
MySQL NDB Cluster 9.7 Release Notes

Abstract

This document contains release notes for the changes in each release of MySQL NDB Cluster that uses version 9.7 of the [NDB \(NDBCLUSTER\)](#) storage engine.

Each NDB Cluster 9.7 release is based on a mainline MySQL Server release and a particular version of the [NDB](#) storage engine, as shown in the version string returned by executing `SELECT VERSION()` in the `mysql` client, or by executing the `ndb_mgm` client `SHOW` or `STATUS` command; for more information, see [MySQL NDB Cluster 9.7](#).

For general information about features added in NDB Cluster 9.7, see [What is New in MySQL NDB Cluster 9.7](#). For a complete list of all bug fixes and feature changes in MySQL NDB Cluster, please refer to the changelog section for each individual NDB Cluster release.

For additional MySQL 9.7 documentation, see the [MySQL 9.7 Reference Manual](#), which includes an overview of features added in MySQL 9.7 that are not specific to NDB Cluster ([What Is New in MySQL 9.7](#)), and discussion of upgrade issues that you may encounter for upgrades from MySQL 9.6 to MySQL 9.7 ([Changes in MySQL 9.7](#)). For a complete list of all bug fixes and feature changes made in MySQL 9.7 that are not specific to [NDB](#), see [MySQL 9.7 Release Notes](#).

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 (<https://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 legal information, see the [Legal Notices](#).

For help with using MySQL, please visit the [MySQL Forums](#), where you can discuss your issues with other MySQL users.

Document generated on: 2026-06-10 (revision: 31185)

Table of Contents

| | |
|---|----|
| Preface and Legal Notices | 1 |
| Changes in MySQL NDB Cluster 9.7.0 (2026-04-23) | 3 |
| Changes in MySQL NDB Cluster 9.6.0 (2026-01-22) | 3 |
| Changes in MySQL NDB Cluster 9.5.0 (2025-10-23) | 4 |
| Changes in MySQL NDB Cluster 9.4.0 (2025-07-23) | 6 |
| Changes in MySQL NDB Cluster 9.3.0 (2025-04-16) | 9 |
| Changes in MySQL NDB Cluster 9.2.0 (2025-01-22) | 12 |
| Changes in MySQL NDB Cluster 9.1.0 (2024-10-16) | 15 |
| Changes in MySQL NDB Cluster 9.0.1 (2024-07-23) | 17 |
| Changes in MySQL NDB Cluster 9.0.0 (2024-07-02) | 17 |
| Index | 21 |

Preface and Legal Notices

This document contains release notes for the changes in each release of MySQL NDB Cluster that uses version 9.7 of the [NDB](#) storage engine.

Legal Notices

Copyright © 1997, 2026, Oracle and/or its affiliates.

License Restrictions

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

Hazardous Applications Notice

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Trademark Notice

Oracle, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

Third-Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Use of This Documentation

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

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 for Accessibility

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 NDB Cluster 9.7.0 (2026-04-23)

MySQL NDB Cluster 9.7.0 is a new Innovation release of NDB Cluster, based on MySQL Server 9.7 and including features in version 9.6 of the [NDB](#) storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 9.7. NDB Cluster 9.7 source code and binaries can be obtained from <https://dev.mysql.com/downloads/cluster/>.

For an overview of major changes made in NDB Cluster 9.7, see [What is New in MySQL NDB Cluster 9.7](#).

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 9 through MySQL 9.7.0 (see [Changes in MySQL NDB Cluster 9.7.0 \(2026-04-23\)](#)).



Note

These release notes were created with the assistance of MySQL HeatWave GenAI.

Changes in MySQL NDB Cluster 9.6.0 (2026-01-22)

MySQL NDB Cluster 9.6.0 is a new Innovation release of NDB Cluster, based on MySQL Server 9.6 and including features in version 9.6 of the [NDB](#) storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 9.6. NDB Cluster 9.6 source code and binaries can be obtained from <https://dev.mysql.com/downloads/cluster/>.

For an overview of major changes made in NDB Cluster 9.6, see [What is New in MySQL NDB Cluster 9.7](#).

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 9 through MySQL 9.6.0 (see [Changes in MySQL NDB Cluster 9.6.0 \(2026-01-22\)](#)).

**Note**

These release notes were created with the assistance of MySQL HeatWave GenAI.

- [Functionality Added or Changed](#)
- [Bugs Fixed](#)

Functionality Added or Changed

- Reporting of File Close errors has been improved. (Bug #38726643)
- The option `--skip-fk-checks` was added to `ndb_restore`. This option modifies the behavior of `--rebuild-indexes` so that, when foreign keys are re-enabled, the existing data in the table is not checked for consistency. (Bug #38593666)
- The logs generated for backup and restore operations are improved in this release. (Bug #38592288)

Bugs Fixed

- The NDB purge process responsible for removing rows from `mysql.ndb_binlog_index` when a binary log file is purged, assumed that `autocommit` is enabled, and each DML operation commits immediately. However, if `autocommit` is disabled globally, the purge process inherited this value and purge operations were not committed.

As of this release, `autocommit` is enabled for the purge session, even when the global setting is disabled. (Bug #38488185)

- NDB data nodes failed to start if either of the `NDBCNTR` or `DBDIH` system files were empty. (Bug #38424959)
- A data node failure occurred due to a reference validity check which failed. (Bug #38422448)
- BackupRecords in a participant node were not released from the pool when a backup was aborted, causing subsequent backup operations to fail. (Bug #38151265)
- Certain Transporter error messages did not include information on the source node. (Bug #37922543)
- The `ndb_restore` log message for total log bytes restored was incorrect (Bug #37697971)
- In the NDB management server, the TLS INFO statistics `upgraded` and `tls` were incremented if TLS authentication had failed. They should only be incremented on success. (Bug #36414579)
- `ndb_log_bin` could be set dynamically, but had no effect. This variable is not dynamic and is only checked when the binlog thread starts. (Bug #32860560)

Changes in MySQL NDB Cluster 9.5.0 (2025-10-23)

MySQL NDB Cluster 9.5.0 is a new Innovation release of NDB Cluster, based on MySQL Server 9.6 and including features in version 9.5 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 9.5. NDB Cluster 9.5 source code and binaries can be obtained from <https://dev.mysql.com/downloads/cluster/>.

For an overview of major changes made in NDB Cluster 9.5, see [What is New in MySQL NDB Cluster 9.7](#).

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 9 through MySQL 9.5.0 (see [Changes in MySQL NDB Cluster 9.5.0 \(2025-10-23\)](#)).



Note

These release notes were created with the assistance of MySQL HeatWave GenAI.

