MySQL NDB Cluster 8.0 Release Notes

Abstract

This document contains release notes for the changes in each release of MySQL NDB Cluster that uses version 8.0 of the NDB (NDBCLUSTER) storage engine.

Each NDB Cluster 8.0 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 8.0.

For general information about features added in NDB Cluster 8.0, see What is New in NDB Cluster. 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 8.0 documentation, see the MySQL 8.0 Reference Manual, which includes an overview of features added in MySQL 8.0 that are not specific to NDB Cluster (What Is New in MySQL 8.0), and discussion of upgrade issues that you may encounter for upgrades from MySQL 5.6 to MySQL 8.0 (Changes in MySQL 8.0). For a complete list of all bug fixes and feature changes made in MySQL 8.0 that are not specific to NDB, see MySQL 8.0 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: 2020-04-03 (revision: 20069)

Table of Contents

Preface and Legal Notices .......................................................................................................................... 2
Changes in MySQL NDB Cluster 8.0.21 (Not yet released, General Availability) ........................................ 3
Changes in MySQL NDB Cluster 8.0.20 (Not yet released, General Availability) ........................................ 3
Changes in MySQL NDB Cluster 8.0.19 (2020-01-13, General Availability) ................................................. 4
Changes in MySQL NDB Cluster 8.0.18 (2019-10-14, Release Candidate) ................................................ 11
Changes in MySQL NDB Cluster 8.0.17 (2019-07-22, Release Candidate) ................................................ 19
Changes in MySQL NDB Cluster 8.0.16 (2019-04-25, Development Milestone) ........................................ 24
Changes in MySQL NDB Cluster 8.0.15 (Not released) .............................................................................. 33
Changes in MySQL NDB Cluster 8.0.14 (2019-01-21, Development Milestone) ........................................ 33
Changes in MySQL NDB Cluster 8.0.13 (2018-10-23, Development Milestone) ........................................ 37
Release Series Changelogs: MySQL NDB Cluster 8.0 ................................................................................. 46
    Changes in MySQL NDB Cluster 8.0.18 (2019-10-14, Release Candidate) ........................................ 46
    Changes in MySQL NDB Cluster 8.0.17 (2019-07-22, Release Candidate) ........................................ 53
    Changes in MySQL NDB Cluster 8.0.16 (2019-04-25, Development Milestone) ............................... 58
    Changes in MySQL NDB Cluster 8.0.14 (2019-01-21, Development Milestone) ............................... 66
Preface and Legal Notices

This document contains release notes for the changes in each release of MySQL NDB Cluster that uses version 8.0 of the NDB storage engine.

Legal Notices

Copyright © 1997, 2020, Oracle and/or its affiliates. All rights reserved.

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.

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.

If this is software 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 installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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.

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

Intel and Intel Xeon 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, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

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.
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 https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Changes in MySQL NDB Cluster 8.0.21 (Not yet released, General Availability)

MySQL NDB Cluster 8.0.21 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in NDB Cluster.

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 8.0 through MySQL 8.0.21 (see Changes in MySQL 8.0.21 (Not yet released, General Availability)).

Version 8.0.21-ndb-8.0.21 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL NDB Cluster 8.0.20 (Not yet released, General Availability)

MySQL NDB Cluster 8.0.20 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.
MySQL NDB Cluster 8.0 Release Notes

For an overview of changes made in NDB Cluster 8.0, see What is New in NDB Cluster.

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 8.0 through MySQL 8.0.20 (see Changes in MySQL 8.0.20 (Not yet released, General Availability)).

Version 8.0.20-ndb-8.0.20 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL NDB Cluster 8.0.19 (2020-01-13, General Availability)

MySQL NDB Cluster 8.0.19 is a new release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in NDB Cluster.

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 8.0 through MySQL 8.0.19 (see Changes in MySQL 8.0.19 (2020-01-13, General Availability)).

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** The default value for the `ndb_autoincrement_prefetchSz` server system variable has been increased to 512. (Bug #30316314)

- **Important Change:** NDB now supports more than 2 replicas (up to a maximum of 4). Setting `NoOfReplicas=3` or `NoOfReplicas=4` is now fully covered in our internal testing and thus supported for use in production. (Bug #97479, Bug #97579, Bug #25261716, Bug #30501414, Bug #30528105)

- **Important Change:** Added the `TransactionMemory` data node configuration parameter which simplifies configuration of data node memory allocation for transaction operations. This is part of ongoing work on pooling of transactional and Local Data Manager (LDM) memory.

This new parameter is intended to supersede several older transactional memory parameters which have been deprecated (and which are thus subject to removal in a future release of NDB Cluster). In particular, the following parameters are incompatible with `TransactionMemory` and cannot be set in the `config.ini` configuration file if this parameter has been set:

- `MaxNoOfConcurrentIndexOperations`
- `MaxNoOfFiredTriggers`
- `MaxNoOfLocalOperations`
- `MaxNoOfLocalScans`

If you attempt to set any of these incompatible parameters concurrently with `TransactionMemory`, the cluster management server cannot start.
For more information, see the description of the `TransactionMemory` parameter and `Parameters incompatible with TransactionMemory`. See also `Data Node Memory Management`, for information about how memory resources are allocated by NDB Cluster data nodes. (Bug #96995, Bug #30344471)

**Important Change**: The maximum or default values for several NDB Cluster data node configuration parameters have been changed in this release. These changes are listed here:

- The maximum value for `DataMemory` is increased from 1 terabyte to 16 TB.
- The maximum value for `DiskPageBufferMemory` is also increased from 1 TB to 16 TB.
- The default value for `StringMemory` is decreased to 5 percent. Previously, this was 25 percent.
- The default value for `LcpScanProgressTimeout` is increased from 60 seconds to 180 seconds.

**Performance**: Read from any replica, which greatly improves the performance of table reads at a very low cost to table write performance, is now enabled by default for all NDB tables. This means both that the default value for the `ndb_read_backup` system variable is now ON, and that the value of the `NDB_TABLE` comment option `READ_BACKUP` is 1 when creating a new NDB table. (Previously, the default values were OFF and 0, respectively.)

For more information, see Setting NDB_TABLE Options, as well as the description of the `ndb_read_backup` system variable.

**NDB Disk Data**: The latency of checkpoints for Disk Data files has been reduced when using non-volatile memory devices such as solid-state drives (especially those using NVMe for data transfer), separate physical drives for Disk Data files, or both. As part of this work, two new data node configuration parameters, listed here, have been introduced:

- `MaxDiskDataLatency` sets a maximum on allowed latency for disk access, aborting transactions exceeding this amount of time to complete.
- `DiskDataUsingSameDisk` makes it possible to take advantage of keeping Disk Data files on separate disks by increasing the rate at which Disk Data checkpoints can be made.

This release also adds three new tables to the `ndbinfo` database. These tables, listed here, can assist with performance monitoring of Disk Data checkpointing:

- `diskstat` provides information about Disk Data tablespace reads, writes, and page requests during the previous 1 second.
- `diskstats_1sec` provides information similar to that given by the `diskstat` table, but does so for each of the last 20 seconds.
- `pgman_time_track_stats` table reports on the latency of disk operations affecting Disk Data tablespaces.

For additional information, see Disk Data latency parameters.

- Added the `ndb_metadata_sync` server system variable, which simplifies knowing when metadata synchronization has completed successfully. Setting this variable to `true` triggers immediate synchronization of all changes between the NDB dictionary and the MySQL data dictionary without regard to any values set for `ndb_metadata_check` or `ndb_metadata_check_interval`. When synchronization has completed, its value is automatically reset to `false`. (Bug #30406657)

- A stack trace is now written to the data node log on abnormal termination of a data node.
• Automatic synchronization of metadata from the MySQL data dictionary to NDB now includes databases containing NDB tables. With this enhancement, if a table exists in NDB, and the table and the database it belongs to do not exist on a given SQL node, it is no longer necessary to create the database manually. Instead, the database, along with all NDB tables belonging to this database, should be created on the SQL node automatically.

**Bugs Fixed**

• **Incompatible Change:** `ndb_restore` no longer restores shared users and grants to the `mysql.ndb_sql_metadata` table by default. A new command-line option `--include-stored-grants` is added to override this behavior and enable restoring of shared user and grant data and metadata.

As part of this fix, `ndb_restore` can now also correctly handle an ordered index on a system table. (Bug #30237657)

References: See also: Bug #29534239, Bug #30459246.

• **Incompatible Change:** The minimum value for the `RedoOverCommitCounter` data node configuration parameter has been increased from 0 to 1. The minimum value for the `RedoOverCommitLimit` data node configuration parameter has also been increased from 0 to 1.

You should check the cluster global configuration file and make any necessary adjustments to values set for these parameters before upgrading. (Bug #29752703)

• **macOS:** On MacOS, SQL nodes sometimes shut down unexpectedly during the binary log setup phase when starting the cluster. This occurred when there existed schemas whose names used uppercase letters and `lower_case_table_names` was set to 2. This caused acquisition of metadata locks to be attempted using keys having the incorrect lettercase, and, subsequently, these locks to fail. (Bug #30192373)

• **Microsoft Windows; NDB Disk Data:** On Windows, restarting a data node other than the master when using Disk Data tables led to a failure in TSMAN. (Bug #97436, Bug #30484272)

• **Solaris:** When debugging, `ndbmtd` consumed all available swap space on Solaris 11.4 SRU 12 and later. (Bug #30446577)

• **Solaris:** The byte order used for numeric values stored in the `mysql.ndb_sql_metadata` table was incorrect on Solaris/Sparc. This could be seen when using `ndb_select_all` or `ndb_restore --print`. (Bug #30265016)

• **NDB Disk Data:** After dropping a disk data table on one SQL node, trying to execute a query against `INFORMATION_SCHEMA.FILES` on a different SQL node stalled at `Waiting for tablespace metadata lock`. (Bug #30152258)

References: See also: Bug #29871406.

• **NDB Disk Data:** `ALTER TABLESPACE ... ADD DATAFILE` could sometimes hang while trying to acquire a metadata lock. (Bug #29871406)

The fix made in NDB 8.0.18 for an issue in which a transaction was committed prematurely aborted the transaction if the table definition had changed midway, but failed in testing to free memory allocated by `getExtraMetadata()`. Now this memory is properly freed before aborting the transaction. (Bug #30576983)

References: This issue is a regression of: Bug #29911440.
• Excessive allocation of attribute buffer when initializing data in DBTC led to preallocation of api connection records failing due to unexpectedly running out of memory. (Bug #30570264)

• Improved error handling in the case where NDB attempted to update a local user having the NDB_STORED_USER privilege but which could not be found in the ndb_sql_metadata table. (Bug #30556487)

• Failure of a transaction during execution of an ALTER TABLE ... ALGORITHM=COPY statement following the rename of the new table to the name of the original table but before dropping the original table caused mysqld to exit prematurely. (Bug #30548209)

• Non-MSI builds on Windows using -DWITH_NDBCLUSTER did not succeed unless the WiX toolkit was installed. (Bug #30536837)

• The allowed_values output from ndb_config --xml --configinfo for the Arbitration data node configuration parameter in NDB 8.0.18 was not consistent with that obtained in previous releases. (Bug #30529220)

References: See also: Bug #30505003.

• A faulty ndbrequire() introduced when implementing partial local checkpoints assumed that m_participatingLQH must be clear when receiving START_LCP_REQ, which is not necessarily true when a failure happens for the master after sending START_LCP_REQ and before handling any START_LCP_CONF signals. (Bug #30523457)

• A local checkpoint sometimes hung when the master node failed while sending an LCP_COMPLETE_REP signal and it was sent to some nodes, but not all of them. (Bug #30520818)

• Added the DUMP 9988 and DUMP 9989 commands. (Bug #30520103)

• The management server did not handle all cases of NODE_FAILREP correctly. (Bug #30520066)

• With SharedGlobalMemory set to 0, some resources did not meet required minimums. (Bug #30411835)

• Execution of ndb_restore --rebuild-indexes together with the --rewrite-database and --exclude-missing-tables options did not create indexes for any tables in the target database. (Bug #30411122)

• When writing the schema operation into the ndb_schema table failed, the states in the NDB_SCHEMA object were not cleared, which led to the SQL node shutting down when it tried to free the object. (Bug #30402362)

References: See also: Bug #30371590.

• If a transaction was aborted while getting a page from the disk page buffer and the disk system was overloaded, the transaction hung indefinitely. This could also cause restarts to hang and node failure handling to fail. (Bug #30397083, Bug #30360681)

References: See also: Bug #30152258.

• When synchronizing extent pages it was possible for the current local checkpoint (LCP) to stall indefinitely if a CONTINUEB signal for handling the LCP was still outstanding when receiving the FSWRITECONF signal for the last page written in the extent synchronization page. The LCP could also be restarted if another page was written from the data pages. It was also possible that this issue caused PREF_LCP pages to be written at times when they should not have been. (Bug #30397083)

• Data node failures with the error Another node failed during system restart... occurred during a partial restart. (Bug #30368622)
• Automatic synchronization could potentially trigger an increase in the number of locks being taken on a
particular metadata object at a given time, such as when a synchronization attempt coincided with a DDL
or DML statement involving the same metadata object; competing locks could lead to the NDB deadlock
detection logic penalizing the user action rather than the background synchronization. We fix this by
changing all exclusive metadata lock acquisition attempts during auto-synchronization so that they use a
timeout of 0 (rather than the 10 seconds previously allowed), which avoids deadlock detection and gives
priority to the user action. (Bug #30358470)

• If a `SYNC_EXTENT_PAGES_REQ` signal was received by `PGMAN` while dropping a log file group as part of
a partial local checkpoint, and thus dropping the page locked by this block for processing next, the LCP
terminated due to trying to access the page after it had already been dropped. (Bug #30305315)

• The wrong number of bytes was reported in the cluster log for a completed local checkpoint. (Bug
#30274618)

References: See also: Bug #29942998.

• Added the new `ndb_mgm` client debugging commands `DUMP 2356` and `DUMP 2357`. (Bug #30265415)

• Executing `ndb_drop_table` using the `--help` option caused this program to terminate prematurely,
and without producing any help output. (Bug #30259264)

• A `mysqld` trying to connect to the cluster, and thus trying to acquire the global schema lock (GSL)
during setup, ignored the setting for `ndb-wait-setup` and hung indefinitely when the GSL had already
been acquired by another `mysqld`, such as when it was executing an `ALTER TABLE` statement. (Bug
#30242141)

• When a table containing self-referential foreign key (in other words, a foreign key referencing another
column of the same table) was altered using the `COPY` algorithm, the foreign key definition was removed.
(Bug #30233405)

In MySQL 8.0, names of foreign keys explicitly provided by user are generated automatically in the SQL
layer and stored in the data dictionary. Such names are of the form `[table_name]_ibfk_[#]` which
align with the names generated by the InnoDB storage engine in MySQL 5.7. NDB 8.0.18 introduced
a change in behavior by NDB such that it also uses the generated names, but in some cases, such as
when tables were renamed, NDB still generated and used its own format for such names internally rather
than those generated by the SQL layer and stored in the data dictionary, which led to the following
issues:

• Discrepancies in `SHOW CREATE TABLE` output and the contents of
  `INFORMATION_SCHEMAREFERENTIAL_CONSTRAINTS`

• Improper metadata locking for foreign keys

• Confusing names for foreign keys in error messages

Now NDB also renames the foreign keys in such cases, using the names provided by the MySQL server,
to align fully with those used by InnoDB. (Bug #30210839)

References: See also: Bug #96508, Bug #30171959.

• When a table referenced by a foreign key was renamed, participating SQL nodes did not properly update
the foreign key definitions for the referencing table in their data dictionaries during schema distribution.
(Bug #30191068)

• Data node handling of failures of other data nodes could sometimes not be synchronized properly, such
that two or more data nodes could see different nodes as the master node. (Bug #30188414)
• Some scan operations failed due to the presence of an old assert in `DbtupBuffer.cpp` that checked whether API nodes were using a version of the software previous to NDB 6.4. This was no longer necessary or correct, and has been removed. (Bug #30188411)

• When executing a global schema lock (GSL), NDB used a single `Ndb_table_guard` object for successive retries when attempting to obtain a table object reference; it was not possible for this to succeed after failing on the first attempt, since `Ndb_table_guard` assumes that the underlying object pointer is determined once only—at initialisation—with the previously retrieved pointer being returned from a cached reference thereafter.

  This resulted in infinite waits to obtain the GSL, causing the binlog injector thread to hang so that `mysqld` considered all NDB tables to be read-only. To avoid this problem, NDB now uses a fresh instance of `Ndb_table_guard` for each such retry. (Bug #30120858)

  References: This issue is a regression of: Bug #30086352.

• When upgrading an SQL node to NDB 8.0 from a previous release series, the `.frm` file whose contents are read and then installed in the data dictionary does not contain any information about foreign keys. This meant that foreign key information was not installed in the SQL node's data dictionary. This is fixed by using the foreign key information available in the NDB data dictionary to update the local MySQL data dictionary during table metadata upgrade. (Bug #30071043)

• Restoring tables with the `--disable-indexes` option resulted in the wrong table definition being installed in the MySQL data dictionary. This is because the serialized dictionary information (SDI) packed into the NDB dictionary's table definition is used to create the table object; the SDI definition is updated only when the DDL change is done through the MySQL server. Installation of the wrong table definition meant that the table could not be opened until the indexes were re-created in the NDB dictionary again using `--rebuild-indexes`.

  This is fixed by extending auto-synchronization such that it compares the SDI to the NDB dictionary table information and fails in cases in which the column definitions do not match. Mismatches involving indexes only are treated as temporary errors, with the table in question being detected again during the next round of change detection. (Bug #30000202, Bug #30414514)

• Restoring tables for which `MAX_ROWS` was used to alter partitioning from a backup made from NDB 7.4 to a cluster running NDB 7.6 did not work correctly. This is fixed by ensuring that the upgrade code handling `PartitionBalance` supplies a valid table specification to the NDB dictionary. (Bug #29955656)

• The number of data bytes for the summary event written in the cluster log when a backup completed was truncated to 32 bits, so that there was a significant mismatch between the number of log records and the number of data records printed in the log for this event. (Bug #29942998)

• `mysqld` sometimes aborted during a long `ALTER TABLE` operation that timed out. (Bug #29894768)

  References: See also: Bug #29192097.

• When an SQL node connected to NDB, it did not know whether it had previously connected to that cluster, and thus could not determine whether its data dictionary information was merely out of date, or completely invalid. This issue is solved by implementing a unique schema version identifier (schema UUID) to the `ndb_schema` table in NDB as well as to the `ndb_schema` table object in the data dictionary. Now, whenever a `mysqld` connects to a cluster as an SQL node, it can compare the schema UUID stored in its data dictionary against that which is stored in the `ndb_schema` table, and so know whether it is connecting for the first time. If so, the SQL node removes any entries that may be in its data dictionary. (Bug #29894166)

  References: See also: Bug #27543602.
• Improved log messages generated by table discovery and table metadata upgrades. (Bug #29894127)

• Using 2 LDM threads on a 2-node cluster with 10 threads per node could result in a partition imbalance, such that one of the LDM threads on each node was the primary for zero fragments. Trying to restore a multi-threaded backup from this cluster failed because the datafile for one LDM contained only the 12-byte data file header, which ndb_restore was unable to read. The same problem could occur in other cases, such as when taking a backup immediately after adding an empty node online.

It was found that this occurred when ODirect was enabled for an EOF backup data file write whose size was less than 512 bytes and the backup was in the STOPPING state. This normally occurs only for an aborted backup, but could also happen for a successful backup for which an LDM had no fragments. We fix the issue by introducing an additional check to ensure that writes are skipped only if the backup actually contains an error which should cause it to abort. (Bug #29892660)

References: See also: Bug #30371389.

• For NDB tables, ALTER TABLE ... ALTER INDEX did not work with ALGORITHM=INPLACE. (Bug #29700197)

• ndb_restore failed in testing on 32-bit platforms. This issue is fixed by increasing the size of the thread stack used by this tool from 64 KB to 128 KB. (Bug #29699887)

References: See also: Bug #30406046.

• An unplanned shutdown of the cluster occurred due to an error in DBTUP while deleting rows from a table following an online upgrade. (Bug #29616383)

• In some cases the SignalSender class, used as part of the implementation of ndb_mgmd and ndbinfo, buffered excessive numbers of unneeded SUB_GCP_COMPLETE_REP and API_REGCONF signals, leading to unnecessary consumption of memory. (Bug #29520353)

References: See also: Bug #20075747, Bug #29474136.

• The setting for the BackupLogBufferSize configuration parameter was not honored. (Bug #29415012)

• When mysqld was run with the --upgrade=FORCE option, it reported the following issues:

[Warning] Table 'mysql.ndb_apply_status' requires repair.
[ERROR] Table 'mysql.ndb_apply_status' repair failed.

This was because --upgrade=FORCE causes a bootstrap system thread to run CHECK TABLE FOR UPGRADE, but ha_ndbcluster::open() refused to open the table before schema synchronization had completed, which eventually led to the reported conditions. (Bug #29305977)

References: See also: Bug #29205142.

• When using explicit SHM connections, with ShmSize set to a value larger than the system's available shared memory, mysqld hung indefinitely on startup and produced no useful error messages. (Bug #28875553)

• The maximum global checkpoint (GCP) commit lag and GCP save timeout are recalculated whenever a node shuts down, to take into account the change in number of data nodes. This could lead to the unintentional shutdown of a viable node when the threshold decreased below the previous value. (Bug #27664092)

References: See also: Bug #26364729.
• A transaction which inserts a child row may run concurrently with a transaction which deletes the parent row for that child. One of the transactions should be aborted in this case, lest an orphaned child row result.

Before committing an insert on a child row, a read of the parent row is triggered to confirm that the parent exists. Similarly, before committing a delete on a parent row, a read or scan is performed to confirm that no child rows exist. When insert and delete transactions were run concurrently, their prepare and commit operations could interact in such a way that both transactions committed. This occurred because the triggered reads were performed using LM_CommittedRead locks (see NdbOperation::LockMode), which are not strong enough to prevent such error scenarios.

This problem is fixed by using the stronger LM_SimpleRead lock mode for both triggered reads. The use of LM_SimpleRead rather than LM_CommittedRead locks ensures that at least one transaction aborts in every possible scenario involving transactions which concurrently insert into child rows and delete from parent rows. (Bug #22180583)

• Concurrent SELECT and ALTER TABLE statements on the same SQL node could sometimes block one another while waiting for locks to be released. (Bug #17812505, Bug #30383887)

• Failure handling in schema synchronization involves pushing warnings and errors to the binary logging thread. Schema synchronization is also retried in case of certain failures which could lead to an accumulation of warnings in the thread. Now such warnings and errors are cleared following each attempt at schema synchronization. (Bug #2991036)

• An INCL_NODECONF signal from any local blocks should be ignored when a node has failed, except in order to reset c_nodeStartSlave.nodeId. (Bug #96550, Bug #30187779)

• When returning Error 1022, NDB did not print the name of the affected table. (Bug #74218, Bug #19763093)

References: See also: Bug #29700174.

Changes in MySQL NDB Cluster 8.0.18 (2019-10-14, Release Candidate)

MySQL NDB Cluster 8.0.18 is a new development release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in NDB Cluster.

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 8.0 through MySQL 8.0.18 (see Changes in MySQL 8.0.18 (2019-10-14, General Availability)).

• Functionality Added or Changed

• Bugs Fixed

Functionality Added or Changed

• Important Change: The 63-byte limit on NDB database and table names has been removed. These identifiers may now take up to 64 bytes, as when using other MySQL storage engines. For more information, see Previous NDB Cluster Issues Resolved in NDB Cluster 8.0. (Bug #27447958)
• **Important Change:** Implemented the `NDB_STORED_USER` privilege, which enables sharing of users, roles, and privileges across all SQL nodes attached to a given NDB Cluster. This replaces the distributed grant tables mechanism from NDB 7.6 and earlier versions of NDB Cluster, which was removed in NDB 8.0.16 due to its incompatibility with changes made to the MySQL privilege system in MySQL 8.0.

A user or role which has this privilege is propagated, along with its (other) privileges to a MySQL server (SQL node) as soon as it connects to the cluster. Changes made to the privileges of the user or role are synchronized immediately with all connected SQL nodes.

`NDB_STORED_USER` can be granted to users and roles other than reserved accounts such as `mysql.session@localhost` or `mysql.infoschema@localhost`. A role can be shared, but assigning a shared role to a user does not cause this user to be shared; the `NDB_STORED_USER` privilege must be granted to the user explicitly in order for the user to be shared between NDB Cluster SQL nodes.

The `NDB_STORED_USER` privilege is always global and must be granted using `ON *.*`. This privilege is recognized only if the MySQL server enables support for the `NDBCLUSTER` storage engine.

For usage information, see the description of `NDB_STORED_USER`. Distributed MySQL Privileges with `NDB_STORED_USER`, has additional information on how `NDB_STORED_USER` and privilege synchronization work. For information on how this change may affect upgrades to NDB 8.0 from previous versions, see Upgrading and Downgrading NDB Cluster.

References: See also: Bug #29862601, Bug #29996547.

• **Important Change:** The maximum row size for an `NDB` table is increased from 14000 to 30000 bytes.

As before, only the first 264 bytes of a `BLOB` or `TEXT` column count towards this total.

The maximum offset for a fixed-width column of an `NDB` table is 8188 bytes; this is also unchanged from previous NDB Cluster releases.

For more information, see Limits Associated with Database Objects in NDB Cluster.

References: See also: Bug #29485977, Bug #29024275.

• **Important Change:** A new binary format has been implemented for the NDB management server's cached configuration file, which is intended to support much larger numbers of nodes in a cluster than previously. Prior to this release, the configuration file supported a maximum of 16381 sections; this number is increased to 4G.

Upgrades to the new format should not require any manual intervention, as the management server (and other cluster nodes) can still read the old format. For downgrades from this release or a later one to NDB 8.0.17 or earlier, it is necessary to remove the binary configuration files prior to starting the old management server binary, or start it using the `--initial` option.

For more information, see Upgrading and Downgrading NDB Cluster.
• **Important Change:** The maximum number of data nodes supported in a single NDB cluster is raised in this release from 48 to 144. The range of supported data node IDs is increased in conjunction with this enhancement to 1-144, inclusive.

In previous releases, recommended node IDs for management nodes were 49 and 50. These values are still supported, but, if used, limit the maximum number of data nodes to 142. For this reason, the recommended node ID values for management servers are now 145 and 146.

The maximum total supported number of nodes of all types in a given cluster is 255. This total is unchanged from previous releases.

For a cluster running more than 48 data nodes, it is not possible to downgrade directly to a previous release that supports only 48 data nodes. In such cases, it is necessary to reduce the number of data nodes to 48 or fewer, and to make sure that all data nodes use node IDs that are less than 49.

• **NDB Cluster APIs:** An alternative constructor for `NdbInterpretedCode` is now provided, which accepts an `NdbRecord` in place of a `Table` object. (Bug #29852377)

• **NDB Cluster APIs:** `NdbScanFilter::cmp()` and the following `NdbInterpretedCode` comparison methods can be now used to compare table column values:

  • `branch_col_eq()`
  • `branch_col_ge()`
  • `branch_col_gt()`
  • `branch_col_le()`
  • `branch_col_lt()`
  • `branch_col_ne()`

When using any of these methods, the table column values to be compared must be of exactly the same type, including with respect to length, precision, and scale. In addition, in all cases, `NULL` is always considered by these methods to be less than any other value. You should also be aware that, when used to compare table column values, `NdbScanFilter::cmp()` does not support all possible values of `BinaryCondition`.

