`ALTER TABLE` Improvements in MARIADB Server

Marko Mäkelä
Lead Developer InnoDB
MariaDB Corporation

FEBRUARY 25-27, 2019 • NEW YORK

Generic `ALTER TABLE` in MariaDB

CREATE TABLE …; INSERT…SELECT; RENAME …; DROP TABLE …;

- Retroactively named `ALGORITHM=COPY` in MySQL 5.6 and MariaDB 10.0
- Until MariaDB 10.2.13 (MDEV-11415), lots of unnecessary undo logging (and the infamous "commit every 10,000 rows" hack to speed up crash recovery).
- Inserting into each index one record at a time (very inefficient).
- No sort buffer is being used inside InnoDB (other than the change buffer)
- Writes a large amount of redo log for the second copy of the table.

History of Native `ALTER TABLE` in InnoDB

Starting with InnoDB Plugin for MySQL 5.1

- “Fast index creation”: `ADD [UNIQUE] INDEX`, `ADD PRIMARY KEY`
- `ALGORITHM=INPLACE` starting with MySQL 5.6 and MariaDB 10.0
  - Misleading name “inplace”; **some operations may rebuild the table!**
    - `(ADD|DROP) COLUMN`, `ADD PRIMARY KEY`, `CHANGE…[NOT] NULL`
  - Some operations are instantaneous: rename column, change `DEFAULT`, …
  - Sometimes sloppily called “online” even when no concurrent DML is allowed

`ALTER ONLINE TABLE`

- InnoDB supports two classes of operations in online `ALTER TABLE`:
  - `ADD [UNIQUE] INDEX`: create indexes without copying the table
  - **online table rebuild:** `ADD PRIMARY KEY` or `ADD`, `DROP`, `MODIFY` columns
- InnoDB refuses `ALTER ONLINE TABLE` or `ALTER TABLE…LOCK=NONE` if:
  - A `FULLTEXT` or `SPATIAL` index is being created
  - The table needs to be rebuilt while `FULLTEXT` or `SPATIAL` index are present

Instant `ALTER TABLE` in InnoDB

MariaDB

OPEN WORKS

FEBRUARY 25-27, 2019 • NEW YORK

Instant `ALTER TABLE` Operations up to 10.3

- 10.0: Renaming columns, changing `DEFAULT` value
- 10.2: Extend `VARCHAR` in some cases: not `VARCHAR(255)` to `VARCHAR(256)`
- 10.3: `ADD COLUMN` (as the last column only), `DROP CONSTRAINT`
- 10.3.8 (MDEV-16330): Add or remove `SYSTEM VERSIONING` of a column
- 10.3.10 (MDEV-16328): change `page_compression_level`
- 10.3.x (MDEV-13301): Rename indexes (by `DROP INDEX`, `ADD INDEX`)

10.4: Instant Change of Collation or Charset

Change character set or collation without copying table

- Change the collation only, e.g., `latin1_swedish_ci` to `latin1_german_ci`
- Change ~~`ascii` to almost anything~~, `utf8mb3` to `utf8mb4`, ~~`ucs2` to `utf16`~~, …
    - Unless the collation is compatible, we must drop/add any indexes on the columns.
    - Unfortunately, columns declared as ascii or ucs2 allow invalid data
- The table **may have to be copied** in order to change the maximum length from 128‥255 **bytes** to more than 255 **bytes**;
  Example: Change `CHAR(85)` or `VARCHAR(85)` from `utf8mb3` to `utf8mb4`

Sponsored by ServiceNow

MariaDB

Instant Column Extension for InnoDB Tables

No change to file formats or data; for any `ROW_FORMAT`

- 10.2: Any extension of `VARCHAR` except from ≤255 bytes to >255 bytes
- 10.4: Any extension of `VARCHAR` from ≤127 bytes or `ROW_RORMAT=REDUNDANT`
- 10.x: Any extension of `CHAR` containing UTF-8 (or other variable-length charset), or internally stored as variable-length
- These operations are compatible with old versions of MariaDB or MySQL.