- [Deprecation and Removal Notes](#)
- [Functionality Added or Changed](#)
- [Bugs Fixed](#)

Deprecation and Removal Notes

- Support for `Ndb.cfg` configuration files, deprecated in MySQL 9.1, has been removed; such files are no longer read by any NDB Cluster executable.

For further information, see [NDB Cluster Connection Strings](#). (WL #16302)

Functionality Added or Changed

- Data node logging is improved with the following:
 - Local logs for data node join and leave events.
 - API disconnection handling logs.
 (Bug #38414245)
- MySQL NDB Cluster now generates log entries for redo logging issues, including `redo log buffer exhaustion`, `redo log space exhaustion`, and `file change problems`, which can cause transactions to be aborted, queued, or delayed. A secondary overload control mechanism monitors the rate at which the redo log part's Redo buffer is flushed to disk and estimates the time required to complete writing all data in the part's Redo buffer. (Bug #37903091)
- `LogLevelBackup` and `LogLevelSchema` Data node options were added in this release. (Bug #28004136)

Bugs Fixed

- If a data node restarted during a backup, the remaining data nodes could hang and the backup process stopped. The backup was marked as failed, and other nodes did not terminate their backup process, preventing further backups from being initiated. (Bug #38316252)
- It was not possible to restore backups with `BackupID` values greater than 2147483647, with the `ndb_restore` tool. Errors were returned similar to the following:

```
ndb_restore: [Warning] option 'backupid': signed value
3000000000 adjusted to 2147483647. Failed to find backup
2147483647.
```

(Bug #38260769)

- When running `CREATE` or `INSERT` statements on a NDB Binlog server with `sql_log_bin = 0`, replication would fail due to a table not existing. (Bug #37953883)
- Concurrent execution of `TRUNCATE TABLE` statements and starting an Ndb backup could cause the statements to hang indefinitely, affecting cluster availability. Errors were returned similar to the following:

```
ERROR 1296 (HY000): Got error 761 'Unable to drop table as
backup is in progress' from NDBCLUSTER
```

(Bug #37486661)

- Setting log level options for data nodes in MySQL NDB Cluster, such as [LogLevelCheckpoint](#), [LogLevelStatistic](#), [LogLevelInfo](#), [LogLevelCongestion](#), [LogLevelError](#), and [LogLevelConnection](#) to a higher value does not provide additional log information beyond their default values. Errors were returned similar to the following:

```
Ndb kernel thread 0 is stuck in: Job Handling elapsed=100,
Watchdog: Warning overslept
```

(Bug #17847063)

- The MySQL NDB Cluster's example code for [MGMAPI_LOGEVENT](#) was intended to compile as plain C, but had a .cpp extension and included a C++ header. The example file has been renamed to [main.c](#) and the C++ header has been removed. (Bug #12678874)

Changes in MySQL NDB Cluster 9.4.0 (2025-07-23)

MySQL NDB Cluster 9.4.0 is a new Innovation release of NDB Cluster, based on MySQL Server 9.4 and including features in version 9.4 of the [NDB](#) storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 9.4. NDB Cluster 9.4 source code and binaries can be obtained from <https://dev.mysql.com/downloads/cluster/>.

For an overview of major changes made in NDB Cluster 9.4, see [What is New in MySQL NDB Cluster 9.7](#).

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 9 through MySQL 9.4.0 (see [Changes in MySQL NDB Cluster 9.4.0 \(2025-07-23\)](#)).



Note

These release notes were created with the assistance of MySQL HeatWave GenAI.

- [Functionality Added or Changed](#)
- [Bugs Fixed](#)

Functionality Added or Changed

- **NDB Replication:** It is now possible to divide binary logging for a MySQL Cluster or for individual [NDBCLUSTER](#) tables into equal portions or “slices” between multiple MySQL servers.

For dividing binary logging for an entire cluster into slices, this [NDB](#) release implements two [mysqld](#) startup options. The [--ndb-log-row-slice-count](#) option determines the number of slices, and thus the number of servers sharing binary logging. [--ndb-log-row-slice-id](#) identifies the slice for which this MySQL server is responsible. See the descriptions of these options for more information.

Division of binary logging into slices can be performed for a specific [NDB](#) table by adding rows to the [ndb_replication](#) table with appropriate values for the [binlog_row_slice_count](#) and [binlog_row_slice_id](#) columns which have been added to this table. If you are upgrading an

existing setup, you may need to perform an ALTER TABLE statement, or drop and re-create the table to obtain this functionality. For further information and examples, see [Per-table binary log slicing](#). (WL #15413)

- **MySQL NDB ClusterJ:** The handling of "table not found" conditions in MySQL NDB Cluster has been improved for consistency. Previously, ClusterJ sometimes threw a `ClusterJUserException` and at other times a `ClusterJDatastoreException` when a table was not found. Now, all such conditions are reported using a new exception class, `ClusterJTableException`, after checking the NDB error code. Additionally, a new property, `com.mysql.clusterj.table.wait.msec`, has been introduced to allow configuration of a wait-and-retry loop for "table not found" conditions.

The new feature also allows a better handling of schema changes that take place on a NDB Cluster. (Bug #37884530)

References: See also: Bug #37861635, Bug #118032.

- **MySQL NDB ClusterJ:** A single `SessionFactory` can now handle multiple databases. This is achieved through a new variant of `SessionFactory.getSession()` that takes a database name as its argument. In this new design, an umbrella `SessionFactory` manages a set of `SessionFactory` objects, each of which connects to a single database through a shared pool of connections, so that connections to databases in the same NDB cluster share the same connection.

This new feature is not enabled by default, but requires setting the property `com.mysql.clusterj.multi.database` to true, and `com.mysql.clusterj.connection.pool.size` must not be 0 (default and recommended value is 1). (Bug #37792409)

References: See also: Bug #36974092, Bug #115884.

- **MySQL NDB ClusterJ:** When a user created two `ClusterJ SessionFactory` objects with the same connection string but different database names, two different `Ndb_cluster_connection` objects were created for the two databases, which consumed a lot of extra resources.

Starting from 9.4.0, different `SessionFactory` objects will share the same underlying connection from a global connection pool when connecting to different databases in the same NDB cluster, reducing resource utilization and increasing performance. In this new implementation, each NDB Cluster now has a system name, and the connection to a cluster is by a `Connection.java` interface, which is returned by a new `Session.getConnection()` method. See the [ClusterJ API Reference](#) for details. (Bug #36974092)

References: See also: Bug #37792409, Bug #117893.