For more information, see the descriptions of the individual API methods.

• **NDB Client Programs:** The dependency of the `ndb_delete_all` utility on the `NDBT` library has been removed. This library, used in NDB development for testing, is not required for normal use. The visible change for users is that `ndb_delete_all` no longer prints `NDBT_ProgramExit - status` following completion of its run. Applications that depend upon this behavior should be updated to reflect this change when upgrading to this release.

• `ndb_restore` now reports the specific `NDB` error number and message when it is unable to load a table descriptor from a backup `.ctl` file. This can happen when attempting to restore a backup taken from a later version of the NDB Cluster software to a cluster running an earlier version—for example, when the backup includes a table using a character set which is unknown to the version of `ndb_restore` being used to restore it. (Bug #30184265)

• The output from `DUMP 1000` in the `ndb_mgm` client has been extended to provide information regarding total data page usage. (Bug #29841454)

References: See also: Bug #29929996.
• NDB Cluster’s condition pushdown functionality has been extended as follows:
  • Expressions using any previously allowed comparisons are now supported.
  • Comparisons between columns in the same table and of the same type are now supported. The columns must be of exactly the same type.

Example: Suppose there are two tables \textit{t1} and \textit{t2} created as shown here:

\begin{verbatim}
CREATE TABLE t1 (a INT, b INT, c CHAR(10), d CHAR(5)) ENGINE=NDB;
CREATE TABLE t2 LIKE t1;
\end{verbatim}

The following joins can now be pushed down to the data nodes:

\begin{verbatim}
SELECT * FROM t1 JOIN t2 ON t2.a < t1.a+10;
SELECT * FROM t1 JOIN t2 ON t2.a = t1.a+t1.b;
SELECT * FROM t1 JOIN t2 ON t2.a = t1.a+t1.b;
SELECT * FROM t1 JOIN t2 ON t2.d = SUBSTRING(t1.c,1,5);
SELECT * FROM t1 JOIN t2 ON t2.c = CONCAT('foo',t1.d,'ba');
\end{verbatim}

Supported comparisons are $<$, $\leq$, $>$, $=$, and $\neq$. (Bug #29685643)

• NDB Cluster now uses \textit{table_name\_fk\_N} as the naming pattern for internally generated foreign keys, which is similar to the \textit{table_name\_ibfk\_N} pattern used by InnoDB. (Bug #96508, Bug #30171959)

References: See also: Bug #30210839.

• Added the \textit{ndb\_schema\_dist\_lock\_wait\_timeout} system variable to control how long to wait for a schema lock to be released when trying to update the SQL node’s local data dictionary for one or more tables currently in use from the NDB data dictionary’s metadata. If this synchronization has not yet occurred by the end of this time, the SQL node returns a warning that schema distribution did not succeed; the next time that the table for which distribution failed is accessed, NDB tries once again to synchronize the table metadata.

• NDB table objects submitted by the metadata change monitor thread are now automatically checked for any mismatches and synchronized by the NDB binary logging thread. The status variable \textit{Ndb\_metadata\_synced\_count}, added in this release shows the number of objects synchronized automatically; it is possible to see which objects have been synchronized by checking the cluster log. In addition, the new status variable \textit{Ndb\_metadata\_blacklist\_size} indicates the number of objects for which synchronization has failed.

References: See also: Bug #30000202.

• It is now possible to build NDB for 64-bit ARM CPUs from the NDB Cluster sources. Currently, we do not provide any precompiled binaries for this platform.

• Start times for the \textit{ndb\_mgmd} management node daemon have been significantly improved as follows:
  • More efficient handling of properties from configuration data can decrease startup times for the management server by a factor of 6 or more as compared with previous versions.
  • Host names no longer create a bottleneck during startup, making \textit{ndb\_mgmd} start times up to 20 times shorter where these are used.

• Columns of NDB tables can now be renamed online, using \texttt{ALGORITHM=INPLACE}.

References: See also: Bug #28609968.
Bugs Fixed

- **Important Change**: Because the current implementation for node failure handling cannot guarantee that even a single transaction of size `MaxNoOfConcurrentOperations` is completed in each round, this parameter is once again used to set a global limit on the total number of concurrent operations in all transactions within a single transaction coordinator instance. (Bug #96617, Bug #30216204)

- **Partitioning; NDB Disk Data**: Creation of a partitioned disk data table was unsuccessful due to a missing metadata lock on the tablespace specified in the `CREATE TABLE` statement. (Bug #28876892)

- **NDB Disk Data**: Tablespaces and data files are not tightly coupled in NDB, in the sense that they are represented by independent `NdbDictionary` objects. Thus, when metadata is restored using the `ndb_restore` tool, there was no guarantee that the tablespace and its associated datafile objects were restored at the same time. This led to the possibility that the tablespace mismatch was detected and automatically synchronized to the data dictionary before the datafile was restored to NDB. This issue also applied to log file groups and undo files.

To fix this problem, the metadata change monitor now submits tablespaces and log file groups only if their corresponding datafiles and undo files actually exist in NDB. (Bug #30090080)

- **NDB Disk Data**: When a data node failed following creation and population of an NDB table having columns on disk, but prior to execution of a local checkpoint, it was possible to lose row data from the tablespace. (Bug #29506869)

- **NDB Cluster APIs**: The NDB API examples `ndbapi_array_simple.cpp` (see NDB API Simple Array Example) and `ndbapi_array_using_adapter.cpp` (see NDB API Simple Array Example Using Adapter) made assignments directly to a `std::vector` array instead of using `push_back()` calls to do so. (Bug #28956047)

- **MySQL NDB ClusterJ**: If ClusterJ was deployed as a separate module of a multi-module web application, when the application tried to create a new instance of a domain object, the exception `java.lang.IllegalArgumentException: non-public interface is not defined by the given loader` was thrown. It was because ClusterJ always tries to create a proxy class from which the domain object can be instantiated, and the proxy class is an implementation of the domain interface and the protected `DomainTypeHandlerImpl::Finalizable` interface. The class loaders of these two interfaces were different in the case, as they belonged to different modules running on the web server, so that when ClusterJ tried to create the proxy class using the domain object interface's class loader, the above-mentioned exception was thrown. This fix makes the `Finalization` interface public so that the class loader of the web application would be able to access it even if it belongs to a different module from that of the domain interface. (Bug #29895213)

- **MySQL NDB ClusterJ**: ClusterJ sometimes failed with a segmentation fault after reconnecting to an NDB Cluster. This was due to ClusterJ reusing old database metadata objects from the old connection. With the fix, those objects are discarded before a reconnection to the cluster. (Bug #29891983)

- **Faulty calculation of microseconds caused the internal `ndb_milli_sleep()` function to sleep for too short a time. (Bug #30211922)

- **Once a data node is started, 95% of its configured `DataMemory` should be available for normal data, with 5% to spare for use in critical situations. During the node startup process, all of its configured `DataMemory` is usable for data, in order to minimize the risk that restoring the node data fails due to running out of data memory due to some dynamic memory structure using more pages for the same data**
than when the node was stopped. For example, a hash table grows differently during a restart than it did previously, since the order of inserts to the table differs from the historical order.

The issue raised in this bug report occurred when a check that the data memory used plus the spare data memory did not exceed the value set for `DataMemory` failed at the point where the spare memory was reserved. This happened as the state of the data node transitioned from starting to started, when reserving spare pages. After calculating the number of reserved pages to be used for spare memory, and then the number of shared pages (that is, pages from shared global memory) to be used for this, the number of reserved pages already allocated was not taken into consideration. (Bug #30205182)

References: See also: Bug #29616383.

- It was not possible to use `ndb_restore` and a backup taken from an NDB 8.0 cluster to restore to a cluster running NDB 7.6. (Bug #30184658)

References: See also: Bug #30221717.

- When starting, a data node's local sysfile was not updated between the first completed local checkpoint and start phase 50. (Bug #30086352)

- In the `BACKUP` block, the assumption was made that the first record in `c_backups` was the local checkpoint record, which is not always the case. Now `NDB` loops through the records in `c_backups` to find the (correct) LCP record instead. (Bug #30080194)

- During node takeover for the master it was possible to end in the state `LCP_STATUS_IDLE` while the remaining data nodes were reporting their state as `LCP_TAB_SAVED`. This led to failure of the node when attempting to handle reception of a `LCP_COMPLETE_REP` signal since this is not expected when idle. Now in such cases local checkpoint handling is done in a manner that ensures that this node finishes in the proper state (`LCP_TAB_SAVED`). (Bug #30032863)

- When a MySQL Server built with `NDBCLUSTER` support was run on Solaris/x86, it failed during schema distribution. The root cause of the problem was an issue with the Developer Studio compiler used to build binaries for this platform when optimization level `-xO2` was used. This issue is fixed by using optimization level `-xO1` instead for `NDBCLUSTER` built for Solaris/x86. (Bug #30031130)

References: See also: Bug #28585914, Bug #30014295.

- `NDB` used `free()` directly to deallocate `ndb_mgm_configuration` objects instead of calling `ndb_mgm_destroy_configuration()`, which correctly uses `delete` for deallocation. (Bug #29998980)

- Default configuration sections did not have the configuration section types set when unpacked into memory, which caused a memory leak since this meant that the section destructor would not destroy the entries for these sections. (Bug #29965125)

- No error was propagated when `NDB` failed to discover a table due to the table format being old and no longer supported, which could cause the `NDB` handler to retry the discovery operation endlessly and thereby hang. (Bug #29949096, Bug #29934763)

- During upgrade of an NDB Cluster when half of the data nodes were running NDB 7.6 while the remainder were running NDB 8.0, attempting to shut down those nodes which were running NDB 7.6 led to failure of one node with the error `CHECK_FAILEDNODEPTR.P->DBLQHFAI`. (Bug #29912988, Bug #30141203)

- Altering a table in the middle of an ongoing transaction caused a table discovery operation which led to the transaction being committed prematurely; in addition, no error was returned when performing further updates as part of the same transaction.
Now in such cases, the table discovery operation fails, when a transaction is in progress. (Bug #29911440)

- When performing a local checkpoint (LCP), a table’s schema version was intermittently read as 0, which caused NDB LCP handling to treat the table as though it were being dropped. This could effect rebuilding of indexes offline by ndb_restore while the table was in the TABLE_READ_ONLY state. Now the function reading the schema version (getCreateSchemaVersion()) no longer not changes it while the table is read-only. (Bug #29910397)

- When an error occurs on an SQL node during schema distribution, information about this was written in the error log, but no indication was provided by the mysql client that the DDL statement in question was unsuccessful. Now in such cases, one or more generic warnings are displayed by the client to indicate that a given schema distribution operation has not been successful, with further information available in the error log of the originating SQL node. (Bug #29889869)

- Errors and warnings pushed to the execution thread during metadata synchronization and metadata change detection were not properly logged and cleared. (Bug #29874313)

- Altering a normal column to a stored generated column was performed online even though this is not supported. (Bug #29862463)

- A pushed join with ORDER BY did not always return the rows of the result in the specified order. This could occur when the optimizer used an ordered index to provide the ordering and the index used a column from the table that served as the root of the pushed join. (Bug #29860378)

- A number of issues in the Backup block for local checkpoints (LCPs) were found and fixed, including the following:
  - Bytes written to LCP part files were not always included in the LCP byte count.
  - The maximum record size for the buffer used for all LCP part files was not updated in all cases in which the table maximum record size had changed.
  - LCP surfacing could occur for LCP scans at times other than when receiving SCAN_FRAGCONF signals.
  - It was possible in some cases for the table currently being scanned to be altered in the middle of a scan request, which behavior is not supported.

(Bug #29843373)

References: See also: Bug #29485977.

- The requestInfo fields for the long and short forms of the LQHKEYREQ signal had different definitions; bits used for the key length in the short version were reused for flags in the long version, since the key length is implicit in the section length of the long version of the signal but it was possible for long LQHKEYREQ signals to contain a keylength in these same bits, which could be misinterpreted by the receiving local query handler, potentially leading to errors. Checks have now been implemented to make sure that this no longer happens. (Bug #29820838)

- The list of dropped shares could hold only one dropped NDB_SHARE instance for each key, which prevented NDB_SHARE instances with same key from being dropped multiple times while handlers held references to those NDB_SHARE instances. This interfered with keeping track of the memory allocated and being able to release it if mysqld shut down without all handlers having released their references to the shares. To resolve this issue, the dropped share list has been changed to use a list type which
allows more than one NDB_SHARE with the same key to exist at the same time. (Bug #29812659, Bug #29812613)

- Removed an ndb_restore compile-time dependency on table names that was defined by the ndbcluster plugin. (Bug #29801100)

- When creating a table in parallel on multiple SQL nodes, the result was a race condition between checking that the table existed and opening the table, which caused CREATE TABLE IF NOT EXISTS to fail with Error 1. This was the result of two issues, described with their fixes here:

1. Opening a table whose NDB_SHARE did not exist returned the non-descriptive error message ERROR 1296 (HY000): Got error 1 'Unknown error code' from NDBCLUSTER. This is fixed with a warning describing the problem in more detail, along with a more sensible error code.

   It was possible to open a table before schema synchronization was completed. This is fixed with a warning better describing the problem, along with an error indicating that cluster is not yet ready.

   In addition, this fixes a related issue in which creating indexes sometimes also failed with Error 1. (Bug #29793534, Bug #29871321)

- Previously, for a pushed condition, every request sent to NDB for a given table caused the generation of a new instance of NdbInterpretedCode. When joining tables, generation of multiple requests for all tables following the first table in the query plan is very likely; if the pushed condition had no dependencies on prior tables in the query plan, identical instances of NdbInterpretedCode were generated for each request, at a significant cost in wasted CPU cycles. Now such pushed conditions are identified and the required NdbInterpretedCode object is generated only once, and reused for every request sent for this table without the need for generating new code each time.

   This change also makes it possible for Scan Filter too large errors to be detected and set during query optimization, which corrects cases where the query plan shown was inaccurate because the indicated push of a condition later had to be undone during the execution phase. (Bug #29704575)

- Some instances of NdbScanFilter used in pushdown conditions were not generated properly due to FLOAT values being represented internally as having zero length. This led to more than the expected number of rows being returned from NDB, as shown by the value of Ndb_api_read_row_count. While the condition was re-evaluated by mysqld when generation of scan filter failed, the end result was still correct in such cases, but any performance gain expected from pushing the condition was lost. (Bug #29699347)

- When creating a table, NDB did not always determine correctly whether it exceeded the maximum allowed record size. (Bug #29698277)

- NDB index statistics are calculated based on the topology of one fragment of an ordered index; the fragment chosen in any particular index is decided at index creation time, both when the index is originally created, and when a node or system restart has recreated the index locally. This calculation is based in part on the number of fragments in the index, which can change when a table is reorganized. This means that, the next time that the node is restarted, this node may choose a different fragment, so that no fragments, one fragment, or two fragments are used to generate index statistics, resulting in errors from ANALYZE TABLE.

   This issue is solved by modifying the online table reorganization to recalculate the chosen fragment immediately, so that all nodes are aligned before and after any subsequent restart. (Bug #29534647)

- As part of initializing schema distribution, each data node must maintain a subscriber bitmap providing information about the API nodes that are currently subscribed to this data node. Previously, the size of the bitmap was hard-coded to MAX_NODES (256), which meant that large amounts of memory might be
allocated but never used when the cluster had significantly fewer nodes than this value. Now the size of
the bitmap is determined by checking the maximum API node ID used in the cluster configuration file.
(Bug #29270539)

• The removal of the mysql_upgrade utility and its replacement by mysqld --initialize means that
the upgrade procedure is executed much earlier than previously, possibly before NDB is fully ready to
handle queries. This caused migration of the MySQL privilege tables from NDB to InnoDB to fail. (Bug
#29205142)

• During a restart when the data nodes had started but not yet elected a president, the management
server received a node ID already in use error, which resulted in excessive retries and logging.
This is fixed by introducing a new error 1705 Not ready for connection allocation yet for this case.

During a restart when the data nodes had not yet completed node failure handling, a spurious Failed
to allocate nodeID error was returned. This is fixed by adding a check to detect an incomplete
node start and to return error 1703 Node failure handling not completed instead.

As part of this fix, the frequency of retries has been reduced for not ready to alloc nodeID errors,
an error insert has been added to simulate a slow restart for testing purposes, and log messages have
been reworded to indicate that the relevant node ID allocation errors are minor and only temporary. (Bug
#27484514)

• NDB on Windows and MacOSX platforms did not always treat table names using mixed case consistently
with lower_case_table_names = 2. (Bug #27307793)

• The process of selecting the transaction coordinator checked for “live” data nodes but not necessarily for
those that were actually available. (Bug #27160203)

• The automatic metadata synchronization mechanism requires the binary logging thread to
acquire the global schema lock before an object can be safely synchronized. When another
thread had acquired this lock at the same time, the binary logging thread waited for up to
TransactionDeadlockDetectionTimeout milliseconds and then returned failure if it was
unsuccessful in acquiring the lock, which was unnecessary and which negatively impacted performance.

This has been fixed by ensuring that the binary logging thread acquires the global schema lock, or else
returns with an error, immediately. As part of this work, a new OperationOptions flag OO_NOWAIT
has also been implemented in the NDB API.

Changes in MySQL NDB Cluster 8.0.17 (2019-07-22, Release Candidate)

MySQL NDB Cluster 8.0.17 is a new development release of NDB 8.0, based on MySQL Server 8.0 and
including features in version 8.0 of the NDB storage engine, as well as fixing recently discovered bugs in
previous NDB Cluster releases.

Obtaining NDB Cluster 8.0.  NDB Cluster 8.0 source code and binaries can be obtained from https://
dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in NDB Cluster.

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 8.0 through MySQL 8.0.17 (see
Changes in MySQL 8.0.17 (2019-07-22, General Availability)).

• Functionality Added or Changed
• **Bugs Fixed**

**Functionality Added or Changed**

• Schema operation timeout detection has been moved from the schema distribution client to the schema distribution coordinator, which now checks ongoing schema operations for timeout at regular intervals, marks participants that have timed out, emits suitable warnings when a schema operation timeout occurs, and prints a list of any ongoing schema operations at regular intervals.

As part of this work, a new option `--ndb-schema-dist-timeout` makes it possible to set the number of seconds for a given SQL node to wait until a schema operation is marked as having timed out. (Bug #29556148)

• Added the status variable `Ndb_trans_hint_count_session`, which shows the number of transactions started in the current session that used hints. Compare this with `Ndb_api_trans_start_count_session` to get the proportion of all NDB transactions in the current session that have been able to use hinting. (Bug #29127040)

• When the cluster is in single user mode, the output of the `ndb_mgm SHOW` command now indicates which API or SQL node has exclusive access while this mode is in effect. (Bug #16275500)

**Bugs Fixed**

• **Important Change:** Attempting to drop, using the `mysql` client, an NDB table that existed in the MySQL data dictionary but not in NDB caused `mysqld` to fail with an error. This situation could occur when an NDB table was dropped using the `ndb_drop_table` tool or in an NDB API application using `dropTable()`. Now in such cases, `mysqld` drops the table from the MySQL data dictionary without raising an error. (Bug #29125206)

• **Important Change:** The dependency of `ndb_restore` on the NDBT library, which is used for internal testing only, has been removed. This means that the program no longer prints `NDBT_ProgramExit: ...` when terminating. Applications that depend upon this behavior should be updated to reflect this change when upgrading to this release.

• **Packaging:** Added debug symbol packages to NDB distributions for .deb-based platforms which do not generate these automatically. (Bug #29040024)

• **NDB Disk Data:** If, for some reason, a disk data table exists in the NDB data dictionary but not in that of the MySQL server, the data dictionary is synchronized by installing the object. This can occur either during the schema synchronization phase when a MySQL server connects to an NDB Cluster, or during table discovery through a DML query or DDL statement.

For disk data tables which used a tablespace for storage, the tablespace ID is stored as part of the data dictionary object, but this was not set during synchronization. (Bug #29597249)

• **NDB Disk Data:** Concurrent Disk Data table and tablespace DDL statements executed on the same SQL node caused a metadata lock deadlock. A DDL statement requires that an exclusive lock be taken on the object being modified and every such lock in turn requires that the global schema lock be acquired in NDB.

To fix this issue, NDB now tracks when a global schema lock corresponding to an exclusive lock on a tablespace is taken. If a different global schema lock request fails while the first lock, NDB assumes that there is a deadlock. In this case, the deadlock is handled by having the new request release all locks it previously acquired, then retrying them at a later point. (Bug #29394407)

References: See also: Bug #29175268.
• **NDB Disk Data:** Following execution of `ALTER TABLESPACE`, SQL statements on an existing table using the affected tablespace failed with error 3508 *Dictionary object id (id) does not exist* where the object ID shown refers to the tablespace. Schema distribution of `ALTER TABLESPACE` involves dropping the old object from the data dictionary on a participating SQL node and creating a new one with a different dictionary object id, but the table object in the SQL node’s data dictionary still used the old tablespace ID which rendered it unusable on the participants.

To correct this problem, tables using the tablespace are now retrieved and stored prior to the creation of the new tablespace, and then updated the new object ID of the tablespace after it has been created in the data dictionary. (Bug #29389168)

• **NDB Replication:** The `ndb_apply_status` table was created using the deprecated syntax `VARCHAR(255) BINARY`. `VARBINARY(255)` is now used instead for creating this table. (Bug #29807585)

• **NDB Replication:** Errors raised from replication settings by a `CREATE TABLE` statement were not properly checked, leading the user to believe (incorrectly) that the table was valid for this purpose. (Bug #29697052)

• **NDB Replication:** `NDB` did not handle binary logging of virtual generated columns of type `BLOB` correctly. Now such columns are always regarded as having zero length.

• **NDB Cluster APIs:** The memcached sources included with the NDB distribution would not build with `-Werror=format-security`. Now warnings are no longer treated as errors when compiling these files. (Bug #29512411)

• **NDB Cluster APIs:** It was not possible to scan a table whose `SingleUserMode` property had been set to `SingleUserModeReadWrite` or `SingleUserModeReadOnly`. (Bug #29493714)

• **NDB Cluster APIs:** The MGM API `ndb_logevent_get_next2()` function did not behave correctly on Windows and 32-bit Linux platforms. (Bug #94917, Bug #29609070)

• The version of Python expected by `ndb_setup.py` was not specified clearly on some platforms. (Bug #29818645)

• Lack of `SharedGlobalMemory` was incorrectly reported as lack of undo buffer memory, even though the cluster used no disk data tables. (Bug #29806771)

References: This issue is a regression of: Bug #92125, Bug #28537319.

• Long `TCKEYREQ` signals did not always use the expected format when invoked from `TCINDXREQ` processing. (Bug #29772731)

• It was possible for an internal `NDB_SCHEMA_OBJECT` to be released too early or not at all; in addition, it was possible to create such an object that reused an existing key. (Bug #29759063)

• `ndb_restore` sometimes used `exit()` rather than `exitHandler()` to terminate the program, which could lead to resources not being properly freed. (Bug #29744353)

• Improved error message printed when the maximum offset for a `FIXED` column is exceeded. (Bug #29714670)

• Communication between the schema distribution client and the schema distribution coordinator is done using `NDB_SCHEMA_OBJECT` as well as by writing rows to the `ndb_schema` table in `NDB`. This allowed for the possibility of a number of different race conditions between when the registration of the schema operation and when the coordinator was notified of it.

This fix addresses the following issues related to the situation just described:
• The coordinator failed to abort active schema operations when the binary logging thread was restarted.

• Schema operations already registered were not aborted properly.

• The distribution client failed to detect correctly when schema distribution was not ready.

• The distribution client, when killed, exited without marking the current schema operation as failed.

• An operation in `NDB_SHARE` could be accessed without the proper locks being in place.

In addition, usage of the `ndb_schema_share` global pointer was removed, and replaced with detecting whether the schema distribution is ready by checking whether an operation for `mysql.ndb_schema` has been created in `NDB_SHARE`. (Bug #29639381)

• With `DataMemory` set to 200 GB, `ndbmtd` failed to start. (Bug #29630367)

• When a backup fails due to `ABORT_BACKUP_ORD` being received while waiting for buffer space, the backup calls `closeScan()` and then sends a `SCAN_FRAGREQ` signal to the `DBLQH` block to close the scan. As part of receiving `SCAN_FRAGCONF` in response, `scanConf()` is called on the operation object for the file record which in turn calls `updateWritePtr()` on the file system buffer (`FsBuffer`). At this point the length sent by `updateWritePtr()` should be 0, but in this case was not, which meant that the buffer did not have enough space even though it did not, the problem being that the size is calculated as `scanStop - scanStart` and these values were held over since the previous `SCAN_FRAGCONF` was received, and were not reset due to being out of buffer space.

To avoid this problem, we now set `scanStart = scanStop` in `confirmBufferData()` (formerly `scanConfExtra()`) which is called as part of processing the `SCAN_FRAGCONF`, indirectly by `scanConf()` for the backup and first local checkpoint files, and directly for the LCP files which use only the operation record for the data buffer. (Bug #29601253)

• The setting for `MaxDMLOperationsPerTransaction` was not validated in a timely fashion, leading to data node failure rather than a management server error in the event that its value exceeded that of `MaxNoOfConcurrentOperations`. (Bug #29549572)

• Data nodes could fail due to an assert in the `DBTC` block under certain circumstances in resource-constrained environments. (Bug #29528188)

• When the `DBSPJ` block called the internal function `lookup_resume()` to schedule a previously enqueued operation, it used a correlation ID which could have been produced from its immediate ancestor in the execution order, and not its parent in the query tree as assumed. This could happen during execution of a `SELECT STRAIGHT_JOIN` query.

