINYINT(1)` columns when `tinyInt1isBit=true` (the default) can be switched between `Types.BOOLEAN` and `Types.BIT` using the new configuration property `transformedBitIsBoolean`, which defaults to `false`. If set to `false` (the default), `DatabaseMetaData.getColumns()` and `ResultSetMetaData.getColumnType()` will return `Types.BOOLEAN` for `TINYINT(1)` columns. If `true`, `Types.BOOLEAN` will be returned instead. Regardless of this configuration property, if `tinyInt1isBit` is enabled, columns with the type `TINYINT(1)` will be returned as `java.lang.Boolean` instances from `ResultSet.getObject(...)`, and `ResultSetMetaData.getColumnClassName()` will return `java.lang.Boolean`. (Bug #10485)
- `SQLException` thrown when retrieving `YEAR(2)` with `ResultSet.getString()`. The driver will now always treat `YEAR` types as `java.sql.Dates` and return the correct values for `getString()`. Alternatively, the `yearIsDateType` connection property can be set to `false` and the values will be treated as `SHORT`s. (Bug #10485)
- Driver doesn't support `{?=CALL(...)}` for calling stored functions. This involved adding support for function retrieval to `DatabaseMetaData.getProcedures()` and `getProcedureColumns()` as well. (Bug #10310)
- Unsigned `SMALLINT` treated as signed for `ResultSet.getInt()`, fixed all cases for `UNSIGNED` integer values and server-side prepared statements, as well as `ResultSet.getObject()` for `UNSIGNED TINYINT`. (Bug #10156)
- Made `ServerPreparedStatement.asSql()` work correctly so auto-explain functionality would work with server-side prepared statements. (Bug #10155)
- Double quotation marks not recognized when parsing client-side prepared statements. (Bug #10155)
- Made JDBC2-compliant wrappers public to enable access to vendor extensions. (Bug #10155)
- `DatabaseMetaData.supportsMultipleOpenResults()` now returns `true`. The driver has supported this for some time, DBMD just missed that fact. (Bug #10155)
- Cleaned up logging of profiler events, moved code to dump a profiler event as a string to `com.mysql.jdbc.log.LogUtils` so that third parties can use it. (Bug #10155)
- Made `enableStreamingResults()` visible on `com.mysql.jdbc.jdbc2.optional.StatementWrapper`. (Bug #10155)
- Actually write manifest file to correct place so it ends up in the binary jar file. (Bug #10144)
- Added `createDatabaseIfNotExist` property (default is `false`), which will cause the driver to ask the server to create the database specified in the URL if it doesn't exist. You must have the appropriate privileges for database creation for this to work. (Bug #10144)
- Memory leak in `ServerPreparedStatement` if `serverPrepare()` fails. (Bug #10144)
- `com.mysql.jdbc.PreparedStatement.ParseInfo` does unnecessary call to `toCharArray()`. (Bug #9064)
- Driver now correctly uses CP932 if available on the server for Windows-31J, CP932 and MS932 java encoding names, otherwise it resorts to SJIS, which is only a close approximation. Currently

only MySQL-5.0.3 and newer (and MySQL-4.1.12 or .13, depending on when the character set gets backported) can reliably support any variant of CP932.

- Overhaul of character set configuration, everything now lives in a properties file.

Changes in MySQL Connector/J 3.1.8 (2005-04-14)

Bugs Fixed

- Should accept `null` for catalog (meaning use current) in DBMD methods, even though it is not JDBC-compliant for legacy's sake. Disable by setting connection property `nullCatalogMeansCurrent` to `false` (which will be the default value in C/J 3.2.x). (Bug #9917)
- Fixed driver not returning `true` for `-1` when `ResultSet.getBoolean()` was called on result sets returned from server-side prepared statements. (Bug #9778)
- Added a `Manifest.MF` file with implementation information to the `.jar` file. (Bug #9778)
- More tests in `Field.isOpaqueBinary()` to distinguish opaque binary (that is, fields with type `CHAR(n)` and `CHARACTER SET BINARY`) from output of various scalar and aggregate functions that return strings. (Bug #9778)
- `DBMD.getTables()` shouldn't return tables if views are asked for, even if the database version doesn't support views. (Bug #9778)
- Should accept `null` for name patterns in DBMD (meaning "%"), even though it isn't JDBC compliant, for legacy's sake. Disable by setting connection property `nullNamePatternMatchesAll` to `false` (which will be the default value in C/J 3.2.x). (Bug #9769)
- The performance metrics feature now gathers information about number of tables referenced in a SELECT. (Bug #9704)
- The logging system is now automatically configured. If the value has been set by the user, using the URL property `logger` or the system property `com.mysql.jdbc.logger`, then use that, otherwise, autodetect it using the following steps:

  1. Log4j, if it is available,
  2. Then JDK1.4 logging,
  3. Then fallback to our `STDERR` logging.

  (Bug #9704)
- `Statement.getMoreResults()` could throw NPE when existing result set was `.close()`d. (Bug #9704)
- Stored procedures with `DECIMAL` parameters with storage specifications that contained "," in them failed. (Bug #9682)
- `PreparedStatement.setObject(int, Object, int type, int scale)` now uses scale value for `BigDecimal` instances. (Bug #9682)
- Added support for the c3p0 connection pool's (http://c3p0.sf.net/) validation/connection checker interface which uses the lightweight `COM_PING` call to the server if available. To use it, configure your c3p0 connection pool's `connectionTesterClassName` property to use `com.mysql.jdbc.integration.c3p0.MysqlConnectionTester`. (Bug #9320)
- `PreparedStatement.getMetaData()` inserts blank row in database under certain conditions when not using server-side prepared statements. (Bug #9320)
- Better detection of `LIMIT` inside/outside of quoted strings so that the driver can more correctly determine whether a prepared statement can be prepared on the server or not. (Bug #9320)

- `Connection.canHandleAsPreparedStatement()` now makes “best effort” to distinguish `LIMIT` clauses with placeholders in them from ones without to have fewer false positives when generating work-arounds for statements the server cannot currently handle as server-side prepared statements. (Bug #9320)
- Fixed `build.xml` to not compile `log4j` logging if `log4j` not available. (Bug #9320)
- Added finalizers to `ResultSet` and `Statement` implementations to be JDBC spec-compliant, which requires that if not explicitly closed, these resources should be closed upon garbage collection. (Bug #9319)
- Stored procedures with same name in different databases confuse the driver when it tries to determine parameter counts/types. (Bug #9319)
- A continuation of Bug #8868, where functions used in queries that should return nonstring types when resolved by temporary tables suddenly become opaque binary strings (work-around for server limitation). Also fixed fields with type of `CHAR(n) CHARACTER SET BINARY` to return correct/matching classes for `RSMD.getColumnClassName()` and `ResultSet.getObject()`. (Bug #9236)
- Cannot use `UTF-8` for characterSetResults configuration property. (Bug #9206)
- `PreparedStatement.addBatch()` doesn't work with server-side prepared statements and streaming `BINARY` data. (Bug #9040)
- `ServerPreparedStatements` now correctly “stream” `BLOB`/`CLOB` data to the server. You can configure the threshold chunk size using the JDBC URL property `blobSendChunkSize` (the default is 1MB). (Bug #8868)
- `DATE_FORMAT()` queries returned as `BLOB`s from `getObject()`. (Bug #8868)
- Server-side session variables can be preset at connection time by passing them as a comma-delimited list for the connection property `sessionVariables`. (Bug #8868)
- `BlobFromLocator` now uses correct identifier quoting when generating prepared statements. (Bug #8868)
- Fixed regression in `ping()` for users using `autoReconnect=true`. (Bug #8868)
- Check for empty strings (`''`) when converting `CHAR`/`VARCHAR` column data to numbers, throw exception if `emptyStringsConvertToZero` configuration property is set to `false` (for backward-compatibility with 3.0, it is now set to `true` by default, but will most likely default to `false` in 3.2). (Bug #8803)
- `DATA_TYPE` column from `DBMD.getBestRowIdentifier()` causes `ArrayIndexOutOfBoundsException` when accessed (and in fact, didn't return any value). (Bug #8803)
- `DBMD.supportsMixedCase*Identifiers()` returns wrong value on servers running on case-sensitive file systems. (Bug #8800)
- `DBMD.supportsResultSetConcurrency()` not returning `true` for forward-only/read-only result sets (we obviously support this). (Bug #8792)
- Fixed `ResultSet.getTime()` on a `NULL` value for server-side prepared statements throws NPE.
- Made `Connection.ping()` a public method.
- Added support for new precision-math `DECIMAL` type in MySQL 5.0.3 and up.
- Fixed `DatabaseMetaData.getTables()` returning views when they were not asked for as one of the requested table types.

Changes in MySQL Connector/J 3.1.7 (2005-02-18)

Bugs Fixed

- `PreparedStatements` not creating streaming result sets. (Bug #8487)
- Don't pass `NULL` to `String.valueOf()` in `ResultSet.getNativeConvertToString()`, as it stringifies it (that is, returns `null`), which is not correct for the method in question. (Bug #8487)
- Fixed NPE in `ResultSet.realClose()` when using usage advisor and result set was already closed. (Bug #8428)
- `ResultSet.getString()` doesn't maintain format stored on server, bug fix only enabled when `noDatetimeStringSync` property is set to `true` (the default is `false`). (Bug #8428)
- Added support for `BIT` type in MySQL-5.0.3. The driver will treat `BIT(1-8)` as the JDBC standard `BIT` type (which maps to `java.lang.Boolean`), as the server does not currently send enough information to determine the size of a bitfield when < 9 bits are declared. `BIT(>9)` will be treated as `VARBINARY`, and will return `byte[]` when `getObject()` is called. (Bug #8424)
- Added `useLocalSessionState` configuration property, when set to `true` the JDBC driver trusts that the application is well-behaved and only sets autocommit and transaction isolation levels using the methods provided on `java.sql.Connection`, and therefore can manipulate these values in many cases without incurring round-trips to the database server. (Bug #8424)
- Added `enableStreamingResults()` to `Statement` for connection pool implementations that check `Statement.setFetchSize()` for specification-compliant values. Call `Statement.setFetchSize(>=0)` to disable the streaming results for that statement. (Bug #8424)
- `ResultSet.getBigDecimal()` throws exception when rounding would need to occur to set scale. The driver now chooses a rounding mode of “half up” if nonrounding `BigDecimal.setScale()` fails. (Bug #8424)
- Fixed synchronization issue with `ServerPreparedStatement.serverPrepare()` that could cause deadlocks/crashes if connection was shared between threads. (Bug #8096)
- Emulated locators corrupt binary data when using server-side prepared statements. (Bug #8096)
- Infinite recursion when “falling back” to master in failover configuration. (Bug #7952)
- Disable multi-statements (if enabled) for MySQL-4.1 versions prior to version 4.1.10 if the query cache is enabled, as the server returns wrong results in this configuration. (Bug #7952)
- Removed `dontUnpackBinaryResults` functionality, the driver now always stores results from server-side prepared statements as is from the server and unpacks them on demand. (Bug #7952)
- Fixed duplicated code in `configureClientCharset()` that prevented `useOldUTF8Behavior=true` from working properly. (Bug #7952)
- Added `holdResultsOpenOverStatementClose` property (default is `false`), that keeps result sets open over statement.close() or new execution on same statement (suggested by Kevin Burton). (Bug #7715)
- Detect new `sql_mode` variable in string form (it used to be integer) and adjust quoting method for strings appropriately. (Bug #7715)
- Timestamps converted incorrectly to strings with server-side prepared statements and updatable result sets. (Bug #7715)
- Timestamp key column data needed `_binary` stripped for `UpdatableResultSet.refreshRow()`. (Bug #7686)