- **MySQL NDB ClusterJ:** ClusterJ has also been improved by having the implementations of the `finalize()` method replaced by the Cleaner API. (Bug #30719755)
- **MySQL NDB ClusterJ:** Enhancements have been made to improve the scalability of ClusterJ applications when using the `session.newInstance` and `session.release` interfaces. A new session cache, which is a cache of Ndb objects maintained in `DbFactoruImpl`, has been introduced, reducing an application's overhead and improving its performance. The cache size can be controlled using the `com.mysql.clusterj.max.cached.instances` configuration parameter, which enables the session cache by default with a maximum size of 100. (Bug #102510, Bug #32474777)
- Timestamps in NDB node logs are now printed by default using microsecond resolution. Data nodes can enable this feature explicitly with the data node `--ndb-log-timestamps=UTC` option; management nodes can also do so using the `ndb_mgmd` option `--ndb-log-timestamps=UTC`. For backwards compatible behavior, set this option to `LEGACY`, which uses the system time zone and resolution in seconds, as in releases prior to NDB 8.4.6; in NDB 9.4, `UTC` is the default.

For SQL nodes, use the `--log-timestamps` option; be aware that this `mysqld` option does not support `LEGACY` as a value.

See [NDB Cluster Log Messages](#), for more information. (Bug #37924338)

- MySQL Cluster now supports explicit default expressions for columns of [NDB](#) tables used with the MySQL server. For example, you can now create a an [NDB](#) table like the one shown here:

```
CREATE TABLE t1 (
  i INT          DEFAULT 0,
  c VARCHAR(10) DEFAULT '',
  f FLOAT        DEFAULT (RAND() * RAND()),
  b BINARY(16)   DEFAULT (UUID_TO_BIN(UUID())),
  d DATE         DEFAULT (CURRENT_DATE + INTERVAL 1 YEAR)
) ENGINE=NDBCLUSTER;
```

For information about supported column data types and default behavior, see [Data Type Default Values](#).



Note

Default column value expressions are supported only when [NDB](#) tables are accessed using the MySQL server (`mysqld`), and are not visible to [NDB API](#) applications.

(WL #16678)

Bugs Fixed

- NDB Client Programs:** `ndb_restore`, when applying the log, allowed an infinite number of retries due to temporary errors, rather than limiting these to 11 as expected. (Bug #37883579)

References: This issue is a regression of: Bug #31546136.

- NDB Client Programs:** When restoring from backup with `DefaultOperationRedoProblemAction=ABORT`, an error in data node handling of a redo log part overload condition resulted in an incorrect error code (626 in this case) being sent back to `ndb_restore`, which caused `ndb_restore` to exit prematurely since it did not expect such an error. (Bug #37687485)

References: This issue is a regression of: Bug #13219, Bug #13930, Bug #13980.

- MySQL NDB ClusterJ:** Handling of schema changes failed when multiple threads were involved. To improve the handling, `session.unloadSchema()` has been changed, and three new public methods have been defined: `boolean isStaleMetadata()`, `boolean isSchemaChangePending()`, `void awaitSchemaChange()`. When ClusterJ detects that a schema change has caused a domain type handler to become invalid, it throws an `ClusterJDatastoreException`, and the application can now do the following:
 - Call `isStaleMetadata()`.
 - If `isStaleMetadata()` returns true, the exception was caused by stale metadata due to schema changes. The thread can then call `session.unloadSchema()`. When multiple threads are involved, the first thread to enter `unloadSchema()` will acquire a lock and refresh the metadata. Any other threads calling `getDomainTypeHandler()` for the domain object while the refresh is in progress receive an exception. The threads can then retry the data operation when the refresh is finished.

After the table metadata has been refreshed, ClusterJ writes a log message containing the former and current version numbers of the table metadata.

- If `isStaleMetadata()` returns false, then user can call `isSchemaChangePending()` to see if another thread is trying to refresh the metadata. If that is the case, the user can use `awaitSchemaChange()` to wait for the lock to be released before it retries the data operation.

Our thanks to Salman Niazi of Hopsworks for contributing the changes and tests for `Session.unloadSchema()`, which make this feature possible. (Bug #37861635, Bug #37856493)

- **MySQL NDB ClusterJ:** Added support for using `LONGVARCHAR` and `DATE` types as primary keys in Cluster/J.

Our thanks to Mikael Ronström of Hopsworks for the contribution. (Bug #37782638)

- **MySQL NDB ClusterJ:** Logging for ClusterJ has been improved by adopting the pattern in `java.util.logging` of Java 1.8, in which a log message is constructed only when the corresponding log level has been enabled, thus improving the efficiency of logging. (Bug #37722667)
- **MySQL NDB ClusterJ:** When users attempted to set a `NOT NULL` string column to `NULL`, an empty string was inserted.

Our thanks to Salman Niazi of Hopsworks for contributing the fix for the issue. (Bug #117205, Bug #37476251)

- Improved password handling for file system encryption.

Our thanks to Axel Svensson for the contribution. (Bug #37909595)

- `CREATE TABLESPACE`, when the specified logfile group did not exist, was rejected with error `723 No such table existed`, which did not correctly identify the source of the problem with the SQL statement. Now in such cases, the server issues Error `789 Logfile group not found`. (Bug #37802388)
- Warning `1296 The temporary named table ... already exists` showed the table and schema names in the wrong order.

Our thanks to Axel Svensson for the contribution. (Bug #117918, Bug #37807409)

Changes in MySQL NDB Cluster 9.3.0 (2025-04-16)

MySQL NDB Cluster 9.3.0 is a new Innovation release of NDB Cluster, based on MySQL Server 9.3 and including features in version 9.3 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 9.3. NDB Cluster 9.3 source code and binaries can be obtained from <https://dev.mysql.com/downloads/cluster/>.

For an overview of major changes made in NDB Cluster 9.3, see [What is New in MySQL NDB Cluster 9.7](#).

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 9 through MySQL 9.3.0 (see [Changes in MySQL NDB Cluster 9.3.0 \(2025-04-16\)](#)).

- [Compilation Notes](#)
- [Functionality Added or Changed](#)
- [Bugs Fixed](#)

Compilation Notes

- The ability to build the source without NDB using the internal script `storage/ndb/compile-cluster` was adversely affected by work done in NDB 8.0.31 making the `ndbcluster` plugin part of its default build. (Bug #117215, Bug #37484376)

Functionality Added or Changed

- **MySQL NDB ClusterJ:** For MySQL NDB Cluster 9.3.0 and later, Java 11 or above is required to build and run ClusterJ. (Bug #37213478)
- Added the `Ndb_schema_participant_count` status variable. This variable provides the count of MySQL servers which are currently participating in NDB Cluster schema change distribution. (Bug #37529202)
- The function `ndb_mgm_set_ignore_sigpipe()` has been deprecated, and the MGM API no longer makes SIGPIPE to be ignored by all applications connected to a management node. SIGPIPE can be disabled on BSD platforms that support the socket option `SO_NOSIGPIPE`, and for other platforms (and also all applications on all platforms), a signal handler can be used to ignore SIGPIPE. (Bug #35570336)

Bugs Fixed

- **NDB Replication:** Timestamps written to the binary log included fractional parts (microseconds) although the query being logged did not use any high-precision functionality. This problem was caused by not resetting the state indicating that fractional parts had been used.