Instant `ALTER TABLE` Operations in 10.4

Specific to the original `ROW_FORMAT=REDUNDANT`

- Instantly remove `NOT NULL` attribute, or extend any `VARCHAR`.
- **Cancelled (MDEV-18627):** Extend fixed-size columns (treat as variable-size)
  - `TINYINT→SMALLINT→MEDIUMINT→INT→BIGINT`; `CHAR`; `VARCHAR→CHAR`
- Uses 6+*c* or 6+2*c* bytes of record header, storing all c columns as variable-length.
  - Later formats (MySQL 5.0.3+): $5+\lceil\log_2(n+1)\rceil+v$ to $5+\lceil\log_2(n+1)\rceil+2v$ bytes ($v\le c$, $n\le c$); using extra space for variable-length or `NULL`able columns only. Minimum is 5 bytes.

Short History of InnoDB `ROW_FORMAT`

- Originally, InnoDB had a record header of 6+*c* or 6+2*c* bytes.
  - Basically, each column was encoded as variable-length and allowing `NULL`.
- MySQL 5.0.3 retroactively named the original format `ROW_FORMAT=REDUNDANT` and introduced a new default `ROW_FORMAT=COMPACT`:
  - 5-byte fixed header, "is null" bitmap (except for `NOT NULL` columns), encode the lengths of variable-length fields only (using 1 or 2 bytes per field)
  - `CHAR(`*n*`)` on UTF-8 is encoded like `VARCHAR` (*n* to 3*n* or 4*n* bytes)
  - **Must copy table** to remove `NOT NULL` or to extend fixed-length columns.
- InnoDB Plugin for MySQL 5.1 introduced `DYNAMIC` and (dead end) `COMPRESSED`:
  - Based on `COMPACT`, but not storing 768-byte prefix of off-page columns.
- `innodb_default_row_format=DYNAMIC` since MariaDB 10.2

File Format Changes to Avoid Rebuild for Instant `ALTER TABLE`

MariaDB
OPEN WORKS
FEBRUARY 25-27, 2019 • NEW YORK

`ALTER TABLE` Improvements in MariaDB 10.3

- MDEV-13134 introduced syntax to avoid “surprise rebuilds”: `ALGORITHM=(INSTANT|NOCOPY)` and `SET alter_algorithm=(instant|nocopy)`
- MDEV-11369 introduced instant `ADD COLUMN`, limited to appending last
  - Both Alibaba and Tencent had developed something similar based on MySQL 5.6.
  - MariaDB supports also `DEFAULT` value expressions, with values stored in one place, in a hidden *metadata record* at the start of the clustered index.
  - Does not support `ROW_FORMAT=COMPRESSED`.

Example of Instant `ADD COLUMN`

```
CREATE TABLE t(id INT PRIMARY KEY, u INT UNIQUE) ENGINE=InnoDB;
INSERT INTO t(id,u) VALUES(1,1),(2,2),(3,3);
ALTER TABLE t ADD COLUMN
(d DATETIME DEFAULT current_timestamp(),
 t TEXT CHARSET utf8 DEFAULT 'The quick brown fox',
 p POINT NOT NULL DEFAULT ST_GeomFromText('POINT(0 0)'));
UPDATE t SET t=NULL WHERE id=3;
```

id	u
1	1
2	2
3	3

MariaDB

Example of Instant ADD COLUMN

```
CREATE TABLE t(id INT PRIMARY KEY, u INT UNIQUE) ENGINE=InnoDB;
INSERT INTO t(id,u) VALUES(1,1),(2,2),(3,3);
ALTER TABLE t ADD COLUMN
(d DATETIME DEFAULT current_timestamp(),
 t TEXT CHARSET utf8 DEFAULT 'The quick brown fox',
 p POINT NOT NULL DEFAULT ST_GeomFromText('POINT(0 0)'));
UPDATE t SET t=NULL WHERE id=3;
```

id	u	d	t	p
		2017-11-10 12:14:00	'The quick brown fox'	POINT(0 0)
1	1	2017-11-10 12:14:00	'The quick brown fox'	POINT(0 0)
2	2	2017-11-10 12:14:00	'The quick brown fox'	POINT(0 0)
3	3	2017-11-10 12:14:00	'The quick brown fox'	POINT(0 0)