- Connector/J now supports the `NO_BACKSLASH_ESCAPES` server SQL mode. (Bug #7374, Bug #16730296)
- Choose correct “direction” to apply time adjustments when both client and server are in GMT time zone when using `ResultSet.get(..., cal)` and `PreparedStatement.set(...., cal)`. (Bug #4718)
- Remove `_binary` introducer from parameters used as in/out parameters in `CallableStatement`. (Bug #4718)
- Always return `byte[]`s for output parameters registered as `*BINARY`. (Bug #4718)
- By default, the driver now scans SQL you are preparing using all variants of `Connection.prepareStatement()` to determine if it is a supported type of statement to prepare on the server side, and if it is not supported by the server, it instead prepares it as a client-side emulated prepared statement. You can disable this by passing `emulateUnsupportedPstmts=false` in your JDBC URL. (Bug #4718)
- Added `dontTrackOpenResources` option (default is `false`, to be JDBC compliant), which helps with memory use for nonwell-behaved apps (that is, applications that don't close `Statement` objects when they should). (Bug #4718)
- Send correct value for “boolean” `true` to server for `PreparedStatement.setObject(n, "true", Types.BIT)`. (Bug #4718)
- Fixed bug with Connection not caching statements from `prepareStatement()` when the statement wasn't a server-side prepared statement. (Bug #4718)

Changes in MySQL Connector/J 3.1.6 (2004-12-23)

Bugs Fixed

- `DBMD.getProcedures()` doesn't respect catalog parameter. (Bug #7026)
- Fixed hang on `SocketInputStream.read()` with `Statement.setMaxRows()` and multiple result sets when driver has to truncate result set directly, rather than tacking a `LIMIT` *n* on the end of it.

Changes in MySQL Connector/J 3.1.5 (2004-12-02)

Bugs Fixed

- Use 1MB packet for sending file for `LOAD DATA LOCAL INFILE` if that is < `max_allowed_packet` on server. (Bug #6537)
- `SUM()` on `DECIMAL` with server-side prepared statement ignores scale if zero-padding is needed (this ends up being due to conversion to `DOUBLE` by server, which when converted to a string to parse into `BigDecimal`, loses all “padding” zeros). (Bug #6537)
- Use `DatabaseMetaData.getIdentifierQuoteString()` when building DBMD queries. (Bug #6537)
- Use our own implementation of buffered input streams to get around blocking behavior of `java.io.BufferedInputStream`. Disable this with `useReadAheadInput=false`. (Bug #6399)
- Make auto-deserialization of `java.lang.Objects` stored in `BLOB` columns configurable using `autoDeserialize` property (defaults to `false`). (Bug #6399)
- `ResultSetMetaData.getColumnDisplaySize()` returns incorrect values for multibyte charsets. (Bug #6399)
- Re-work `Field.isOpaqueBinary()` to detect `CHAR(`*n*`) CHARACTER SET BINARY` to support fixed-length binary fields for `ResultSet.getObject()`. (Bug #6399)

- Failing to connect to the server when one of the addresses for the given host name is IPV6 (which the server does not yet bind on). The driver now loops through *all* IP addresses for a given host, and stops on the first one that `accepts()` a `socket.connect()`. (Bug #6348)
- Removed unwanted new `Throwable()` in `ResultSet` constructor due to bad merge (caused a new object instance that was never used for every result set created). Found while profiling for Bug #6359. (Bug #6225)
- `ServerSidePreparedStatement` allocating short-lived objects unnecessarily. (Bug #6225)
- Use null-safe-equals for key comparisons in updatable result sets. (Bug #6225)
- Fixed too-early creation of `StringBuffer` in `EscapeProcessor.escapeSQL()`, also return `String` when escaping not needed (to avoid unnecessary object allocations). Found while profiling for Bug #6359. (Bug #6225)
- `UNSIGNED BIGINT` unpacked incorrectly from server-side prepared statement result sets. (Bug #5729)
- Added experimental configuration property `dontUnpackBinaryResults`, which delays unpacking binary result set values until they're asked for, and only creates object instances for nonnumeric values (it is set to `false` by default). For some usecase/jvm combinations, this is friendlier on the garbage collector. (Bug #5706)
- Don't throw exceptions for `Connection.releaseSavepoint()`. (Bug #5706)
- Inefficient detection of pre-existing string instances in `ResultSet.getNativeString()`. (Bug #5706)
- Use a per-session `Calendar` instance by default when decoding dates from `ServerPreparedStatements` (set to old, less performant behavior by setting property `dynamicCalendars=true`). (Bug #5706)
- Fixed batched updates with server prepared statements weren't looking if the types had changed for a given batched set of parameters compared to the previous set, causing the server to return the error “Wrong arguments to mysql_stmt_execute()”. (Bug #5235)
- Handle case when string representation of timestamp contains trailing “`.`” with no numbers following it. (Bug #5235)
- Server-side prepared statements did not honor `zeroDateTimeBehavior` property, and would cause class-cast exceptions when using `ResultSet.getObject()`, as the all-zero string was always returned. (Bug #5235)
- Fix comparisons made between string constants and dynamic strings that are converted with either `toUpperCase()` or `toLowerCase()` to use `Locale.ENGLISH`, as some locales “override” case rules for English. Also use `StringUtils.indexOfIgnoreCase()` instead of `.toUpperCase().indexOf()`, avoids creating a very short-lived transient `String` instance.

Changes in MySQL Connector/J 3.1.4 (2004-09-04)

Bugs Fixed

- Fixed `ServerPreparedStatement` to read prepared statement metadata off the wire, even though it is currently a placeholder instead of using `MysqlIO.clearInputStream()` which didn't work at various times because data wasn't available to read from the server yet. This fixes sporadic errors users were having with `ServerPreparedStatements` throwing `ArrayIndexOutOfBoundExceptions`. (Bug #5032)
- Added three ways to deal with all-zero datetimes when reading them from a `ResultSet`: `exception` (the default), which throws an `SQLException` with an SQLState of `S1009`;

`convertToNull`, which returns `NULL` instead of the date; and `round`, which rounds the date to the nearest closest value which is `'0001-01-01'`. (Bug #5032)

- The driver is more strict about truncation of numerics on `ResultSet.get*()`, and will throw an `SQLException` when truncation is detected. You can disable this by setting `jdbcCompliantTruncation` to `false` (it is enabled by default, as this functionality is required for JDBC compliance). (Bug #5032)
- You can now use URLs in `LOAD DATA LOCAL INFILE` statements, and the driver will use Java's built-in handlers for retrieving the data and sending it to the server. This feature is not enabled by default, you must set the `allowUrlInLocalInfile` connection property to `true`. (Bug #5032)
- `ResultSet.getObject()` doesn't return type `Boolean` for pseudo-bit types from prepared statements on 4.1.x (shortcut for avoiding extra type conversion when using binary-encoded result sets obscured test in `getObject()` for “pseudo” bit type). (Bug #5032)
- Use `com.mysql.jdbc.Message`'s classloader when loading resource bundle, should fix sporadic issues when the caller's classloader can't locate the resource bundle. (Bug #5032)
- `ServerPreparedStatements` dealing with return of `DECIMAL` type don't work. (Bug #5012)
- Track packet sequence numbers if `enablePacketDebug=true`, and throw an exception if packets received out-of-order. (Bug #4689)
- `ResultSet.wasNull()` does not work for primitives if a previous `null` was returned. (Bug #4689)
- Optimized integer number parsing, enable “old” slower integer parsing using JDK classes using `useFastIntParsing=false` property. (Bug #4642)
- Added `useOnlyServerErrorMessages` property, which causes message text in exceptions generated by the server to only contain the text sent by the server (as opposed to the SQLState's “standard” description, followed by the server's error message). This property is set to `true` by default. (Bug #4642)
- `ServerPreparedStatement.execute*()` sometimes threw `ArrayIndexOutOfBoundsException` when unpacking field metadata. (Bug #4642)
- Connector/J 3.1.3 beta does not handle integers correctly (caused by changes to support unsigned reads in `Buffer.readInt()` -> `Buffer.readShort()`). (Bug #4510)
- Added support in `DatabaseMetaData.getTables()` and `getTableTypes()` for views, which are now available in MySQL server 5.0.x. (Bug #4510)
- `ResultSet.getObject()` returns wrong type for strings when using prepared statements. (Bug #4482)
- Calling `MysqlPooledConnection.close()` twice (even though an application error), caused NPE. Fixed. (Bug #4482)

Changes in MySQL Connector/J 3.1.3 (2004-07-07)

Bugs Fixed

- Support new time zone variables in MySQL-4.1.3 when `useTimezone=true`. (Bug #4311)
- Error in retrieval of `mediumint` column with prepared statements and binary protocol. (Bug #4311)
- Support for unsigned numerics as return types from prepared statements. This also causes a change in `ResultSet.getObject()` for the `bigint unsigned` type, which used to return `BigDecimal` instances, it now returns instances of `java.lang.BigInteger`. (Bug #4311)
- Externalized more messages (on-going effort). (Bug #4119)

- Null bitmask sent for server-side prepared statements was incorrect. (Bug #4119)
- Added constants for MySQL error numbers (publicly accessible, see `com.mysql.jdbc.MysqlErrorNumbers`), and the ability to generate the mappings of vendor error codes to SQLStates that the driver uses (for documentation purposes). (Bug #4119)
- Added packet debugging code (see the `enablePacketDebug` property documentation). (Bug #4119)
- Use SQL Standard SQL states by default, unless `useSqlStateCodes` property is set to `false`. (Bug #4119)
- Mangle output parameter names for `CallableStatements` so they will not clash with user variable names.
- Added support for `INOUT` parameters in `CallableStatements`.