We fix this by ensuring that the indicator for the use of microseconds in a given query after having run it is always reset. This avoids the possibility of a later query writing a timestamp which includes microseconds to the binary log when the query as executed did not use microseconds. (Bug #37112446)
- **ndbinfo Information Database:** Certain queries against `ndbinfo` tables were not handled correctly. (Bug #37372650)
- **NDB Client Programs:** With a data node in an unstarted state, such as immediately after executing `node_id RESTART -n` in the `ndb_mgm` client, issuing `ALL REPORT BACKUPSTATUS` in the client subsequently led to an unplanned shutdown of the cluster. (Bug #37505513)
- **MySQL NDB ClusterJ:** The ClusterJ log file only reported the configured, requested node ID for a cluster connection (which was often zero). With this fix, after a connection has been established, ClusterJ reports the actual assigned node ID in the log. (Bug #37556172)
- **MySQL NDB ClusterJ:** A potential circular reference from `NdbRecordSmartValueHandlerImpl` that can cause delays in garbage collection has been removed. (Bug #37361267)
- **MySQL NDB ClusterJ:** Setting the connection property `com.mysql.clusterj.byte.buffer.pool.sizes` to "512, 51200" caused ClusterJ application to fail with a fatal exception thrown by `java.nio.ByteBuffer`. (Bug #37188154)
- **MySQL NDB ClusterJ:** When using a debug build of ClusterJ to run any tests in the testsuite, it exited with the error "1 thread(s) did not exit." (Bug #36383937)
- **MySQL NDB ClusterJ:** Running a ClusterJ application with Java 10 resulted in `java.lang.ClassNotFoundException`, because the class `java.internal.ref.Cleaner` is not available in Java 10. With this fix, the `java.lang.ref.Cleaner` class is used instead for resource cleanup. (Bug #29931569)
- The bundled `libxml2` library has been upgraded to version 2.9.13. (Bug #37806165)
- API node failure is detected by one or more data nodes; data nodes detecting API node failure inform all other data nodes of the failure, eventually triggering API node failure handling on each data node.

Each data node handles API node failure independently; once all internal blocks have completed cleanup, the API node failure is considered handled, and, after a timed delay, the `QMGR` block allows the failed API node's node ID to be used for new connections.

`QMGR` monitors API node failure handling, periodically generating warning logs for API node failure handling that has not completed (approximately every 30 seconds). These logs indicate which blocks have yet to complete failure handling.

This enhancement improves logging in handling stalls particularly with regard to the `DBTC` block, which must roll back or commit and complete the API node's transactions, and release the associated `COMMIT` and `ACK` markers. In addition, the time to wait for API node failure handling is now configurable as the `ApiFailureHandlingTimeout` data node configuration parameter; after this number of seconds, handling is escalated to a data node restart. (Bug #37524092)

References: See also: Bug #37469364.

- When a data node hangs during shutdown reasons for this may include: I/O problems on the node, in which case the thread shutting down hangs while operating on error and trace files; or an error in the shutdown logic, where the thread shutting down raises a Unix signal, and causes a deadlock. When such issues occur, users might observe watchdog warnings in the logs, referring to the last signal processed; this could be misleading in cases where there was actually a (different) preceding cause which had triggered the shutdown.

To help pinpoint the origin of such problems if they occur, we have made the following improvements:

- Added a new watchdog state `shutting down`. This is set early enough in the error handling process that it causes all watchdog logging of shutdown stalls to attribute the delay to a shutdown delay (correctly) rather than problem in execution.
- We have also modified the watchdog mechanism to be aware of shutdown states, and use a more direct path—which is less likely to stall—to force the data node process to stop when needed.

(Bug #37518267)

- When restoring `NDB` tables from backup, it is now possible for `mysqld` to open such tables even if their indexes are not yet available. (Bug #37516858)
- Signal dump code run when handling an unplanned node shutdown sometimes exited unexpectedly when speculatively reading section IDs which might not be present. (Bug #37512526)
- The `LQH_TRANSCONF` signal printer did not validate its input length correctly, which could lead the node process to exit. (Bug #37512477)
- When restoring stored grants (using `ndb_restore --include-stored-grants`) from an `NDB` backup following an initial restart of the data nodes, the `ndb_sql_metadata` table was neither created nor restored. (Bug #37492169)
- Nothing was written to the cluster log to indicate when `PURGE BINARY LOGS` had finished waiting for the purge to complete. (Bug #37489870)
- `WITH_NDB_TLS_SEARCH_PATH` was not set when compiling `NDB Cluster` using `WITHOUT_SERVER`. (Bug #37398657)
- Now, when `ndb_metadata_check` is enabled, we synchronize both schema and tables in the same interval. (Bug #37382551)
- This fix addresses the following two issues:
 1. When a resend could not be started due to a gap created by an out of buffer error in the event stream, forwarding event data for the bucket was not initiated. We fix this by ensure that the takeover process has been initiated before exiting the resend code.
 2. An out of buffer error which occurred during an ongoing resend was not handled. In this case, we now interrupt the resend when such an error is raised.

(Bug #37349305)

- On certain rare occasions, when concurrent calls were made to `release()` and `get()`, instances of `ndb_schema_object` were doubly freed. (Bug #35793818)
- If an out-of-buffer-release (OOBR) process took an excessive amount of time, the reset was performed prematurely, before all buffers were released, thus interfering with concurrent seizing of new pages, beginning new `out_of_buffer` handling, or starting a resend.

We solve this issue by ensuring that resumption of event buffering takes place only after the OOBR process has completed for all buckets. (Bug #20648778)

- Removed the unused InnoDB and NDB handler on `get_tablespace()` method. (Bug #109443, Bug #34916556)

Changes in MySQL NDB Cluster 9.2.0 (2025-01-22)

MySQL NDB Cluster 9.2.0 is a new Innovation release of NDB Cluster, based on MySQL Server 9.2 and including features in version 9.2 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 9.2. NDB Cluster 9.2 source code and binaries can be obtained from <https://dev.mysql.com/downloads/cluster/>.

For an overview of major changes made in NDB Cluster 9.2, see [What is New in MySQL NDB Cluster 9.7](#).

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 9 through MySQL 9.2.0 (see [Changes in MySQL NDB Cluster 9.2.0 \(2025-01-22\)](#)).