MariaDB

Example of Instant ADD COLUMN

```
CREATE TABLE t(id INT PRIMARY KEY, u INT UNIQUE) ENGINE=InnoDB;
INSERT INTO t(id,u) VALUES(1,1),(2,2),(3,3);
ALTER TABLE t ADD COLUMN
(d DATETIME DEFAULT current_timestamp(),
 t TEXT CHARSET utf8 DEFAULT 'The quick brown fox',
 p POINT NOT NULL DEFAULT ST_GeomFromText('POINT(0 0)'));
UPDATE t SET t=NULL WHERE id=3;
```

id	u	d	t	p
		2017-11-10 12:14:00	'The quick brown fox'	POINT(0 0)
1	1	2017-11-10 12:14:00	'The quick brown fox'	POINT(0 0)
2	2	2017-11-10 12:14:00	'The quick brown fox'	POINT(0 0)
3	3	2017-11-10 12:14:00	NULL	POINT(0 0)

10.4: `DROP, (ADD|MODIFY)…(FIRST|AFTER…)`

Extends the 10.3 Instant `ADD COLUMN` metadata record with a BLOB

- Keeps the user record format unchanged; adds metadata for column mapping.
  - Physically, do `ADD COLUMN` last in the clustered index records.
  - `DROP COLUMN` will leave garbage in old records; new records will write `NULL`, empty strings, or dummy fixed-length values.
- The format of secondary indexes remains completely unchanged.
- Replacing `PRIMARY KEY(a,b)` with `PRIMARY KEY(b,a)` must copy the table.

Basic Usage of Instant `ALTER TABLE`

- By default, `ALTER TABLE` is instantaneous when possible
- Use the `FORCE` keyword for the old-fashioned table rebuild, with the old-fashioned (additional) limitations with regard to `FULLTEXT INDEX` and `SPATIAL INDEX`
- `FULLTEXT INDEX` limits the ability to `ADD`, `DROP` columns or change their order

- To monitor the number of avoided table rebuilds via using the metadata record:

```
SELECT variable_value
FROM information_schema.global_status
WHERE variable_name = 'innodb_instant_alter_column';
```

- See also https://mariadb.com/resources/blog/instant-add-column-innodb

Record Changes for Instant `ADD COLUMN`

- An InnoDB table is a collection of indexes:
  - Clustered index (ordered by `PRIMARY KEY` or similar); index-organized table
  - Optional secondary indexes, pointing to clustered index keys
- In the clustered index leaf page records, we `ADD COLUMN` last:
  - (`PRIMARY KEY`, `DB_TRX_ID`, `DB_ROLL_PTR`, non-virtual columns, **added columns**)
- How to tell if added columns are present?
  - `ROW_FORMAT=REDUNDANT` explicitly stores the number of index fields.
  - `ROW_FORMAT=COMPACT`, `ROW_FORMAT=DYNAMIC` will require bigger changes:
    - Record header flag and optional field for “number of added columns”.
    - Must store the original number of fields or columns somewhere.

Page Changes for Instant `ALTER TABLE`

- Root page: `FIL_PAGE_TYPE_INSTANT`; `PAGE_INSTANT` stores the original (smaller, or with DROP COLUMN, bigger) number of clustered index fields
- At the start of the clustered index, store a metadata record with `REC_INFO_MIN_REC_FLAG` and the optional “added columns” header:
  - The number of fields must match the current table definition
  - Values of “added columns” are the values of “missing columns” in user records
  - For `DROP COLUMN`, some original metadata is stored in a metadata BLOB

Better `ALTER TABLE` for Replication and all Storage Engines

MariaDB OPENWORKS

FEBRUARY 25-27, 2019 • NEW YORK

Problems with Online Table Rebuild

Why are tools like GH-OST still used instead of `ALTER ONLINE TABLE`?