Changes in MySQL Connector/J 3.1.2 (2004-06-09)

Bugs Fixed

- Don't enable server-side prepared statements for server version 5.0.0 or 5.0.1, as they aren't compatible with the '4.1.2+' style that the driver uses (the driver expects information to come back that isn't there, so it hangs). (Bug #3804)
- `getWarnings()` returns `SQLWarning` instead of `DataTruncation`. (Bug #3804)
- `getProcedureColumns()` doesn't work with wildcards for procedure name. (Bug #3540)
- `getProcedures()` does not return any procedures in result set. (Bug #3539)
- Fixed `DatabaseMetaData.getProcedures()` when run on MySQL-5.0.0 (output of `SHOW PROCEDURE STATUS` changed between 5.0.0 and 5.0.1. (Bug #3520)
- Added `connectionCollation` property to cause driver to issue `set collation_connection=...` query on connection init if default collation for given charset is not appropriate. (Bug #3520)
- `DBMD.getSQLStateType()` returns incorrect value. (Bug #3520)
- Correctly map output parameters to position given in `prepareCall()` versus. order implied during `registerOutParameter()`. (Bug #3146)
- Cleaned up detection of server properties. (Bug #3146)
- Correctly detect initial character set for servers >= 4.1.0. (Bug #3146)
- Support placeholder for parameter metadata for server >= 4.1.2. (Bug #3146)
- Added `gatherPerformanceMetrics` property, along with properties to control when/where this info gets logged (see docs for more info).
- Fixed case when no parameters could cause a `NullPointerException` in `CallableStatement.setOutputParameters()`.
- Enabled callable statement caching using `cacheCallableStmts` property.
- Fixed sending of split packets for large queries, enabled nio ability to send large packets as well.
- Added `.toString()` functionality to `ServerPreparedStatement`, which should help if you're trying to debug a query that is a prepared statement (it shows SQL as the server would process).
- Added `logSlowQueries` property, along with `slowQueriesThresholdMillis` property to control when a query should be considered “slow.”

- Removed wrapping of exceptions in `MysqlIO.changeUser()`.
- Fixed stored procedure parameter parsing info when size was specified for a parameter (for example, `char()`, `varchar()`).
- `ServerPreparedStatements` weren't actually de-allocating server-side resources when `.close()` was called.
- Fixed case when no output parameters specified for a stored procedure caused a bogus query to be issued to retrieve out parameters, leading to a syntax error from the server.

Changes in MySQL Connector/J 3.1.1 (2004-02-14)

Bugs Fixed

- Use DocBook version of docs for shipped versions of drivers. (Bug #2671)
- `NULL` fields were not being encoded correctly in all cases in server-side prepared statements. (Bug #2671)
- Fixed rare buffer underflow when writing numbers into buffers for sending prepared statement execution requests. (Bug #2671)
- Fixed `ConnectionProperties` that weren't properly exposed through accessors, cleaned up `ConnectionProperties` code. (Bug #2623)
- Class-cast exception when using scrolling result sets and server-side prepared statements. (Bug #2623)
- Merged unbuffered input code from 3.0. (Bug #2623)
- Enabled streaming of result sets from server-side prepared statements. (Bug #2606)
- Server-side prepared statements were not returning data type `YEAR` correctly. (Bug #2606)
- Fixed charset conversion issue in `getTables()`. (Bug #2502)
- Implemented multiple result sets returned from a statement or stored procedure. (Bug #2502)
- Implemented `Connection.prepareCall()`, and `DatabaseMetaData.getProcedures()` and `getProcedureColumns()`. (Bug #2359)
- Merged prepared statement caching, and `.getMetaData()` support from 3.0 branch. (Bug #2359)
- Fixed off-by-1900 error in some cases for years in `TimeUtil.fastDate`/`TimeCreate()` when unpacking results from server-side prepared statements. (Bug #2359)
- Reset `long binary` parameters in `ServerPreparedStatement` when `clearParameters()` is called, by sending `COM_RESET_STMT` to the server. (Bug #2359)
- `NULL` values for numeric types in binary encoded result sets causing `NullPointerExceptions`. (Bug #2359)
- Display where/why a connection was implicitly closed (to aid debugging). (Bug #1673)
- `DatabaseMetaData.getColumns()` is not returning correct column ordinal info for non-`'%'` column name patterns. (Bug #1673)
- Fixed `NullPointerException` in `ServerPreparedStatement.setTimestamp()`, as well as year and month discrepencies in `ServerPreparedStatement.setTimestamp()`, `setDate()`. (Bug #1673)
- Added ability to have multiple database/JVM targets for compliance and regression/unit tests in `build.xml`. (Bug #1673)

- Fixed sending of queries larger than 16M. (Bug #1673)
- Merged fix of data type mapping from MySQL type `FLOAT` to `java.sql.Types.REAL` from 3.0 branch. (Bug #1673)
- Fixed NPE and year/month bad conversions when accessing some datetime functionality in `ServerPreparedStatements` and their resultant result sets. (Bug #1673)
- Added named and indexed input/output parameter support to `CallableStatement`. MySQL-5.0.x or newer. (Bug #1673)
- `CommunicationsException` implemented, that tries to determine why communications was lost with a server, and displays possible reasons when `.getMessage()` is called. (Bug #1673)
- Detect collation of column for `RSMD.isCaseSensitive()`. (Bug #1673)
- Optimized `Buffer.readLenByteArray()` to return shared empty byte array when length is 0.
- Fix support for table aliases when checking for all primary keys in `UpdatableResultSet`.
- Unpack “unknown” data types from server prepared statements as `Strings`.
- Implemented `Statement.getWarnings()` for MySQL-4.1 and newer (using `SHOW WARNINGS`).
- Ensure that warnings are cleared before executing queries on prepared statements, as-per JDBC spec (now that we support warnings).
- Correctly initialize datasource properties from JNDI Refs, including explicitly specified URLs.
- Implemented long data (Blobs, Clobs, InputStreams, Readers) for server prepared statements.
- Deal with 0-length tokens in `EscapeProcessor` (caused by callable statement escape syntax).
- `DatabaseMetaData` now reports `supportsStoredProcedures()` for MySQL versions >= 5.0.0
- Support for `mysql_change_user()`. See the `changeUser()` method in `com.mysql.jdbc.Connection`.
- Removed `useFastDates` connection property.
- Support for NIO. Use `useNIO=true` on platforms that support NIO.
- Check for closed connection on delete/update/insert row operations in `UpdatableResultSet`.
- Support for transaction savepoints (MySQL >= 4.0.14 or 4.1.1).
- Support “old” `profileSql` capitalization in `ConnectionProperties`. This property is deprecated, you should use `profileSQL` if possible.
- Fixed character encoding issues when converting bytes to ASCII when MySQL doesn't provide the character set, and the JVM is set to a multibyte encoding (usually affecting retrieval of numeric values).
- Centralized setting of result set type and concurrency.
- Fixed bug with `UpdatableResultSets` not using client-side prepared statements.
- Default result set type changed to `TYPE_FORWARD_ONLY` (JDBC compliance).
- Fixed `IllegalAccessError` to `Calendar.getTimeInMillis()` in `DateTimeValue` (for JDK < 1.4).
- Allow contents of `PreparedStatement.setBlob()` to be retained between calls to `.execute*()`.

- Fixed stack overflow in `Connection.prepareCall()` (bad merge).
- Refactored how connection properties are set and exposed as `DriverPropertyInfo` as well as `Connection` and `DataSource` properties.
- Reduced number of methods called in average query to be more efficient.
- Prepared `Statements` will be re-prepared on auto-reconnect. Any errors encountered are postponed until first attempt to re-execute the re-prepared statement.

Changes in MySQL Connector/J 3.1.0 (2003-02-18, Alpha)

Bugs Fixed

- Added `useServerPrepStmts` property (default `false`). The driver will use server-side prepared statements when the server version supports them (4.1 and newer) when this property is set to `true`. It is currently set to `false` by default until all bind/fetch functionality has been implemented. Currently only DML prepared statements are implemented for 4.1 server-side prepared statements.
- Added `requireSSL` property.
- Track open `Statements`, close all when `Connection.close()` is called (JDBC compliance).

Changes in MySQL Connector/J 3.0

Changes in MySQL Connector/J 3.0.17 (2005-06-23)

Bugs Fixed

- Workaround for server Bug #9098: Default values of `CURRENT_*` for `DATE`, `TIME`, `DATETIME`, and `TIMESTAMP` columns can't be distinguished from `string` values, so `UpdatableResultSet.moveToInsertRow()` generates bad SQL for inserting default values. (Bug #8812)
- `NON_UNIQUE` column from `DBMD.getIndexInfo()` returned inverted value. (Bug #8812)
- `EUCKR` charset is sent as `SET NAMES euc_kr` which MySQL-4.1 and newer doesn't understand. (Bug #8629)
- Added support for the `EUC_JP_Solaris` character encoding, which maps to a MySQL encoding of `eucjpms` (backported from 3.1 branch). This only works on servers that support `eucjpms`, namely 5.0.3 or later. (Bug #8629)
- Use hex escapes for `PreparedStatement.setBytes()` for double-byte charsets including “aliases” `Windows-31J`, `CP934`, `MS932`. (Bug #8629)
- `DatabaseMetaData.supportsSelectForUpdate()` returns correct value based on server version. (Bug #8629)
- Which requires hex escaping of binary data when using multibyte charsets with prepared statements. (Bug #8064)
- Fixed duplicated code in `configureClientCharset()` that prevented `useOldUTF8Behavior=true` from working properly. (Bug #7952)
- Backported SQLState codes mapping from Connector/J 3.1, enable with `useSqlStateCodes=true` as a connection property, it defaults to `false` in this release, so that we don't break legacy applications (it defaults to `true` starting with Connector/J 3.1). (Bug #7686)
- Timestamp key column data needed `_binary` stripped for `UpdatableResultSet.refreshRow()`. (Bug #7686)

