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 either the MySQL Forums or MySQL Mailing Lists, where you can discuss your issues with other MySQL users.

Document generated on: 2018-12-11 (revision: 16693)

Table of Contents

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

Documentation Accessibility

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

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit

http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Changes in MySQL NDB Cluster 8.0.14 (Not yet released)

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 (Not yet released, General Availability)).

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

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
    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`.

- `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 synchonizations. 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 additonal 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)

- 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 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 `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)

- **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 metatdata 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)

- 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 #25679639)

- 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 semi-join 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 semi-join, 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 semi-join 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 semi-join 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

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

W