ALTER TABLE IMPROVEMENTS IN MARIADB SERVER

Marko Mäkelä
Lead Developer InnoDB
MariaDB Corporation

Generic ALTER TABLE in MySQL & MariaDB

`CREATE...; INSERT...SELECT; RENAME...; DROP`

- Starting with MySQL 5.6 & MariaDB 10.0, called `ALGORITHM=COPY`
- Until MySQL 8.0 & MariaDB 10.2, lots of unnecessary undo logging
  - To speed up crash recovery, there was a hack of “commit every 10,000 rows”.
- Copies data one index record at a time, unsorted

History of InnoDB Native ALTER TABLE (1/2)

- InnoDB Plugin for MySQL 5.1: `ADD [UNIQUE] INDEX`, `ADD PRIMARY KEY`
  - Pre-sorts all data for each index that is being created
- `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`, …
  - Sloppily called “online” even when no concurrent DML is allowed or involved

History of InnoDB Native ALTER TABLE (2/2)

- MySQL 5.7 (and MariaDB 10.2) introduced bulk index creation:
  - Build the indexes one leaf page at a time, without redo logging
  - MariaDB introduced `innodb_log_optimize_ddl=OFF` for backup-friendliness
- Some MySQL 5.6 & 5.7 (MariaDB 10.0 & 10.2) features are half-baked:
  - Native `ALTER TABLE` refuses to create or rebuild multiple `FULLTEXT INDEX`
  - Some combinations of operations involving 5.7 (10.2) virtual columns are refused

ALTER ONLINE TABLE

- InnoDB supports two classes of operations in online `ALTER TABLE`:
  - `ADD [UNIQUE] INDEX`: create indexes without copying the table
  - online rebuild: `ADD PRIMARY KEY` or `ADD`, `DROP`, `MODIFY` columns; `FORCE`
- Not implemented for the bug-ridden `FULLTEXT` or `SPATIAL` indexes
  - `FULLTEXT INDEX` has suffered from hangs and various other issues
  - `SPATIAL INDEX` can return wrong results due to corruption or race conditions

Instant ALTER with Existing Data Format

MariaDB

Instant ALTER TABLE Operations in InnoDB

- 5.6 & 10.0: Renaming columns, changing `DEFAULT` value
- 5.7 & 10.2: Extend `VARCHAR` in some cases (not crossing 255→256 bytes)
- 10.3: Avoid “surprise rebuilds” by `ALGORITHM=(INSTANT|NOCOPY)`
- 10.3: Various metadata changes that do not affect the data format
  - `DROP CONSTRAINT`, enable/disable the `SYSTEM VERSIONING` of a column, …
- 10.4: `CHARSET` utf8mb3→utf8mb4, `COLLATE` (may rebuild indexes)

Extending VARCHAR (or UTF-8 CHAR)

- How MySQL 5.0.3 `ROW_FORMAT=COMPACT` and its variations encode lengths l:
  - If l<128 or l_{max}<256: encode l in 1 byte. Else, encode in 2 bytes (MSB set in 1st byte)
  - MariaDB 10.4: Any extension from l_{max}<128 to l_{max}>255 is allowed!
  - MariaDB 10.4: Any extension in `ROW_FORMAT=REDUNDANT` tables is allowed!
- Change of `CHARSET` will affect the data format if l_{max} in bytes changes from [128,255] to more than 255
  - Instead of `ALGORITHM=INSTANT`, such operation would use `ALGORITHM=COPY`

File Format Changes for Instant ALTER

MariaDB

A Word on Compatibility

- Downgrades are usually not tested, and cannot be guaranteed to work.
  - Users (and customers) may want to downgrade, at least between minor versions.
  - We must avoid unnecessary incompatible changes to file formats.
- If you do not use instant `ADD`/`DROP`/reorder column, you should be able to export files from MariaDB 10.3 or 10.4 to earlier versions.
- The changes to the format must be clearly identified, so that an attempt to import the files into older versions will fail gracefully.

History of Instant ADD COLUMN

- 10.3: `ADD COLUMN` (as the last column only, with constant `DEFAULT` value)
  - No format changes to metadata tables; supports `IMPORT TABLESPACE`
  - Does not support `ROW_FORMAT=COMPRESSED`.
  - Alibaba and Tencent had something similar in their MySQL 5.6 forks.
  - MySQL 8.0 later introduced a more limited version, storing metadata externally
- MariaDB evaluates the `DEFAULT` expressions during `ALTER TABLE` and stores the values in a hidden metadata record at the start of the clustered index.