Now `NDB` checks whether the execution ancestor is different from the query tree parent, and if not, performs a lookup of the query tree parent, and the parent’s correlation ID is enqueued to be executed later. (Bug #29501263)

• When a new master took over, sending a `MASTER_LCP_REQ` signal and executing `MASTER_LCPCONF` from participating nodes, it expected that they had not completed the current local checkpoint under the previous master, which need not be true. (Bug #29487340, Bug #29601546)

• When restoring `TINYBLOB` columns, `ndb_restore` now treats them as having the `BINARY` character set. (Bug #29486538)

• When selecting a sorted result set from a query that included a `LIMIT` clause on a single table, and where the sort was executed as `Using filesort` and the `ref` access method was used on an ordered index, it was possible for the result set to be missing one or more rows. (Bug #29474188)
MySQL NDB Cluster 8.0 Release Notes

- Restoration of epochs by `ndb_restore` failed due to temporary redo errors. Now `ndb_restore` retries epoch updates when such errors occur. (Bug #29466089)

- `ndb_restore` tried to extract an 8-character substring of a table name when checking to determine whether or not the table was a blob table, regardless of the length of the name. (Bug #29465794)

- When a pushed join was used in combination with the `eq_ref` access method it was possible to obtain an incorrect join result due to the 1 row cache mechanism implemented in NDB 8.0.16 as part of the work done in that version to extend NDB condition pushdown by allowing referring values from previous tables. This issue is now fixed by turning off this caching mechanism and reading the row directly from the handler instead, when there is a pushed condition defined on the table. (Bug #29460314)

- Improved and made more efficient the conversion of rows by the `ha_ndbcluster` handler from the format used internally by NDB to that used by the MySQL server for columns that contain neither BLOB nor BIT values, which is the most common case. (Bug #29435461)

- A failed `DROP TABLE` could be attempted an infinite number of times in the event of a temporary error. Now in such cases, the number of retries is limited to 100. (Bug #29355155)

- `ndb_restore --restore-epoch` incorrectly reported the stop GCP as 1 less than the actual position. (Bug #29343655)

- A `SavedEvent` object in the CMVMI kernel block is written into a circular buffer. Such an object is split in two when wrapping at the end of the buffer; NDB looked beyond the end of the buffer instead of in the wrapped data at the buffer’s beginning. (Bug #29336793)

- NDB did not compile with `-DWITH_SYSTEM_LIBS=ON` due to an incorrectly configured dependency on `zlib`. (Bug #29304517)

- Removed a memory leak found when running `ndb_mgmd --config-file` after compiling NDB with Clang 7. (Bug #29284643)

- Removed `clang` compiler warnings caused by usage of extra ; characters outside functions; these are incompatible with C++98. (Bug #29227925)

- Adding a column defined as `TIMESTAMP DEFAULT CURRENT_TIMESTAMP` to an NDB table is not supported with `ALGORITHM=INPLACE`. Attempting to do so now causes an error. (Bug #28128849)

- Added support which was missing in `ndb_restore` for conversions between the following sets of types:
  - BLOB and BINARY or VARBINARY columns
  - TEXT and BLOB columns
  - BLOB columns with unequal lengths
  - BINARY and VARBINARY columns with unequal lengths
  (Bug #28074988)

- Restore points in backups created with the `SNAPSHOTSTART` option (see Using The NDB Cluster Management Client to Create a Backup) were not always consistent with epoch boundaries. (Bug #27566346)

  References: See also: Bug #27497461.

- Neither the `MAX_EXECUTION_TIME` optimizer hint nor the `max_execution_time` system variable was respected for DDL statements or queries against INFORMATION_SCHEMA tables while an NDB global schema lock was in effect. (Bug #27538139)
MySQL NDB Cluster 8.0 Release Notes

• DDL operations were not always performed correctly on database objects including databases and tables, when multi-byte character sets were used for the names of either or both of these. (Bug #27150334)

• `ndb_import` did not always free up all resources used before exiting. (Bug #27130143)

• `NDBCLUSTER` subscription log printouts provided only 2 words of the bitmap (in most cases containing 8 words), which made it difficult to diagnose schema distribution issues. (Bug #22180480)

• For certain tables with very large rows and a very large primary key, `START BACKUP SNAPSHOTEND` while performing inserts into one of these tables or `START BACKUP SNAPSHOTSTART` with concurrent deletes could lead to data node errors.

As part of this fix, `ndb_print_backup_file` can now read backup files created in very old versions of NDB Cluster (6.3 and earlier); in addition, this utility can now also read undo log files. (Bug #94654, Bug #29485977)

• When one of multiple SQL nodes which were connected to the cluster was down and then rejoined the cluster, or a new SQL node joined the cluster, this node did not use the data dictionary correctly, and thus did not always add, alter, or drop databases properly when synchronizing with the existing SQL nodes.

Now, during schema distribution at startup, the SQL node compares all databases on the data nodes with those in its own data dictionary. If any database on the data nodes is found to be missing from the SQL node’s data dictionary, the SQL Node installs it locally using `CREATE DATABASE`; the database is created using the default MySQL Server database properties currently in effect on this SQL node.

Changes in MySQL NDB Cluster 8.0.16 (2019-04-25, Development Milestone)

MySQL NDB Cluster 8.0.16 is a new development release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the `NDB` storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in NDB Cluster.

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 8.0 through MySQL 8.0.16 (see Changes in MySQL 8.0.16 (2019-04-25, General Availability)).

• Deprecation and Removal Notes

• SQL Syntax Notes

• Functionality Added or Changed

• Bugs Fixed

Deprecation and Removal Notes

• **Incompatible Change:** Distribution of privileges amongst MySQL servers connected to NDB Cluster, as implemented in NDB 7.6 and earlier, does not function in NDB 8.0, and most code supporting these has now been removed. When a `mysqld` detects such tables in `NDB`, it creates shadow tables local to itself using the `InnoDB` storage engine; these shadow tables are created on each MySQL server connected to
an NDB cluster. Privilege tables using the NDB storage engine are not employed for access control; once all connected MySQL servers are upgraded, the privilege tables in NDB can be removed safely using ndb_drop_table.

For compatibility reasons, ndb_restore --restore-privilege-tables can still be used to restore distributed privilege tables present in a backup taken from a previous release of NDB Cluster to a cluster running NDB 8.0. These tables are handled as described in the preceding paragraph.

For additional information regarding upgrades from previous NDB Cluster release series to NDB 8.0, see Upgrading and Downgrading NDB Cluster.

**SQL Syntax Notes**

- **Incompatible Change:** For consistency with InnoDB, the NDB storage engine now uses a generated constraint name if the CONSTRAINT symbol clause is not specified, or the CONSTRAINT keyword is specified without a symbol. In previous NDB releases, NDB used the FOREIGN KEY index_name value.

  This change described above may introduce incompatibilities for applications that depend on the previous foreign key constraint naming behavior. (Bug #29173134)

**Functionality Added or Changed**

- **Packaging:** A Docker image for this release can be obtained from https://hub.docker.com/r/mysql/mysql-cluster/. (Bug #96084, Bug #30010921)

- Allocation of resources in the transaction coordinator (see The DBTC Block) is now performed using dynamic memory pools. This means that resource allocation determined by data node configuration parameters such as those discussed in Transaction parameters and Transaction temporary storage is now limited so as not to exceed the total resources available to the transaction coordinator.

  As part of this work, several new data node parameters controlling transactional resources in DBTC, listed here, have also been added. For more information about these new parameters, see Transaction resource allocation parameters. (Bug #29164271, Bug #29194843)

  References: See also: Bug #29131828.

- **NDB** backups can now be performed in a parallel fashion on individual data nodes using multiple local data managers (LDMs). (Previously, backups were done in parallel across data nodes, but were always serial within data node processes.) No special syntax is required for the START BACKUP command in the ndb_mgm client to enable this feature, but all data nodes must be using multiple LDMs. This means that data nodes must be running ndbmtd and they must be configured to use multiple LDMs prior to taking the backup (see Multi-Threading Configuration Parameters (ndbmtd)).

  ndb_restore also now detects such a backup and automatically attempts to restore it in parallel. It is also possible to restore backups taken in parallel to a previous version of NDB Cluster by slightly modifying the usual restore procedure.

  For more information about taking and restoring NDB Cluster backups that were created using parallelism on the data nodes, see Taking an NDB Backup with Parallel Data Nodes, and Restoring from a backup taken in parallel. (Bug #28563639, Bug #28993400)

  • The compile-cluster script included in the NDB source distribution no longer supports in-source builds.

  • Building with CMake3 is now supported by the compile-cluster script included in the NDB source distribution.
• As part of its automatic synchronization mechanism, NDB now implements a metadata change monitor thread for detecting changes made to metadata for data objects such as tables, tablespaces, and log file groups with the MySQL data dictionary. This thread runs in the background, checking every 60 seconds for inconsistencies between the NDB dictionary and the MySQL data dictionary.

The monitor polling interval can be adjusted by setting the value of the `ndb_metadata_check_interval` system variable, and can be disabled altogether by setting `ndb_metadata_check` to OFF. The number of times that inconsistencies have been detected since `mysqld` was last started is shown as the status variable, `Ndb_metadata_detected_count`.

• Condition pushdown is no longer limited to predicate terms referring to column values from the same table to which the condition was being pushed; column values from tables earlier in the query plan can now also be referred to from pushed conditions. This lets the data nodes filter out more rows (in parallel), leaving less work to be performed by a single `mysqld` process, which is expected to provide significant improvements in query performance.

For more information, see Engine Condition Pushdown Optimization.

**Bugs Fixed**

• **Important Change; NDB Disk Data:** `mysqldump` terminated unexpectedly when attempting to dump NDB disk data tables. The underlying reason for this was that `mysqldump` expected to find information relating to undo log buffers in the `EXTRA` column of the `INFORMATION_SCHEMA.FILES` table but this information had been removed in NDB 8.0.13. This information is now restored to the `EXTRA` column. (Bug #28800252)

• **Important Change:** When restoring to a cluster using data node IDs different from those in the original cluster, `ndb_restore` tried to open files corresponding to node ID 0. To keep this from happening, the `--nodeid` and `--backupid` options—neither of which has a default value—are both now explicitly required when invoking `ndb_restore`. (Bug #28813708)

• **Important Change:** Starting with this release, the default value of the `ndb_log_bin` system variable is now `FALSE`. (Bug #27135706)

• **Packaging; MySQL NDB ClusterJ:** `libndbclient` was missing from builds on some platforms. (Bug #28997603)

• **NDB Disk Data:** When a log file group had more than 18 undo logs, it was not possible to restart the cluster. (Bug #251155785)

  References: See also: Bug #28922609.

• **NDB Disk Data:** Concurrent `CREATE TABLE` statements using tablespaces caused deadlocks between metadata locks. This occurred when `Ndb_metadata_change_monitor` acquired exclusive metadata locks on tablespaces and logfile groups after detecting metadata changes, due to the fact that each exclusive metadata lock in turn acquired a global schema lock. This fix attempts to solve that issue by downgrading the locks taken by `Ndb_metadata_change_monitor` to `MDL_SHARED_READ`. (Bug #29175268)

  References: See also: Bug #29394407.

• **NDB Disk Data:** The error message returned when validation of `MaxNoOfOpenFiles` in relation to `InitialNoOfOpenFiles` failed has been improved to make the nature of the problem clearer to users. (Bug #28943749)

• **NDB Disk Data:** Schema distribution of `ALTER TABLESPACE` and `ALTER LOGFILE GROUP` statements failed on a participant MySQL server if the referenced tablespace or log file group did not exist in its data
Now in such cases, the effects of the statement are distributed successfully regardless of any initial mismatch between MySQL servers. (Bug #28866336)

- **NDB Disk Data:** Repeated execution of `ALTER TABLESPACE ... ADD DATAFILE` against the same tablespace caused data nodes to hang and left them, after being killed manually, unable to restart. (Bug #22605467)

- **NDB Replication:** A `DROP DATABASE` operation involving certain very large tables could lead to an unplanned shutdown of the cluster. (Bug #28855062)

- **NDB Replication:** When writes on the master—done in such a way that multiple changes affecting `BLOB` column values belonging to the same primary key were part of the same epoch—were replicated to the slave, Error 1022 occurred due to constraint violations in the `NDB$BLOB_id_part` table. (Bug #28746560)

- **NDB Cluster APIs:** NDB now identifies short-lived transactions not needing the reduction of lock contention provided by `NdbBlob::close()` and no longer invokes this method in cases (such as when autocommit is enabled) in which unlocking merely causes extra work and round trips to be performed prior to committing or aborting the transaction. (Bug #29305592)

  References: See also: Bug #49190, Bug #11757181.

- **NDB Cluster APIs:** When the most recently failed operation was released, the pointer to it held by `NdbTransaction` became invalid and when accessed led to failure of the NDB API application. (Bug #29275244)

  References: See also: Bug #26995027.

- **NDB Cluster APIs:** When the NDB kernel's `SUMA` block sends a `TE_ALTER` event, it does not keep track of when all fragments of the event are sent. When NDB receives the event, it buffers the fragments, and processes the event when all fragments have arrived. An issue could possibly arise for very large table definitions, when the time between transmission and reception could span multiple epochs; during this time, `SUMA` could send a `SUB_GCP_COMPLETE_REP` signal to indicate that it has sent all data for an epoch, even though in this case that is not entirely true since there may be fragments of a `TE_ALTER` event still waiting on the data node to be sent. Reception of the `SUB_GCP_COMPLETE_REP` leads to closing the buffers for that epoch. Thus, when `TE_ALTER` finally arrives, NDB assumes that it is a duplicate from an earlier epoch, and silently discards it.

  We fix the problem by making sure that the `SUMA` kernel block never sends a `SUB_GCP_COMPLETE_REP` for any epoch in which there are unsent fragments for a `SUB_TABLE_DATA` signal.

  This issue could have an impact on NDB API applications making use of `TE_ALTER` events. (SQL nodes do not make any use of `TE_ALTER` events and so they and applications using them were not affected.) (Bug #28836474)

  References: See also: Bug #26995027.

- **When a pushed join executing in the DBSPJ block had to store correlation IDs during query execution, memory for these was allocated for the lifetime of the entire query execution, even though these specific correlation IDs are required only when producing the most recent batch in the result set. Subsequent batches require additional correlation IDs to be stored and allocated; thus, if the query took sufficiently long to complete, this led to exhaustion of query memory (error 20008). Now in such cases, memory is allocated only for the lifetime of the current result batch, and is freed and made available for re-use following completion of the batch. (Bug #29336777)

  References: See also: Bug #26995027.

- **When comparing or hashing a fixed-length string that used a `NO_PAD` collation, any trailing padding characters (typically spaces) were sent to the hashing and comparison functions such that they became significant, even though they were not supposed to be. Now any such trailing spaces are trimmed from a fixed-length string whenever a `NO_PAD` collation is specified.**
Note

Since NO_PAD collations were introduced as part of UCA-9.0 collations in MySQL 8.0, there should be no impact relating to this fix on upgrades to NDB 8.0 from previous GA releases of NDB Cluster.

(Bug #29322313)

- When a NOT IN or NOT BETWEEN predicate was evaluated as a pushed condition, NULL values were not eliminated by the condition as specified in the SQL standard. (Bug #29232744)

  References: See also: Bug #28672214.

- Internally, NDB treats NULL as less than any other value, and predicates of the form column < value or column <= value are checked for possible nulls. Predicates of the form value > column or value >= column were not checked, which could lead to errors. Now in such cases, these predicates are rewritten so that the column comes first, so that they are also checked for the presence of NULL. (Bug #29231709)

  References: See also: Bug #92407, Bug #28643463.

- After folding of constants was implemented in the MySQL Optimizer, a condition containing a DATE or DATETIME literal could no longer be pushed down by NDB. (Bug #29161281)

- When a join condition made a comparison between a column of a temporal data type such as DATE or DATETIME and a constant of the same type, the predicate was pushed if the condition was expressed in the form column operator constant, but not when in inverted order (as constant inverse_operator column). (Bug #29058732)

  When processing a pushed condition, NDB did not detect errors or warnings thrown when a literal value being compared was outside the range of the data type it was being compared with, and thus truncated. This could lead to excess or missing rows in the result. (Bug #29054626)

- If an EQ_REF or REF key in the child of a pushed join referred to any columns of a table not a member of the pushed join, this table was not an NDB table (because its format was of nonnative endianness), and the data type of the column being joined on was stored in an endian-sensitive format, then the key generated was generated, likely resulting in the return of an (invalid) empty join result.

  Since only big endian platforms may store tables in nonnative (little endian) formats, this issue was expected only on such platforms, most notably SPARC, and not on x86 platforms. (Bug #29010641)

- API and data nodes running NDB 7.6 and later could not use an existing parsed configuration from an earlier release series due to being overly strict with regard to having values defined for configuration parameters new to the later release, which placed a restriction on possible upgrade paths. Now NDB 7.6 and later are less strict about having all new parameters specified explicitly in the configuration which they are served, and use hard-coded default values in such cases. (Bug #28993400)

- NDB 7.6 SQL nodes hung when trying to connect to an NDB 8.0 cluster. (Bug #28985685)

- The schema distribution data maintained in the NDB binary logging thread keeping track of the number of subscribers to the NDB schema table always allocated some memory structures for 256 data nodes regardless of the actual number of nodes. Now NDB allocates only as many of these structures as are actually needed. (Bug #28949523)

- Added DUMP 406 (NdbsfsDumpRequests) to provide NDB file system information to global checkpoint and local checkpoint stall reports in the node logs. (Bug #28922609)
• When a joined table was eliminated early as not pushable, it could not be referred to in any subsequent join conditions from other tables without eliminating those conditions from consideration even if those conditions were otherwise pushable. (Bug #28898811)

• When starting or restarting an SQL node and connecting to a cluster where NDB was already started, NDB reported Error 4009 Cluster Failure because it could not acquire a global schema lock. This was because the MySQL Server as part of initialization acquires exclusive metadata locks in order to modify internal data structures, and the ndbcluster plugin acquires the global schema lock. If the connection to NDB was not yet properly set up during mysql initialization, mysql received a warning from ndbcluster when the latter failed to acquire global schema lock, and printed it to the log file, causing an unexpected error in the log. This is fixed by not pushing any warnings to background threads when failure to acquire a global schema lock occurs and pushing the NDB error as a warning instead. (Bug #28898544)

• A race condition between the DBACC and DBLQH kernel blocks occurred when different operations in a transaction on the same row were concurrently being prepared and aborted. This could result in DBTUP attempting to prepare an operation when a preceding operation had been aborted, which was unexpected and could thus lead to undefined behavior including potential data node failures. To solve this issue, DBACC and DBLQH now check that all dependencies are still valid before attempting to prepare an operation.

Note
This fix also supersedes a previous one made for a related issue which was originally reported as Bug #28500861. (Bug #28893633)

• Where a data node was restarted after a configuration change whose result was a decrease in the sum of MaxNoOfTables, MaxNoOfOrderedIndexes, and MaxNoOfUniqueHashIndexes, it sometimes failed with a misleading error message which suggested both a temporary error and a bug, neither of which was the case.

The failure itself is expected, being due to the fact that there is at least one table object with an ID greater than the (new) sum of the parameters just mentioned, and that this table cannot be restored since the maximum value for the ID allowed is limited by that sum. The error message has been changed to reflect this, and now indicates that this is a permanent error due to a problem configuration. (Bug #28884880)

• The ndbinfo.cpustat table reported inaccurate information regarding send threads. (Bug #28884157)

• Execution of an LCP_COMPLETE_REP signal from the master while the LCP status was IDLE led to an assertion. (Bug #28871889)

• NDB now provides on-the-fly .frm file translation during discovery of tables created in versions of the software that did not support the MySQL Data Dictionary. Previously, such translation of tables that had old-style metadata was supported only during schema synchronization during MySQL server startup, but not subsequently, which led to errors when NDB tables having old-style metadata, created by ndb_restore and other such tools after mysql had been started, were accessed using SHOW CREATE TABLE or SELECT; these tables were usable only after restarting mysql. With this fix, the restart is no longer required. (Bug #28841009)

• An in-place upgrade to an NDB 8.0 release from an earlier release did not remove .ndb files, even though these are no longer used in NDB 8.0. (Bug #28832816)
- Removed `storage/ndb/demos` and the demonstration scripts and support files it contained from the source tree. These were obsolete and unmaintained, and did not function with any current version of NDB Cluster.

  Also removed `storage/ndb/include/newtonapi`, which included files relating to an obsolete and unmaintained API not supported in any release of NDB Cluster, as well as references elsewhere to these files. (Bug #28808766)

- There was no version compatibility table for NDB 8.x; this meant that API nodes running NDB 8.0.13 or 7.6.x could not connect to data nodes running NDB 8.0.14. This issue manifested itself for NDB API users as a failure in `wait_until_ready()`. (Bug #28776365)

  References: See also: Bug #18886034, Bug #18874849.

- Issuing a STOP command in the `ndb_mgm` client caused `ndbmtd` processes which had recently been added to the cluster to hang in Phase 4 during shutdown. (Bug #28772867)

- A fix for a previous issue disabled the usage of pushed conditions for lookup type (eq_ref) operations in pushed joins. It was thought at the time that not pushing a lookup condition would not have any measurable impact on performance, since only a single row could be eliminated if the condition failed. The solution implemented at that time did not take into account the possibility that, in a pushed join, a lookup operation could be a parent operation for other lookups, and even scan operations, which meant that eliminating a single row could actually result in an entire branch being eliminated in error. (Bug #28728603)

  References: This issue is a regression of: Bug #27397802.

- When a local checkpoint (LCP) was complete on all data nodes except one, and this node failed, NDB did not continue with the steps required to finish the LCP. This led to the following issues:

  No new LCPs could be started.

  Redo and Undo logs were not trimmed and so grew excessively large, causing an increase in times for recovery from disk. This led to write service failure, which eventually led to cluster shutdown when the head of the redo log met the tail. This placed a limit on cluster uptime.

  Node restarts were no longer possible, due to the fact that a data node restart requires that the node's state be made durable on disk before it can provide redundancy when joining the cluster. For a cluster with two data nodes and two replicas, this meant that a restart of the entire cluster (system restart) was required to fix the issue (this was not necessary for a cluster with two replicas and four or more data nodes). (Bug #28728485, Bug #28698831)

  References: See also: Bug #11757421.

- The pushability of a condition to NDB was limited in that all predicates joined by a logical AND within a given condition had to be pushable to NDB in order for the entire condition to be pushed. In some cases this severely restricted the pushability of conditions. This fix breaks up the condition into its components, and evaluates the pushability of each predicate; if some of the predicates cannot be pushed, they are returned as a remainder condition which can be evaluated by the MySQL server. (Bug #28728007)

- Running ANALYZE TABLE on an NDB table with an index having longer than the supported maximum length caused data nodes to fail. (Bug #28714864)

- It was possible in certain cases for nodes to hang during an initial restart. (Bug #28698831)

  References: See also: Bug #27622643.
• When a condition was pushed to a storage engine, it was re-evaluated by the server, in spite of the fact that only rows matching the pushed condition should ever be returned to the server in such cases. (Bug #28672214)

• In some cases, one and sometimes more data nodes underwent an unplanned shutdown while running ndb_restore. This occurred most often, but was not always restricted to, when restoring to a cluster having a different number of data nodes from the cluster on which the original backup had been taken.

The root cause of this issue was exhaustion of the pool of SafeCounter objects, used by the DBDICT kernel block as part of executing schema transactions, and taken from a per-block-instance pool shared with protocols used for NDB event setup and subscription processing. The concurrency of event setup and subscription processing is such that the SafeCounter pool can be exhausted; event and subscription processing can handle pool exhaustion, but schema transaction processing could not, which could result in the node shutdown experienced during restoration.

This problem is solved by giving DBDICT schema transactions an isolated pool of reserved SafeCounters which cannot be exhausted by concurrent NDB event activity. (Bug #28595915)

• When a backup aborted due to buffer exhaustion, synchronization of the signal queues prior to the expected drop of triggers for insert, update, and delete operations resulted in abort signals being processed before the STOP_BACKUP phase could continue. The abort changed the backup status to ABORT_BACKUP_ORD, which led to an unplanned shutdown of the data node since resuming STOP_BACKUP requires that the state be STOP_BACKUP_REQ. Now the backup status is not set to STOP_BACKUP_REQ (requesting the backup to continue) until after signal queue synchronization is complete. (Bug #28563639)

• The output of ndb_config --configinfo --xml --query-all now shows that configuration changes for the ThreadConfig and MaxNoOfExecutionThreads data node parameters require system initial restarts (restart="system" initial="true"). (Bug #28494286)

• After a commit failed due to an error, mysqld shut down unexpectedly while trying to get the name of the table involved. This was due to an issue in the internal function ndbcluster_print_error(). (Bug #28435082)

• API nodes should observe that a node is moving through SL_STOPPING phases (graceful stop) and stop using the node for new transactions, which minimizes potential disruption in the later phases of the node shutdown process. API nodes were only informed of node state changes via periodic heartbeat signals, and so might not be able to avoid interacting with the node shutting down. This generated unnecessary failures when the heartbeat interval was long. Now when a data node is being gracefully stopped, all API nodes are notified directly, allowing them to experience minimal disruption. (Bug #28380808)

• ndb_config --diff-default failed when trying to read a parameter whose default value was the empty string (""). (Bug #27972537)

• ndb_restore did not restore autoincrement values correctly when one or more staging tables were in use. As part of this fix, we also in such cases block applying of the SYSTAB_0 backup log, whose content continued to be applied directly based on the table ID, which could overwrite the autoincrement values stored in SYSTAB_0 for unrelated tables. (Bug #27917769, Bug #27831990)

References: See also: Bug #27832033.

• ndb_restore employed a mechanism for restoring autoincrement values which was not atomic, and thus could yield incorrect autoincrement values being restored when multiple instances of ndb_restore were used in parallel. (Bug #27832033)

References: See also: Bug #27917769, Bug #27831990.
MySQL NDB Cluster 8.0 Release Notes

- Executing `SELECT * FROM INFORMATION_SCHEMA.TABLES` caused SQL nodes to restart in some cases. (Bug #27613173)

- When tables with `BLOB` columns were dropped and then re-created with a different number of `BLOB` columns the event definitions for monitoring table changes could become inconsistent in certain error situations involving communication errors when the expected cleanup of the corresponding events was not performed. In particular, when the new versions of the tables had more `BLOB` columns than the original tables, some events could be missing. (Bug #27072756)

- When query memory was exhausted in the `DBSPJ` kernel block while storing correlation IDs for deferred operations, the query was aborted with error status 20000 `Query aborted due to out of query memory`. (Bug #26995027)

  References: See also: Bug #86537.

- When running a cluster with 4 or more data nodes under very high loads, data nodes could sometimes fail with Error 899 `Rowid already allocated`. (Bug #25960230)

- `mysqld` shut down unexpectedly when a purge of the binary log was requested before the server had completely started, and it was thus not yet ready to delete rows from the `ndb_binlog_index` table. Now when this occurs, requests for any needed purges of the `ndb_binlog_index` table are saved in a queue and held for execution when the server has completely started. (Bug #25817834)

- `MaxBufferedEpochs` is used on data nodes to avoid excessive buffering of row changes due to lagging `NDB` event API subscribers; when epoch acknowledgements from one or more subscribers lag by this number of epochs, an asynchronous disconnection is triggered, allowing the data node to release the buffer space used for subscriptions. Since this disconnection is asynchronous, it may be the case that it has not completed before additional new epochs are completed on the data node, resulting in new epochs not being able to seize GCP completion records, generating warnings such as those shown here:

  ![Example Warning](image)

  And leading to the following warning:

  ![Example Warning](image)

  This fix performs the following modifications:

  - Modifies the size of the GCP completion record pool to ensure that there is always some extra headroom to account for the asynchronous nature of the disconnect processing previously described, thus avoiding `c_gcp_list` seize failures.

  - Modifies the wording of the `MaxBufferedEpochs` warning to avoid the contradictory phrase “100 > 100”.

    (Bug #20344149)

- Asynchronous disconnection of `mysqld` from the cluster caused any subsequent attempt to start an NDB API transaction to fail. If this occurred during a bulk delete operation, the SQL layer called `HA::end_bulk_delete()`, whose implementation by `ha_ndbcluster` assumed that a transaction had been started, and could fail if this was not the case. This problem is fixed by checking that the transaction pointer used by this method is set before referencing it. (Bug #20116393)
MySQL NDB Cluster 8.0 Release Notes

• Removed warnings raised when compiling NDB with Clang 6. (Bug #93634, Bug #29112560)

• When executing the redo log in debug mode it was possible for a data node to fail when deallocating a row. (Bug #93273, Bug #28955797)

• An NDB table having both a foreign key on another NDB table using ON DELETE CASCADE and one or more TEXT or BLOB columns leaked memory.