- [Compilation Notes](#)
- [Bugs Fixed](#)

Compilation Notes

- **macOS:** A `uint64_t` value used with `%zu` caused a `[-Wformat]` compiler warning on MacOS. (Bug #37174692)
- Removed a warning in `storage/ndb/src/common/util/cstrbuf.cpp`. (Bug #37049014)
- The file `storage/ndb/tools/restore/consumer.cpp` contained dead code (only), and has been removed. (Bug #36237561)

Bugs Fixed

- **Microsoft Windows:** Successive iterations of the sequence `ndb_sign_keys --create-key` followed by `ndb_sign_keys --promote` were unsuccessful on Windows. (Bug #36951132)
- **NDB Disk Data:** `mysqld` did not use a disk scan for NDB tables with 256 disk columns or more. (Bug #37201922)
- **NDB Replication:** The replication applier normally retries temporary errors occurring while applying transactions. Such retry logic is not performed for transactions containing row events where the `STMT_END_F` flag is missing; instead, the statement is committed in an additional step while applying the subsequent `COMMIT` query event when there are still locked tables. Problems arose when committing this statement, because temporary errors were not handled properly. Replica skip error functionality was also affected in that it attempted to skip only the error that occurred when a transaction was committed a second time.

The binary log contains an epoch transaction with writes from multiple server IDs on the source. The replica then uses `IGNORE_SERVER_IDS` (<last_server_id_in_binlog>) to cause the `STMT_END_F` to be filtered away, thus committing the statement from the COMMIT query log event on the applier. Holding a lock on one of the rows to be updated by the applier triggered error handling, which caused replication to stop with an error, with no retries being performed.

We now handle such errors, logging all messages in diagnostics areas (as is already done for row log events) and then retrying the transaction. (Bug #37331118)

- **NDB Replication:** When a MySQL server performing binary logging connects to an NDB Cluster, it checks for existing binary logs; if it finds any, it writes an `Incident` event to a log file of its own so that any downstream replicas can detect the potential for lost events. Problems arose under some circumstances because it was possible for the timestamps of events logged in this file to be out of order; the `Incident` event was written following other events but had a smaller timestamp than these preceding events. We fix this issue by ensuring that a fresh timestamp is used prior to writing an incident to the binary log on startup rather than one which may have been obtained and held for some time previously. (Bug #37228735)
- **NDB Cluster APIs:** The `Ndb_cluster_connection` destructor calls `g_eventLogger::stopAsync()` in order to release the buffers used by the asynchronous logging mechanism as well as to stop the threads responsible for this logging. When the `g_eventLogger` object was deleted before the `Ndb_cluster_connection` destructor was called, the application terminated after trying to use a method on a null object. This could happen in either of two ways:
 - An API program deleted the logger object before deleting the `Ndb_cluster_connection`.
 - `ndb_end()` was called before the `Ndb_cluster_connection` was deleted.

We solve this issue by skipping the call to `stopAsync()` in the `Ndb_cluster_connection` destructor when `g_eventLogger` is `NULL`. This fix also adds a warning to inform API users that deleting `g_eventLogger` before calling the `Ndb_cluster_connection` destructor is incorrect usage.

For more information, see [API Initialization and Cleanup](#). (Bug #37300558)

- **NDB Cluster APIs:** Removed known causes of API node versus data node state misalignments, and improved the handling of state misalignments when detected. In one such case, separate handling of scan errors in the NDB kernel and those originating in API programs led to cleanup not being performed after some scans. Handling of `DBTC` and API state alignment errors has been improved by this set of fixes, as well as scan protocol timeout handling in `DBSPJ`; now, when such misalignments in state are detected, the involved API nodes are disconnected rather than the data node detecting it being forced to shut down. (Bug #20430083, Bug #22782511, Bug #23528433, Bug #28505289, Bug #36273474, Bug #36395384, Bug #36838756, Bug #37022773, Bug #37022901, Bug #37023549)

References: See also: Bug #22782511, Bug #23528433, Bug #36273474, Bug #36395384, Bug #36838756.

- **ndbinfo Information Database:** At table create and drop time, access of `ndbinfo` tables such as `operations_per_fragment` and `memory_per_fragment` sometimes examined data which was not valid.

To fix this, during scans of these `ndbinfo` tables, we ignore any fragments from tables in transient states at such times due to being created or dropped. (Bug #37140331)

- Work done previously to support opening NDB tables with missing indexes was intended to allow the features of the MySQL server to be used to solve problems in cases where indexes cannot be rebuilt due to unmet constraints. With missing indexes, some of the SQL handler functionality is unavailable—for example, the use of indexes to select rows for modification efficiently, or to identify duplicates when processing modifications, or to push joins relying on indexes. This could lead to the unplanned shutdown of an NDB Cluster SQL node.

In such cases, the server now simply returns an error. (Bug #37299071)

- Recent refactoring of the transporter layer added the reporting of the presence of socket shutdown errors, but not their nature. This led to confusion in the common case where a socket shutdown is requested, but the socket is already closed by the peer. To avoid such confusion, this logging has been removed. (Bug #37243135)

References: This issue is a regression of: Bug #35750771.

- It was not possible to create an [NDB](#) table with 256 or more [BLOB](#) columns when also specifying a reduced inline size, as in the following SQL statement:

```
CREATE TABLE t1 (
  pk INT PRIMARY KEY,
  b1 BLOB COMMENT 'NDB_COLUMN=BLOB_INLINE_SIZE=100',
  b2 BLOB COMMENT 'NDB_COLUMN=BLOB_INLINE_SIZE=100',
  ...,
  b256 BLOB COMMENT 'NDB_COLUMN=BLOB_INLINE_SIZE=100'
) ENGINE=NDBCLUSTER;
```

(Bug #37201818)

- In some cases, the occurrence of node failures during shutdown led to the cluster becoming unrecoverable without manual intervention.

We fix this by modifying global checkpoint ID (GCI) information propagation ([CopyGCI](#) mechanism) to reject propagation of any set of GCI information which does not describe the ability to recover the cluster automatically as part of a system restart. (Bug #37163647)

References: See also: Bug #37162636.

- In some cases, node failures during an otherwise graceful shutdown could lead to a cluster becoming unrecoverable without manual intervention. This fix modifies the generic GCI info propagation mechanism ([CopyGCI](#)) to reject propagating any set of GCI information which does not describe the ability to recover a cluster automatically. (Bug #37162636)
- Improved variable names used in [start_resend\(\)](#), and enhanced related debug messages to users and developers with additional information. (Bug #37157987)
- In certain cases, a [COPY_FRAGREQ](#) signal did not honor a fragment scan lock. (Bug #37125935)
- In cases where [NDB](#) experienced an API protocol timeout when attempting to close a scan operation, it considered the [DBTC ApiConnectRecord](#) involved to be lost for further use, at least until the API disconnected and API failure handling within [DBTC](#) reclaimed the record.