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)

MariaDB 10.4: Instant DROP & reorder

- After instant `DROP COLUMN`, we must keep storing dummy (garbage) values.
  - A mapping of columns and clustered index fields is stored in the metadata record.
  - The mapping also enables instant (`ADD`|`CHANGE`|`MODIFY`)…(`FIRST`|`AFTER`…).
  - May be refused due to the presence of `FULLTEXT INDEX` or virtual columns.
- Internally, clustered index fields for added columns are appended to the end.
- The format of secondary indexes remains completely unchanged.

Basic Usage of Instant ALTER TABLE

- By default, `ALTER TABLE` is instantaneous when possible.
  - Use the `FORCE` keyword if you want to rebuild the table, with the associated limitations regarding `FULLTEXT INDEX` and `SPATIAL INDEX`.
  - See also https://mariadb.com/resources/blog/instant-add-column-innodb
- 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';
```

Better ALTER TABLE for Replication and All Storage Engines

MariaDB

Problems with Online InnoDB Table Rebuild

- Replicas will only start applying `ALTER TABLE` after the master finished
  - Large tables cause a huge replication lag; the fix MDEV-11675 is targeting 10.5
- Log of concurrent changes must be buffered; the size is hard to predict
  - Written **before** DML `COMMIT`; ‘transient’ duplicate key errors cause failures
- 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

- Planned feature: MDEV-515: InnoDB bulk insert into empty table or partition
  - Speeds up replaying `mysqldump` and many `INSERT`, `REPLACE`, `LOAD DATA`
  - Works also for generic `ALTER TABLE…ALGORITHM=COPY`
  - Also for MDEV-16329 Cross-Engine `ALTER ONLINE TABLE`
- For recovery, just write 1 undo log record “truncate on rollback”
- Build indexes pre-sorted, page by page, like `CREATE INDEX` does

Theoretical Limits of Avoiding Copying in ALTER TABLE

MariaDB

Deferred Conversions and Format Tagging

- Payload format changes can be instantaneous if they relax constraints:
  - Change `INT UNSIGNED` to `BIGINT` (unsigned to wider signed integer)
  - Change “anything” to `utf8` or `utf16`; e.g.: `_latin1 0xe4` ≙ `_utf8 0xc3a4`
    - Must validate `ascii` and `ucs2` data due to bugs that allowed invalid data!
- 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.

ALGORITHM=NOCOPY with Validation (1/2)

- Avoid copying, but perform a table scan to validate the data.
  - Hard to avoid locking the entire table; maybe triggers could be involved?
  - `ALTER IGNORE TABLE` could involve `UPDATE` of offending data.
- Example: `i BIGINT NULL→INT UNSIGNED NOT NULL` might be OK
- 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)

1. Scan the table to validate all rows, e.g., to `MODIFY i INT UNSIGNED`:
   - `ALTER IGNORE` would `UPDATE` offending data, e.g.: `SET i=NULL WHERE i<0`
2. Execute any `DROP INDEX` or `ADD INDEX`
   - Also rebuild any secondary indexes whose format would be affected
3. Execute any additional operations (such as instant `DROP COLUMN`)
4. Update the data dictionary

Summary

- MariaDB 10.3 and 10.4 changed the data format to allow instantaneous (`ADD`|`MODIFY`) `COLUMN`…(`FIRST`|`AFTER`…), `DROP COLUMN`.
- `ALTER TABLE`…`FORCE;` will request a rebuild in the ‘canonical’ fixed format.
- You can avoid “surprise rebuilds” (unexpected DoS via excessive I/O) by:
  - Specifying `ALGORITHM=INSTANT` or `ALGORITHM=NOCOPY`
  - `SET alter_algorithm=instant;` or `SET alter_algorithm=nocopy;`
  - If the “efficiency constraint” cannot be fulfilled, the `ALTER TABLE` will be refused.


OPENWORKS
2020
MAY 4-6
CONRAD NEW YORK
EARLY BIRD REGISTRATION OPEN:
MARIADB.COM/OPENWORKS
MariaDB