  As part of this fix, ON DELETE CASCADE is no longer supported for foreign keys on NDB tables when the child table contains a column that uses any of the BLOB or TEXT types. (Bug #89511, Bug #27484882)

Changes in MySQL NDB Cluster 8.0.15 (Not released)

MySQL NDB Cluster 8.0.15 was not released. NDB Cluster 8.0.14 is followed by NDB Cluster 8.0.16; users of NDB 8.0.14 should upgrade to 8.0.16 when the latter version becomes available.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in NDB Cluster.

Changes in MySQL NDB Cluster 8.0.14 (2019-01-21, Development Milestone)

MySQL NDB Cluster 8.0.14 is a new development release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in NDB Cluster.

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 8.0 through MySQL 8.0.14 (see Changes in MySQL 8.0.14 (2019-01-21, General Availability)).

• Functionality Added or Changed
  • Bugs Fixed

Functionality Added or Changed

• Performance: This release introduces a number of significant improvements in the performance of scans; these are listed here:

  • Row checksums help detect hardware issues, but do so at the expense of performance. NDB now offers the possibility of disabling these by setting the new ndb_row_checksum server system variable to 0; doing this means that row checksums are not used for new or altered tables. This can have a significant impact (5 to 10 percent, in some cases) on performance for all types of queries. This variable is set to 1 by default, to provide compatibility with the previous behavior.

  • A query consisting of a scan can execute for a longer time in the LDM threads when the queue is not busy.

  • Previously, columns were read before checking a pushed condition; now checking of a pushed condition is done before reading any columns.
• Performance of pushed joins should see significant improvement when using range scans as part of join execution.

• **NDB Disk Data:** NDB now implements schema distribution of disk data objects including tablespaces and log file groups by SQL nodes when they connect to a cluster, just as it does for NDB databases and in-memory tables. This eliminates a possible mismatch between the MySQL data dictionary and the NDB dictionary following a native backup and restore that could arise when disk data tablespaces and undo log file groups were restored to the NDB dictionary, but not to the MySQL Server's data dictionary.

• **NDB Disk Data:** NDB now makes use of the MySQL data dictionary to ensure correct distribution of tablespaces and log file groups across all cluster SQL nodes when connecting to the cluster.

• The extra metadata property for NDB tables is now used to store information from the MySQL data dictionary. Because this information is significantly larger than the binary representation previously stored here (a .frm file, no longer used), the hard-coded size limit for this extra metadata has been increased.

  This change can have an impact on downgrades: Trying to read NDB tables created in NDB 8.0.14 and later may cause data nodes running NDB 8.0.13 or earlier to fail on startup with NDB error code 2355 *Failure to restore schema: Permanent error, external action needed: Resource configuration error.* This can happen if the table’s metadata exceeds 6K in size, which was the old limit. Tables created in NDB 8.0.13 and earlier can be read by later versions without any issues.

  For more information, see Changes in NDB table extra metadata, and See also MySQL Data Dictionary. (Bug #27230681)

**Bugs Fixed**

• **Packaging:** Expected NDB header files were in the *devel* RPM package instead of *libndbclient-devel*. (Bug #84580, Bug #26448330)

• **ndbmemcache:** *libndbclient.so* was not able to find and load *libssl.so*, which could cause issues with ndbmemcache and Java-based programs using NDB. (Bug #26824659)

  References: See also: Bug #27882088, Bug #28410275.

• **MySQL NDB ClusterJ:** The *ndb.clusterj* test for NDB 8.0.13 failed when being run more than once. This was deal to a new, stricter rule with NDB 8.0.13 that did not allow temporary files being left behind in the variable folder of *mysql-test-run (mtr).* With this fix, the temporary files are deleted before the test is executed. (Bug #28279038)

• **MySQL NDB ClusterJ:** A *NullPointerException* was thrown when a full table scan was performed with ClusterJ on tables containing either a BLOB or a TEXT field. It was because the proper object initializations were omitted, and they have now been added by this fix. (Bug #28199372, Bug #91242)

• The *version_comment* system variable was not correctly configured in *mysqld* binaries and returned a generic pattern instead of the proper value. This affected all NDB Cluster binary releases with the exception of *.deb* packages. (Bug #29054235)

• Trying to build from source using `-DWITH_NDBCLUSTER` and `-Werror` failed with GCC 8. (Bug #28707282)

• When copying deleted rows from a live node to a node just starting, it is possible for one or more of these rows to have a global checkpoint index equal to zero. If this happened at the same time that a full local checkpoint was started due to the undo log getting full, the `LCP_SKIP` bit was set for a row having GCI = 0, leading to an unplanned shutdown of the data node. (Bug #28372628)
• **ndbmtd** sometimes experienced a hang when exiting due to log thread shutdown. (Bug #28027150)

• **NDB** has an upper limit of 128 characters for a fully qualified table name. Due to the fact that **mysql** names NDB tables using the format `database_name/catalog_name/table_name`, where `catalog_name` is always `def`, it is possible for statements such as **CREATE TABLE** to fail in spite of the fact that neither the table name nor the database name exceeds the 63-character limit imposed by **NDB**. The error raised in such cases was misleading and has been replaced. (Bug #27769521)

References: See also: Bug #27769801.

• When the **SUMA** kernel block receives a **SUB_STOP_REQ** signal, it executes the signal then replies with **SUB_STOP_CONF**. (After this response is relayed back to the API, the API is open to send more **SUB_STOP_REQ** signals.) After sending the **SUB_STOP_CONF**, SUMA drops the subscription if no subscribers are present, which involves sending multiple **DROP_TRIG_IMPL_REQ** messages to **DBTUP**. LocalProxy can handle up to 21 of these requests in parallel; any more than this are queued in the Short Time Queue. When execution of a **DROP_TRIG_IMPL_REQ** was delayed, there was a chance for the queue to become overloaded, leading to a data node shutdown with **Error in short time queue**.

This issue is fixed by delaying the execution of the **SUB_STOP_REQ** signal if **DBTUP** is already handling **DROP_TRIG_IMPL_REQ** signals at full capacity, rather than queueing up the **DROP_TRIG_IMPL_REQ** signals. (Bug #26574003)

• **ndb_restore** returned -1 instead of the expected exit code in the event of an index rebuild failure. (Bug #25112726)

• When starting, a data node copies metadata, while a local checkpoint updates metadata. To avoid any conflict, any ongoing LCP activity is paused while metadata is being copied. An issue arose when a local checkpoint was paused on a given node, and another node that was also restarting checked for a complete LCP on this node; the check actually caused the LCP to be completed before copying of metadata was complete and so ended the pause prematurely. Now in such cases, the LCP completion check waits to complete a paused LCP until copying of metadata is finished and the pause ends as expected, within the LCP in which it began. (Bug #24827685)

• **ndbout** and **ndberr** became invalid after exiting from **mgmd_run()**, and redirecting to them before the next call to **mgmd_run()** caused a segmentation fault, during an **ndb_mgmd** service restart. This fix ensures that **ndbout** and **ndberr** remain valid at all times. (Bug #17732772, Bug #28536919)

• **NdbScanFilter** did not always handle **NULL** according to the SQL standard, which could result in sending non-qualifying rows to be filtered (otherwise not necessary) by the MySQL server. (Bug #92407, Bug #28643463)

References: See also: Bug #93977, Bug #29231709.

• The internal function **ndb_my_error()** was used in **ndbcluster_get_tablespace_statistics()** and **prepare_inplace_alter_table()** to report errors when the function failed to interact with **NDB**. The function was expected to push the NDB error as warning on the stack and then set an error by translating the NDB error to a MySQL error and then finally call **my_error()** with the translated error. When calling **my_error()**, the function extracts a format string that may contain placeholders and use the format string in a function similar to **sprintf()**, which in this case could read arbitrary memory leading to a segmentation fault, due to the fact that **my_error()** was called without any arguments.

The fix is always to push the NDB error as a warning and then set an error with a provided message. A new helper function has been added to **Thd_ndb** to be used in place of **ndb_my_error()**. (Bug #92244, Bug #28575934)

• Running out of undo log buffer memory was reported using error 921 Out of transaction memory ... (increase **SharedGlobalMemory**).
This problem is fixed by introducing a new error code 923 \textit{Out of undo buffer memory (increase UNDO_BUFFER_SIZE)}. (Bug \#92125, Bug \#28537319)

• When moving an \texttt{OperationRec} from the serial to the parallel queue, \texttt{Dbacc::startNext()} failed to update the \texttt{Operationrec::OP_ACC_LOCK_MODE} flag which is required to reflect the accumulated \texttt{OP_LOCK_MODE} of all previous operations in the parallel queue. This inconsistency in the ACC lock queues caused the scan lock takeover mechanism to fail, as it incorrectly concluded that a lock to take over was not held. The same failure caused an assert when aborting an operation that was a member of such an inconsistent parallel lock queue. (Bug \#92100, Bug \#28530928)

• \texttt{ndb_restore} did not free all memory used after being called to restore a table that already existed. (Bug \#92085, Bug \#28525898)

• A data node failed during startup due to the arrival of a \texttt{SCAN_FRAGREQ} signal during the restore phase. This signal originated from a scan begun before the node had previously failed and which should have been aborted due to the involvement of the failed node in it. (Bug \#92059, Bug \#28518448)

• \texttt{DBTUP} sent the error \textit{Tuple corruption detected} when a read operation attempted to read the value of a tuple inserted within the same transaction. (Bug \#92009, Bug \#28500861)

References: See also: Bug \#28893633.

• False constraint violation errors could occur when executing updates on self-referential foreign keys. (Bug \#91965, Bug \#28486390)

References: See also: Bug \#90644, Bug \#27930382.

• An \texttt{NDB} internal trigger definition could be dropped while pending instances of the trigger remained to be executed, by attempting to look up the definition for a trigger which had already been released. This caused unpredictable and thus unsafe behavior possibly leading to data node failure. The root cause of the issue lay in an invalid assumption in the code relating to determining whether a given trigger had been released; the issue is fixed by ensuring that the behavior of \texttt{NDB}, when a trigger definition is determined to have been released, is consistent, and that it meets expectations. (Bug \#91894, Bug \#28451957)

• In some cases, a workload that included a high number of concurrent inserts caused data node failures when using debug builds. (Bug \#91764, Bug \#28387450, Bug \#29055038)

• During an initial node restart with disk data tables present and \texttt{TwoPassInitialNodeRestartCopy} enabled, \texttt{DBTUP} used an unsafe scan in disk order. Such scans are no longer employed in this case. (Bug \#91724, Bug \#28378227)

• Checking for old LCP files tested the table version, but this was not always dependable. Now, instead of relying on the table version, the check regards as invalid any LCP file having a maxGCI smaller than its createGci. (Bug \#91637, Bug \#28346565)

• In certain cases, a cascade update trigger was fired repeatedly on the same record, which eventually consumed all available concurrent operations, leading to Error 233 \textit{Out of operation records in transaction coordinator (increase MaxNoOfConcurrentOperations)}. If MaxNoOfConcurrentOperations was set to a value sufficiently high to avoid this, the issue manifested as data nodes consuming very large amounts of CPU, very likely eventually leading to a timeout. (Bug \#91472, Bug \#28262259)
Changes in MySQL NDB Cluster 8.0.13 (2018-10-23, Development Milestone)

MySQL NDB Cluster 8.0.13 is a new development release of NDB 8.0, based on MySQL Server 8.0 and including features in version 8.0 of the NDB storage engine, as well as fixing recently discovered bugs in previous NDB Cluster releases.

Obtaining NDB Cluster 8.0. NDB Cluster 8.0 source code and binaries can be obtained from https://dev.mysql.com/downloads/cluster/.

For an overview of changes made in NDB Cluster 8.0, see What is New in NDB Cluster.

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 8.0 through MySQL 8.0.13 (see Changes in MySQL 8.0.13 (2018-10-22, General Availability)).

• Functionality Added or Changed

• Bugs Fixed

Functionality Added or Changed

• Important Change; NDB Disk Data: The following changes are made in the display of information about Disk Data files in the INFORMATION_SCHEMA_FILES table:

  • Tablespaces and log file groups are no longer represented in the FILES table. (These constructs are not actually files.)

  • Each data file is now represented by a single row in the FILES table. Each undo log file is also now represented in this table by one row only. (Previously, a row was displayed for each copy of each of these files on each data node.)

  • For rows corresponding to data files or undo log files, node ID and undo log buffer information is no longer displayed in the EXTRA column of the FILES table.

  ! Important

  This change is reverted in NDB 8.0.15. (Bug #92796, Bug #28800252)

• Important Change; NDB Client Programs: Removed the deprecated --ndb option for perror. Use ndb_perror to obtain error message information from NDB error codes instead. (Bug #81705, Bug #23523957)

References: See also: Bug #81704, Bug #23523926.

• Important Change: Beginning with this release, MySQL NDB Cluster is being developed in parallel with the standard MySQL 8.0 server under a new unified release model with the following features:

  • NDB 8.0 is developed in, built from, and released with the MySQL 8.0 source code tree.

  • The numbering scheme for NDB Cluster 8.0 releases follows the scheme for MySQL 8.0, starting with the current MySQL release (8.0.13).

  • Building the source with NDB support appends -cluster to the version string returned by mysql -V, as shown here:

    shell> mysql -V
NDB binaries continue to display both the MySQL Server version and the NDB engine version, like this:

```shell
shell> ndb_mgm -V
MySQL distrib mysql-8.0.13 ndb-8.0.13-dmr, for Linux (x86_64)
```

In MySQL Cluster NDB 8.0, these two version numbers are always the same.

To build the MySQL 8.0.13 (or later) source with NDB Cluster support, use the CMake option `DWITH_NDBCLUSTER`.

- **NDB Cluster APIs:** Added the Table methods `getExtraMetadata()` and `setExtraMetadata()`.
- **INFORMATION_SCHEMA** tables now are populated with tablespace statistics for MySQL Cluster tables. (Bug #27167728)
- It is now possible to specify a set of cores to be used for I/O threads performing offline multithreaded builds of ordered indexes, as opposed to normal I/O duties such as file I/O, compression, or decompression. “Offline” in this context refers to building of ordered indexes performed when the parent table is not being written to; such building takes place when an NDB cluster performs a node or system restart, or as part of restoring a cluster from backup using `ndb_restore --rebuild-indexes`.

In addition, the default behaviour for offline index build work is modified to use all cores available to `ndbmtd`, rather limiting itself to the core reserved for the I/O thread. Doing so can improve restart and restore times and performance, availability, and the user experience.

This enhancement is implemented as follows:

1. The default value for `BuildIndexThreads` is changed from 0 to 128. This means that offline ordered index builds are now multithreaded by default.
2. The default value for `TwoPassInitialNodeRestartCopy` is changed from `false` to `true`. This means that an initial node restart first copies all data from a “live” node to one that is starting —without creating any indexes—builds ordered indexes offline, and then again synchronizes its data with the live node, that is, synchronizing twice and building indexes offline between the two synchronizations. This causes an initial node restart to behave more like the normal restart of a node, and reduces the time required for building indexes.
3. A new thread type (`idxbld`) is defined for the `ThreadConfig` configuration parameter, to allow locking of offline index build threads to specific CPUs.

In addition, NDB now distinguishes the thread types that are accessible to “ThreadConfig” by the following two criteria:

1. Whether the thread is an execution thread. Threads of types `main`, `ldm`, `recv`, `rep`, `tc`, and `send` are execution threads; thread types `io`, `watchdog`, and `idxbld` are not.
2. Whether the allocation of the thread to a given task is permanent or temporary. Currently all thread types except `idxbld` are permanent.

For additional information, see the descriptions of the parameters in the Manual. (Bug #25835748, Bug #26928111)

- When performing an NDB backup, the `ndbinfo.logbuffers` table now displays information regarding buffer usage by the backup process on each data node. This is implemented as rows reflecting two new log types in addition to `REDO` and `DD-UNDO`. One of these rows has the log type `BACKUP-DATA`,
which shows the amount of data buffer used during backup to copy fragments to backup files. The other row has the log type BACKUP-LOG, which displays the amount of log buffer used during the backup to record changes made after the backup has started. One each of these log_type rows is shown in the logbuffers table for each data node in the cluster. Rows having these two log types are present in the table only while an NDB backup is currently in progress. (Bug #25822988)

• Added the ODirectSyncFlag configuration parameter for data nodes. When enabled, the data node treats all completed filesystem writes to the redo log as though they had been performed using fsync.  

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>This parameter has no effect if at least one of the following conditions is true:</td>
</tr>
<tr>
<td>• ODirect is not enabled.</td>
</tr>
<tr>
<td>• InitFragmentLogFiles is set to SPARSE.</td>
</tr>
</tbody>
</table>

(Bug #25428560)

• Added the --logbuffer-size option for ndbd and ndbmt.d, for use in debugging with a large number of log messages. This controls the size of the data node log buffer; the default (32K) is intended for normal operations. (Bug #89679, Bug #27550943)

• Prior to NDB 8.0, all string hashing was based on first transforming the string into a normalized form, then MD5-hashing the resulting binary image. This could give rise to some performance problems, for the following reasons:
  • The normalized string is always space padded to its full length. For a VARCHAR, this often involved adding more spaces than there were characters in the original string.
  • The string libraries were not optimized for this space padding, and added considerable overhead in some use cases.
  • The padding semantics varied between character sets, some of which were not padded to their full length.
  • The transformed string could become quite large, even without space padding; some Unicode 9.0 collations can transform a single code point into 100 bytes of character data or more.
  • Subsequent MD5 hashing consisted mainly of padding with spaces, and was not particularly efficient, possibly causing additional performance penalties by flush significant portions of the L1 cache.

Collations provide their own hash functions, which hash the string directly without first creating a normalized string. In addition, for Unicode 9.0 collations, the hashes are computed without padding. NDB now takes advantage of this built-in function whenever hashing a string identified as using a Unicode 9.0 collation.

Since, for other collations there are existing databases which are hash partitioned on the transformed string, NDB continues to employ the previous method for hashing strings that use these, to maintain compatibility. (Bug #89609, Bug #27523758)

References: See also: Bug #89590, Bug #27515000, Bug #89604, Bug #27522732.

• A table created in NDB 7.6 and earlier contains metadata in the form of a compressed .frm file, which is no longer supported in MySQL 8.0. To facilitate online upgrades to NDB 8.0, NDB performs on-the-fly translation of this metadata and writes it into the MySQL Server's data dictionary, which enables the
mysql in NDB Cluster 8.0 to work with the table without preventing subsequent use of the table by a previous version of the NDB software.

**Important**

Once a table’s structure has been modified in NDB 8.0, its metadata is stored using the Data Dictionary, and it can no longer be accessed by NDB 7.6 and earlier.

This enhancement also makes it possible to restore an NDB backup made using an earlier version to a cluster running NDB 8.0 (or later).

## Bugs Fixed

- **Important Change; NDB Disk Data:** It was possible to issue a `CREATE TABLE` statement that referred to a nonexistent tablespace. Now such a statement fails with an error. (Bug #85859, Bug #25860404)

- **Important Change; NDB Replication:** Because the MySQL Server now executes `RESET MASTER` with a global read lock, the behavior of this statement when used with NDB Cluster has changed in the following two respects:
  - It is no longer guaranteed to be synchronous; that is, it is now possible that a read coming immediately before `RESET MASTER` is issued may not be logged until after the binary log has been rotated.
  - It now behaves identically, regardless of whether the statement is issued on the same SQL node that is writing the binary log, or on a different SQL node in the same cluster.

**Note**

SHOW BINLOG EVENTS, FLUSH LOGS, and most data definition statements continue, as they did in previous NDB versions, to operate in a synchronous fashion. (Bug #89976, Bug #27665565)

- **Important Change:** NDB supports any of the following three values for the `CREATE TABLE` statement’s `ROW_FORMAT` option: DEFAULT, FIXED, and DYNAMIC. Formerly, any values other than these were accepted but resulted in DYNAMIC being used. Now a `CREATE TABLE` statement that attempts to create an NDB table fails with an error if `ROW_FORMAT` is used, and does not have one of the three values listed. (Bug #88803, Bug #27230898)

- **Microsoft Windows; ndbinfo Information Database:** The process ID of the monitor process used on Windows platforms by RESTART to spawn and restart a mysql is now shown in the ndbinfo.processes table as an angel_pid. (Bug #90235, Bug #27767237)

- **NDB Cluster APIs:** The example NDB API programs ndbapi_array_simple and ndbapi_array_using_adapter did not perform cleanup following execution; in addition, the example program ndbapi_simple_dual did not check to see whether the table used by this example already existed. Due to these issues, none of these examples could be run more than once in succession. The issues just described have been corrected in the example sources, and the relevant code listings in the NDB API documentation have been updated to match. (Bug #27009386)

- **NDB Cluster APIs:** A previous fix for an issue, in which the failure of multiple data nodes during a partial restart could cause API nodes to fail, did not properly check the validity of the associated NdbReceiver object before proceeding. Now in such cases an invalid object triggers handling for invalid signals, rather than a node failure. (Bug #25902137)
References: This issue is a regression of: Bug #25092498.

• **NDB Cluster APIs**: Incorrect results, usually an empty result set, were returned when `setBound()` was used to specify a **NULL** bound. This issue appears to have been caused by a problem in gcc, limited to cases using the old version of this method (which does not employ `NdbRecord`), and is fixed by rewriting the problematic internal logic in the old implementation. (Bug #89468, Bug #27461752)

• **NDB Cluster APIs**: Released NDB API objects are kept in one or more `Ndb_free_list` structures for later reuse. Each list also keeps track of all objects seized from it, and makes sure that these are eventually released back to it. In the event that the internal function `NdbScanOperation::init()` failed, it was possible for an `NdbApiSignal` already allocated by the `NdbOperation` to be leaked. Now in such cases, `NdbScanOperation::release()` is called to release any objects allocated by the failed `NdbScanOperation` before it is returned to the free list.

This fix also handles a similar issue with `NdbOperation::init()`, where a failed call could also leak a signal. (Bug #89249, Bug #27389894)

• **NDB Cluster APIs**: Removed the unused `TFSentinel` implementation class, which raised compiler warnings on 32-bit systems. (Bug #89005, Bug #27302881)

• **NDB Cluster APIs**: The success of thread creation by API calls was not always checked, which could lead to timeouts in some cases. (Bug #88784, Bug #27225714)

• **NDB Cluster APIs**: The file `storage/ndb/src/ndbapi/ndberror.c` was renamed to `ndberror.cpp`. (Bug #87725, Bug #26781567)

• **ndbinfo Information Database**: Counts of committed rows and committed operations per fragment used by some tables in `ndbinfo` were taken from the `DBACC` block, but due to the fact that commit signals can arrive out of order, transient counter values could be negative. This could happen if, for example, a transaction contained several interleaved insert and delete operations on the same row; in such cases, commit signals for delete operations could arrive before those for the corresponding insert operations, leading to a failure in `DBACC`.

This issue is fixed by using the counts of committed rows which are kept in `DBTUP`, which do not have this problem. (Bug #88087, Bug #26968613)

• **NDB Client Programs**: When passed an invalid connection string, the `ndb_mgm` client did not always free up all memory used before exiting. (Bug #90179, Bug #27737906)

• **NDB Client Programs**: `ndb_show_tables` did not always free up all memory which it used. (Bug #90152, Bug #27727544)

• **NDB Client Programs**: On Unix platforms, the Auto-Installer failed to stop the cluster when `ndb_mgmd` was installed in a directory other than the default. (Bug #89624, Bug #27531186)

• **NDB Client Programs**: The Auto-Installer did not provide a mechanism for setting the `ServerPort` parameter. (Bug #89623, Bug #27539823)

• **MySQL NDB ClusterJ**: When a table containing a **BLOB** or a **TEXT** field was being queried with ClusterJ for a record that did not exist, an exception ("The method is not valid in current blob state") was thrown. (Bug #28536926)

• **MySQL NDB ClusterJ**: ClusterJ quit unexpectedly as there was no error handling in the `scanIndex()` function of the `ClusterTransactionImpl` class for a null returned to it internally by the `scanIndex()` method of the `ndbTransaction` class. (Bug #27297681, Bug #88989)

• Local checkpoints did not always handle **DROP TABLE** operations correctly. (Bug #27926532)
References: This issue is a regression of: Bug #26908347, Bug #26968613.

• In some circumstances, when a transaction was aborted in the DBTC block, there remained links to trigger records from operation records which were not yet reference-counted, but when such an operation record was released the trigger reference count was still decremented. (Bug #27629680)

• An internal buffer being reused immediately after it had been freed could lead to an unplanned data node shutdown. (Bug #27622643)

References: See also: Bug #28698831.

• An NDB online backup consists of data, which is fuzzy, and a redo and undo log. To restore to a consistent state it is necessary to ensure that the log contains all of the changes spanning the capture of the fuzzy data portion and beyond to a consistent snapshot point. This is achieved by waiting for a GCI boundary to be passed after the capture of data is complete, but before stopping change logging and recording the stop GCI in the backup's metadata.

At restore time, the log is replayed up to the stop GCI, restoring the system to the state it had at the consistent stop GCI. A problem arose when, under load, it was possible to select a GCI boundary which occurred too early and did not span all the data captured. This could lead to inconsistencies when restoring the backup; these could be be noticed as broken constraints or corrupted BLOB entries.

Now the stop GCI is chosen is so that it spans the entire duration of the fuzzy data capture process, so that the backup log always contains all data within a given stop GCI. (Bug #27497461)

References: See also: Bug #27566346.

• For NDB tables, when a foreign key was added or dropped as a part of a DDL statement, the foreign key metadata for all parent tables referenced should be reloaded in the handler on all SQL nodes connected to the cluster, but this was done only on the mysqld on which the statement was executed. Due to this, any subsequent queries relying on foreign key metadata from the corresponding parent tables could return inconsistent results. (Bug #27439587)

References: See also: Bug #82989, Bug #24666177.

• ANALYZE TABLE used excessive amounts of CPU on large, low-cardinality tables. (Bug #27438963)

• Queries using very large lists with IN were not handled correctly, which could lead to data node failures. (Bug #27397802)

References: See also: Bug #28728603.

• A data node overload could in some situations lead to an unplanned shutdown of the data node, which led to all data nodes disconnecting from the management and nodes.

This was due to a situation in which API_FAILREQ was not the last received signal prior to the node failure.

As part of this fix, the transaction coordinator's handling of SCAN_TABREQ signals for an ApiConnectRecord in an incorrect state was also improved. (Bug #27381901)

References: See also: Bug #47039, Bug #11755287.

• In a two-node cluster, when the node having the lowest ID was started using --nostart, API clients could not connect, failing with Could not alloc node id at HOST port PORT_NO: No free node id found for mysql(API). (Bug #27225212)
• Changing `MaxNoOfExecutionThreads` without an initial system restart led to an unplanned data node shutdown. (Bug #27169282)

References: This issue is a regression of: Bug #26908347, Bug #26968613.

• In most cases, and especially in error conditions, `NDB` command-line programs failed on exit to free memory used by option handling, and failed to call `ndb_end()`. This is fixed by removing the internal methods `ndb_load_defaults()` and `ndb_free_defaults()` from `storage/ndb/include/util/ndb_opts.h`, and replacing these with an `Ndb_opts` class that automatically frees such resources as part of its destructor. (Bug #26930148)

References: See also: Bug #87396, Bug #26617328.