This has been improved by having the API send a [TCRELEASEREQ](#) signal to [DBTC](#) in such cases, performing API failure handling for a single [ApiConnectRecord](#) within [DBTC](#). (Bug #37023661)

References: See also: Bug #36273474, Bug #36395384, Bug #37022773, Bug #37022901, Bug #37023549.

- For tables using the [NDB](#) storage engine, the column comment option [BLOB_INLINE_SIZE](#) was silently ignored for [TINYBLOB](#) columns, and (silently) defaulted to the hard-coded 256 byte value regardless of the size provided; this was misleading to users.

To fix this problem, we now specifically disallow [BLOB_INLINE_SIZE](#) on [TINYBLOB](#) columns altogether, and [NDB](#) now prints a warning saying that the column size is defaulting to 256 bytes. (Bug #36725332)

- Testing revealed that a fix for a previous issue which added a check of the [ApiConnectRecord](#) failure number against the system's current failure number did not initialize the [ApiConnectRecord](#) failure number in all cases. (Bug #36155195)

References: This issue is a regression of: Bug #36028828.

- When attempting to alter the default value for a column using `ALTER TABLE INPLACE`, the new value was not persisted in the `NDB` dictionary.

This happened even though changing the default value using `INPLACE` is not supported, and a user wishing to change the default value needs to perform a copying `ALTER TABLE` instead.

Now in such cases we check for this issue, detecting and preventing any attempt at an unsupported `INPLACE` change of the column's default value. (Bug #35106114)

- `ndb_config` did not always handle very long file paths correctly.

Our thanks to Dirkjan Bussink for the contribution. (Bug #116748, Bug #37310680)

- Errors of unknown provenance were logged while assigning node IDs during cluster synchronization, leading to user doubt and concern. Logging of the data node `QMGR` block and the `ndb_mgmd` process relating to node ID allocation issues has therefore been improved, to supply more and better information about what is being reported in such cases. (Bug #116351, Bug #37189356)
- A multi-range scan sometimes lost its fragment lock for the second and subsequent ranges of the scan. (Bug #111932, Bug #35660890)

Changes in MySQL NDB Cluster 9.1.0 (2024-10-16)

MySQL NDB Cluster 9.1.0 is a new Innovation release of NDB Cluster, based on MySQL Server 9.1 and including features in version 9.1 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 9.1. NDB Cluster 9.1 source code and binaries can be obtained from <https://dev.mysql.com/downloads/cluster/>.

For an overview of major changes made in NDB Cluster 9.1, see [What is New in MySQL NDB Cluster 9.7](#).

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 9 through MySQL 9.1.0 (see [Changes in MySQL NDB Cluster 9.1.0 \(2024-10-16\)](#)).

- [Deprecation and Removal Notes](#)
- [Functionality Added or Changed](#)
- [Bugs Fixed](#)

Deprecation and Removal Notes

- Use of an `Ndb.cfg` file for setting the connection string for an NDB process was not well documented or supported. With this release, this file is now formally deprecated, and a warning is issued whenever it is used. You should expect support for this file to be removed in a future release of MySQL Cluster. (WL #15765)

Functionality Added or Changed

- The `ndbcluster` plugin subscribes to all changes that occur in `NDB` and writes them epoch by epoch to the binary log. Each epoch received from `NDB` consists of a large number of changes, all of which are written to the binary log transaction cache before flushing them to the binary log. Previously, it was possible to configure the cache size for all threads, which often led to improper resource allocation for a MySQL Server used for writing a binary log of changes for `NDB`.

To enable dimensioning and configuring the system properly, we introduce a new system variable `ndb_log_cache_size` which makes it possible to set the size of the transaction cache used by the NDB binary log injector, so that this size can be set separately for writing the binary log for NDB transactions and (using `binlog_cache_size`) for writing other transactions whose sizes are likely to be smaller. (Bug #36694848)

Bugs Fixed

- **NDB Cluster APIs:** Using `NdbRecord` and `OO_SETVALUE` from the NDB API to write the value of a `Varchar`, `Varbinary`, `Longvarchar`, or `Longvarbinary` column failed with error 829. (Bug #36989337)
- **MySQL NDB ClusterJ:** References to ClusterJPA and OpenJPA have been removed from the comments in the packaging files, as JPA code was already removed from ClusterJ some time ago. (Bug #36725675)
- **MySQL NDB ClusterJ:** `ReconnectTest` in the ClusterJ test suite failed sometimes due to a race condition. The test has been rewritten with proper synchronization. (Bug #28550140)
- Removed node management code from `TRIX` that was not actually used. (Bug #37006547)
- Submitting concurrent shutdown commands for individual nodes using `ndb_mgm SHUTDOWN node_id` or the MGM API sometimes had one or both of the following adverse results:
 - Cluster failure when all nodes in the same node group were stopped
 - Inability to recover when all nodes in the same node group were stopped, and the cluster had more than one node group

This was due to the fact that the (planned) shutdown of a single node assumed that only one such shutdown occurred at a time, but did not actually check this limitation.

We fix this so that concurrent single-node shutdown requests are serialized across the cluster, and any which would cause a cluster outage are rejected. (Bug #36943756)

References: See also: Bug #36839995.

- Shutdown of a data node late in a schema transaction updating index statistics caused the president node to shut down as well. (Bug #36886242)

References: See also: Bug #36877952.

- It was possible for duplicate events to be sent to user applications when a data node was shut down. (Bug #36750146)
- `BLOB_INLINE_SIZE=0` set within a column comment was not honored, and the default for the blob type was used instead (such as 256 bytes for `BLOB`).

See [NDB_COLUMN Options](#), for more information. (Bug #36724336)

- Issues arose when an attempt was made to use a SHM transporter's wakeup socket before it was ready, due in part to error-handling when setting up the SHM transporter, which did not close the socket correctly prior to making another attempt at setup. (Bug #36568752, Bug #36623058)
- An error in a `my.cnf` file could cause the management node to shut down unexpectedly. (Bug #36508565)
- A race condition sometimes occurred between the watchdog thread and the signal execution thread trying to start node failure handling in parallel. (Bug #35728261)

Changes in MySQL NDB Cluster 9.0.1 (2024-07-23)

MySQL NDB Cluster 9.0.1 is a new Innovation release of NDB Cluster, based on MySQL Server 9.0 and including features in version 9.0 of the [NDB](#) storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 9.0.1. NDB Cluster 9.0.1 source code and binaries can be obtained from <https://dev.mysql.com/downloads/cluster/>.

For an overview of major changes made in NDB Cluster 9.0.1, see [What is New in MySQL NDB Cluster 9.7](#).