- Replication slave will only start after commit→huge lag (to be fixed in MDEV-11675)
- The `online_log` needs to be buffered (in memory or temporary files)
  - The size depends on the concurrent DML workload; hard to predict!
  - Written before commit; DML duplicate key errors make also `ALTER TABLE` fail

Watch out for MDEV-16329 Cross-Engine `ALTER ONLINE TABLE`

- Keep engine-native for `ADD [UNIQUE] INDEX` or `ALGORITHM=INSTANT`

Speeding up Bulk Operations in InnoDB

Needed for MDEV-16329 Cross-Engine `ALTER ONLINE TABLE`

MDEV-515: InnoDB bulk insert into empty table or partition

- Speeds up `mysqldump` and many `INSERT`, `REPLACE`, `LOAD DATA`
- Works also for generic `ALTER TABLE…ALGORITHM=COPY`
- For recovery, just write 1 undo log record “truncate on rollback”
- Avoid or reduce redo logging; build indexes pre-sorted, page by page
  - Similar to `CREATE INDEX` in MariaDB 10.2+

Theoretical Limits of Avoiding Copying in `ALTER TABLE`

MariaDB

OPEN WORKS

FEBRUARY 25-27, 2019 • NEW YORK

Format Tagging for Lazy Conversions

Avoid rebuilding or copying the table when changing data encodings

- Format changes can be instantaneous if they relax constraints:
    - Change virtually anything to `utf8` or `utf16`; e.g.: `_latin1` 0xe4 ≙ `_utf8` 0xc3a4
    - Change `INT UNSIGNED` to `BIGINT` (unsigned to wider signed integer)
- These could be implemented with a per-record or per-page “format version tag” and by converting records to the newest version whenever the data is being read.
- Affected secondary indexes must be rebuilt.

File Format Changes for Format Tagging

User data records (or pages) must indicate their physical format

- “Format version number” that points to something in the hidden metadata record?
- A prototype with “dual-format” clustered index leaf pages was implemented and rejected due to the `ROW_FORMAT=REDUNDANT` storage overhead
- For any `ROW_FORMAT`, we need additional metadata to indicate how to convert data when reading or searching: e.g., `latin1` to `utf8`, `INT` to `BIGINT`
- **Do we want this?** Could add significant memory and time overhead to DML!

`ALGORITHM=NOCOPY` with Validation (1/2)

Avoid copying the table even if the data could be incompatible

- Perform a **locking table scan** to validate the data.
  - Example: `i BIGINT NULL` to `INT UNSIGNED NOT NULL` is OK if `i>=0`
  - `ALTER` **`ONLINE`** `TABLE` actually conflicts with `ALGORITHM=`**`NOCOPY`** in this case!
  - `ALTER IGNORE TABLE` would involve `UPDATE` of offending data.
- Affected **secondary indexes must be rebuilt** if the physical format changes
  - `ADD CONSTRAINT` … (`CHECK`|`FOREIGN KEY`) does not change format!

`ALGORITHM=NOCOPY` with Validation (2/2)

The Lifetime of an `ALTER TABLE` Transaction

1. Check constraints for each row, e.g., `MODIFY i INT UNSIGNED`:
   - `ALTER IGNORE` would `UPDATE` offending data, e.g.: `SET i=NULL WHERE i<0`
2. `DROP INDEX` and `ADD INDEX` of affected indexes, or user-specified ones
3. Any additional operations that are part of the `ALTER` (say, instant `DROP COLUMN`)
4. Update the data dictionary

Summary

- MariaDB 10.3 and 10.4 changed the InnoDB data format to allow instantaneous (`ADD`|`MODIFY`) `COLUMN`…(`FIRST`|`AFTER`…), `DROP`. You can still `FORCE` a rebuild.
- MariaDB 10.4 supports instant `ALTER TABLE` whenever it is technically possible without changing the storage format further.
- Future MariaDB versions might support instant `ALTER TABLE` or avoid copying whenever technically possible. The current metadata format is extensible.
- Use `ALGORITHM=INSTANT` or `ALGORITHM=NOCOPY` (or `SET alter_algorithm`) to get errors instead of unexpected DoS via excessive I/O.