• A query against the `INFORMATION_SCHEMA.FILES` table returned no results when it included an `ORDER BY` clause. (Bug #26877788)

• ClusterJ failed to connect to a MySQL node that used `utf8mb4_800_ci_ai` as its default character set for connection. Also, ClusterJ quit unexpectedly when handling a table with a character set number of 255 or larger. This fix corrected both issues. (Bug #26027722)

• During a restart, `DBLQH` loads redo log part metadata for each redo log part it manages, from one or more redo log files. Since each file has a limited capacity for metadata, the number of files which must be consulted depends on the size of the redo log part. These files are opened, read, and closed sequentially, but the closing of one file occurs concurrently with the opening of the next.

In cases where closing of the file was slow, it was possible for more than 4 files per redo log part to be open concurrently; since these files were opened using the `OM_WRITE_BUFFER` option, more than 4 chunks of write buffer were allocated per part in such cases. The write buffer pool is not unlimited; if all redo log parts were in a similar state, the pool was exhausted, causing the data node to shut down.

This issue is resolved by avoiding the use of `OM_WRITE_BUFFER` during metadata reload, so that any transient opening of more than 4 redo log files per log file part no longer leads to failure of the data node. (Bug #25965370)

• Under certain conditions, data nodes restarted unnecessarily during execution of `ALTER TABLE... REORGANIZE PARTITION`. (Bug #25675481)

References: See also: Bug #26735618, Bug #27191468.

• Race conditions sometimes occurred during asynchronous disconnection and reconnection of the transporter while other threads concurrently inserted signal data into the send buffers, leading to an unplanned shutdown of the cluster.

As part of the work fixing this issue, the internal templating function used by the Transporter Registry when it prepares a send is refactored to use likely-or-unlikely logic to speed up execution, and to remove a number of duplicate checks for NULL. (Bug #24444908, Bug #25128512)

References: See also: Bug #20112700.

• `ndb_restore` sometimes logged data file and log file progress values much greater than 100%. (Bug #20989106)

• Removed unneeded debug printouts from the internal function `ha_ndbcluster::copy_fk_for_offline_alter()`. (Bug #90991, Bug #28069711)

• The internal function `BitmaskImpl::setRange()` set one bit fewer than specified. (Bug #90648, Bug #27931995)
• Inserting a row into an NDB table having a self-referencing foreign key that referenced a unique index on the table other than the primary key failed with ER_NO_REFERENCED_ROW_2. This was due to the fact that NDB checked foreign key constraints before the unique index was updated, so that the constraint check was unable to use the index for locating the row. Now, in such cases, NDB waits until all unique index values have been updated before checking foreign key constraints on the inserted row. (Bug #90644, Bug #27930382)

References: See also: Bug #91965, Bug #28486390.

• Removed all references to the C++ register storage class in the NDB Cluster sources; use of this specifier, which was deprecated in C++11 and removed in C++17, raised warnings when building with recent compilers. (Bug #90110, Bug #27705985)

• It was not possible to create an NDB table using PARTITION_BALANCE set to FOR_RA_BY_LDM_X_2, FOR_RA_BY_LDM_X_3, or FOR_RA_BY_LDM_X_4. (Bug #89811, Bug #27602352)

References: This issue is a regression of: Bug #81759, Bug #23544301.

• Adding a [tcp] or [shm] section to the global configuration file for a cluster with multiple data nodes caused default TCP connections to be lost to the node using the single section. (Bug #89627, Bug #27532407)

• Removed a memory leak in the configuration file parser. (Bug #89392, Bug #27440614)

• Fixed a number of implicit-fallthrough warnings, warnings raised by uninitialized values, and other warnings encountered when compiling NDB with GCC 7.2.0. (Bug #89254, Bug #89255, Bug #89258, Bug #89259, Bug #89270, Bug #27390714, Bug #27390745, Bug #27390684, Bug #27390816, Bug #27396662)

References: See also: Bug #88136, Bug #26990244.

• Node connection states were not always reported correctly by ClusterMgr immediately after reporting a disconnection. (Bug #89121, Bug #27349285)

• As a result of the reuse of code intended for send threads when performing an assist send, all of the local release send buffers were released to the global pool, which caused the intended level of the local send buffer pool to be ignored. Now send threads and assisting worker threads follow their own policies for maintaining their local buffer pools. (Bug #89119, Bug #27349118)

• When the PGMAN block seized a new Page_request record using seizeLast, its return value was not checked, which could cause access to invalid memory. (Bug #89009, Bug #27303191)

• TCROLLBACKREP signals were not handled correctly by the DBTC kernel block. (Bug #89004, Bug #27302734)

• When sending priority A signals, we now ensure that the number of pending signals is explicitly initialized. (Bug #88986, Bug #27294856)

• The internal function ha_ndbcluster::unpack_record() did not perform proper error handling. (Bug #88587, Bug #27150980)

• CHECKSUM is not supported for NDB tables, but this was not not reflected in the CHECKSUM column of the INFORMATION_SCHEMA.TABLES table, which could potentially assume a random value in such cases. Now the value of this column is always set to NULL for NDB tables, just as it is for InnoDB tables. (Bug #88552, Bug #27143813)

• Removed a memory leak detected when running ndb_mgm -e "CLUSTERLOG ...". (Bug #88517, Bug #27128846)
• When terminating, `ndb_config` did not release all memory which it had used. (Bug #88515, Bug #27128398)

• `ndb_restore` did not free memory properly before exiting. (Bug #88514, Bug #27128361)

• In certain circumstances where multiple `Ndb` objects were being used in parallel from an API node, the block number extracted from a block reference in `DBLQH` was the same as that of a `SUMA` block even though the request was coming from an API node. Due to this ambiguity, `DBLQH` mistook the request from the API node for a request from a `SUMA` block and failed. This is fixed by checking node IDs before checking block numbers. (Bug #88441, Bug #27130570)

• A join entirely within the materialized part of a semijoin was not pushed even if it could have been. In addition, `EXPLAIN` provided no information about why the join was not pushed. (Bug #88224, Bug #27022925)

References: See also: Bug #27067538.

• All known compiler warnings raised by `-Werror` when building the `NDB` source code have been fixed. (Bug #88136, Bug #26990244)

• When the duplicate weedout algorithm was used for evaluating a semijoin, the result had missing rows. (Bug #88117, Bug #26984919)

References: See also: Bug #87992, Bug #26926666.

• `NDB` did not compile with GCC 7. (Bug #88011, Bug #26933472)

• A table used in a loose scan could be used as a child in a pushed join query, leading to possibly incorrect results. (Bug #87992, Bug #26926666)

• When representing a materialized semijoin in the query plan, the MySQL Optimizer inserted extra `QEP_TAB` and `JOIN_TAB` objects to represent access to the materialized subquery result. The join pushdown analyzer did not properly set up its internal data structures for these, leaving them uninitialized instead. This meant that later usage of any item objects referencing the materialized semijoin accessed an initialized `tableno` column when accessing a 64-bit `tableno` bitmask, possibly referring to a point beyond its end, leading to an unplanned shutdown of the SQL node. (Bug #87971, Bug #26919289)

• In some cases, a `SCAN_FRAGCONF` signal was received after a `SCAN_FRAGREQ` with a close flag had already been sent, clearing the timer. When this occurred, the next `SCAN_FRAGREF` to arrive caused time tracking to fail. Now in such cases, a check for a cleared timer is performed prior to processing the `SCAN_FRAGREF` message. (Bug #87942, Bug #26908347)

• While deleting an element in `Dbacc`, or moving it during hash table expansion or reduction, the method used (`getLastAndRemove()`) could return a reference to a removed element on a released page, which could later be referenced from the functions calling it. This was due to a change brought about by the implementation of dynamic index memory in NDB 7.6.2; previously, the page had always belonged to a single `Dbacc` instance, so accessing it was safe. This was no longer the case following the change; a page released in `Dbacc` could be placed directly into the global page pool where any other thread could then allocate it.

Now we make sure that newly released pages in `Dbacc` are kept within the current `Dbacc` instance and not given over directly to the global page pool. In addition, the reference to a released page has been removed; the affected internal method now returns the last element by value, rather than by reference. (Bug #87932, Bug #26906640)

References: See also: Bug #87987, Bug #26925595.
• When creating a table with a nonexistent conflict detection function, NDB returned an improper error message. (Bug #87628, Bug #26730019)

• ndb_top failed to build with the error "HAVE_NCURSESW_H" is not defined. (Bug #87035, Bug #26429281)

• In a MySQL Cluster with one MySQL Server configured to write a binary log failure occurred when creating and using an NDB table with non-stored generated columns. The problem arose only when the product was built with debugging support. (Bug #86084, Bug #25957586)

• It was possible to create or alter a STORAGE MEMORY table using a nonexistent tablespace without any error resulting. Such an operation now fails with Error 3510 ER_TABLESPACE_MISSING_WITH_NAME, as intended. (Bug #82116, Bug #23744378)

• ndb_restore --print-data --hex did not print trailing 0s of LONGVARBINARY values. (Bug #65560, Bug #14198580)

• When the internal function ha_ndbcluster::copy_fk_for_offline_alter() checked dependent objects on a table from which it was supposed to drop a foreign key, it did not perform any filtering for foreign keys, making it possible for it to attempt retrieval of an index or trigger instead, leading to a spurious Error 723 (No such table).

Release Series Changelogs: MySQL NDB Cluster 8.0

This section contains unified changelog information for the NDB Cluster 8.0 release series.

For changelogs covering individual MySQL NDB Cluster 8.0 releases, see NDB Cluster Release Notes.

For general information about features added in MySQL NDB Cluster 8.0, see What is New in NDB Cluster 8.0.

For an overview of features added in MySQL 8.0 that are not specific to NDB Cluster, see What Is New in MySQL 8.0. For a complete list of all bug fixes and feature changes made in MySQL 8.0 that are not specific to NDB Cluster, see the MySQL 8.0 Release Notes.

Changes in MySQL NDB Cluster 8.0.18 (2019-10-14, Release Candidate)

• Functionality Added or Changed

• Bugs Fixed

Functionality Added or Changed

• Important Change: The 63-byte limit on NDB database and table names has been removed. These identifiers may now take up to 64 bytes, as when using other MySQL storage engines. For more information, see Previous NDB Cluster Issues Resolved in NDB Cluster 8.0. (Bug #27447958)

• Important Change: Implemented the NDB_STORED_USER privilege, which enables sharing of users, roles, and privileges across all SQL nodes attached to a given NDB Cluster. This replaces the distributed grant tables mechanism from NDB 7.6 and earlier versions of NDB Cluster, which was removed in NDB 8.0.16 due to its incompatibility with changes made to the MySQL privilege system in MySQL 8.0.

A user or role which has this privilege is propagated, along with its (other) privileges to a MySQL server (SQL node) as soon as it connects to the cluster. Changes made to the privileges of the user or role are synchronized immediately with all connected SQL nodes.

NDB_STORED_USER can be granted to users and roles other than reserved accounts such as mysql.session@localhost or mysql.infoschema@localhost. A role can be shared, but
assigning a shared role to a user does not cause this user to be shared; the `NDB_STORED_USER` privilege must be granted to the user explicitly in order for the user to be shared between NDB Cluster SQL nodes.

The `NDB_STORED_USER` privilege is always global and must be granted using `ON *.*`. This privilege is recognized only if the MySQL server enables support for the `NDBCLUSTER` storage engine.

For usage information, see the description of `NDB_STORED_USER`. Distributed MySQL Privileges with `NDB_STORED_USER`, has additional information on how `NDB_STORED_USER` and privilege synchronization work. For information on how this change may affect upgrades to NDB 8.0 from previous versions, see Upgrading and Downgrading NDB Cluster.

References: See also: Bug #29862601, Bug #29996547.

• **Important Change:** The maximum row size for an `NDB` table is increased from 14000 to 30000 bytes.

As before, only the first 264 bytes of a `BLOB` or `TEXT` column count towards this total.

The maximum offset for a fixed-width column of an `NDB` table is 8188 bytes; this is also unchanged from previous NDB Cluster releases.

For more information, see Limits Associated with Database Objects in NDB Cluster.

References: See also: Bug #29485977, Bug #29024275.

• **Important Change:** A new binary format has been implemented for the NDB management server's cached configuration file, which is intended to support much larger numbers of nodes in a cluster than previously. Prior to this release, the configuration file supported a maximum of 16381 sections; this number is increased to 4G.

Upgrades to the new format should not require any manual intervention, as the management server (and other cluster nodes) can still read the old format. For downgrades from this release or a later one to NDB 8.0.17 or earlier, it is necessary to remove the binary configuration files prior to starting the old management server binary, or start it using the `--initial` option.

For more information, see Upgrading and Downgrading NDB Cluster.

• **Important Change:** The maximum number of data nodes supported in a single NDB cluster is raised in this release from 48 to 144. The range of supported data node IDs is increased in conjunction with this enhancement to 1-144, inclusive.

In previous releases, recommended node IDs for management nodes were 49 and 50. These values are still supported, but, if used, limit the maximum number of data nodes to 142. For this reason, the recommended node ID values for management servers are now 145 and 146.

The maximum total supported number of nodes of all types in a given cluster is 255. This total is unchanged from previous releases.

For a cluster running more than 48 data nodes, it is not possible to downgrade directly to a previous release that supports only 48 data nodes. In such cases, it is necessary to reduce the number of data nodes to 48 or fewer, and to make sure that all data nodes use node IDs that are less than 49.

• **NDB Cluster APIs:** An alternative constructor for `NdbInterpretedCode` is now provided, which accepts an `NdbRecord` in place of a `Table` object. (Bug #29852377)

• **NDB Cluster APIs:** `NdbScanFilter::cmp()` and the following `NdbInterpretedCode` comparison methods can be now used to compare table column values:
• `branch_col_eq()`
• `branch_col_ge()`
• `branch_col_gt()`
• `branch_col_le()`
• `branch_col_lt()`
• `branch_col_ne()`

When using any of these methods, the table column values to be compared must be of exactly the same type, including with respect to length, precision, and scale. In addition, in all cases, `NULL` is always considered by these methods to be less than any other value. You should also be aware that, when used to compare table column values, `NdbScanFilter::cmp()` does not support all possible values of `BinaryCondition`.

For more information, see the descriptions of the individual API methods.

• **NDB Client Programs:** The dependency of the `ndb_delete_all` utility on the `NDBT` library has been removed. This library, used in `NDB` development for testing, is not required for normal use. The visible change for users is that `ndb_delete_all` no longer prints `NDBT_ProgramExit - status` following completion of its run. Applications that depend upon this behavior should be updated to reflect this change when upgrading to this release.

• `ndb_restore` now reports the specific `NDB` error number and message when it is unable to load a table descriptor from a backup `.ctl` file. This can happen when attempting to restore a backup taken from a later version of the `NDB` Cluster software to a cluster running an earlier version—for example, when the backup includes a table using a character set which is unknown to the version of `ndb_restore` being used to restore it. (Bug #30184265)

• The output from `DUMP 1000` in the `ndb_mgm` client has been extended to provide information regarding total data page usage. (Bug #29841454)

References: See also: Bug #29929996.

• **NDB Cluster’s condition pushdown functionality has been extended as follows:**
  
  • Expressions using any previously allowed comparisons are now supported.
  
  • Comparisons between columns in the same table and of the same type are now supported. The columns must be of exactly the same type.

  **Example:** Suppose there are two tables `t1` and `t2` created as shown here:
  
  ```sql
  CREATE TABLE t1 (a INT, b INT, c CHAR(10), d CHAR(5)) ENGINE=NDB;
  CREATE TABLE t2 LIKE t1;
  ```

  The following joins can now be pushed down to the data nodes:
  
  ```sql
  SELECT * FROM t1 JOIN t2 ON t2.a < t1.a+10;
  SELECT * FROM t1 JOIN t2 ON t2.a = t1.a+t1.b;
  SELECT * FROM t1 JOIN t2 ON t2.a = t1.a+t1.b;
  ```
  
  ```sql
  SELECT * FROM t1 JOIN t2 ON t2.d = SUBSTRING(t1.c,1,5);
  SELECT * FROM t1 JOIN t2 ON t2.c = CONCAT('foo',t1.d,'ba');
  ```

  Supported comparisons are `<`, `<=`, `>`, `>=`, `=`, and `<>`. (Bug #29685643)
• NDB Cluster now uses `table_name_fk_N` as the naming pattern for internally generated foreign keys, which is similar to the `table_name_ibfk_N` pattern used by InnoDB. (Bug #96508, Bug #30171959)

References: See also: Bug #30210839.

• Added the `ndb_schema_dist_lock_wait_timeout` system variable to control how long to wait for a schema lock to be released when trying to update the SQL node's local data dictionary for one or more tables currently in use from the NDB data dictionary's metadata. If this synchronization has not yet occurred by the end of this time, the SQL node returns a warning that schema distribution did not succeed; the next time that the table for which distribution failed is accessed, NDB tries once again to synchronize the table metadata.

• NDB table objects submitted by the metadata change monitor thread are now automatically checked for any mismatches and synchronized by the NDB binary logging thread. The status variable `Ndb_metadata_synced_count` added in this release shows the number of objects synchronized automatically; it is possible to see which objects have been synchronized by checking the cluster log. In addition, the new status variable `Ndb_metadata_blacklist_size` indicates the number of objects for which synchronization has failed.

References: See also: Bug #30000202.

• It is now possible to build NDB for 64-bit ARM CPUs from the NDB Cluster sources. Currently, we do not provide any precompiled binaries for this platform.

• Start times for the `ndb_mgmd` management node daemon have been significantly improved as follows:
  • More efficient handling of properties from configuration data can decrease startup times for the management server by a factor of 6 or more as compared with previous versions.
  • Host names not present in the management server's `hosts` file no longer create a bottleneck during startup, making `ndb_mgmd` start times up to 20 times shorter where these are used.
  • Columns of NDB tables can now be renamed online, using `ALGORITHM=INPLACE`.

References: See also: Bug #28609968.

**Bugs Fixed**

• **Important Change:** Because the current implementation for node failure handling cannot guarantee that even a single transaction of size `MaxNoOfConcurrentOperations` is completed in each round, this parameter is once again used to set a global limit on the total number of concurrent operations in all transactions within a single transaction coordinator instance. (Bug #96617, Bug #30216204)

• **Partitioning; NDB Disk Data:** Creation of a partitioned disk data table was unsuccessful due to a missing metadata lock on the tablespace specified in the `CREATE TABLE` statement. (Bug #28876892)

• **NDB Disk Data:** Tablespaces and data files are not tightly coupled in NDB, in the sense that they are represented by independent `NdbDictionary` objects. Thus, when metadata is restored using the `ndb_restore` tool, there was no guarantee that the tablespace and its associated datafile objects were restored at the same time. This led to the possibility that the tablespace mismatch was detected and automatically synchronized to the data dictionary before the datafile was restored to NDB. This issue also applied to log file groups and undo files.

To fix this problem, the metadata change monitor now submits tablespaces and logfile groups only if their corresponding datafiles and undofiles actually exist in NDB. (Bug #30090080)
MySQL NDB Cluster 8.0 Release Notes

- **NDB Disk Data**: When a data node failed following creation and population of an NDB table having columns on disk, but prior to execution of a local checkpoint, it was possible to lose row data from the tablespace. (Bug #29506869)

- **NDB Cluster APIs**: The NDB API examples `ndbapi_array_simple.cpp` (see NDB API Simple Array Example) and `ndbapi_array_using_adapter.cpp` (see NDB API Simple Array Example Using Adapter) made assignments directly to a `std::vector` array instead of using `push_back()` calls to do so. (Bug #28956047)

- Faulty calculation of microseconds caused the internal `ndb_milli_sleep()` function to sleep for too short a time. (Bug #30211922)

- Once a data node is started, 95% of its configured DataMemory should be available for normal data, with 5% to spare for use in critical situations. During the node startup process, all of its configured DataMemory is usable for data, in order to minimize the risk that restoring the node data fails due to running out of data memory due to some dynamic memory structure using more pages for the same data than when the node was stopped. For example, a hash table grows differently during a restart than it did previously, since the order of inserts to the table differs from the historical order.

The issue raised in this bug report occurred when a check that the data memory used plus the spare data memory did not exceed the value set for DataMemory failed at the point where the spare memory was reserved. This happened as the state of the data node transitioned from starting to started, when reserving spare pages. After calculating the number of reserved pages to be used for spare memory, and then the number of shared pages (that is, pages from shared global memory) to be used for this, the number of reserved pages already allocated was not taken into consideration. (Bug #30205182)

References: See also: Bug #29616383.

- It was not possible to use `ndb_restore` and a backup taken from an NDB 8.0 cluster to restore to a cluster running NDB 7.6. (Bug #30184658)

References: See also: Bug #30221717.

- When starting, a data node’s local sysfile was not updated between the first completed local checkpoint and start phase 50. (Bug #30086352)

- In the BACKUP block, the assumption was made that the first record in `c_backups` was the local checkpoint record, which is not always the case. Now NDB loops through the records in `c_backups` to find the (correct) LCP record instead. (Bug #30080194)

- During node takeover for the master it was possible to end in the state LCP_STATUS_IDLE while the remaining data nodes were reporting their state as LCP_TAB_SAVED. This led to failure of the node when attempting to handle reception of a LCP_COMPLETE_REP signal since this is not expected when idle. Now in such cases local checkpoint handling is done in a manner that ensures that this node finishes in the proper state (LCP_TAB_SAVED). (Bug #30032863)

- When a MySQL Server built with NDBCLUSTER support was run on Solaris/x86, it failed during schema distribution. The root cause of the problem was an issue with the Developer Studio compiler used to build binaries for this platform when optimization level `-xO2` was used. This issue is fixed by using optimization level `-xO1` instead for NDBCLUSTER built for Solaris/x86. (Bug #30031130)

References: See also: Bug #28585914, Bug #30014295.

- NDB used `free()` directly to deallocate `ndb_mgm_configuration` objects instead of calling `ndb_mgm_destroy_configuration()`, which correctly uses `delete` for deallocation. (Bug #29998980)
• Default configuration sections did not have the configuration section types set when unpacked into memory, which caused a memory leak since this meant that the section destructor would not destroy the entries for these sections. (Bug #29965125)

• No error was propagated when NDB failed to discover a table due to the table format being old and no longer supported, which could cause the NDB handler to retry the discovery operation endlessly and thereby hang. (Bug #29949096, Bug #29934763)

• During upgrade of an NDB Cluster when half of the data nodes were running NDB 7.6 while the remainder were running NDB 8.0, attempting to shut down those nodes which were running NDB 7.6 led to failure of one node with the error CHECK FAILEDNODEPTR.P→DBLQHFAI. (Bug #29912988, Bug #30141203)

• Altering a table in the middle of an ongoing transaction caused a table discovery operation which led to the transaction being committed prematurely; in addition, no error was returned when performing further updates as part of the same transaction.

Now in such cases, the table discovery operation fails, when a transaction is in progress. (Bug #29911440)

• When performing a local checkpoint (LCP), a table's schema version was intermittently read as 0, which caused NDB LCP handling to treat the table as though it were being dropped. This could affect rebuilding of indexes offline by ndb_restore while the table was in the TABLE_READ_ONLY state. Now the function reading the schema version (getCreateSchemaVersion()) no longer not changes it while the table is read-only. (Bug #29910397)

• When an error occurs on an SQL node during schema distribution, information about this was written in the error log, but no indication was provided by the mysql client that the DDL statement in question was unsuccessful. Now in such cases, one or more generic warnings are displayed by the client to indicate that a given schema distribution operation has not been successful, with further information available in the error log of the originating SQL node. (Bug #29889869)

• Errors and warnings pushed to the execution thread during metadata synchronization and metadata change detection were not properly logged and cleared. (Bug #29874313)

• Altering a normal column to a stored generated column was performed online even though this is not supported. (Bug #29862463)

• A pushed join with ORDER BY did not always return the rows of the result in the specified order. This could occur when the optimizer used an ordered index to provide the ordering and the index used a column from the table that served as the root of the pushed join. (Bug #29860378)

• A number of issues in the Backup block for local checkpoints (LCPs) were found and fixed, including the following:
  • Bytes written to LCP part files were not always included in the LCP byte count.
  • The maximum record size for the buffer used for all LCP part files was not updated in all cases in which the table maximum record size had changed.
  • LCP surfacing could occur for LCP scans at times other than when receiving SCAN_FRAGCONF signals.
  • It was possible in some cases for the table currently being scanned to be altered in the middle of a scan request, which behavior is not supported.

(Bug #29843373)
References: See also: Bug #29485977.

• The requestInfo fields for the long and short forms of the LQHKEYREQ signal had different definitions; bits used for the key length in the short version were reused for flags in the long version, since the key length is implicit in the section length of the long version of the signal but it was possible for long LQHKEYREQ signals to contain a keylength in these same bits, which could be misinterpreted by the receiving local query handler, potentially leading to errors. Checks have now been implemented to make sure that this no longer happens. (Bug #29820838)

• The list of dropped shares could hold only one dropped NDB_SHARE instance for each key, which prevented NDB_SHARE instances with same key from being dropped multiple times while handlers held references to those NDB_SHARE instances. This interfered with keeping track of the memory allocated and being able to release it if mysql shut down without all handlers having released their references to the shares. To resolve this issue, the dropped share list has been changed to use a list type which allows more than one NDB_SHARE with the same key to exist at the same time. (Bug #29812659, Bug #29812613)

• Removed an ndb_restore compile-time dependency on table names that was defined by the ndbcluster plugin. (Bug #29801100)

• When creating a table in parallel on multiple SQL nodes, the result was a race condition between checking that the table existed and opening the table, which caused CREATE TABLE IF NOT EXISTS to fail with Error 1. This was the result of two issues, described with their fixes here:

  1. Opening a table whose NDB_SHARE did not exist returned the non-descriptive error message ERROR 1296 (HY000): Got error 1 'Unknown error code' from NDBCLUSTER. This is fixed with a warning describing the problem in more detail, along with a more sensible error code.

     It was possible to open a table before schema synchronization was completed. This is fixed with a warning better describing the problem, along with an error indicating that cluster is not yet ready.

In addition, this fixes a related issue in which creating indexes sometimes also failed with Error 1. (Bug #29793534, Bug #29871321)

• Previously, for a pushed condition, every request sent to NDB for a given table caused the generation of a new instance of NdbInterpretedCode. When joining tables, generation of multiple requests for all tables following the first table in the query plan is very likely; if the pushed condition had no dependencies on prior tables in the query plan, identical instances of NdbInterpretedCode were generated for each request, at a significant cost in wasted CPU cycles. Now such pushed conditions are identified and the required NdbInterpretedCode object is generated only once, and reused for every request sent for this table without the need for generating new code each time.

This change also makes it possible for Scan Filter too large errors to be detected and set during query optimization, which corrects cases where the query plan shown was inaccurate because the indicated push of a condition later had to be undone during the execution phase. (Bug #29704575)

• Some instances of NdbScanFilter used in pushdown conditions were not generated properly due to FLOAT values being represented internally as having zero length. This led to more than the expected number of rows being returned from NDB, as shown by the value of Ndb_api_read_row_count. While the condition was re-evaluated by mysql when generation of scan filter failed, the end result was still correct in such cases, but any performance gain expected from pushing the condition was lost. (Bug #29699347)

• When creating a table, NDB did not always determine correctly whether it exceeded the maximum allowed record size. (Bug #29698277)
• **NDB** index statistics are calculated based on the topology of one fragment of an ordered index; the fragment chosen in any particular index is decided at index creation time, both when the index is originally created, and when a node or system restart has recreated the index locally. This calculation is based in part on the number of fragments in the index, which can change when a table is reorganized. This means that, the next time that the node is restarted, this node may choose a different fragment, so that no fragments, one fragment, or two fragments are used to generate index statistics, resulting in errors from **ANALYZE TABLE**.