- `MS932`, `SHIFT_JIS`, and `Windows_31J` not recognized as aliases for `sjis`. (Bug #7607)
- Handle streaming result sets with more than 2 billion rows properly by fixing wraparound of row number counter. (Bug #7601)
- `PreparedStatement.fixDecimalExponent()` adding extra `+`, making number unparseable by MySQL server. (Bug #7601)
- Escape sequence {fn convert(..., type)} now supports ODBC-style types that are prepended by `SQL_`. (Bug #7601)
- Statements created from a pooled connection were returning physical connection instead of logical connection when `getConnection()` was called. (Bug #7316)
- Support new protocol type `MYSQL_TYPE_VARCHAR`. (Bug #7081)
- Added `useOldUTF8Behavior`' configuration property, which causes JDBC driver to act like it did with MySQL-4.0.x and earlier when the character encoding is `utf-8` when connected to MySQL-4.1 or newer. (Bug #7081)
- `DatabaseMetaData.getIndexInfo()` ignored `unique` parameter. (Bug #7081)
- `PreparedStatement.fixDecimalExponent()` adding extra `+`, making number unparseable by MySQL server. (Bug #7061)
- `PreparedStatements` don't encode Big5 (and other multibyte) character sets correctly in static SQL strings. (Bug #7033)
- Connections starting up failed-over (due to down master) never retry master. (Bug #6966)
- Adding `CP943` to aliases for `sjis`. (Bug #6549, Bug #7607)
- `Timestamp`/`Time` conversion goes in the wrong “direction” when `useTimeZone=true` and server time zone differs from client time zone. (Bug #5874)

Changes in MySQL Connector/J 3.0.16 (2004-11-15)

Bugs Fixed

- Made `TINYINT(1)` -> `BIT`/`Boolean` conversion configurable using `tinyInt1isBit` property (default `true` to be JDBC compliant out of the box). (Bug #5664)
- Off-by-one bug in `Buffer.readString(`*`string`*`)`. (Bug #5664)
- `ResultSet.updateByte()` when on insert row throws `ArrayOutOfBoundsException`. (Bug #5664)
- Fixed regression where `useUnbufferedInput` was defaulting to `false`. (Bug #5664)
- `ResultSet.getTimestamp()` on a column with `TIME` in it fails. (Bug #5664)
- Fixed `DatabaseMetaData.getTypes()` returning incorrect (this is, nonnegative) scale for the `NUMERIC` type. (Bug #5664)
- Only set `character_set_results` during connection establishment if server version >= 4.1.1. (Bug #5664)
- Fixed `ResultSetMetaData.isReadOnly()` to detect nonwritable columns when connected to MySQL-4.1 or newer, based on existence of “original” table and column names.
- Re-issue character set configuration commands when re-using pooled connections or `Connection.changeUser()` when connected to MySQL-4.1 or newer.

Changes in MySQL Connector/J 3.0.15 (2004-09-04)

Bugs Fixed

- `ResultSet.getMetaData()` should not return incorrectly initialized metadata if the result set has been closed, but should instead throw an `SQLException`. Also fixed for `getRow()` and `getWarnings()` and traversal methods by calling `checkClosed()` before operating on instance-level fields that are nullified during `.close()`. (Bug #5069)
- Use `_binary` introducer for `PreparedStatement.setBytes()` and `set*Stream()` when connected to MySQL-4.1.x or newer to avoid misinterpretation during character conversion. (Bug #5069)
- Parse new time zone variables from 4.1.x servers. (Bug #5069)
- `ResultSet` should release `Field[]` instance in `.close()`. (Bug #5022)
- `RSMD.getPrecision()` returning 0 for nonnumeric types (should return max length in chars for nonbinary types, max length in bytes for binary types). This fix also fixes mapping of `RSMD.getColumnType()` and `RSMD.getColumnTypeName()` for the `BLOB` types based on the length sent from the server (the server doesn't distinguish between `TINYBLOB`, `BLOB`, `MEDIUMBLOB` or `LONGBLOB` at the network protocol level). (Bug #4880)
- “Production” is now “GA” (General Availability) in naming scheme of distributions. (Bug #4860, Bug #4138)
- `DBMD.getColumns()` returns incorrect JDBC type for unsigned columns. This affects type mappings for all numeric types in the `RSMD.getColumnType()` and `RSMD.getColumnTypeNames()` methods as well, to ensure that “like” types from `DBMD.getColumns()` match up with what `RSMD.getColumnType()` and `getColumnTypeNames()` return. (Bug #4860, Bug #4138)
- Calling `.close()` twice on a `PooledConnection` causes NPE. (Bug #4808)
- `DOUBLE` mapped twice in `DBMD.getTypeInfo()`. (Bug #4742)
- Added FLOSS license exemption. (Bug #4742)
- Removed redundant calls to `checkRowPos()` in `ResultSet`. (Bug #4334)
- Failover for `autoReconnect` not using port numbers for any hosts, and not retrying all hosts.

  

  Warning

  This required a change to the `SocketFactory connect()` method signature, which is now `public Socket connect(String host, int portNumber, Properties props)`; therefore, any third-party socket factories will have to be changed to support this signature.

  (Bug #4334)
- Logical connections created by `MysqlConnectionPoolDataSource` will now issue a `rollback()` when they are closed and sent back to the pool. If your application server/connection pool already does this for you, you can set the `rollbackOnPooledClose` property to `false` to avoid the overhead of an extra `rollback()`. (Bug #4334)
- `StringUtils.escapeEasternUnicodeByteStream` was still broken for GBK. (Bug #4010)

Changes in MySQL Connector/J 3.0.14 (2004-05-28)

Bugs Fixed

- Fixed URL parsing error.

Changes in MySQL Connector/J 3.0.13 (2004-05-27)

Bugs Fixed

- `No Database Selected` when using `MysqlConnectionPoolDataSource`. (Bug #3920)
- `PreparedStatement.getGeneratedKeys()` method returns only 1 result for batched insertions. (Bug #3873)
- Using a `MySQLDatasource` without server name fails. (Bug #3848)

Changes in MySQL Connector/J 3.0.12 (2004-05-18)

Bugs Fixed

- Inconsistent reporting of data type. The server still doesn't return all types for *BLOBs *TEXT correctly, so the driver won't return those correctly. (Bug #3570)
- `UpdatableResultSet` not picking up default values for `moveToInsertRow()`. (Bug #3557)
- Not specifying database in URL caused `MalformedURL` exception. (Bug #3554)
- Auto-convert MySQL encoding names to Java encoding names if used for `characterEncoding` property. (Bug #3554)
- Use `junit.textui.TestRunner` for all unit tests (to enable them to be run from the command line outside of Ant or Eclipse). (Bug #3554)
- Added encoding names that are recognized on some JVMs to fix case where they were reverse-mapped to MySQL encoding names incorrectly. (Bug #3554)
- Made `StringRegressionTest` 4.1-unicode aware. (Bug #3520)
- Fixed regression in `PreparedStatement.setString()` and eastern character encodings. (Bug #3520)
- `DBMD.getSQLStateType()` returns incorrect value. (Bug #3520)
- Renamed `StringUtils.escapeSJISByteStream()` to more appropriate `escapeEasternUnicodeByteStream()`. (Bug #3511)
- `StringUtils.escapeSJISByteStream()` not covering all eastern double-byte charsets correctly. (Bug #3511)
- Return creating statement for `ResultSets` created by `getGeneratedKeys()`. (Bug #2957)
- Use `SET character_set_results` during initialization to enable any charset to be returned to the driver for result sets. (Bug #2670)
- Don't truncate `BLOB` or `CLOB` values when using `setBytes()` and `setBinary/CharacterStream()`. (Bug #2670)
- Dynamically configure character set mappings for field-level character sets on MySQL-4.1.0 and newer using `SHOW COLLATION` when connecting. (Bug #2670)
- Map `binary` character set to `US-ASCII` to support `DATETIME` charset recognition for servers >= 4.1.2. (Bug #2670)
- Use `charsetnr` returned during connect to encode queries before issuing `SET NAMES` on MySQL >= 4.1.0. (Bug #2670)

- Add helper methods to `ResultSetMetaData` (`getColumnCharacterEncoding()` and `getColumnCharacterSet()`) to permit end users to see what charset the driver thinks it should be using for the column. (Bug #2670)
- Only set `character_set_results` for MySQL >= 4.1.0. (Bug #2670)
- Allow `url` parameter for `MysqlDataSource` and `MysqlConnectionPool DataSource` so that passing of other properties is possible from inside appservers.
- Don't escape SJIS/GBK/BIG5 when using MySQL-4.1 or newer.
- Backport documentation tooling from 3.1 branch.
- Added `failOverReadOnly` property, to enable the user to configure the state of the connection (read-only/writable) when failed over.
- Add unsigned attribute to `DatabaseMetaData.getColumns()` output in the `TYPE_NAME` column.
- Map duplicate key and foreign key errors to SQLState of `23000`.
- Backported “change user” and “reset server state” functionality from 3.1 branch, to enable clients of `MysqlConnectionPoolDataSource` to reset server state on `getConnection()` on a pooled connection.
- Allow `java.util.Date` to be sent in as parameter to `PreparedStatement.setObject()`, converting it to a `Timestamp` to maintain full precision. . (Bug #103)

Changes in MySQL Connector/J 3.0.11 (2004-02-19)

Bugs Fixed

- Return `java.lang.Double` for `FLOAT` type from `ResultSetMetaData.getColumnClassName()`. (Bug #2855)
- Return `[B` instead of `java.lang.Object` for `BINARY`, `VARBINARY` and `LONGVARBINARY` types from `ResultSetMetaData.getColumnClassName()` (JDBC compliance). (Bug #2855)
- Issue connection events on all instances created from a `ConnectionPoolDataSource`. (Bug #2855)
- Return `java.lang.Integer` for `TINYINT` and `SMALLINT` types from `ResultSetMetaData.getColumnClassName()`. (Bug #2852)
- Added `useUnbufferedInput` parameter, and now use it by default (due to JVM issue http://developer.java.sun.com/developer/bugParade/bugs/4401235.html) (Bug #2578)
- Fixed failover always going to last host in list. (Bug #2578)
- Detect `on`/`off` or `1`, `2`, `3` form of `lower_case_table_names` value on server. (Bug #2578)
- `AutoReconnect` time was growing faster than exponentially. (Bug #2447)
- Trigger a `SET NAMES utf8` when encoding is forced to `utf8` *or* `utf-8` using the `characterEncoding` property. Previously, only the Java-style encoding name of `utf-8` would trigger this.