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 9 through MySQL 9.0.1 (see [Changes in MySQL NDB Cluster 9.0.1 \(2024-07-23\)](#)).

Version 9.0.1-ndb-9.0.1 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL NDB Cluster 9.0.0 (2024-07-02)

MySQL NDB Cluster 9.0.0 is a new Innovation release of NDB Cluster, based on MySQL Server 9.0 and including features in version 9.0 of the [NDB](#) storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining MySQL NDB Cluster 9.0. NDB Cluster 9.0 source code and binaries can be obtained from <https://dev.mysql.com/downloads/cluster/>.

For an overview of major changes made in NDB Cluster 9.0, see [What is New in MySQL NDB Cluster 9.7](#).

This release also incorporates all bug fixes and changes made in previous NDB Cluster releases, as well as all bug fixes and feature changes which were added in mainline MySQL 9 through MySQL 9.0.0 (see [Changes in MySQL NDB Cluster 9.0.0 \(2024-07-02\)](#)).



Important

This release is no longer available for download. It was removed due to a critical issue that could stop the server from restarting following the creation of a very large number of tables (8001 or more). Please upgrade to MySQL Cluster 9.0.1 instead.

- [Deprecation and Removal Notes](#)
- [Performance Schema Notes](#)
- [Functionality Added or Changed](#)
- [Bugs Fixed](#)

Deprecation and Removal Notes

- **NDB Cluster APIs:** `Node.js` support in MySQL NDB Cluster is now deprecated, and you should expect its removal in a future release.

You can also find the NDB Cluster `jones-ndb` driver for `Node.js` at <https://github.com/mysql/mysql-js>, where it remains available to interested users. (WL #16245)

- **NDB Client Programs:** The `ndb_size.pl` utility is now deprecated and is no longer supported. You can expect it to be removed from a future version of the NDB Cluster distribution; for this reason, you should now modify any applications which depend on it accordingly. (WL #16456)

- Use of an `Ndb.cfg` file for setting the connection string for an NDB process was not well documented or supported. With this release, this file is now formally deprecated, and you should expect support for it to be removed in a future release of MySQL Cluster. (WL #15765)

Performance Schema Notes

- **NDB Replication:** Previously, information about the NDB replication applier was available to the user only as a set of server status variables, and only for the default replication channel.

This release implements a new Performance Schema `ndb_replication_applier_status` table, which can be thought of as an NDB-specific extension to the existing `replication_applier_status` table. This enhancement provides per-channel information about an applier's status, including information relating to NDB Cluster conflict detection and resolution, in table form.

For more information about this table, see [The `ndb_replication_applier_status` Table](#). For information about replication in NDB Cluster, see [NDB Cluster Replication](#). (WL #16013)

Functionality Added or Changed

- **Important Change:** Now, when the removal of a data node file or directory fails with a file does not exist (`ENOENT`) error, this is treated as a successful removal.
- **ndbinfo Information Database:** Added a `type` column to the `transporter_details` table in the `ndbinfo` information database. This column shows the type of connection used by the transporter, which is either of `TCP` or `SHM`.
- **NDB Client Programs:** Added the `--CA-days` option to `ndb_sign_keys` to make it possible to specify a certificate's lifetime. (Bug #36549567)
- **NDB Client Programs:** When started, `ndbd` now produces a warning in the data node log like this one:

```
2024-05-28 13:32:16 [ndbd] WARNING -- Running ndbd with a single thread of
signal execution. For multi-threaded signal execution run the ndbmttd binary.
```

(Bug #36326896)

Bugs Fixed

- **NDB Replication:** When subscribing to changes in the `mysql.ndb_apply_status` table, different settings were used depending on whether `ndb_log_apply_status` was `ON` or `OFF`. Since `ndb_log_apply_status` can be changed at runtime and subscriptions are not recreated at that time, changing these settings at runtime did not have the desired effect.

The difference between enabling `ndb_log_apply_status` dynamically at runtime and doing so from the start of the MySQL process was in the format used when writing the `ndb_apply_status` updates to the binary log. When `ndb_log_apply_status` was enabled at runtime, writes were still done using the `UPDATE` format when `WRITE` was intended.

To fix this inconsistency and make the behavior more distinct, we now always use `WRITE` format in such cases; using the `WRITE` format also makes the binary log image slightly smaller and is thus preferred. In addition, the cleanup of old events has been improved, which improves the cleanup of failed attempts to create tables and events. (Bug #36453684)

- **NDB Replication:** The binary log index purge callback was skipped for the replica applier, which caused orphan rows to be left behind in the `ndb_binlog_index` table. (Bug #20573020, Bug #35847745, Bug #36378551, Bug #36420628, Bug #36423593, Bug #36485220, Bug #36492736)
- **NDB Cluster APIs:** TLS connection errors were printed even though TLS was not specified for connections.

To fix this issue, following an ignored TLS error, we explicitly reset the error condition in the management handle to `NO_ERROR`. (Bug #36354973)

- **NDB Cluster APIs:** It was possible to employ the following NDB API methods without them being used as `const`, although this alternative usage had long been deprecated (and was not actually documented):

- `Dictionary::listEvents()`
- `Dictionary::listIndexes()`
- `Dictionary::listObjects()`
- `NdbOperation::getNdbErrorLine()`

Now, each of these methods must always be invoked as `const`. (Bug #36165876)

- **NDB Client Programs:** In some cases, it was not possible to load certificates generated using `ndb_sign_keys`. (Bug #36430004)
- **NDB Client Programs:** `ndb_redo_log_reader` could not read data from encrypted files. (Bug #36313482)
- **NDB Client Programs:** The following command-line options did not function correctly for the `ndb_redo_log_reader` utility program:

- `--mbyte`
- `--page`
- `--pageindex`

(Bug #36313427)

- **NDB Client Programs:** `ndb_redo_log_reader` exited with `Record type = 0 not implemented` when reaching an unused page, all zero bytes, or a page which was only partially used (typically a page consisting of the page header only). (Bug #36313259)
- **NDB Client Programs:** `ndb_restore` did not restore a foreign key whose columns differed in order from those of the parent key.

Our thanks to Axel Svensson for the contribution. (Bug #114147, Bug #36345882)

- The destructor for `NDB_SCHEMA_OBJECT` makes several assertions about the state of the schema object, but the state was protected by a mutex, and the destructor did not acquire this mutex before testing the state.