This issue is solved by modifying the online table reorganization to recalculate the chosen fragment immediately, so that all nodes are aligned before and after any subsequent restart. (Bug #29534647)

• As part of initializing schema distribution, each data node must maintain a subscriber bitmap providing information about the API nodes that are currently subscribed to this data node. Previously, the size of the bitmap was hard-coded to **MAX_NODES** (256), which meant that large amounts of memory might be allocated but never used when the cluster had significantly fewer nodes than this value. Now the size of the bitmap is determined by checking the maximum API node ID used in the cluster configuration file. (Bug #29270539)

• The removal of the **mysql_upgrade** utility and its replacement by **mysqld --initialize** means that the upgrade procedure is executed much earlier than previously, possibly before **NDB** is fully ready to handle queries. This caused migration of the MySQL privilege tables from **NDB** to **InnoDB** to fail. (Bug #29205142)

• During a restart when the data nodes had started but not yet elected a president, the management server received a **node ID already in use** error, which resulted in excessive retries and logging. This is fixed by introducing a new error 1705 Not ready for connection allocation yet for this case.

During a restart when the data nodes had not yet completed node failure handling, a spurious **Failed to allocate nodeID** error was returned. This is fixed by adding a check to detect an incomplete node start and to return error 1703 Node failure handling not completed instead.

As part of this fix, the frequency of retries has been reduced for not ready to alloc nodeID errors, an error insert has been added to simulate a slow restart for testing purposes, and log messages have been reworded to indicate that the relevant node ID allocation errors are minor and only temporary. (Bug #27484514)

• **NDB** on Windows and MacOSX platforms did not always treat table names using mixed case consistently with **lower_case_table_names** = 2. (Bug #27307793)

• The process of selecting the transaction coordinator checked for “live” data nodes but not necessarily for those that were actually available. (Bug #27160203)

• The automatic metadata synchronization mechanism requires the binary logging thread to acquire the global schema lock before an object can be safely synchronized. When another thread had acquired this lock at the same time, the binary logging thread waited for up to **TransactionDeadlockDetectionTimeout** milliseconds and then returned failure if it was unsuccessful in acquiring the lock, which was unnecessary and which negatively impacted performance.

This has been fixed by ensuring that the binary logging thread acquires the global schema lock, or else returns with an error, immediately. As part of this work, a new **OperationOptions** flag **OO_NOWAIT** has also been implemented in the NDB API.

**Changes in MySQL NDB Cluster 8.0.17 (2019-07-22, Release Candidate)**

• Functionality Added or Changed
Bugs Fixed

Functionality Added or Changed

- Schema operation timeout detection has been moved from the schema distribution client to the schema distribution coordinator, which now checks ongoing schema operations for timeout at regular intervals, marks participants that have timed out, emits suitable warnings when a schema operation timeout occurs, and prints a list of any ongoing schema operations at regular intervals.

  As part of this work, a new option `--ndb-schema-dist-timeout` makes it possible to set the number of seconds for a given SQL node to wait until a schema operation is marked as having timed out. (Bug #29556148)

- Added the status variable `Ndb_trans_hint_count_session`, which shows the number of transactions started in the current session that used hints. Compare this with `Ndb_api_trans_start_count_session` to get the proportion of all NDB transactions in the current session that have been able to use hinting. (Bug #29127040)

- When the cluster is in single user mode, the output of the `ndb_mgm SHOW` command now indicates which API or SQL node has exclusive access while this mode is in effect. (Bug #16275500)

Bugs Fixed

- **Important Change:** Attempting to drop, using the `mysql` client, an NDB table that existed in the MySQL data dictionary but not in NDB caused `mysqld` to fail with an error. This situation could occur when an NDB table was dropped using the `ndb_drop_table` tool or in an NDB API application using `dropTable()`. Now in such cases, `mysqld` drops the table from the MySQL data dictionary without raising an error. (Bug #29125206)

- **Important Change:** The dependency of `ndb_restore` on the NDBT library, which is used for internal testing only, has been removed. This means that the program no longer prints `NDBT_ProgramExit: ...` when terminating. Applications that depend upon this behavior should be updated to reflect this change when upgrading to this release.

- **Packaging:** Added debug symbol packages to NDB distributions for `.deb`-based platforms which do not generate these automatically. (Bug #29040024)

- **NDB Disk Data:** If, for some reason, a disk data table exists in the NDB data dictionary but not in that of the MySQL server, the data dictionary is synchronized by installing the object. This can occur either during the schema synchronization phase when a MySQL server connects to an NDB Cluster, or during table discovery through a DML query or DDL statement.

  For disk data tables which used a tablespace for storage, the tablespace ID is stored as part of the data dictionary object, but this was not set during synchronization. (Bug #29597249)

- **NDB Disk Data:** Concurrent Disk Data table and tablespace DDL statements executed on the same SQL node caused a metadata lock deadlock. A DDL statement requires that an exclusive lock be taken on the object being modified and every such lock in turn requires that the global schema lock be acquired in NDB.

  To fix this issue, NDB now tracks when a global schema lock corresponding to an exclusive lock on a tablespace is taken. If a different global schema lock request fails while the first lock, NDB assumes that there is a deadlock. In this case, the deadlock is handled by having the new request release all locks it previously acquired, then retrying them at a later point. (Bug #29394407)

References: See also: Bug #29175268.
• **NDB Disk Data:** Following execution of `ALTER TABLESPACE`, SQL statements on an existing table using the affected tablespace failed with error 3508 `Dictionary object id (id) does not exist` where the object ID shown refers to the tablespace. Schema distribution of `ALTER TABLESPACE` involves dropping the old object from the data dictionary on a participating SQL node and creating a new one with a different dictionary object id, but the table object in the SQL node’s data dictionary still used the old tablespace ID which rendered it unusable on the participants.

To correct this problem, tables using the tablespace are now retrieved and stored prior to the creation of the new tablespace, and then updated the new object ID of the tablespace after it has been created in the data dictionary. (Bug #29389168)

• **NDB Cluster APIs:** The memcached sources included with the NDB distribution would not build with `-Werror=format-security`. Now warnings are no longer treated as errors when compiling these files. (Bug #29512411)

• **NDB Cluster APIs:** It was not possible to scan a table whose `SingleUserMode` property had been set to `SingleUserModeReadWrite` or `SingleUserModeReadOnly`. (Bug #29493714)

• **NDB Cluster APIs:** The MGM API `ndb_logevent_get_next2()` function did not behave correctly on Windows and 32-bit Linux platforms. (Bug #94917, Bug #29609070)

• The version of Python expected by `ndb_setup.py` was not specified clearly on some platforms. (Bug #29818645)

• Lack of `SharedGlobalMemory` was incorrectly reported as lack of undo buffer memory, even though the cluster used no disk data tables. (Bug #29806771)

References: This issue is a regression of: Bug #92125, Bug #28537319.

• Long `TCKEYREQ` signals did not always use the expected format when invoked from `TCINDEXREQ` processing. (Bug #29772731)

• It was possible for an internal `NDB_SCHEMA_OBJECT` to be released too early or not at all; in addition, it was possible to create such an object that reused an existing key. (Bug #29759063)

• `ndb_restore` sometimes used `exit()` rather than `exitHandler()` to terminate the program, which could lead to resources not being properly freed. (Bug #29744353)

• Improved error message printed when the maximum offset for a `FIXED` column is exceeded. (Bug #29714670)

• Communication between the schema distribution client and the schema distribution coordinator is done using `NDB_SCHEMA_OBJECT` as well as by writing rows to the `ndb_schema` table in NDB. This allowed for the possibility of a number of different race conditions between when the registration of the schema operation and when the coordinator was notified of it.

This fix addresses the following issues related to the situation just described:

• The coordinator failed to abort active schema operations when the binary logging thread was restarted.

• Schema operations already registered were not aborted properly.

• The distribution client failed to detect correctly when schema distribution was not ready.

• The distribution client, when killed, exited without marking the current schema operation as failed.

• An operation in `NDB_SHARE` could be accessed without the proper locks being in place.
In addition, usage of the `ndb_schema_share` global pointer was removed, and replaced with detecting whether the schema distribution is ready by checking whether an operation for `mysql.ndb_schema` has been created in `NDB_SHARE`. (Bug #29639381)

- With `DataMemory` set to 200 GB, `ndbmtd` failed to start. (Bug #29630367)
- When a backup fails due to `ABORT_BACKUP_ORD` being received while waiting for buffer space, the backup calls `closeScan()` and then sends a `SCAN_FRAGREQ` signal to the `DBLQH` block to close the scan. As part of receiving `SCAN_FRAGCONF` in response, `scanConf()` is called on the operation object for the file record which in turn calls `updateWritePtr()` on the file system buffer (`FsBuffer`). At this point the length sent by `updateWritePtr()` should be 0, but in this case was not, which meant that the buffer did not have enough space even though it did not, the problem being that the size is calculated as `scanStop - scanStart` and these values were held over since the previous `SCAN_FRAGCONF` was received, and were not reset due to being out of buffer space.

To avoid this problem, we now set `scanStart = scanStop` in `confirmBufferData()` (formerly `scanConfExtra()`) which is called as part of processing the `SCAN_FRAGCONF`, indirectly by `scanConf()` for the backup and first local checkpoint files, and directly for the LCP files which use only the operation record for the data buffer. (Bug #29601253)

- The setting for `MaxDMLOperationsPerTransaction` was not validated in a timely fashion, leading to data node failure rather than a management server error in the event that its value exceeded that of `MaxNoOfConcurrentOperations`. (Bug #29549572)

- Data nodes could fail due to an assert in the `DBTC` block under certain circumstances in resource-constrained environments. (Bug #29528188)

- When the `DBSPJ` block called the internal function `lookup_resume()` to schedule a previously enqueued operation, it used a correlation ID which could have been produced from its immediate ancestor in the execution order, and not its parent in the query tree as assumed. This could happen during execution of a `SELECT STRAIGHT_JOIN` query.

Now NDB checks whether the execution ancestor is different from the query tree parent, and if not, performs a lookup of the query tree parent, and the parent's correlation ID is enqueued to be executed later. (Bug #29501263)

- When a new master took over, sending a `MASTER_LCP_REQ` signal and executing `MASTER_LCPCONF` from participating nodes, it expected that they had not completed the current local checkpoint under the previous master, which need not be true. (Bug #29487340, Bug #29601546)

- When restoring `TINYBLOB` columns, `ndb_restore` now treats them as having the `BINARY` character set. (Bug #29486538)

- When selecting a sorted result set from a query that included a `LIMIT` clause on a single table, and where the sort was executed as `Using filesort` and the `ref` access method was used on an ordered index, it was possible for the result set to be missing one or more rows. (Bug #29474188)

- Restoration of epochs by `ndb_restore` failed due to temporary redo errors. Now `ndb_restore` retries epoch updates when such errors occur. (Bug #29466089)

- `ndb_restore` tried to extract an 8-character substring of a table name when checking to determine whether or not the table was a blob table, regardless of the length of the name. (Bug #29465794)

- When a pushed join was used in combination with the `eq_ref` access method it was possible to obtain an incorrect join result due to the 1 row cache mechanism implemented in NDB 8.0.16 as part of the work done in that version to extend NDB condition pushdown by allowing referring values from previous
tables. This issue is now fixed by turning off this caching mechanism and reading the row directly from the handler instead, when there is a pushed condition defined on the table. (Bug #29460314)

- Improved and made more efficient the conversion of rows by the ha_ndbcluster handler from the format used internally by NDB to that used by the MySQL server for columns that contain neither BLOB nor BIT values, which is the most common case. (Bug #29435461)

- A failed DROP TABLE could be attempted an infinite number of times in the event of a temporary error. Now in such cases, the number of retries is limited to 100. (Bug #29355155)

- ndb_restore --restore-epoch incorrectly reported the stop GCP as 1 less than the actual position. (Bug #29343655)

- A SavedEvent object in the CMVMI kernel block is written into a circular buffer. Such an object is split in two when wrapping at the end of the buffer; NDB looked beyond the end of the buffer instead of in the wrapped data at the buffer's beginning. (Bug #29336793)

- NDB did not compile with -DWITH_SYSTEM_LIBS=ON due to an incorrectly configured dependency on zlib. (Bug #29304517)

- Removed a memory leak found when running ndb_mgmd --config-file after compiling NDB with Clang 7. (Bug #29284643)

- Removed clang compiler warnings caused by usage of extra ; characters outside functions; these are incompatible with C++98. (Bug #29227925)

- Adding a column defined as TIMESTAMP DEFAULT CURRENT_TIMESTAMP to an NDB table is not supported with ALGORITHM=INPLACE. Attempting to do so now causes an error. (Bug #28128849)

- Added support which was missing in ndb_restore for conversions between the following sets of types:
  - BLOB and BINARY or VARBINARY columns
  - TEXT and BLOB columns
  - BLOB columns with unequal lengths
  - BINARY and VARBINARY columns with unequal lengths

(Bug #28074988)

- Neither the MAX_EXECUTION_TIME optimizer hint nor the max_execution_time system variable was respected for DDL statements or queries against INFORMATION_SCHEMA tables while an NDB global schema lock was in effect. (Bug #27538139)

- DDL operations were not always performed correctly on database objects including databases and tables, when multi-byte character sets were used for the names of either or both of these. (Bug #27150334)

- ndb_import did not always free up all resources used before exiting. (Bug #27130143)

- NDBCLUSTER subscription log printouts provided only 2 words of the bitmap (in most cases containing 8 words), which made it difficult to diagnose schema distribution issues. (Bug #22180480)
• For certain tables with very large rows and a very large primary key, `START BACKUP SNAPSHOTEND` while performing inserts into one of these tables or `START BACKUP SNAPSHOTSTART` with concurrent deletes could lead to data node errors.

As part of this fix, `ndb_print_backup_file` can now read backup files created in very old versions of NDB Cluster (6.3 and earlier); in addition, this utility can now also read undo log files. (Bug #94654, Bug #29485977)

• When one of multiple SQL nodes which were connected to the cluster was down and then rejoined the cluster, or a new SQL node joined the cluster, this node did not use the data dictionary correctly, and thus did not always add, alter, or drop databases properly when synchronizing with the existing SQL nodes.

Now, during schema distribution at startup, the SQL node compares all databases on the data nodes with those in its own data dictionary. If any database on the data nodes is found to be missing from the SQL node’s data dictionary, the SQL Node installs it locally using `CREATE DATABASE`; the database is created using the default MySQL Server database properties currently in effect on this SQL node.

Changes in MySQL NDB Cluster 8.0.16 (2019-04-25, Development Milestone)

• Deprecation and Removal Notes

• SQL Syntax Notes

• Functionality Added or Changed

• Bugs Fixed

Deprecation and Removal Notes

• **Incompatible Change**: Distribution of privileges amongst MySQL servers connected to NDB Cluster, as implemented in NDB 7.6 and earlier, does not function in NDB 8.0, and most code supporting these has now been removed. When a `mysqld` detects such tables in NDB, it creates shadow tables local to itself using the InnoDB storage engine; these shadow tables are created on each MySQL server connected to an NDB cluster. Privilege tables using the NDB storage engine are not employed for access control; once all connected MySQL servers are upgraded, the privilege tables in NDB can be removed safely using `ndb_drop_table`.

For compatibility reasons, `ndb_restore --restore-privilege-tables` can still be used to restore distributed privilege tables present in a backup taken from a previous release of NDB Cluster to a cluster running NDB 8.0. These tables are handled as described in the preceding paragraph.

For additional information regarding upgrades from previous NDB Cluster release series to NDB 8.0, see Upgrading and Downgrading NDB Cluster.

SQL Syntax Notes

• **Incompatible Change**: For consistency with InnoDB, the NDB storage engine now uses a generated constraint name if the `CONSTRAINT symbol` clause is not specified, or the `CONSTRAINT` keyword is specified without a `symbol`. In previous NDB releases, NDB used the `FOREIGN KEY index_name` value.

This change described above may introduce incompatibilities for applications that depend on the previous foreign key constraint naming behavior. (Bug #29173134)
Functionality Added or Changed

- **Packaging:** A Docker image for this release can be obtained from https://hub.docker.com/r/mysql/mysql-cluster/. (Bug #96084, Bug #30010921)

- Allocation of resources in the transaction coordinator (see The DBTC Block) is now performed using dynamic memory pools. This means that resource allocation determined by data node configuration parameters such as those discussed in Transaction parameters and Transaction temporary storage is now limited so as not to exceed the total resources available to the transaction coordinator.

As part of this work, several new data node parameters controlling transactional resources in DBTC, listed here, have also been added. For more information about these new parameters, see Transaction resource allocation parameters. (Bug #29164271, Bug #29194843)

References: See also: Bug #29131828.

- **NDB** backups can now be performed in a parallel fashion on individual data nodes using multiple local data managers (LDMs). (Previously, backups were done in parallel across data nodes, but were always serial within data node processes.) No special syntax is required for the `START BACKUP` command in the `ndb_mgm` client to enable this feature, but all data nodes must be using multiple LDMs. This means that data nodes must be running `ndbmtd` and they must be configured to use multiple LDMs prior to taking the backup (see Multi-Threading Configuration Parameters (`ndbmtd`)).

  `ndb_restore` also now detects such a backup and automatically attempts to restore it in parallel. It is also possible to restore backups taken in parallel to a previous version of NDB Cluster by slightly modifying the usual restore procedure.

For more information about taking and restoring NDB Cluster backups that were created using parallelism on the data nodes, see Taking an NDB Backup with Parallel Data Nodes, and Restoring from a backup taken in parallel. (Bug #28563639, Bug #28993400)

- The `compile-cluster` script included in the `NDB` source distribution no longer supports in-source builds.

- Building with `CMake3` is now supported by the `compile-cluster` script included in the `NDB` source distribution.

- As part of its automatic synchronization mechanism, NDB now implements a metadata change monitor thread for detecting changes made to metadata for data objects such as tables, tablespaces, and log file groups with the MySQL data dictionary. This thread runs in the background, checking every 60 seconds for inconsistencies between the NDB dictionary and the MySQL data dictionary.

  The monitor polling interval can be adjusted by setting the value of the `ndb_metadata_check_interval` system variable, and can be disabled altogether by setting `ndb_metadata_check` to OFF. The number of times that inconsistencies have been detected since `mysqld` was last started is shown as the status variable, `Ndb_metadata_detected_count`.

- Condition pushdown is no longer limited to predicate terms referring to column values from the same table to which the condition was being pushed; column values from tables earlier in the query plan can now also be referred to from pushed conditions. This lets the data nodes filter out more rows (in parallel), leaving less work to be performed by a single `mysqld` process, which is expected to provide significant improvements in query performance.

For more information, see Engine Condition Pushdown Optimization.
**Bugs Fixed**

- **Important Change; NDB Disk Data:** `mysqldump` terminated unexpectedly when attempting to dump NDB disk data tables. The underlying reason for this was that `mysqldump` expected to find information relating to undo log buffers in the `EXTRA` column of the `INFORMATION_SCHEMA.FILES` table but this information had been removed in NDB 8.0.13. This information is now restored to the `EXTRA` column. (Bug #28800252)

- **Important Change:** When restoring to a cluster using data node IDs different from those in the original cluster, `ndb_restore` tried to open files corresponding to node ID 0. To keep this from happening, the `--nodeid` and `--backupid` options—neither of which has a default value—are both now explicitly required when invoking `ndb_restore`. (Bug #28813708)

- **Important Change:** Starting with this release, the default value of the `ndb_log_bin` system variable is now `FALSE`. (Bug #27135706)

- **NDB Disk Data:** When a log file group had more than 18 undo logs, it was not possible to restart the cluster. (Bug #251155785)

  References: See also: Bug #28922609.

- **NDB Disk Data:** Concurrent `CREATE TABLE` statements using tablespaces caused deadlocks between metadata locks. This occurred when `Ndb_metadata_change_monitor` acquired exclusive metadata locks on tablespaces and logfile groups after detecting metadata changes, due to the fact that each exclusive metadata lock in turn acquired a global schema lock. This fix attempts to solve that issue by downgrading the locks taken by `Ndb_metadata_change_monitor` to `MDL_SHARED_READ`. (Bug #29175268)

  References: See also: Bug #29394407.

- **NDB Disk Data:** The error message returned when validation of `MaxNoOfOpenFiles` in relation to `InitialNoOfOpenFiles` failed has been improved to make the nature of the problem clearer to users. (Bug #28943749)

- **NDB Disk Data:** Schema distribution of `ALTER TABLESPACE` and `ALTER LOGFILE GROUP` statements failed on a participant MySQL server if the referenced tablespace or log file group did not exist in its data dictionary. Now in such cases, the effects of the statement are distributed successfully regardless of any initial mismatch between MySQL servers. (Bug #28866336)

- **NDB Disk Data:** Repeated execution of `ALTER TABLESPACE ... ADD DATAFILE` against the same tablespace caused data nodes to hang and left them, after being killed manually, unable to restart. (Bug #22605467)

- **NDB Cluster APIs:** NDB now identifies short-lived transactions not needing the reduction of lock contention provided by `NdbBlob::close()` and no longer invokes this method in cases (such as when autocommit is enabled) in which unlocking merely causes extra work and round trips to be performed prior to committing or aborting the transaction. (Bug #29305592)

  References: See also: Bug #49190, Bug #11757181.

- **NDB Cluster APIs:** When the most recently failed operation was released, the pointer to it held by `NdbTransaction` became invalid and when accessed led to failure of the NDB API application. (Bug #29275244)

- **NDB Cluster APIs:** When the NDB kernel's `SUMA` block sends a `TE_ALTER` event, it does not keep track of when all fragments of the event are sent. When NDB receives the event, it buffers the fragments, and processes the event when all fragments have arrived. An issue could possibly arise for very large table
MySQL NDB Cluster 8.0 Release Notes

definitions, when the time between transmission and reception could span multiple epochs; during this
time, SUMA could send a SUB_GCP_COMPLETE_REP signal to indicate that it has sent all data for an
epoch, even though in this case that is not entirely true since there may be fragments of a TE_ALTER
event still waiting on the data node to be sent. Reception of the SUB_GCP_COMPLETE_REP leads to
closing the buffers for that epoch. Thus, when TE_ALTER finally arrives, NDB assumes that it is a
duplicate from an earlier epoch, and silently discards it.

We fix the problem by making sure that the SUMA kernel block never sends a SUB_GCP_COMPLETE_REP
for any epoch in which there are unsent fragments for a SUB_TABLE_DATA signal.

This issue could have an impact on NDB API applications making use of TE_ALTER events. (SQL nodes
do not make any use of TE_ALTER events and so they and applications using them were not affected.)
(Bug #28836474)

• When a pushed join executing in the DBSPJ block had to store correlation IDs during query execution,
memory for these was allocated for the lifetime of the entire query execution, even though these specific
correlation IDs are required only when producing the most recent batch in the result set. Subsequent
batches require additional correlation IDs to be stored and allocated; thus, if the query took sufficiently
long to complete, this led to exhaustion of query memory (error 20008). Now in such cases, memory
is allocated only for the lifetime of the current result batch, and is freed and made available for re-use
following completion of the batch. (Bug #29336777)

References: See also: Bug #26995027.

• When comparing or hashing a fixed-length string that used a NO_PAD collation, any trailing padding
characters (typically spaces) were sent to the hashing and comparison functions such that they became
significant, even though they were not supposed to be. Now any such trailing spaces are trimmed from a
fixed-length string whenever a NO_PAD collation is specified.

Note

Since NO_PAD collations were introduced as part of UCA-9.0 collations in MySQL
8.0, there should be no impact relating to this fix on upgrades to NDB 8.0 from
previous GA releases of NDB Cluster.

(Bug #29322313)

• When a NOT IN or NOT BETWEEN predicate was evaluated as a pushed condition, NULL values were
not eliminated by the condition as specified in the SQL standard. (Bug #29232744)

References: See also: Bug #28672214.

• Internally, NDB treats NULL as less than any other value, and predicates of the form column < value
or column <= value are checked for possible nulls. Predicates of the form value > column or
value >= column were not checked, which could lead to errors. Now in such cases, these predicates
are rewritten so that the column comes first, so that they are also checked for the presence of NULL.
(Bug #29231709)

References: See also: Bug #92407, Bug #28643463.

• After folding of constants was implemented in the MySQL Optimizer, a condition containing a DATE or
DATETIME literal could no longer be pushed down by NDB. (Bug #29161281)

• When a join condition made a comparison between a column of a temporal data type such as
DATE or DATETIME and a constant of the same type, the predicate was pushed if the condition was
expressed in the form column operator constant, but not when in inverted order (as constant
inverse_operator column). (Bug #29058732)
• When processing a pushed condition, NDB did not detect errors or warnings thrown when a literal value being compared was outside the range of the data type it was being compared with, and thus truncated. This could lead to excess or missing rows in the result. (Bug #29054626)

• If an EQ_REF or REF key in the child of a pushed join referred to any columns of a table not a member of the pushed join, this table was not an NDB table (because its format was of nonnative endianness), and the data type of the column being joined on was stored in an endian-sensitive format, then the key generated was generated, likely resulting in the return of an (invalid) empty join result.

Since only big endian platforms may store tables in nonnative (little endian) formats, this issue was expected only on such platforms, most notably SPARC, and not on x86 platforms. (Bug #29010641)

• API and data nodes running NDB 7.6 and later could not use an existing parsed configuration from an earlier release series due to being overly strict with regard to having values defined for configuration parameters new to the later release, which placed a restriction on possible upgrade paths. Now NDB 7.6 and later are less strict about having all new parameters specified explicitly in the configuration which they are served, and use hard-coded default values in such cases. (Bug #289993400)

• NDB 7.6 SQL nodes hung when trying to connect to an NDB 8.0 cluster. (Bug #28985685)

• The schema distribution data maintained in the NDB binary logging thread keeping track of the number of subscribers to the NDB schema table always allocated some memory structures for 256 data nodes regardless of the actual number of nodes. Now NDB allocates only as many of these structures as are actually needed. (Bug #28949523)

• Added DUMP 406 (NdbfsDumpRequests) to provide NDB file system information to global checkpoint and local checkpoint stall reports in the node logs. (Bug #28922609)

• When a joined table was eliminated early as not pushable, it could not be referred to in any subsequent join conditions from other tables without eliminating those conditions from consideration even if those conditions were otherwise pushable. (Bug #28898811)

• When starting or restarting an SQL node and connecting to a cluster where NDB was already started, NDB reported Error 4009 Cluster Failure because it could not acquire a global schema lock. This was because the MySQL Server as part of initialization acquires exclusive metadata locks in order to modify internal data structures, and the ndbcluster plugin acquires the global schema lock. If the connection to NDB was not yet properly set up during mysqld initialization, mysqld received a warning from ndbcluster when the latter failed to acquire global schema lock, and printed it to the log file, causing an unexpected error in the log. This is fixed by not pushing any warnings to background threads when failure to acquire a global schema lock occurs and pushing the NDB error as a warning instead. (Bug #28898544)

• A race condition between the DBACC and DBLQH kernel blocks occurred when different operations in a transaction on the same row were concurrently being prepared and aborted. This could result in DBTUP attempting to prepare an operation when a preceding operation had been aborted, which was unexpected and could thus lead to undefined behavior including potential data node failures. To solve this issue, DBACC and DBLQH now check that all dependencies are still valid before attempting to prepare an operation.

Note
This fix also supersedes a previous one made for a related issue which was originally reported as Bug #28500861.