Changes in MySQL Connector/J 3.0.10 (2004-01-13)

Bugs Fixed

- Enable caching of the parsing stage of prepared statements using the `cachePrepStmts`, `prepStmtCacheSize`, and `prepStmtCacheSqlLimit` properties (disabled by default). (Bug #2006)

- Fixed security exception when used in Applets (applets can't read the system property `file.encoding` which is needed for `LOAD DATA LOCAL INFILE`). (Bug #2006)
- Speed up parsing of `PreparedStatements`, try to use one-pass whenever possible. (Bug #2006)
- Fixed exception `Unknown character set 'danish'` on connect with JDK-1.4.0 (Bug #2006)
- Fixed mappings in SQLError to report deadlocks with SQLStates of `41000`. (Bug #2006)
- Removed static synchronization bottleneck from instance factory method of `SingleByteCharsetConverter`. (Bug #2006)
- Removed static synchronization bottleneck from `PreparedStatement.setTimestamp()`. (Bug #2006)
- `ResultSet.findColumn()` should use first matching column name when there are duplicate column names in `SELECT` query (JDBC-compliance). (Bug #2006)
- `maxRows` property would affect internal statements, so check it for all statement creation internal to the driver, and set to 0 when it is not. (Bug #2006)
- Use constants for SQLStates. (Bug #2006)
- Map charset `ko18_ru` to `ko18r` when connected to MySQL-4.1.0 or newer. (Bug #2006)
- Ensure that `Buffer.writeString()` saves room for the `\0`. (Bug #2006)
- `ArrayIndexOutOfBounds` when parameter number == number of parameters + 1. (Bug #1958)
- Connection property `maxRows` not honored. (Bug #1933)
- Statements being created too many times in `DBMD.extractForeignKeyFromCreateTable()`. (Bug #1925)
- Support escape sequence {fn convert ... }. (Bug #1914)
- Implement `ResultSet.updateClob()`. (Bug #1913)
- Autoreconnect code didn't set catalog upon reconnect if it had been changed. (Bug #1913)
- `ResultSet.getObject()` on `TINYINT` and `SMALLINT` columns should return Java type `Integer`. (Bug #1913)
- Added more descriptive error message `Server Configuration Denies Access to DataSource`, as well as retrieval of message from server. (Bug #1913)
- `ResultSetMetaData.isCaseSensitive()` returned wrong value for `CHAR`/`VARCHAR` columns. (Bug #1913)
- Added `alwaysClearStream` connection property, which causes the driver to always empty any remaining data on the input stream before each query. (Bug #1913)
- `DatabaseMetaData.getSystemFunction()` returning bad function `VResultsSion`. (Bug #1775)
- Foreign Keys column sequence is not consistent in `DatabaseMetaData.getImported/Exported/CrossReference()`. (Bug #1731)
- Fix for `ArrayIndexOutOfBounds` exception when using `Statement.setMaxRows()`. (Bug #1695)
- Subsequent call to `ResultSet.updateFoo()` causes NPE if result set is not updatable. (Bug #1630)

- Fix for 4.1.1-style authentication with no password. (Bug #1630)
- Cross-database updatable result sets are not checked for updatability correctly. (Bug #1592)
- `DatabaseMetaData.getColumns()` should return `Types.LONGVARCHAR` for MySQL `LONGTEXT` type. (Bug #1592)
- Fixed regression of `Statement.getGeneratedKeys()` and `REPLACE` statements. (Bug #1576)
- Barge blobs and split packets not being read correctly. (Bug #1576)
- Backported fix for aliased tables and `UpdatableResultSets` in `checkUpdatability()` method from 3.1 branch. (Bug #1534)
- “Friendlier” exception message for `PacketTooLargeException`. (Bug #1534)
- Don't count quoted IDs when inside a 'string' in `PreparedStatement` parsing. (Bug #1511)

Changes in MySQL Connector/J 3.0.9 (2003-10-07)

Bugs Fixed

- `ResultSet.get/setString` mashing char 127. (Bug #1247)
- Added property to “clobber” streaming results, by setting the `clobberStreamingResults` property to `true` (the default is `false`). This will cause a “streaming” `ResultSet` to be automatically closed, and any outstanding data still streaming from the server to be discarded if another query is executed before all the data has been read from the server. (Bug #1247)
- Added `com.mysql.jdbc.util.BaseBugReport` to help creation of testcases for bug reports. (Bug #1247)
- Backported authentication changes for 4.1.1 and newer from 3.1 branch. (Bug #1247)
- Made `databaseName`, `portNumber`, and `serverName` optional parameters for `MysqlDataSourceFactory`. (Bug #1246)
- Optimized `CLOB.setChracterStream()`. (Bug #1131)
- Fixed `CLOB.truncate()`. (Bug #1130)
- Faster date handling code in `ResultSet` and `PreparedStatement` (no longer uses `Date` methods that synchronize on static calendars).
- Fixed deadlock issue with `Statement.setMaxRows()`. (Bug #1099)
- `DatabaseMetaData.getColumns()` getting confused about the keyword “set” in character columns. (Bug #1099)
- Clip +/- INF (to smallest and largest representative values for the type in MySQL) and NaN (to 0) for `setDouble`/`setFloat()`, and issue a warning on the statement when the server does not support +/- INF or NaN. (Bug #884)
- Don't fire connection closed events when closing pooled connections, or on `PooledConnection.getConnection()` with already open connections. (Bug #884)
- Double-escaping of `'\'` when charset is SJIS or GBK and `'\'` appears in nonescaped input. (Bug #879)
- When emptying input stream of unused rows for “streaming” result sets, have the current thread `yield()` every 100 rows to not monopolize CPU time. (Bug #879)
- Issue exception on `ResultSet.getXXX()` on empty result set (wasn't caught in some cases). (Bug #848)

- Don't hide messages from exceptions thrown in I/O layers. (Bug #848)
- Fixed regression in large split-packet handling. (Bug #848)
- Better diagnostic error messages in exceptions for “streaming” result sets. (Bug #848)
- Don't change timestamp TZ twice if `useTimezone==true`. (Bug #774)
- Fixed test for end of buffer in `Buffer.readString()`.
- Don't wrap `SQLExceptions` in `RowDataDynamic`. (Bug #688)
- Don't try and reset isolation level on reconnect if MySQL doesn't support them. (Bug #688)
- The `insertRow` in an `UpdatableResultSet` is now loaded with the default column values when `moveToInsertRow()` is called. (Bug #688)
- `DatabaseMetaData.getColumns()` wasn't returning `NULL` for default values that are specified as `NULL`. (Bug #688)
- Change default statement type/concurrency to `TYPE_FORWARD_ONLY` and `CONCUR_READ_ONLY` (spec compliance). (Bug #688)
- Fix `UpdatableResultSet` to return values for `get`*`XXX`*`()` when on insert row. (Bug #675)
- Support `InnoDB` constraint names when extracting foreign key information in `DatabaseMetaData` (implementing ideas from Parwinder Sekhon). (Bug #664, Bug #517)
- Backported 4.1 protocol changes from 3.1 branch (server-side SQL states, new field information, larger client capability flags, connect-with-database, and so forth). (Bug #664, Bug #517)
- `refreshRow` didn't work when primary key values contained values that needed to be escaped (they ended up being doubly escaped). (Bug #661)
- Fixed `ResultSet.previous()` behavior to move current position to before result set when on first row of result set. (Bug #496)
- Fixed `Statement` and `PreparedStatement` issuing bogus queries when `setMaxRows()` had been used and a `LIMIT` clause was present in the query. (Bug #496)

Changes in MySQL Connector/J 3.0.8 (2003-05-23)

Bugs Fixed

- Use JVM charset with file names and `LOAD DATA [LOCAL] INFILE`.
- Fix infinite loop with `Connection.cleanup()`.
- Changed Ant target `compile-core` to `compile-driver`, and made testsuite compilation a separate target.
- Fixed result set not getting set for `Statement.executeUpdate()`, which affected `getGeneratedKeys()` and `getUpdateCount()` in some cases.
- Return list of generated keys when using multi-value `INSERTS` with `Statement.getGeneratedKeys()`.
- Fixed SJIS encoding bug, thanks to Naoto Sato. (Bug #378)
- Fix problem detecting server character set in some cases. (Bug #378)
- Allow multiple calls to `Statement.close()`. (Bug #378)
- Return correct number of generated keys when using `REPLACE` statements. (Bug #378)

- Unicode character 0xFFFF in a string would cause the driver to throw an `ArrayOutOfBoundsException`. . (Bug #378)
- Fix row data decoding error when using *very* large packets. (Bug #378)
- Optimized row data decoding. (Bug #378)
- Issue exception when operating on an already closed prepared statement. (Bug #378)
- Optimized usage of `EscapeProcessor`. (Bug #378)
- Allow bogus URLs in `Driver.getPropertyInfo()`.

Changes in MySQL Connector/J 3.0.7 (2003-04-08)

Bugs Fixed

- Fixed charset issues with database metadata (charset was not getting set correctly).
- You can now toggle profiling on/off using `Connection.setProfileSql(boolean)`.
- 4.1 Column Metadata fixes.
- Fixed `MysqlPooledConnection.close()` calling wrong event type.
- Fixed `StringIndexOutOfBoundsException` in `PreparedStatement.setClob()`.
- `IOExceptions` during a transaction now cause the `Connection` to be closed.
- Remove synchronization from `Driver.connect()` and `Driver.acceptsUrl()`.
- Fixed missing conversion for `YEAR` type in `ResultSetMetaData.getColumnTypeName()`.
- Updatable `ResultSets` can now be created for aliased tables/columns when connected to MySQL-4.1 or newer.
- Fixed `LOAD DATA LOCAL INFILE` bug when file > `max_allowed_packet`.
- Don't pick up indexes that start with `pri` as primary keys for `DBMD.getPrimaryKeys()`.
- Ensure that packet size from `alignPacketSize()` does not exceed `max_allowed_packet` (JVM bug)
- Don't reset `Connection.isReadOnly()` when autoReconnecting.
- Fixed escaping of 0x5c (`'\'`) character for GBK and Big5 charsets.
- Fixed `ResultSet.getTimestamp()` when underlying field is of type `DATE`.
- Throw `SQLExceptions` when trying to do operations on a forcefully closed `Connection` (that is, when a communication link failure occurs).

Changes in MySQL Connector/J 3.0.6 (2003-02-18)

Bugs Fixed

- Backported 4.1 charset field info changes from Connector/J 3.1.
- Fixed `Statement.setMaxRows()` to stop sending `LIMIT` type queries when not needed (performance).
- Fixed `DBMD.getTypeInfo()` and `DBMD.getColumns()` returning different value for precision in `TEXT` and `BLOB` types.

