

Another server status variable `Ndb_conflict_fn_max` provides a count of the number of times that a row was not applied on the current SQL node due to “greatest timestamp wins” conflict resolution since the last time that `mysqld` was started.

The number of times that a row was not applied as the result of “same timestamp wins” conflict resolution on a given `mysqld` since the last time it was restarted is given by the global status variable `Ndb_conflict_fn_old`. In addition to incrementing `Ndb_conflict_fn_old`, the primary key of the row that was not used is inserted into an *exceptions table*, as explained elsewhere in this section.

See also [NDB Cluster Status Variables](#).

Examples

The following examples assume that you have already a working NDB Cluster replication setup, as described in [Section 23.6.5, “Preparing the NDB Cluster for Replication”](#), and [Section 23.6.6, “Starting NDB Cluster Replication \(Single Replication Channel\)”](#).

NDB\$MAX() example. Suppose you wish to enable “greatest timestamp wins” conflict resolution on table `test.t1`, using column `mycol` as the “timestamp”. This can be done using the following steps:

1. Make sure that you have started the source `mysqld` with `--ndb-log-update-as-write=OFF`.
2. On the source, perform this `INSERT` statement:

```
INSERT INTO mysql.ndb_replication
VALUES ('test', 't1', 0, NULL, 'NDB$MAX(mycol)');
```

Inserting a 0 into the `server_id` indicates that all SQL nodes accessing this table should use conflict resolution. If you want to use conflict resolution on a specific `mysqld` only, use the actual server ID.

Inserting `NULL` into the `binlog_type` column has the same effect as inserting 0 (`NBT_DEFAULT`); the server default is used.

3. Create the `test.t1` table:

```
CREATE TABLE test.t1 (
  columns
  mycol INT UNSIGNED,
  columns
) ENGINE=NDB;
```

Now, when updates are performed on this table, conflict resolution is applied, and the version of the row having the greatest value for `mycol` is written to the replica.



Note

Other `binlog_type` options—such as `NBT_UPDATED_ONLY_USE_UPDATE` should be used to control logging on the source using the `ndb_replication` table rather than by using command-line options.

NDB\$OLD() example. Suppose an NDB table such as the one defined here is being replicated, and you wish to enable “same timestamp wins” conflict resolution for updates to this table:

```
CREATE TABLE test.t2 (
  a INT UNSIGNED NOT NULL,
  b CHAR(25) NOT NULL,
  columns,
  mycol INT UNSIGNED NOT NULL,
  columns,
  PRIMARY KEY pk (a, b)
```