(Bug #28893633)
MySQL NDB Cluster 8.0 Release Notes

• Where a data node was restarted after a configuration change whose result was a decrease in the sum of `MaxNoOfTables`, `MaxNoOfOrderedIndexes`, and `MaxNoOfUniqueHashIndexes`, it sometimes failed with a misleading error message which suggested both a temporary error and a bug, neither of which was the case.

The failure itself is expected, being due to the fact that there is at least one table object with an ID greater than the (new) sum of the parameters just mentioned, and that this table cannot be restored since the maximum value for the ID allowed is limited by that sum. The error message has been changed to reflect this, and now indicates that this is a permanent error due to a problem configuration. (Bug #28884880)

• The `ndbinfo.cpustat` table reported inaccurate information regarding send threads. (Bug #28884157)

• Execution of an LCPCOMPLETE_REP signal from the master while the LCP status was IDLE led to an assertion. (Bug #28871889)

• NDB now provides on-the-fly `.frm` file translation during discovery of tables created in versions of the software that did not support the MySQL Data Dictionary. Previously, such translation of tables that had old-style metadata was supported only during schema synchronization during MySQL server startup, but not subsequently, which led to errors when NDB tables having old-style metadata, created by `ndb_restore` and other such tools after `mysqld` had been started, were accessed using `SHOW CREATE TABLE` or `SELECT`; these tables were usable only after restarting `mysqld`. With this fix, the restart is no longer required. (Bug #28841009)

• An in-place upgrade to an NDB 8.0 release from an earlier release did not remove `.ndb` files, even though these are no longer used in NDB 8.0. (Bug #28832816)

• Removed `storage/ndb/demos` and the demonstration scripts and support files it contained from the source tree. These were obsolete and unmaintained, and did not function with any current version of NDB Cluster.

Also removed `storage/ndb/include/newtonapi`, which included files relating to an obsolete and unmaintained API not supported in any release of NDB Cluster, as well as references elsewhere to these files. (Bug #28808766)

• There was no version compatibility table for NDB 8.x; this meant that API nodes running NDB 8.0.13 or 7.6.x could not connect to data nodes running NDB 8.0.14. This issue manifested itself for NDB API users as a failure in `wait_until_ready()`. (Bug #28776365)

References: See also: Bug #18886034, Bug #18874849.

• Issuing a `STOP` command in the `ndb_mgm` client caused `ndbmtd` processes which had recently been added to the cluster to hang in Phase 4 during shutdown. (Bug #28772867)

• A fix for a previous issue disabled the usage of pushed conditions for lookup type (`eq_ref`) operations in pushed joins. It was thought at the time that not pushing a lookup condition would not have any measurable impact on performance, since only a single row could be eliminated if the condition failed. The solution implemented at that time did not take into account the possibility that, in a pushed join, a lookup operation could be a parent operation for other lookups, and even scan operations, which meant that eliminating a single row could actually result in an entire branch being eliminated in error. (Bug #28728603)

References: This issue is a regression of: Bug #27397802.

• When a local checkpoint (LCP) was complete on all data nodes except one, and this node failed, NDB did not continue with the steps required to finish the LCP. This led to the following issues:
No new LCPs could be started.

Redo and Undo logs were not trimmed and so grew excessively large, causing an increase in times for recovery from disk. This led to write service failure, which eventually led to cluster shutdown when the head of the redo log met the tail. This placed a limit on cluster uptime.

Node restarts were no longer possible, due to the fact that a data node restart requires that the node's state be made durable on disk before it can provide redundancy when joining the cluster. For a cluster with two data nodes and two replicas, this meant that a restart of the entire cluster (system restart) was required to fix the issue (this was not necessary for a cluster with two replicas and four or more data nodes). (Bug #28728485, Bug #28698831)

References: See also: Bug #11757421.

• The pushability of a condition to NDB was limited in that all predicates joined by a logical AND within a given condition had to be pushable to NDB in order for the entire condition to be pushed. In some cases this severely restricted the pushability of conditions. This fix breaks up the condition into its components, and evaluates the pushability of each predicate; if some of the predicates cannot be pushed, they are returned as a remainder condition which can be evaluated by the MySQL server. (Bug #28728007)

• Running ANALYZE TABLE on an NDB table with an index having longer than the supported maximum length caused data nodes to fail. (Bug #28714864)

• It was possible in certain cases for nodes to hang during an initial restart. (Bug #28698831)

References: See also: Bug #27622643.

• When a condition was pushed to a storage engine, it was re-evaluated by the server, in spite of the fact that only rows matching the pushed condition should ever be returned to the server in such cases. (Bug #28672214)

• In some cases, one and sometimes more data nodes underwent an unplanned shutdown while running ndb_restore. This occurred most often, but was not always restricted to, when restoring to a cluster having a different number of data nodes from the cluster on which the original backup had been taken.

The root cause of this issue was exhaustion of the pool of SafeCounter objects, used by the DBDICT kernel block as part of executing schema transactions, and taken from a per-block-instance pool shared with protocols used for NDB event setup and subscription processing. The concurrency of event setup and subscription processing is such that the SafeCounter pool can be exhausted; event and subscription processing can handle pool exhaustion, but schema transaction processing could not, which could result in the node shutdown experienced during restoration.

This problem is solved by giving DBDICT schema transactions an isolated pool of reserved SafeCounters which cannot be exhausted by concurrent NDB event activity. (Bug #28595915)

• When a backup aborted due to buffer exhaustion, synchronization of the signal queues prior to the expected drop of triggers for insert, update, and delete operations resulted in abort signals being processed before the STOP_BACKUP phase could continue. The abort changed the backup status to ABORT_BACKUP_ORD, which led to an unplanned shutdown of the data node since resuming STOP_BACKUP requires that the state be STOP_BACKUP_REQ. Now the backup status is not set to STOP_BACKUP_REQ (requesting the backup to continue) until after signal queue synchronization is complete. (Bug #28563639)

• The output of ndb_config --configinfo --xml --query-all now shows that configuration changes for the threadConfig and MaxNoOfExecutionThreads data node parameters require system initial restarts (restart="system" initial="true"). (Bug #28494286)
• After a commit failed due to an error, `mysqld` shut down unexpectedly while trying to get the name of the table involved. This was due to an issue in the internal function `ndbcluster_print_error()`. (Bug #28435082)

• API nodes should observe that a node is moving through `SL_STOPPING` phases (graceful stop) and stop using the node for new transactions, which minimizes potential disruption in the later phases of the node shutdown process. API nodes were only informed of node state changes via periodic heartbeat signals, and so might not be able to avoid interacting with the node shutting down. This generated unnecessary failures when the heartbeat interval was long. Now when a data node is being gracefully stopped, all API nodes are notified directly, allowing them to experience minimal disruption. (Bug #28380808)

• `ndb_config --diff-default` failed when trying to read a parameter whose default value was the empty string (`""`). (Bug #27972537)

• `ndb_restore` did not restore autoincrement values correctly when one or more staging tables were in use. As part of this fix, we also in such cases block applying of the `SYSTAB_0` backup log, whose content continued to be applied directly based on the table ID, which could overwrite the autoincrement values stored in `SYSTAB_0` for unrelated tables. (Bug #27917769, Bug #27831990)

References: See also: Bug #27832033.

• `ndb_restore` employed a mechanism for restoring autoincrement values which was not atomic, and thus could yield incorrect autoincrement values being restored when multiple instances of `ndb_restore` were used in parallel. (Bug #27832033)

References: See also: Bug #27917769, Bug #27831990.

• Executing `SELECT * FROM INFORMATION_SCHEMA.TABLES` caused SQL nodes to restart in some cases. (Bug #27613173)

• When tables with `BLOB` columns were dropped and then re-created with a different number of `BLOB` columns the event definitions for monitoring table changes could become inconsistent in certain error situations involving communication errors when the expected cleanup of the corresponding events was not performed. In particular, when the new versions of the tables had more `BLOB` columns than the original tables, some events could be missing. (Bug #27072756)

• When query memory was exhausted in the `DBSPJ` kernel block while storing correlation IDs for deferred operations, the query was aborted with error status 20000 `Query aborted due to out of query memory`. (Bug #26995027)

References: See also: Bug #86537.

• When running a cluster with 4 or more data nodes under very high loads, data nodes could sometimes fail with Error 899 `Rowid already allocated`. (Bug #25960230)

• `mysqld` shut down unexpectedly when a purge of the binary log was requested before the server had completely started, and it was thus not yet ready to delete rows from the `ndb_binlog_index` table. Now when this occurs, requests for any needed purges of the `ndb_binlog_index` table are saved in a queue and held for execution when the server has completely started. (Bug #25817834)

• `MaxBufferedEpochs` is used on data nodes to avoid excessive buffering of row changes due to lagging `NDB` event API subscribers; when epoch acknowledgements from one or more subscribers lag by this number of epochs, an asynchronous disconnection is triggered, allowing the data node to release the buffer space used for subscriptions. Since this disconnection is asynchronous, it may be the case that it has not completed before additional new epochs are completed on the data node, resulting in new epochs not being able to seize GCP completion records, generating warnings such as those shown here:
MySQL NDB Cluster 8.0 Release Notes

And leading to the following warning:

```
Disconnecting node %u because it has exceeded MaxBufferedEpochs
(100 > 100), epoch ....
```

This fix performs the following modifications:

- Modifies the size of the GCP completion record pool to ensure that there is always some extra headroom to account for the asynchronous nature of the disconnect processing previously described, thus avoiding `c_gcp_list` seize failures.
- Modifies the wording of the `MaxBufferedEpochs` warning to avoid the contradictory phrase “100 > 100”.

(Bug #20344149)

- Asynchronous disconnection of `mysqlid` from the cluster caused any subsequent attempt to start an NDB API transaction to fail. If this occurred during a bulk delete operation, the SQL layer called `HA::end_bulk_delete()`, whose implementation by `ha_ndbcluster` assumed that a transaction had been started, and could fail if this was not the case. This problem is fixed by checking that the transaction pointer used by this method is set before referencing it. (Bug #20116393)
- Removed warnings raised when compiling `NDB` with Clang 6. (Bug #93634, Bug #29112560)
- When executing the redo log in debug mode it was possible for a data node to fail when deallocating a row. (Bug #93273, Bug #28955797)
- An `NDB` table having both a foreign key on another `NDB` table using `ON DELETE CASCADE` and one or more `TEXT` or `BLOB` columns leaked memory.

As part of this fix, `ON DELETE CASCADE` is no longer supported for foreign keys on `NDB` tables when the child table contains a column that uses any of the `BLOB` or `TEXT` types. (Bug #89511, Bug #27484882)

Changes in MySQL NDB Cluster 8.0.14 (2019-01-21, Development Milestone)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Performance**: This release introduces a number of significant improvements in the performance of scans; these are listed here:
  - Row checksums help detect hardware issues, but do so at the expense of performance. `NDB` now offers the possibility of disabling these by setting the new `ndb_row_checksum` server system variable to 0; doing this means that row checksums are not used for new or altered tables. This can have a significant impact (5 to 10 percent, in some cases) on performance for all types of queries. This variable is set to 1 by default, to provide compatibility with the previous behavior.
  - A query consisting of a scan can execute for a longer time in the LDM threads when the queue is not busy.
• Previously, columns were read before checking a pushed condition; now checking of a pushed condition is done before reading any columns.

• Performance of pushed joins should see significant improvement when using range scans as part of join execution.

• **NDB Disk Data:** NDB now implements schema distribution of disk data objects including tablespaces and log file groups by SQL nodes when they connect to a cluster, just as it does for NDB databases and in-memory tables. This eliminates a possible mismatch between the MySQL data dictionary and the NDB dictionary following a native backup and restore that could arise when disk data tablespaces and undo log file groups were restored to the NDB dictionary, but not to the MySQL Server's data dictionary.

• **NDB Disk Data:** NDB now makes use of the MySQL data dictionary to ensure correct distribution of tablespaces and log file groups across all cluster SQL nodes when connecting to the cluster.

• The extra metadata property for NDB tables is now used to store information from the MySQL data dictionary. Because this information is significantly larger than the binary representation previously stored here (a .frm file, no longer used), the hard-coded size limit for this extra metadata has been increased.

This change can have an impact on downgrades: Trying to read NDB tables created in NDB 8.0.14 and later may cause data nodes running NDB 8.0.13 or earlier to fail on startup with NDB error code 2355 Failure to restore schema: Permanent error, external action needed: Resource configuration error. This can happen if the table's metadata exceeds 6K in size, which was the old limit. Tables created in NDB 8.0.13 and earlier can be read by later versions without any issues.

For more information, see Changes in NDB table extra metadata, and See also MySQL Data Dictionary. (Bug #27230681)

### Bugs Fixed

• **Packaging:** Expected NDB header files were in the devel RPM package instead of libndbclient-devel. (Bug #84580, Bug #26448330)

• The version_comment system variable was not correctly configured in mysqld binaries and returned a generic pattern instead of the proper value. This affected all NDB Cluster binary releases with the exception of .deb packages. (Bug #29054235)

• Trying to build from source using -DWITH_NDBCLUSTER and -Werror failed with GCC 8. (Bug #28707282)

• When copying deleted rows from a live node to a node just starting, it is possible for one or more of these rows to have a global checkpoint index equal to zero. If this happened at the same time that a full local checkpoint was started due to the undo log getting full, the LCP_SKIP bit was set for a row having GCI = 0, leading to an unplanned shutdown of the data node. (Bug #28372628)

• ndbmtd sometimes experienced a hang when exiting due to log thread shutdown. (Bug #28027150)

• NDB has an upper limit of 128 characters for a fully qualified table name. Due to the fact that mysqld names NDB tables using the format database_name/catalog_name/table_name, where catalog_name is always def, it is possible for statements such as CREATE TABLE to fail in spite of the fact that neither the table name nor the database name exceeds the 63-character limit imposed by NDB. The error raised in such cases was misleading and has been replaced. (Bug #27769521)

References: See also: Bug #27769801.
MySQL NDB Cluster 8.0 Release Notes

- When the SUMA kernel block receives a SUB_STOP_REQ signal, it executes the signal then replies with SUB_STOP_CONF. (After this response is relayed back to the API, the API is open to send more SUB_STOP_REQ signals.) After sending the SUB_STOP_CONF, SUMA drops the subscription if no subscribers are present, which involves sending multiple DROP_TRIG_IMPL_REQ messages to DBTUP. LocalProxy can handle up to 21 of these requests in parallel; any more than this are queued in the Short Time Queue. When execution of a DROP_TRIG_IMPL_REQ was delayed, there was a chance for the queue to become overloaded, leading to a data node shutdown with Error in short time queue. This issue is fixed by delaying the execution of the SUB_STOP_REQ signal if DBTUP is already handling DROP_TRIG_IMPL_REQ signals at full capacity, rather than queueing up the DROP_TRIG_IMPL_REQ signals. (Bug #26574003)

- ndb_restore returned -1 instead of the expected exit code in the event of an index rebuild failure. (Bug #25112726)

- When starting, a data node copies metadata, while a local checkpoint updates metadata. To avoid any conflict, any ongoing LCP activity is paused while metadata is being copied. An issue arose when a local checkpoint was paused on a given node, and another node that was also restarting checked for a complete LCP on this node; the check actually caused the LCP to be completed before copying of metadata was complete and so ended the pause prematurely. Now in such cases, the LCP completion check waits to complete a paused LCP until copying of metadata is finished and the pause ends as expected, within the LCP in which it began. (Bug #24827685)

- ndbout and ndberr became invalid after exiting from mgmd_run(), and redirecting to them before the next call to mgmd_run() caused a segmentation fault, during an ndb_mgmd service restart. This fix ensures that ndbout and ndberr remain valid at all times. (Bug #17732772, Bug #28536919)

- NdbScanFilter did not always handle NULL according to the SQL standard, which could result in sending non-qualifying rows to be filtered (otherwise not necessary) by the MySQL server. (Bug #92407, Bug #28643463)

References: See also: Bug #93977, Bug #29231709.

- The internal function ndb_my_error() was used in ndbcluster_get_tablespace_statistics() and prepare_inplace_alter_table() to report errors when the function failed to interact with NDB. The function was expected to push the NDB error as warning on the stack and then set an error by translating the NDB error to a MySQL error and then finally call my_error() with the translated error. When calling my_error(), the function extracts a format string that may contain placeholders and use the format string in a function similar to sprintf(), which in this case could read arbitrary memory leading to a segmentation fault, due to the fact that my_error() was called without any arguments. The fix is always to push the NDB error as a warning and then set an error with a provided message. A new helper function has been added to Thd_ndb to be used in place of ndb_my_error(). (Bug #92244, Bug #28575934)

- Running out of undo log buffer memory was reported using error 921 Out of transaction memory ... (increase SharedGlobalMemory).

This problem is fixed by introducing a new error code 923 Out of undo buffer memory (increase UNDO_BUFFER_SIZE). (Bug #92125, Bug #28537319)

- When moving an OperationRec from the serial to the parallel queue, Dbacc::startNext() failed to update the Operationrec::OP_ACC_LOCK_MODE flag which is required to reflect the accumulated OP_LOCK_MODE of all previous operations in the parallel queue. This inconsistency in the ACC lock queues caused the scan lock takeover mechanism to fail, as it incorrectly concluded that a lock to take over was not held. The same failure caused an assert when aborting an operation that was a member of such an inconsistent parallel lock queue. (Bug #92100, Bug #28530928)
MySQL NDB Cluster 8.0 Release Notes

• `ndb_restore` did not free all memory used after being called to restore a table that already existed. (Bug #92085, Bug #28525898)

• A data node failed during startup due to the arrival of a `SCAN_FRAGREQ` signal during the restore phase. This signal originated from a scan begun before the node had previously failed and which should have been aborted due to the involvement of the failed node in it. (Bug #92059, Bug #28518448)

• `DBTUP` sent the error `Tuple corruption detected` when a read operation attempted to read the value of a tuple inserted within the same transaction. (Bug #92009, Bug #28500861)

References: See also: Bug #28893633.

• False constraint violation errors could occur when executing updates on self-referential foreign keys. (Bug #91965, Bug #28486390)

References: See also: Bug #90644, Bug #27930382.

• An `NDB` internal trigger definition could be dropped while pending instances of the trigger remained to be executed, by attempting to look up the definition for a trigger which had already been released. This caused unpredictable and thus unsafe behavior possibly leading to data node failure. The root cause of the issue lay in an invalid assumption in the code relating to determining whether a given trigger had been released; the issue is fixed by ensuring that the behavior of `NDB`, when a trigger definition is determined to have been released, is consistent, and that it meets expectations. (Bug #91894, Bug #28451957)

• In some cases, a workload that included a high number of concurrent inserts caused data node failures when using debug builds. (Bug #91764, Bug #28387450, Bug #29055038)

• During an initial node restart with disk data tables present and `TwoPassInitialNodeRestartCopy` enabled, `DBTUP` used an unsafe scan in disk order. Such scans are no longer employed in this case. (Bug #91724, Bug #28378227)

• Checking for old LCP files tested the table version, but this was not always dependable. Now, instead of relying on the table version, the check regards as invalid any LCP file having a `maxGCI` smaller than its `createGci`. (Bug #91637, Bug #28346565)

• In certain cases, a cascade update trigger was fired repeatedly on the same record, which eventually consumed all available concurrent operations, leading to Error 233 `Out of operation records in transaction coordinator (increase MaxNoOfConcurrentOperations)`. If `MaxNoOfConcurrentOperations` was set to a value sufficiently high to avoid this, the issue manifested as data nodes consuming very large amounts of CPU, very likely eventually leading to a timeout. (Bug #91472, Bug #28262259)

Changes in MySQL NDB Cluster 8.0.13 (2018-10-23, Development Milestone)

• Functionality Added or Changed

• Bugs Fixed

Functionality Added or Changed

• **Important Change; NDB Disk Data**: The following changes are made in the display of information about Disk Data files in the `INFORMATION_SCHEMA.FILES` table:

  • Tablespaces and log file groups are no longer represented in the `FILES` table. (These constructs are not actually files.)
Each data file is now represented by a single row in the `FILES` table. Each undo log file is also now represented in this table by one row only. (Previously, a row was displayed for each copy of each of these files on each data node.)

For rows corresponding to data files or undo log files, node ID and undo log buffer information is no longer displayed in the `EXTRA` column of the `FILES` table.

**Important**
This change is reverted in NDB 8.0.15. (Bug #92796, Bug #28800252)

- **Important Change; NDB Client Programs:** Removed the deprecated `--ndb` option for `perror`. Use `ndb_perror` to obtain error message information from NDB error codes instead. (Bug #81705, Bug #23523957)

  References: See also: Bug #81704, Bug #23523926.

- **Important Change:** Beginning with this release, MySQL NDB Cluster is being developed in parallel with the standard MySQL 8.0 server under a new unified release model with the following features:
  - NDB 8.0 is developed in, built from, and released with the MySQL 8.0 source code tree.
  - The numbering scheme for NDB Cluster 8.0 releases follows the scheme for MySQL 8.0, starting with the current MySQL release (8.0.13).
  - Building the source with NDB support appends `--cluster` to the version string returned by `mysql -V`, as shown here:

```shell
mysql -V
mysql Ver 8.0.13-cluster for Linux on x86_64 (Source distribution)
```

NDB binaries continue to display both the MySQL Server version and the NDB engine version, like this:

```shell
ndb_mgm -V
MySQL distrib mysql-8.0.13 ndb-8.0.13-dmr, for Linux (x86_64)
```

In MySQL Cluster NDB 8.0, these two version numbers are always the same.

To build the MySQL 8.0.13 (or later) source with NDB Cluster support, use the CMake option `DWITH_NDBCLUSTER`.

- **NDB Cluster APIs:** Added the `Table` methods `getExtraMetadata()` and `setExtraMetadata()`.

- **INFORMATION_SCHEMA** tables now are populated with tablespace statistics for MySQL Cluster tables. (Bug #27167728)

- It is now possible to specify a set of cores to be used for I/O threads performing offline multithreaded builds of ordered indexes, as opposed to normal I/O duties such as file I/O, compression, or decompression. “Offline” in this context refers to building of ordered indexes performed when the parent
table is not being written to; such building takes place when an NDB cluster performs a node or system restart, or as part of restoring a cluster from backup using `ndb_restore --rebuild-indexes`.

In addition, the default behaviour for offline index build work is modified to use all cores available to ndbmtd, rather limiting itself to the core reserved for the I/O thread. Doing so can improve restart and restore times and performance, availability, and the user experience.

This enhancement is implemented as follows:

1. The default value for `BuildIndexThreads` is changed from 0 to 128. This means that offline ordered index builds are now multithreaded by default.

2. The default value for `TwoPassInitialNodeRestartCopy` is changed from `false` to `true`. This means that an initial node restart first copies all data from a “live” node to one that is starting —without creating any indexes—builds ordered indexes offline, and then again synchronizes its data with the live node, that is, synchronizing twice and building indexes offline between the two synchronizations. This causes an initial node restart to behave more like the normal restart of a node, and reduces the time required for building indexes.

3. A new thread type (`idxbld`) is defined for the `ThreadConfig` configuration parameter, to allow locking of offline index build threads to specific CPUs.

In addition, NDB now distinguishes the thread types that are accessible to “ThreadConfig” by the following two criteria:

1. Whether the thread is an execution thread. Threads of types `main`, `ldm`, `recv`, `rep`, `tc`, and `send` are execution threads; thread types `io`, `watchdog`, and `idxbld` are not.

2. Whether the allocation of the thread to a given task is permanent or temporary. Currently all thread types except `idxbld` are permanent.

For additional information, see the descriptions of the parameters in the Manual. (Bug #25835748, Bug #26928111)

- When performing an NDB backup, the `ndbinfo.logbuffers` table now displays information regarding buffer usage by the backup process on each data node. This is implemented as rows reflecting two new log types in addition to `REDO` and `DD-UNDO`. One of these rows has the log type `BACKUP-DATA`, which shows the amount of data buffer used during backup to copy fragments to backup files. The other row has the log type `BACKUP-LOG`, which displays the amount of log buffer used during the backup to record changes made after the backup has started. One each of these `log_type` rows is shown in the `logbuffers` table for each data node in the cluster. Rows having these two log types are present in the table only while an NDB backup is currently in progress. (Bug #25822988)

- Added the `ODirectSyncFlag` configuration parameter for data nodes. When enabled, the data node treats all completed filesystem writes to the redo log as though they had been performed using `fsync`.

---

### Note

This parameter has no effect if at least one of the following conditions is true:

- `ODirect` is not enabled.

- `InitFragmentLogFiles` is set to `SPARSE`.

(Bug #25428560)
MySQL NDB Cluster 8.0 Release Notes

- Added the `--logbuffer-size` option for `ndbd` and `ndbmtd`, for use in debugging with a large number of log messages. This controls the size of the data node log buffer; the default (32K) is intended for normal operations. (Bug #89679, Bug #27550943)

- Prior to NDB 8.0, all string hashing was based on first transforming the string into a normalized form, then MD5-hashing the resulting binary image. This could give rise to some performance problems, for the following reasons:
  - The normalized string is always space padded to its full length. For a `VARCHAR`, this often involved adding more spaces than there were characters in the original string.
  - The string libraries were not optimized for this space padding, and added considerable overhead in some use cases.
  - The padding semantics varied between character sets, some of which were not padded to their full length.
  - The transformed string could become quite large, even without space padding; some Unicode 9.0 collations can transform a single code point into 100 bytes of character data or more.
  - Subsequent MD5 hashing consisted mainly of padding with spaces, and was not particularly efficient, possibly causing additional performance penalties by flush significant portions of the L1 cache.

Collations provide their own hash functions, which hash the string directly without first creating a normalized string. In addition, for Unicode 9.0 collations, the hashes are computed without padding. NDB now takes advantage of this built-in function whenever hashing a string identified as using a Unicode 9.0 collation.

Since, for other collations there are existing databases which are hash partitioned on the transformed string, NDB continues to employ the previous method for hashing strings that use these, to maintain compatibility. (Bug #89609, Bug #27523758)

References: See also: Bug #89590, Bug #27515000, Bug #89604, Bug #27522732.

- A table created in NDB 7.6 and earlier contains metadata in the form of a compressed `.frm` file, which is no longer supported in MySQL 8.0. To facilitate online upgrades to NDB 8.0, NDB performs on-the-fly translation of this metadata and writes it into the MySQL Server's data dictionary, which enables the `mysqld` in NDB Cluster 8.0 to work with the table without preventing subsequent use of the table by a previous version of the NDB software.

  **Important**

  Once a table's structure has been modified in NDB 8.0, its metadata is stored using the Data Dictionary, and it can no longer be accessed by NDB 7.6 and earlier.

  This enhancement also makes it possible to restore an NDB backup made using an earlier version to a cluster running NDB 8.0 (or later).

**Bugs Fixed**

- **Important Change; NDB Disk Data:** It was possible to issue a `CREATE TABLE` statement that referred to a nonexistent tablespace. Now such a statement fails with an error. (Bug #85859, Bug #25860404)

- **Important Change:** NDB supports any of the following three values for the `CREATE TABLE` statement's `ROW_FORMAT` option: `DEFAULT`, `FIXED`, and `DYNAMIC`. Formerly, any values other than these were accepted but resulted in `DYNAMIC` being used. Now a `CREATE TABLE` statement that attempts to create
an **NDB** table fails with an error if `ROW_FORMAT` is used, and does not have one of the three values listed. (Bug #88803, Bug #27230898)

- **Microsoft Windows; ndbinfo Information Database**: The process ID of the monitor process used on Windows platforms by `RESTART` to spawn and restart a `mysqld` is now shown in the `ndbinfo.processes` table as an `angel_pid`. (Bug #90235, Bug #27767237)

- **NDB Cluster APIs**: The example NDB API programs `ndbapi_array_simple` and `ndbapi_array_using_adapter` did not perform cleanup following execution; in addition, the example program `ndbapi_simple_dual` did not check to see whether the table used by this example already existed. Due to these issues, none of these examples could be run more than once in succession.