We fix this by acquiring the mutex within the destructor. (Bug #36568964)

- **NDB** now writes a message to the MySQL server log before and after logging an incident in the binary log. (Bug #36548269)
- Removed a memory leak in `/util/NodeCertificate.cpp`. (Bug #36537931)
- Removed a memory leak from `src/ndbapi/NdbDictionaryImpl.cpp`. (Bug #36532102)
- The internal method `CertLifetime::set_set_cert_lifetime(X509 *cert)` should set the not-before and not-after times in the certificate to the same as those stored in the `CertLifetime` object, but instead it set the not-before time to the current time, and the not-after time to be of the same duration as the object. (Bug #36514834)
- Removed a possible use-after-free warning in `ConfigObject::copy_current()`. (Bug #36497108)

- When a thread acquires and releases the global schema lock required for schema changes and reads, the associated log message did not identify who performed the operation.

To fix this issue, we now do the following:

- Prepend the message in the log with the identification of the NDB Cluster component or user session responsible.
- Provide information about the related Performance Schema thread so that it can be traced.

(Bug #36446730)

References: See also: Bug #36446604.

- Metadata changes were not logged with their associated thread IDs. (Bug #36446604)

References: See also: Bug #36446730.

- When building NDB using `lld`, the build terminated prematurely with the error message `ld.lld: error: version script assignment of 'local' to symbol 'my_init' failed: symbol not defined` while attempting to link `libndbclient.so`. (Bug #36431274)
- TLS did not fail cleanly on systems which used OpenSSL 1.0, which is unsupported. Now in such cases, users get a clear error message advising that an upgrade to OpenSSL 1.1 or later is required to use TLS with NDB Cluster. (Bug #36426461)
- The included `libxml2` library was updated to version 2.9.13. (Bug #36417013)
- NDB Cluster's pushdown join functionality expects pushed conditions to filter exactly, so that no rows that do not match the condition must be returned, and all rows that do match the condition must be returned. When the condition contained a BINARY value compared to a BINARY column this was not always true; if the value was shorter than the column size, it could compare as equal to a column value despite having different lengths, if the condition was pushed down to NDB.

Now, when deciding whether a condition is pushable, we also make sure that the BINARY value length exactly matches the BINARY column's size. In addition, when binary string values were used in conditions with BINARY or VARBINARY columns, the actual length of a given string value was not used but rather an overestimate of its length. This is now changed; this should allow more conditions comparing short string values with VARBINARY columns to be pushed down than before this fix was made. (Bug #36390313, Bug #36513270)

References: See also: Bug #36399759, Bug #36400256. This issue is a regression of: Bug #36364619.

- Setting `AutomaticThreadConfig` and `NumCPUs` when running single-threaded data nodes (`ndbd`) sometimes led to unrecoverable errors. Now `ndbd` ignores settings for these parameters, which are intended to apply only to multi-threaded data nodes (`ndbmt`). (Bug #36388981)
- Improved the error message returned when trying to add a primary key to an NDBCLUSTER table using `ALGORITHM=INPLACE`. (Bug #36382071)

References: See also: Bug #30766579.

- The handling of the LQH operation pool which occurs as part of TC takeover skipped the last element in either of the underlying physical pools (static or dynamic). If this element was in use, holding an operation record for a transaction belonging to a transaction coordinator on the failed node, it was not returned, resulting in an incomplete takeover which sometimes left operations behind. Such operations interfered with subsequent transactions and the copying process (`CopyFrag`) used by the failed node to recover.

To fix this problem, we avoid skipping the final record while iterating through the LQH operation records during TC takeover. (Bug #36363119)

- The `libssh` library was updated to version 0.10.4. (Bug #36135621)
- When distribution awareness was not in use, the cluster tended to choose the same data node as the transaction coordinator repeatedly. (Bug #35840020, Bug #36554026)
- In certain cases, management nodes were unable to allocate node IDs to restarted data and SQL nodes. (Bug #35658072)
- Setting `ODirect` in the cluster's configuration caused excess logging when verifying that `ODirect` was actually settable for all paths. (Bug #34754817)
- In some cases, when trying to perform an online add index operation on an `NDB` table with no explicit primary key (see [Limitations of NDB online operations](#)), the resulting error message did not make the nature of the problem clear. (Bug #30766579)

References: See also: Bug #36382071.

Index

A

API timeouts, 12
ApiConnectRecord, 12
ApiFailureHandlingTimeout, 9
AutomaticThreadConfig, 17

B

backup, 9
BINARY, 17
binary log, 17
binary log rotation, 12
binary logging, 6
BLOB_INLINE_SIZE, 12, 15

C

CA-days, 17
certificates, 17
Cleaner, 9
compile-cluster, 9
compiling, 17
condition pushdown, 17
configuration, 15
const, 17
CopyFrag, 17
CopyGCI, 12
CREATE TABLESPACE, 6

D

DBLQH, 17
DBSPJ, 12
DBTC, 12, 17
DEFAULT, 6, 12
deprecation, 6

E

encryption, 6
errors, 6

F

failure handling, 15

G

global schema lock, 17

I

Important Change, 17
include-stored-grants, 9
index statistics, 15
INPLACE, 12, 17
InvocationHandler, 9

L

libssh, 17
libxml2, 9, 17
lld, 17
log level, 5
logging, 6, 12, 17
LQH_TRANSCONF, 9

M

macOS, 12
MGM API, 9
Microsoft Windows, 12
MySQL NDB ClusterJ, 6, 9, 15
mysql.ndb_apply_status table, 17

N

NDB Client Programs, 6, 9, 17
NDB Cluster, 4, 5, 6, 9, 12, 15, 17
NDB Cluster APIs, 12, 15, 17
NDB Disk Data, 12
NDB Replication, 6, 9, 12, 17
Ndb.cfg, 5
ndbd, 17
NdbDictionaryImpl.cpp, 17
ndbinfo Information Database, 9, 12, 17
NdbRecord, 15
Ndb_cluster_connection, 12
ndb_config, 12
ndb_log_apply_status, 17
ndb_log_cache_size, 15
ndb_metadata_check, 9
ndb_redo_log_reader, 17
ndb_replication table, 6
ndb_replication_applier_status, 17
ndb_restore, 17
ndb_schema_object, 9
NDB_SCHEMA_OBJECT, 17
Ndb_schema_participant_count, 9
ndb_sign_keys, 17
ndb_size.pl, 17
node IDs, 12
node shutdown, 15
Node.js, 17

null, 6
NumCPUs, 17

O

ODirect, 17
online operations, 17
OOBR, 9
OpenJPA, 15
openSSL, 17
out of buffer errors, 9

P

PURGE BINARY LOGS, 9

Q

QMGR, 9

R

REPORT BACKUPSTATUS, 9
restarts, 17

S

scans, 12
SHM, 15
signal dump, 9
STMT_END_F, 12
system restarts, 12

T

TCREASEREQ, 12
testsuite, 9
TLS, 9, 17
transaction coordinator, 17
transporter_details, 17
TRIX, 15

W

watchdog, 9