- Fixed `SQLExceptions` getting swallowed on initial connect.
- Fixed `ResultSetMetaData` to return `""` when catalog not known. Fixes `NullPointerExceptions` with Sun's `CachedRowSet`.
- Allow ignoring of warning for “non transactional tables” during rollback (compliance/usability) by setting `ignoreNonTxTables` property to `true`.
- Clean up `Statement` query/method mismatch tests (that is, `INSERT` not permitted with `.executeQuery()`).
- Fixed `ResultSetMetaData.isWritable()` to return correct value.
- More checks added in `ResultSet` traversal method to catch when in closed state.
- Implemented `Blob.setBytes()`. You still need to pass the resultant `Blob` back into an updatable `ResultSet` or `PreparedStatement` to persist the changes, because MySQL does not support “locators”.
- Add “window” of different `NULL` sorting behavior to `DBMD.nullsAreSortedAtStart` (4.0.2 to 4.0.10, true; otherwise, no).

Changes in MySQL Connector/J 3.0.5 (2003-01-22)

Bugs Fixed

- Fixed `ResultSet.isBeforeFirst()` for empty result sets.
- Added missing `LONGTEXT` type to `DBMD.getColumns()`.
- Implemented an empty `TypeMap` for `Connection.getTypeMap()` so that some third-party apps work with MySQL (IBM WebSphere 5.0 Connection pool).
- Added update options for foreign key metadata.
- Fixed `Buffer.fastSkipLenString()` causing `ArrayIndexOutOfBounds` exceptions with some queries when unpacking fields.
- Quote table names in `DatabaseMetaData.getColumns()`, `getPrimaryKeys()`, `getIndexInfo()`, `getBestRowIdentifier()`.
- Retrieve `TX_ISOLATION` from database for `Connection.getTransactionIsolation()` when the MySQL version supports it, instead of an instance variable.
- Greatly reduce memory required for `setBinaryStream()` in `PreparedStatements`.

Changes in MySQL Connector/J 3.0.4 (2003-01-06)

Bugs Fixed

- Streamlined character conversion and `byte[]` handling in `PreparedStatements` for `setByte()`.
- Fixed `PreparedStatement.executeBatch()` parameter overwriting.
- Added quoted identifiers to database names for `Connection.setCatalog`.
- Added support for 4.0.8-style large packets.
- Reduce memory footprint of `PreparedStatements` by sharing outbound packet with `MysqlIO`.
- Added `strictUpdates` property to enable control of amount of checking for “correctness” of updatable result sets. Set this to `false` if you want faster updatable result sets and you know that

you create them from `SELECT` statements on tables with primary keys and that you have selected all primary keys in your query.

- Added support for quoted identifiers in `PreparedStatement` parser.

Changes in MySQL Connector/J 3.0.3 (2002-12-17)

Bugs Fixed

- Allow user to alter behavior of `Statement`/ `PreparedStatement.executeBatch()` using `continueBatchOnError` property (defaults to `true`).
- More robust escape tokenizer: Recognize `--` comments, and permit nested escape sequences (see `testsuite.EscapeProcessingTest`).
- Fixed `Buffer.isLastDataPacket()` for 4.1 and newer servers.
- `NamedPipeSocketFactory` now works (only intended for Windows), see `README` for instructions.
- Changed `charsToByte` in `SingleByteCharConverter` to be nonstatic.
- Use nonaliased table/column names and database names to fully qualify tables and columns in `UpdatableResultSet` (requires MySQL-4.1 or newer).
- `LOAD DATA LOCAL INFILE ...` now works, if your server is configured to permit it. Can be turned off with the `allowLoadLocalInfile` property (see the `README`).
- Implemented `Connection.nativeSQL()`.
- Fixed `ResultSetMetaData.getColumnTypeName()` returning `BLOB` for `TEXT` and `TEXT` for `BLOB` types.
- Fixed charset handling in `Fields.java`.
- Because of above, implemented `ResultSetMetaData.isAutoIncrement()` to use `Field.isAutoIncrement()`.
- Substitute `'?'` for unknown character conversions in single-byte character sets instead of `'\0'`.
- Added `CLIENT_LONG_FLAG` to be able to get more column flags (`isAutoIncrement()` being the most important).
- Honor `lower_case_table_names` when enabled in the server when doing table name comparisons in `DatabaseMetaData` methods.
- `DBMD.getImported/ExportedKeys()` now handles multiple foreign keys per table.
- More robust implementation of updatable result sets. Checks that *all* primary keys of the table have been selected.
- Some MySQL-4.1 protocol support (extended field info from selects).
- Check for connection closed in more `Connection` methods (`createStatement`, `prepareStatement`, `setTransactionIsolation`, `setAutoCommit`).
- Fixed `ResultSetMetaData.getPrecision()` returning incorrect values for some floating-point types.
- Changed `SingleByteCharConverter` to use lazy initialization of each converter.

Changes in MySQL Connector/J 3.0.2 (2002-11-08)

Bugs Fixed

- Implemented `Clob.setString()`.
- Added `com.mysql.jdbc.MiniAdmin` class, which enables you to send `shutdown` command to MySQL server. This is intended to be used when “embedding” Java and MySQL server together in an end-user application.
- Added SSL support. See `README` for information on how to use it.
- All `DBMD` result set columns describing schemas now return `NULL` to be more compliant with the behavior of other JDBC drivers for other database systems (MySQL does not support schemas).
- Use `SHOW CREATE TABLE` when possible for determining foreign key information for `DatabaseMetaData`. Also enables cascade options for `DELETE` information to be returned.
- Implemented `Clob.setCharacterStream()`.
- Failover and `autoReconnect` work only when the connection is in an `autoCommit(false)` state, to stay transaction-safe.
- Fixed `DBMD.supportsResultSetConcurrency()` so that it returns `true` for `ResultSet.TYPE_SCROLL_INSENSITIVE` and `ResultSet.CONCUR_READ_ONLY` or `ResultSet.CONCUR_UPDATABLE`.
- Implemented `Clob.setAsciiStream()`.
- Removed duplicate code from `UpdatableResultSet` (it can be inherited from `ResultSet`, the extra code for each method to handle updatability I thought might someday be necessary has not been needed).
- Fixed `UnsupportedEncodingException` thrown when “forcing” a character encoding using properties.
- Fixed incorrect conversion in `ResultSet.getLong()`.
- Implemented `ResultSet.updateBlob()`.
- Removed some not-needed temporary object creation by smarter use of `Strings` in `EscapeProcessor`, `Connection` and `DatabaseMetaData` classes.
- Escape `0x5c` character in strings for the SJIS charset.
- `PreparedStatement` now honors stream lengths in setBinary/Ascii/Character Stream() unless you set the connection property `useStreamLengthsInPrepStmts` to `false`.
- Fixed issue with updatable result sets and `PreparedStatements` not working.
- Fixed start position off-by-1 error in `Clob.getSubString()`.
- Added `connectTimeout` parameter that enables users of JDK-1.4 and newer to specify a maximum time to wait to establish a connection.
- Fixed various non-ASCII character encoding issues.
- Fixed `ResultSet.isLast()` for empty result sets (should return `false`).
- Added driver property `useHostsInPrivileges`. Defaults to `true`. Affects whether or not `@hostname` will be used in `DBMD.getColumn/TablePrivileges`.
- Fixed `ResultSet.setFetchDirection(FETCH_UNKNOWN)`.
- Added `queriesBeforeRetryMaster` property that specifies how many queries to issue when failed over before attempting to reconnect to the master (defaults to 50).

- Fixed issue when calling `Statement.setFetchSize()` when using arbitrary values.
- Properly restore connection properties when autoReconnecting or failing-over, including `autoCommit` state, and isolation level.
- Implemented `Clob.truncate()`.

Changes in MySQL Connector/J 3.0.1 (2002-09-21)

Bugs Fixed

- Charsets now automatically detected. Optimized code for single-byte character set conversion.
- Fixed `ResultSetMetaData.isSigned()` for `TINYINT` and `BIGINT`.
- Fixed `RowDataStatic.getAt()` off-by-one bug.
- Fixed `ResultSet.getRow()` off-by-one bug.
- Massive code clean-up to follow Java coding conventions (the time had come).
- Implemented `ResultSet.getCharacterStream()`.
- Added limited `Clob` functionality (`ResultSet.getClob()`, `PreparedStatement.setClob()`, `PreparedStatement.setObject(Clob)`.
- `Connection.isClosed()` no longer “pings” the server.
- `Connection.close()` issues `rollback()` when `getAutoCommit()` is `false`.
- Added `socketTimeout` parameter to URL.
- Added `LOCAL TEMPORARY` to table types in `DatabaseMetaData.getTableTypes()`.
- Added `paranoid` parameter, which sanitizes error messages by removing “sensitive” information from them (such as host names, ports, or user names), as well as clearing “sensitive” data structures when possible.

Changes in MySQL Connector/J 3.0.0 (2002-07-31)

Bugs Fixed

- General source-code cleanup.
- The driver now only works with JDK-1.2 or newer.
- Fix and sort primary key names in `DBMetaData` (SF bugs 582086 and 582086).
- `ResultSet.getTimestamp()` now works for `DATE` types (SF bug 559134).
- Float types now reported as `java.sql.Types.FLOAT` (SF bug 579573).
- Support for streaming (row-by-row) result sets (see `README`) Thanks to Doron.
- Testsuite now uses Junit (which you can get from http://www.junit.org.
- JDBC Compliance: Passes all tests besides stored procedure tests.
- `ResultSet.getDate/Time/Timestamp` now recognizes all forms of invalid values that have been set to all zeros by MySQL (SF bug 586058).
- Added multi-host failover support (see `README`).

- Repackaging: New driver name is `com.mysql.jdbc.Driver`, old name still works, though (the driver is now provided by MySQL-AB).
- Support for large packets (new addition to MySQL-4.0 protocol), see `README` for more information.
- Better checking for closed connections in `Statement` and `PreparedStatement`.
- Performance improvements in string handling and field metadata creation (lazily instantiated) contributed by Alex Twisleton-Wykeham-Fiennes.
- JDBC-3.0 functionality including `Statement/PreparedStatement.getGeneratedKeys()` and `ResultSet.getURL()`.
- Overall speed improvements using controlling transient object creation in `MysqlIO` class when reading packets.
- **!!! LICENSE CHANGE !!!** The driver is now GPL.
- Performance enhancements: Driver is now 50–100% faster in most situations, and creates fewer temporary objects.