  The issues just described have been corrected in the example sources, and the relevant code listings in the NDB API documentation have been updated to match. (Bug #27009386)

- **NDB Cluster APIs**: A previous fix for an issue, in which the failure of multiple data nodes during a partial restart could cause API nodes to fail, did not properly check the validity of the associated `NdbReceiver` object before proceeding. Now in such cases an invalid object triggers handling for invalid signals, rather than a node failure. (Bug #25902137)

  References: This issue is a regression of: Bug #25092498.

- **NDB Cluster APIs**: Incorrect results, usually an empty result set, were returned when `setBound()` was used to specify a NULL bound. This issue appears to have been caused by a problem in gcc, limited to cases using the old version of this method (which does not employ `NdbRecord`), and is fixed by rewriting the problematic internal logic in the old implementation. (Bug #89468, Bug #27461752)

- **NDB Cluster APIs**: Released NDB API objects are kept in one or more `Ndb_free_list` structures for later reuse. Each list also keeps track of all objects seized from it, and makes sure that these are eventually released back to it. In the event that the internal function `NdbScanOperation::init()` failed, it was possible for an `NdbApiSignal` already allocated by the `NdbOperation` to be leaked. Now in such cases, `NdbScanOperation::release()` is called to release any objects allocated by the failed `NdbScanOperation` before it is returned to the free list.

  This fix also handles a similar issue with `NdbOperation::init()`, where a failed call could also leak a signal. (Bug #89249, Bug #27389894)

- **NDB Cluster APIs**: Removed the unused `TFSentinel` implementation class, which raised compiler warnings on 32-bit systems. (Bug #89005, Bug #27302881)

- **NDB Cluster APIs**: The success of thread creation by API calls was not always checked, which could lead to timeouts in some cases. (Bug #88784, Bug #27225714)

- **NDB Cluster APIs**: The file `storage/ndb/src/ndbapi/ndberror.c` was renamed to `ndberror.cpp`. (Bug #87725, Bug #26781567)

- **NDB Client Programs**: When passed an invalid connection string, the `ndb_mgm` client did not always free up all memory used before exiting. (Bug #90179, Bug #27737906)

- **NDB Client Programs**: `ndb_show_tables` did not always free up all memory which it used. (Bug #90152, Bug #27727544)

  References: This issue is a regression of: Bug #26908347, Bug #26968613.
MySQL NDB Cluster 8.0 Release Notes

• In some circumstances, when a transaction was aborted in the DBTC block, there remained links to trigger records from operation records which were not yet reference-counted, but when such an operation record was released the trigger reference count was still decremented. (Bug #27629680)

• An internal buffer being reused immediately after it had been freed could lead to an unplanned data node shutdown. (Bug #27622643)

  References: See also: Bug #28698831.

• An NDB online backup consists of data, which is fuzzy, and a redo and undo log. To restore to a consistent state it is necessary to ensure that the log contains all of the changes spanning the capture of the fuzzy data portion and beyond to a consistent snapshot point. This is achieved by waiting for a GCI boundary to be passed after the capture of data is complete, but before stopping change logging and recording the stop GCI in the backup's metadata.

  At restore time, the log is replayed up to the stop GCI, restoring the system to the state it had at the consistent stop GCI. A problem arose when, under load, it was possible to select a GCI boundary which occurred too early and did not span all the data captured. This could lead to inconsistencies when restoring the backup; these could be be noticed as broken constraints or corrupted BLOB entries.

  Now the stop GCI is chosen is so that it spans the entire duration of the fuzzy data capture process, so that the backup log always contains all data within a given stop GCI. (Bug #27497461)

  References: See also: Bug #27566346.

• For NDB tables, when a foreign key was added or dropped as a part of a DDL statement, the foreign key metadata for all parent tables referenced should be reloaded in the handler on all SQL nodes connected to the cluster, but this was done only on the mysqld on which the statement was executed. Due to this, any subsequent queries relying on foreign key metadata from the corresponding parent tables could return inconsistent results. (Bug #27439587)

  References: See also: Bug #82989, Bug #24666177.

• ANALYZE TABLE used excessive amounts of CPU on large, low-cardinality tables. (Bug #27438963)

• Queries using very large lists with IN were not handled correctly, which could lead to data node failures. (Bug #27397802)

  References: See also: Bug #28728603.

• A data node overload could in some situations lead to an unplanned shutdown of the data node, which led to all data nodes disconnecting from the management and nodes.

  This was due to a situation in which API_FAILREQ was not the last received signal prior to the node failure.

  As part of this fix, the transaction coordinator's handling of SCAN_TABREQ signals for an ApiConnectRecord in an incorrect state was also improved. (Bug #27381901)

  References: See also: Bug #47039, Bug #11755287.

• In a two-node cluster, when the node having the lowest ID was started using --nostart, API clients could not connect, failing with Could not alloc node id at HOST port PORT_NO: No free node id found for mysqld(API). (Bug #27225212)

• Changing MaxNoOfExecutionThreads without an initial system restart led to an unplanned data node shutdown. (Bug #27169282)
References: This issue is a regression of: Bug #26908347, Bug #26968613.

- In most cases, and especially in error conditions, NDB command-line programs failed on exit to free memory used by option handling, and failed to call ndb_end(). This is fixed by removing the internal methods `ndb_load_defaults()` and `ndb_free_defaults()` from `storage/ndb/include/util/ndb_opts.h`, and replacing these with an `Ndb_opts` class that automatically frees such resources as part of its destructor. (Bug #26930148)

  References: See also: Bug #87396, Bug #26617328.

- A query against the `INFORMATION_SCHEMA.FILES` table returned no results when it included an `ORDER BY` clause. (Bug #26877788)

- During a restart, DBLQH loads redo log part metadata for each redo log part it manages, from one or more redo log files. Since each file has a limited capacity for metadata, the number of files which must be consulted depends on the size of the redo log part. These files are opened, read, and closed sequentially, but the closing of one file occurs concurrently with the opening of the next.

  In cases where closing of the file was slow, it was possible for more than 4 files per redo log part to be open concurrently; since these files were opened using the `OM_WRITE_BUFFER` option, more than 4 chunks of write buffer were allocated per part in such cases. The write buffer pool is not unlimited; if all redo log parts were in a similar state, the pool was exhausted, causing the data node to shut down.

  This issue is resolved by avoiding the use of `OM_WRITE_BUFFER` during metadata reload, so that any transient opening of more than 4 redo log files per log file part no longer leads to failure of the data node. (Bug #25965370)

- Under certain conditions, data nodes restarted unnecessarily during execution of `ALTER TABLE... REORGANIZE PARTITION`. (Bug #25675481)

  References: See also: Bug #26735618, Bug #27191468.

- Race conditions sometimes occurred during asynchronous disconnection and reconnection of the transporter while other threads concurrently inserted signal data into the send buffers, leading to an unplanned shutdown of the cluster.

  As part of the work fixing this issue, the internal templating function used by the Transporter Registry when it prepares a send is refactored to use likely-or-unlikely logic to speed up execution, and to remove a number of duplicate checks for NULL. (Bug #24444908, Bug #25128512)

  References: See also: Bug #20112700.

- `ndb_restore` sometimes logged data file and log file progress values much greater than 100%. (Bug #20989106)

- Removed unneeded debug printouts from the internal function `ha_ndbcluster::copy_fk_for_offline_alter()`. (Bug #90991, Bug #28069711)

- The internal function `BitmaskImpl::setRange()` set one bit fewer than specified. (Bug #90648, Bug #27931995)

- Inserting a row into an NDB table having a self-referencing foreign key that referenced a unique index on the table other than the primary key failed with ER_NO_REFERENCED_ROW_2. This was due to the fact that NDB checked foreign key constraints before the unique index was updated, so that the constraint check was unable to use the index for locating the row. Now, in such cases, NDB waits until all unique index values have been updated before checking foreign key constraints on the inserted row. (Bug #90644, Bug #27930382)
References: See also: Bug #91965, Bug #28486390.

• Removed all references to the C++ register storage class in the NDB Cluster sources; use of this specifier, which was deprecated in C++11 and removed in C++17, raised warnings when building with recent compilers. (Bug #90110, Bug #27705985)

• It was not possible to create an NDB table using PARTITION_BALANCE set to FOR_RA_BY_LDM_X_2, FOR_RA_BY_LDM_X_3, or FOR_RA_BY_LDM_X_4. (Bug #89811, Bug #27602352)

References: This issue is a regression of: Bug #81759, Bug #23544301.

• Adding a [tcp] or [shm] section to the global configuration file for a cluster with multiple data nodes caused default TCP connections to be lost to the node using the single section. (Bug #89627, Bug #27532407)

• Removed a memory leak in the configuration file parser. (Bug #89392, Bug #27440614)

• Fixed a number of implicit-fallthrough warnings, warnings raised by uninitialized values, and other warnings encountered when compiling NDB with GCC 7.2.0. (Bug #89254, Bug #89255, Bug #89258, Bug #89259, Bug #89270, Bug #27390714, Bug #27390745, Bug #27390684, Bug #27390816, Bug #27396662)

References: See also: Bug #88136, Bug #26990244.

• Node connection states were not always reported correctly by ClusterMgr immediately after reporting a disconnection. (Bug #89121, Bug #27349285)

• As a result of the reuse of code intended for send threads when performing an assist send, all of the local release send buffers were released to the global pool, which caused the intended level of the local send buffer pool to be ignored. Now send threads and assisting worker threads follow their own policies for maintaining their local buffer pools. (Bug #89119, Bug #27349118)

• When the PGMAN block seized a new Page_request record using seizeLast, its return value was not checked, which could cause access to invalid memory. (Bug #89009, Bug #27303191)

• TCROLLBACKREP signals were not handled correctly by the DBTC kernel block. (Bug #89004, Bug #27302734)

• When sending priority A signals, we now ensure that the number of pending signals is explicitly initialized. (Bug #88986, Bug #27294856)

• The internal function ha_ndbcluster::unpack_record() did not perform proper error handling. (Bug #88587, Bug #27150980)

• CHECKSUM is not supported for NDB tables, but this was not not reflected in the CHECKSUM column of the INFORMATION_SCHEMA.TABLES table, which could potentially assume a random value in such cases. Now the value of this column is always set to NULL for NDB tables, just as it is for InnoDB tables. (Bug #88552, Bug #27143813)

• Removed a memory leak detected when running ndb_mgm -e "CLUSTERLOG ...". (Bug #88517, Bug #27128846)

• When terminating, ndb_config did not release all memory which it had used. (Bug #88515, Bug #27128398)

• ndb_restore did not free memory properly before exiting. (Bug #88514, Bug #27128361)
MySQL NDB Cluster 8.0 Release Notes

• In certain circumstances where multiple Ndb objects were being used in parallel from an API node, the block number extracted from a block reference in DBLQH was the same as that of a SUMA block even though the request was coming from an API node. Due to this ambiguity, DBLQH mistook the request from the API node for a request from a SUMA block and failed. This is fixed by checking node IDs before checking block numbers. (Bug #88441, Bug #27130570)

• A join entirely within the materialized part of a semijoin was not pushed even if it could have been. In addition, EXPLAIN provided no information about why the join was not pushed. (Bug #88224, Bug #27022925)

References: See also: Bug #27067538.

• All known compiler warnings raised by -Werror when building the NDB source code have been fixed. (Bug #88136, Bug #26990244)

• When the duplicate weedout algorithm was used for evaluating a semijoin, the result had missing rows. (Bug #88117, Bug #26984919)

References: See also: Bug #87992, Bug #26926666.

• NDB did not compile with GCC 7. (Bug #88011, Bug #26933472)

• A table used in a loose scan could be used as a child in a pushed join query, leading to possibly incorrect results. (Bug #87992, Bug #26926666)

• When representing a materialized semijoin in the query plan, the MySQL Optimizer inserted extra QEP_TAB and JOIN_TAB objects to represent access to the materialized subquery result. The join pushdown analyzer did not properly set up its internal data structures for these, leaving them uninitialized instead. This meant that later usage of any item objects referencing the materialized semijoin accessed an initialized tableno column when accessing a 64-bit tableno bitmask, possibly referring to a point beyond its end, leading to an unplanned shutdown of the SQL node. (Bug #87971, Bug #26919289)

• In some cases, a SCAN_FRAGCONF signal was received after a SCAN_FRAGREQ with a close flag had already been sent, clearing the timer. When this occurred, the next SCAN_FRAGREF to arrive caused time tracking to fail. Now in such cases, a check for a cleared timer is performed prior to processing the SCAN_FRAGREF message. (Bug #87942, Bug #26908347)

• While deleting an element in Dbacc, or moving it during hash table expansion or reduction, the method used (getLastAndRemove()) could return a reference to a removed element on a released page, which could later be referenced from the functions calling it. This was due to a change brought about by the implementation of dynamic index memory in NDB 7.6.2; previously, the page had always belonged to a single Dbacc instance, so accessing it was safe. This was no longer the case following the change; a page released in Dbacc could be placed directly into the global page pool where any other thread could then allocate it.

Now we make sure that newly released pages in Dbacc are kept within the current Dbacc instance and not given over directly to the global page pool. In addition, the reference to a released page has been removed; the affected internal method now returns the last element by value, rather than by reference. (Bug #87932, Bug #26906640)

References: See also: Bug #87987, Bug #26925595.

• When creating a table with a nonexistent conflict detection function, NDB returned an improper error message. (Bug #87628, Bug #26730019)

• ndb_top failed to build with the error "HAVE_NCURSESW_H" is not defined. (Bug #87035, Bug #26429281)
• In a MySQL Cluster with one MySQL Server configured to write a binary log failure occurred when creating and using an NDB table with non-stored generated columns. The problem arose only when the product was built with debugging support. (Bug #86084, Bug #25957586)

• It was possible to create or alter a STORAGE MEMORY table using a nonexistent tablespace without any error resulting. Such an operation now fails with Error 3510 ER_TABLESPACE_MISSING_WITH_NAME, as intended. (Bug #82116, Bug #23744378)

• ndb_restore --print-data --hex did not print trailing 0s of LONGVARBINARY values. (Bug #65560, Bug #14198580)

• When the internal function ha_ndbcluster::copy_fk_for_offline_alter() checked dependent objects on a table from which it was supposed to drop a foreign key, it did not perform any filtering for foreign keys, making it possible for it to attempt retrieval of an index or trigger instead, leading to a spurious Error 723 (No such table).

Index

Symbols
--diff-default, 24, 58
--disable-indexes, 4
--help, 4
--include-stored-grants, 4
--initial, 24, 58
--ndb-schema-dist-timeout, 19, 53
--nostart, 37, 69
--rebuild-indexes, 4
--restore-epoch, 19, 53
--upgrade=force, 4
.ctl files, 11, 46
.deb, 19, 53
.frm, 24, 58

A
aborted transactions, 37, 69
ABORT_BACKUP_ORD, 19, 53
ADD DATAFILE, 24, 58
ALGORITHM=COPY, 4
ALTER, 4
ALTER LOGFILE GROUP, 24, 58
ALTER TABLE, 4, 11, 46
ALTER TABLESPACE, 4, 19, 24, 53, 58
ANALYZE TABLE, 24, 58
Arbitration, 4
ARM, 11, 46
autoincrement, 24, 58
automatic synchronization, 4

B
backup, 4, 19, 24, 37, 58, 69
BACKUP, 11, 46
Backup block, 11, 46
BackupLogBufferSize, 4
backups, 24, 58  
BINARY, 19  
binary log, 37, 69  
BitmaskImpl::setRange(), 37, 69  
BLOB, 19, 24, 33, 37, 58  
BuildIndexThreads, 37, 69  

C  
changes  
  NDB Cluster, 46  
character sets, 11, 46  
checkpoints, 37, 69  
CHECKSUM, 37, 69  
clang, 24, 58  
close(), 24, 58  
ClusterTransactionImpl, 37  
CMake3, 24, 58  
CMVMI, 19, 53  
collations, 37, 69  
COLUMN_FORMAT, 19, 53  
comparison methods, 11, 46  
compilation, 11, 46  
compiling, 11, 19, 24, 37, 46, 53, 58, 69  
concurrent operations, 24, 58  
condition pushdown, 11, 24, 46, 58  
cfg.ini, 37, 69  
configuration, 4, 11, 24, 46, 58  
configuration handling, 11, 46  
correlation IDs, 24, 58  
cpustat, 24, 58  
CREATE TABLE, 11, 19, 24, 33, 37, 46, 58, 66, 69  

D  
data dictionary, 4, 19, 24, 33, 37, 53, 58, 66, 69  
data files, 11, 46  
data node, 33, 66  
data node shutdown, 37, 69  
data nodes, 4, 11, 37, 46, 69  
data pages, 11, 46  
DataMemory, 11, 46  
DATETIME, 24, 58  
DBACC, 37  
Dbacc::getLastAndRemove(), 37, 69  
Dbacc::startNext(), 33, 66  
Dbqlh::sendKeyinfo20(), 37, 69  
DBSPJ, 19, 24, 53, 58  
DBTC, 19, 24, 37, 53, 58, 69  
Dbtc::initData(), 4  
DBTUP, 4, 33, 37, 66  
DbtupBuffer.cpp, 4  
DDL, 19, 24, 53, 58  
demos, 24, 58  
disconnection, 24, 58
DiskDataUsingSameDisk, 4
diskstat, 4
diskstats_1sec, 4
distributed privileges, 11, 24, 46, 58
distribution, 37, 69
DROP DATABASE, 24
DROP TABLE, 19, 24, 37, 53, 58, 69
DROP_TRIG_IMPL_REQ, 33, 66
DUMP, 4, 11, 46
DUMP 9988, 4
DUMP 9989, 4
duplicate weedout, 37, 69
dynamic index memory, 37, 69

Element deletion, 37, 69
endianness, 24, 58
ePOCHS, 19, 53
eq_ref, 19, 24, 53, 58
error codes, 33, 66
error handling, 11, 46
error messages, 4, 11, 46
errors, 4, 11, 19, 33, 37, 46, 53, 66, 69
ER_NO_REFERENCED_ROW_2, 37, 69
ER_TABLESPACE_MISSING_WITH_NAME, 37, 69
examples, 11, 46
examples (NDB API), 37, 69
exit(), 19, 53
EXPLAIN, 37, 69

FILES, 37, 69
FILES table, 4
FIXED, 19, 53
foreign key constraint, 24, 58
foreign keys, 4, 11, 33, 37, 46, 66, 69
frm files, 37, 69

gcc, 37, 69
GCC 8, 33, 66
GCI, 33, 66
GCI boundary, 37, 69
GCP, 4
generated columns, 11, 46
getCreateSchemaVersion(), 11, 46
getExtraMetadata(), 37, 69
global schema lock, 4, 11, 24, 46, 58

hashing, 37, 69
ha_ndbcluster, 19, 53
ha_ndbcluster::copy_fk_for_offline_alter(), 37, 69
ha_ndbcluster::unpack_record(), 37, 69
host names, 11, 46

I
ID allocation, 4
identifiers, 11, 46
idxbld, 37, 69
Important Change, 4, 11, 19, 24, 37, 46, 53, 58, 69
IN, 37, 69
INCL_NODECONF, 4
Incompatible Change, 4, 24, 58
index length, 24, 58
index statistics, 11, 46
INFORMATION_SCHEMA, 37, 69
INFORMATION_SCHEMA.FILES, 24, 37, 58, 69
INFORMATION_SCHEMA.TABLES, 24, 58
InitFragmentLogFiles, 37, 69
InitialNoOfOpenFiles, 24, 58
INPLACE, 4, 19, 53
insert operations, 33, 66
InstallDirectory, 37
invisible indexes, 4

J
joins, 11, 46
JOIN_TAB, 37, 69

L
LCP, 4, 11, 24, 46, 58
LCP pause, 33, 66
LCP_COMPLETE_REP, 4, 24, 58
LCP_STATUS_IDLE, 24, 58
LCP_TAB_SAVED, 11, 46
LDM, 4
less than, 24, 58
libndbclient-devel, 33, 66
libndbclient.so, 33
libssl.so, 33
LIMIT, 19, 53
lock contention, 24, 58
locking, 4, 11, 46
locks, 4, 19, 24, 53, 58
log buffer, 37, 69
log file groups, 33, 66
logbuffers, 37, 69
LOGFILE GROUP, 24, 58
logging, 4
long signals, 19, 53
LONGVARBINARY, 37, 69
LooseScan, 37, 69
lower_case_table_names, 4, 11, 46
LQHKEYREQ, 11, 46
NdbScanFilter, 11, 33, 46, 66
NdbScanOperation, 37, 69
NDBT, 11, 19, 46, 53
NdbTransaction, 24, 58
Ndb_api_read_row_count, 11, 46
ndb_apply_status, 19
ndb_autoincrement_prefetch_sz, 4
ndb_binlog_index, 24, 58
ndb_config, 4, 24, 37, 58, 69
ndb_delete_all, 11, 46
ndb_drop_table, 4, 19, 53
ndb_import, 19, 53
ndb_logevent_get_next2(), 19, 53
ndb_log_bin, 24, 58
Ndb_metadata_blacklist_size, 11, 46
Ndb_metadata_change_monitor, 24, 58
ndb_metadata_check, 4, 24, 58
ndb_metadata_check_interval, 24, 58
Ndb_metadata_detected_count, 24, 58
ndb_metadata_sync, 4
Ndb_metadata_synced_count, 11, 46
ndb_mgm, 19, 37, 53, 69
ndb_mgmd, 11, 19, 33, 37, 46, 53, 66
ndb_milli_sleep(), 11, 46
ndb_my_error(), 33, 66
Ndb_opts, 37, 69
ndb_read_backup, 4
ndb_restore, 4, 11, 19, 24, 33, 37, 46, 53, 58, 66, 69
ndb_row_checksum, 33, 66
ndb_schema, 4
ndb_schema_dist_lock_wait_timeout, 11, 46
NDB_SCHEMA_OBJECT, 19, 53
ndb_schema_share, 19, 53
ndb_setup.py, 19, 53
NDB_SHARE, 11, 46
ndb_show_tables, 37, 69
NDB_STORED_USER, 4, 11, 46
ndb_top, 37, 69
Ndb_trans_hint_count_session, 19, 53
ndb_wait_setup, 4
newtonapi, 24, 58
NO PAD collations, 24, 58
node ID allocation, 11, 46
node IDs, 11, 46
node takeover, 11, 46
NODE_FAILREP, 4
NoOfReplicas, 4
NOT BETWEEN, 24, 58
NOT IN, 24, 58
NULL, 24, 33, 37, 58, 66, 69

O
ODirect, 37, 69
ODirectSyncFlag, 37, 69
OM_WRITE_BUFFER, 37, 69
ON DELETE CASCADE, 24, 58
online DDL, 11, 46
online upgrades, 37, 69
OO_NOWAIT, 11, 46
optimizer, 19, 53
option handling, 37, 69
ORDER BY, 11, 37, 46, 69
O_DIRECT, 4

P
Packaging, 19, 24, 33, 53, 58, 66
parallelism, 24, 58
partial LCP, 4
partial restarts, 4
Partitioning, 11, 46
PARTITION_BALANCE, 37, 69
Performance, 4, 33, 66
performance, 19, 53
PGMAN, 4, 37, 69
pgman_time_track_stats, 4
processes table, 37, 69
pushdown, 11, 24, 46, 58
pushed conditions, 24, 58
pushed joins, 11, 24, 46, 58

Q
QEP_TAB, 37, 69
query memory, 24, 58

R
race, 37, 69
race condition, 24, 58
race conditions, 19, 53
READ_BACKUP, 4
redo log, 24, 37, 58, 69
redo log part metadata, 37, 69
RedoOverCommitCounter, 4
RedoOverCommitLimit, 4
ref, 19, 53
register, 37, 69
REORGANIZE PARTITION, 37, 69
RESET MASTER, 37
resource allocation, 11, 24, 46, 58
resource usage, 37, 69
RESTART, 24, 58
restarts, 11, 37, 46, 69
row ID, 24, 58
ROW_FORMAT, 37, 69

S
SafeCounter, 24, 58
SavedEvent, 19, 53
scanIndex(), 37
scans, 19, 33, 53, 66
SCAN_FRACONF, 37, 69
SCAN_FRAREFER, 37, 69
SCAN_FRAREQ, 33, 37, 66, 69
schema distribution, 11, 19, 46, 53
schema synchronization, 4, 33, 66
schema UUID, 4
SELECT, 4, 24, 58
semijoin, 37, 69
send buffer, 37, 69
send buffers, 37, 69
ServerPort, 37
setExtraMetadata(), 37, 69
SharedGlobalMemory, 4, 19, 53
SHM, 37, 69
ShmSize, 4
SHOW, 19, 53
signals, 37, 69
SignalSender, 4
SINGLE USER MODE, 19, 53
SingleUserMode, 19, 53
SNAPSHOTEND, 19, 53
SNAPSHOTSTART, 19, 53
Solaris, 4
Solaris/x86, 11, 46
SPJ, 37, 69
SQL node, 24, 58
START_LCP_REQ, 4
STOP, 24, 58
stop GCI, 37, 69
STORAGE MEMORY, 37, 69
STRAIGHT_JOIN, 19, 53
subscription logs, 19, 53
SUB_GCP_COMPLETE_REP, 24, 58
SUB_STOP_REQ, 33, 66
SUMA, 33, 37, 66, 69
swap, 4
synchronization, 11, 19, 46, 53
sysfiles, 11, 46
SYSTAB_0, 24, 58

table discovery, 11, 24, 46, 58
table reorganization, 11, 46
Table::getExtraMetadata(), 4
tableno, 37, 69
tablespace, 11, 19, 33, 46, 53, 66
TABLE_READ_ONLY, 11, 46
takeover, 19, 53
TCKEYREQ, 19, 53
TCROLLBACKREP, 37, 69
temporal data types, 24, 58
temporary errors, 19, 53
testing, 33
TEXT, 24, 33, 37, 58
TE_ALTER, 24, 58
TFSentinel, 37, 69
thread creation, 37, 69
ThreadConfig, 24, 37, 58, 69
TIMESTAMP, 19, 53
TINYBLOB, 19, 53
transaction coordinator, 11, 46
TransactionMemory, 4
transactions, 4, 11, 24, 46, 58
TransporterRegistry::prepareSendTemplate(), 37, 69
triggers (NDB), 33, 66
truncation, 24, 58
tuple corruption, 33, 66
TwoPassInitialNodeRestartCopy, 33, 37, 66, 69
type conversion, 19, 53

U
UCA-9.0, 24, 58
undo files, 24, 58
undo log file groups, 33, 66
upgrades, 4, 11, 24, 33, 46, 58, 66
upgrades and downgrades, 11, 46
using filesort, 19, 53

V
version_comment, 33, 66
VIRTUAL, 19

W
wait_until_ready(), 24, 58
warnings, 37, 69
Windows, 11, 46
WiX, 4