Changes in MySQL Connector/J 2.0

Changes in MySQL Connector/J 2.0.14 (2002-05-16)

Bugs Fixed

- `ResultSet.getDouble()` now uses code built into JDK to be more precise (but slower).
- Fixed typo for `relaxAutoCommit` parameter.
- `LogicalHandle.isClosed()` calls through to physical connection.
- Added SQL profiling (to `STDERR`). Set `profileSql=true` in your JDBC URL. See `README` for more information.
- `PreparedStatement` now releases resources on `.close()`. (SF bug 553268)
- More code cleanup.
- Quoted identifiers not used if server version does not support them. Also, if server started with `--ansi` or `--sql-mode=ANSI_QUOTES`, "`"` will be used as an identifier quote character, otherwise "`'`" will be used.

Changes in MySQL Connector/J 2.0.13 (2002-04-24)

Bugs Fixed

- Fixed unicode chars being read incorrectly. (SF bug 541088)
- Faster blob escaping for `PrepStmt`.
- Added `setURL()` to `MySQLXADataSource`. (SF bug 546019)
- Added `set`/`getPortNumber()` to `DataSource(s)`. (SF bug 548167)
- `PreparedStatement.toString()` fixed. (SF bug 534026)
- More code cleanup.
- Rudimentary version of `Statement.getGeneratedKeys()` from JDBC-3.0 now implemented (you need to be using JDK-1.4 for this to work, I believe).

- `DBMetaData.getIndexInfo()` - bad PAGES fixed. (SF BUG 542201)
- `ResultSetMetaData.getColumnClassName()` now implemented.

Changes in MySQL Connector/J 2.0.12 (2002-04-07)

Bugs Fixed

- Fixed `testsuite.Traversal` `afterLast()` bug, thanks to Igor Lastric.
- Added new types to `getTypeInfo()`, fixed existing types thanks to Al Davis and Kid Kalanon.
- Fixed time zone off-by-1-hour bug in `PreparedStatement` (538286, 528785).
- Added identifier quoting to all `DatabaseMetaData` methods that need them (should fix 518108).
- Added support for `BIT` types (51870) to `PreparedStatement`.
- `ResultSet.insertRow()` should now detect auto_increment fields in most cases and use that value in the new row. This detection will not work in multi-valued keys, however, due to the fact that the MySQL protocol does not return this information.
- Relaxed synchronization in all classes, should fix 520615 and 520393.
- `DataSources` - fixed `setUrl` bug (511614, 525565), wrong datasource class name (532816, 528767).
- Added support for `YEAR` type (533556).
- Fixes for `ResultSet` updatability in `PreparedStatement`.
- `ResultSet`: Fixed updatability (values being set to `null` if not updated).
- Added `getTable/ColumnPrivileges()` to DBMD (fixes 484502).
- Added `getIdleFor()` method to `Connection` and `MysqlLogicalHandle`.
- `ResultSet.refreshRow()` implemented.
- Fixed `getRow()` bug (527165) in `ResultSet`.
- General code cleanup.

Changes in MySQL Connector/J 2.0.11 (2002-01-27)

Bugs Fixed

- Full synchronization of `Statement.java`.
- Fixed missing `DELETE_RULE` value in `DBMD.getImported/ExportedKeys()` and `getCrossReference()`.
- More changes to fix `Unexpected end of input stream` errors when reading `BLOB` values. This should be the last fix.

Changes in MySQL Connector/J 2.0.10 (2002-01-24)

Bugs Fixed

- Fixed null-pointer-exceptions when using `MysqlConnectionPoolDataSource` with Websphere 4 (bug 505839).
- Fixed spurious `Unexpected end of input stream` errors in `MysqlIO` (bug 507456).

Changes in MySQL Connector/J 2.0.9 (2002-01-13)

Bugs Fixed

- Fixed extra memory allocation in `MysqlIO.readPacket()` (bug 488663).
- Added detection of network connection being closed when reading packets (thanks to Todd Lizambri).
- Fixed casting bug in `PreparedStatement` (bug 488663).
- `DataSource` implementations moved to `org.gjt.mm.mysql.jdbc2.optional` package, and (initial) implementations of `PooledConnectionDataSource` and `XADataSource` are in place (thanks to Todd Wolff for the implementation and testing of `PooledConnectionDataSource` with IBM WebSphere 4).
- Fixed quoting error with escape processor (bug 486265).
- Removed concatenation support from driver (the `||` operator), as older versions of VisualAge seem to be the only thing that use it, and it conflicts with the logical `||` operator. You will need to start `mysqld` with the `--ansi` flag to use the `||` operator as concatenation (bug 491680).
- `Ant` build was corrupting included `jar` files, fixed (bug 487669).
- Report batch update support through `DatabaseMetaData` (bug 495101).
- Implementation of `DatabaseMetaData.getExported/ImportedKeys()` and `getCrossReference()`.
- Fixed off-by-one-hour error in `PreparedStatement.setTimestamp()` (bug 491577).
- Full synchronization on methods modifying instance and class-shared references, driver should be entirely thread-safe now (please let me know if you have problems).

Changes in MySQL Connector/J 2.0.8 (2001-11-25)

Bugs Fixed

- `XADataSource`/`ConnectionPoolDataSource` code (experimental)
- `DatabaseMetaData.getPrimaryKeys()` and `getBestRowIdentifier()` are now more robust in identifying primary keys (matches regardless of case or abbreviation/full spelling of `Primary Key` in `Key_type` column).
- Batch updates now supported (thanks to some inspiration from Daniel Rall).
- `PreparedStatement.setAnyNumericType()` now handles positive exponents correctly (adds `+` so MySQL can understand it).

Changes in MySQL Connector/J 2.0.7 (2001-10-24)

Bugs Fixed

- Character sets read from database if `useUnicode=true` and `characterEncoding` is not set. (thanks to Dmitry Vereshchagin)
- Initial transaction isolation level read from database (if available). (thanks to Dmitry Vereshchagin)
- Fixed `PreparedStatement` generating SQL that would end up with syntax errors for some queries.
- `PreparedStatement.setCharacterStream()` now implemented

- Capitalize type names when `capitalizeTypeNames=true` is passed in URL or properties (for WebObjects. (thanks to Anjo Krank)
- `ResultSet.getBlob()` now returns `null` if column value was `null`.
- Fixed `ResultSetMetaData.getPrecision()` returning one less than actual on newer versions of MySQL.
- Fixed dangling socket problem when in high availability (`autoReconnect=true`) mode, and finalizer for `Connection` will close any dangling sockets on GC.
- Fixed time zone issue in `PreparedStatement.setTimestamp()`. (thanks to Erik Olofsson)
- PreparedStatement.setDouble() now uses full-precision doubles (reverting a fix made earlier to truncate them).
- Fixed `DatabaseMetaData.supportsTransactions()`, and `supportsTransactionIsolationLevel()` and `getTypeInfo()` `SQL_DATETIME_SUB` and `SQL_DATA_TYPE` fields not being readable.
- Updatable result sets now correctly handle `NULL` values in fields.
- PreparedStatement.setBoolean() will use 1/0 for values if your MySQL version is 3.21.23 or higher.
- Fixed `ResultSet.isAfterLast()` always returning `false`.

Changes in MySQL Connector/J 2.0.6 (2001-06-16)

Bugs Fixed

- Fixed `PreparedStatement` parameter checking.
- Fixed case-sensitive column names in `ResultSet.java`.

Changes in MySQL Connector/J 2.0.5 (2001-06-13)

Bugs Fixed

- `ResultSet.insertRow()` works now, even if not all columns are set (they will be set to `NULL`).
- Added `Byte` to `PreparedStatement.setObject()`.
- Fixed data parsing of `TIMESTAMP` values with 2-digit years.
- Added `ISOLATION` level support to `Connection.setIsolationLevel()`
- `DataBaseMetaData.getCrossReference()` no longer `ArrayIndexOOB`.
- `ResultSet.getBoolean()` now recognizes `-1` as `true`.
- `ResultSet` has +/-Inf/inf support.
- `getObject()` on `ResultSet` correctly does `TINYINT`->`Byte` and `SMALLINT`->`Short`.
- Fixed `ResultSetMetaData.getColumnTypeName` for `TEXT`/`BLOB`.
- Fixed `ArrayIndexOutOfBounds` when sending large `BLOB` queries. (Max size packet was not being set)
- Fixed NPE on `PreparedStatement.executeUpdate()` when all columns have not been set.
- Fixed `ResultSet.getBlob()` `ArrayIndex` out-of-bounds.

Changes in MySQL Connector/J 2.0.3 (2000-12-03)

Bugs Fixed

- Fixed composite key problem with updatable result sets.
- Faster ASCII string operations.
- Fixed off-by-one error in `java.sql.Blob` implementation code.
- Fixed incorrect detection of `MAX_ALLOWED_PACKET`, so sending large blobs should work now.
- Added detection of -/+INF for doubles.
- Added `ultraDevHack` URL parameter, set to `true` to enable (broken) Macromedia UltraDev to use the driver.
- Implemented `getBigDecimal()` without scale component for JDBC2.

Changes in MySQL Connector/J 2.0.1 (2000-04-06)

Bugs Fixed

- Columns that are of type `TEXT` now return as `Strings` when you use `getObject()`.
- Cleaned up exception handling when driver connects.
- Fixed `RSMD.isWritable()` returning wrong value. Thanks to Moritz Maass.
- `DatabaseMetaData.getPrimaryKeys()` now works correctly with respect to `key_seq`. Thanks to Brian Slesinsky.
- Fixed many JDBC-2.0 traversal, positioning bugs, especially with respect to empty result sets. Thanks to Ron Smits, Nick Brook, Cessar Garcia and Carlos Martinez.
- No escape processing is done on `PreparedStatements` anymore per JDBC spec.
- Fixed some issues with updatability support in `ResultSet` when using multiple primary keys.

Changes in MySQL Connector/J 2.0.0pre5 (21 February 2000)

- Fixed Bad Handshake problem.

Changes in MySQL Connector/J 2.0.0pre4 (10 January 2000)

- Fixes to ResultSet for insertRow() - Thanks to Cesar Garcia
- Fix to Driver to recognize JDBC-2.0 by loading a JDBC-2.0 class, instead of relying on JDK version numbers. Thanks to John Baker.
- Fixed ResultSet to return correct row numbers
- Statement.getUpdateCount() now returns rows matched, instead of rows actually updated, which is more SQL-92 like.

10-29-99

- Statement/PreparedStatement.getMoreResults() bug fixed. Thanks to Noel J. Bergman.
- Added Short as a type to PreparedStatement.setObject(). Thanks to Jeff Crowder
- Driver now automagically configures maximum/preferred packet sizes by querying server.

- Autoreconnect code uses fast ping command if server supports it.
- Fixed various bugs with respect to packet sizing when reading from the server and when alloc'ing to write to the server.

Changes in MySQL Connector/J 2.0.0pre (17 August 1999)

- Now compiles under JDK-1.2. The driver supports both JDK-1.1 and JDK-1.2 at the same time through a core set of classes. The driver will load the appropriate interface classes at runtime by figuring out which JVM version you are using.
- Fixes for result sets with all nulls in the first row. (Pointed out by Tim Endres)
- Fixes to column numbers in SQLExceptions in ResultSet (Thanks to Blas Rodriguez Somoza)
- The database no longer needs to specified to connect. (Thanks to Christian Motschke)

Changes in MySQL Connector/J 1.2 and lower

Changes in MySQL Connector/J 1.2b (04 July 1999)

- Better Documentation (in progress), in doc/mm.doc/book1.html
- DBMD now permits null for a column name pattern (not in spec), which it changes to '%'.
- DBMD now has correct types/lengths for getXXX().
- ResultSet.getDate(), getTime(), and getTimestamp() fixes. (contributed by Alan Wilken)
- EscapeProcessor now handles \{ \} and { or } inside quotation marks correctly. (thanks to Alik for some ideas on how to fix it)
- Fixes to properties handling in Connection. (contributed by Juho Tikkala)
- ResultSet.getObject() now returns null for NULL columns in the table, rather than bombing out. (thanks to Ben Grosman)
- ResultSet.getObject() now returns Strings for types from MySQL that it doesn't know about. (Suggested by Chris Perdue)
- Removed DataInput/Output streams, not needed, 1/2 number of method calls per IO operation.
- Use default character encoding if one is not specified. This is a work-around for broken JVMs, because according to spec, EVERY JVM must support "ISO8859_1", but they do not.
- Fixed Connection to use the platform character encoding instead of "ISO8859_1" if one isn't explicitly set. This fixes problems people were having loading the character- converter classes that didn't always exist (JVM bug). (thanks to Fritz Elfert for pointing out this problem)
- Changed MysqlIO to re-use packets where possible to reduce memory usage.
- Fixed escape-processor bugs pertaining to {} inside quotation marks.

Changes in MySQL Connector/J 1.2a (14 April 1999)

- Fixed character-set support for non-Javasoft JVMs (thanks to many people for pointing it out)
- Fixed ResultSet.getBoolean() to recognize 'y' & 'n' as well as '1' & '0' as boolean flags. (thanks to Tim Pizey)
- Fixed ResultSet.getTimestamp() to give better performance. (thanks to Richard Swift)

- Fixed getByte() for numeric types. (thanks to Ray Bellis)
- Fixed DatabaseMetaData.getTypeInfo() for DATE type. (thanks to Paul Johnston)
- Fixed EscapeProcessor for "fn" calls. (thanks to Piyush Shah at locomotive.org)
- Fixed EscapeProcessor to not do extraneous work if there are no escape codes. (thanks to Ryan Gustafson)
- Fixed Driver to parse URLs of the form "jdbc:mysql://host:port" (thanks to Richard Lobb)

Changes in MySQL Connector/J 1.1i (24 March 1999)

- Fixed Timestamps for PreparedStatements
- Fixed null pointer exceptions in RSMD and RS
- Re-compiled with jikes for valid class files (thanks ms!)

Changes in MySQL Connector/J 1.1h (08 March 1999)

- Fixed escape processor to deal with unmatched { and } (thanks to Craig Coles)
- Fixed escape processor to create more portable (between DATETIME and TIMESTAMP types) representations so that it will work with BETWEEN clauses. (thanks to Craig Longman)
- MysqlIO.quit() now closes the socket connection. Before, after many failed connections some OS's would run out of file descriptors. (thanks to Michael Brinkman)
- Fixed NullPointerException in Driver.getPropertyInfo. (thanks to Dave Potts)
- Fixes to MysqlDefs to allow all *text fields to be retrieved as Strings. (thanks to Chris at Leverage)
- Fixed setDouble in PreparedStatement for large numbers to avoid sending scientific notation to the database. (thanks to J.S. Ferguson)
- Fixed getScale() and getPrecision() in RSMD. (contrib'd by James Klicman)
- Fixed getObject() when field was DECIMAL or NUMERIC (thanks to Bert Hobbs)
- DBMD.getTables() bombed when passed a null table-name pattern. Fixed. (thanks to Richard Lobb)
- Added check for "client not authorized" errors during connect. (thanks to Hannes Wallnoefer)

Changes in MySQL Connector/J 1.1g (19 February 1999)

- Result set rows are now byte arrays. Blobs and Unicode work bidirectonally now. The useUnicode and encoding options are implemented now.
- Fixes to PreparedStatement to send binary set by setXXXStream to be sent untouched to the MySQL server.
- Fixes to getDriverPropertyInfo().

Changes in MySQL Connector/J 1.1f (31 December 1998)

- Changed all ResultSet fields to Strings, this should allow Unicode to work, but your JVM must be able to convert between the character sets. This should also make reading data from the server be a bit quicker, because there is now no conversion from StringBuffer to String.
- Changed PreparedStatement.streamToString() to be more efficient (code from Uwe Schaefer).

- URL parsing is more robust (throws SQL exceptions on errors rather than NullPointerExceptions)
- PreparedStatement now can convert Strings to Time/Date values using setObject() (code from Robert Currey).
- IO no longer hangs in Buffer.readInt(), that bug was introduced in 1.1d when changing to all byte-arrays for result sets. (Pointed out by Samo Login)

Changes in MySQL Connector/J 1.1b (03 November 1998)

- Fixes to DatabaseMetaData to allow both IBM VA and J-Builder to work. Let me know how it goes. (thanks to Jac Kersing)
- Fix to ResultSet.getBoolean() for NULL strings (thanks to Barry Lagerweij)
- Beginning of code cleanup, and formatting. Getting ready to branch this off to a parallel JDBC-2.0 source tree.
- Added "final" modifier to critical sections in MysqlIO and Buffer to allow compiler to inline methods for speed.

9-29-98

- If object references passed to setXXX() in PreparedStatement are null, setNull() is automatically called for you. (Thanks for the suggestion goes to Erik Ostrom)
- setObject() in PreparedStatement will now attempt to write a serialized representation of the object to the database for objects of Types.OTHER and objects of unknown type.
- Util now has a static method readObject() which given a ResultSet and a column index will re-instantiate an object serialized in the above manner.

Changes in MySQL Connector/J 1.1 (02 September 1998)

- Got rid of "ugly hack" in MysqlIO.nextRow(). Rather than catch an exception, Buffer.isLastDataPacket() was fixed.
- Connection.getCatalog() and Connection.setCatalog() should work now.
- Statement.setMaxRows() works, as well as setting by property maxRows. Statement.setMaxRows() overrides maxRows set using properties or url parameters.
- Automatic re-connection is available. Because it has to "ping" the database before each query, it is turned off by default. To use it, pass in "autoReconnect=true" in the connection URL. You may also change the number of reconnect tries, and the initial timeout value using "maxReconnects=n" (default 3) and "initialTimeout=n" (seconds, default 2) parameters. The timeout is an exponential backoff type of timeout; for example, if you have initial timeout of 2 seconds, and maxReconnects of 3, then the driver will timeout 2 seconds, 4 seconds, then 16 seconds between each re-connection attempt.

Changes in MySQL Connector/J 1.0 (24 August 1998)

- Fixed handling of blob data in Buffer.java
- Fixed bug with authentication packet being sized too small.
- The JDBC Driver is now under the LGPL

8-14-98

- Fixed Buffer.readLenString() to correctly read data for BLOBS.

- Fixed PreparedStatement.stringToStream to correctly read data for BLOBS.
- Fixed PreparedStatement.setDate() to not add a day. (above fixes thanks to Vincent Partington)
- Added URL parameter parsing (?user=... and so forth).

Changes in MySQL Connector/J 0.9d (04 August 1998)

- Big news! New package name. Tim Endres from ICE Engineering is starting a new source tree for GNU GPL'd Java software. He's graciously given me the org.gjt.mm package directory to use, so now the driver is in the org.gjt.mm.mysql package scheme. I'm "legal" now. Look for more information on Tim's project soon.
- Now using dynamically sized packets to reduce memory usage when sending commands to the DB.
- Small fixes to getTypeInfo() for parameters, and so forth.
- DatabaseMetaData is now fully implemented. Let me know if these drivers work with the various IDEs out there. I've heard that they're working with JBuilder right now.
- Added JavaDoc documentation to the package.
- Package now available in .zip or .tar.gz.

Changes in MySQL Connector/J 0.9 (28 July 1998)

- Implemented getTypeInfo(). Connection.rollback() now throws an SQLException per the JDBC spec.
- Added PreparedStatement that supports all JDBC API methods for PreparedStatement including InputStreams. Please check this out and let me know if anything is broken.
- Fixed a bug in ResultSet that would break some queries that only returned 1 row.
- Fixed bugs in DatabaseMetaData.getTables(), DatabaseMetaData.getColumns() and DatabaseMetaData.getCatalogs().
- Added functionality to Statement that enables executeUpdate() to store values for IDs that are automatically generated for AUTO_INCREMENT fields. Basically, after an executeUpdate(), look at the SQLWarnings for warnings like "LAST_INSERTED_ID = 'some number', COMMAND = 'your SQL query'". If you are using AUTO_INCREMENT fields in your tables and are executing a lot of executeUpdate()s on one Statement, be sure to clearWarnings() every so often to save memory.

Changes in MySQL Connector/J 0.8 (06 July 1998)

- Split MysqlIO and Buffer to separate classes. Some ClassLoaders gave an IllegalAccess error for some fields in those two classes. Now mm.mysql works in applets and all classloaders. Thanks to Joe Ennis <jce@mail.boone.com> for pointing out the problem and working on a fix with me.

Changes in MySQL Connector/J 0.7 (01 July 1998)

- Fixed DatabaseMetadata problems in getColumns() and bug in switch statement in the Field constructor. Thanks to Costin Manolache <costin@tdiinc.com> for pointing these out.

Changes in MySQL Connector/J 0.6 (21 May 1998)

- Incorporated efficiency changes from Richard Swift <Richard.Swift@kanatek.ca> in `MysqlIO.java` and `ResultSet.java`:
- We're now 15% faster than gwe's driver.
- Started working on `DatabaseMetaData`.

- The following methods are implemented:
  - `getTables()`
  - `getTableTypes()`
  - `getColumns()`
  - `getCatalogs()`