`binlog_row_value_options=PARTIAL_JSON`. If a replication source has this variable set, partial updates received from that source are handled and applied by a replica regardless of the replica's own setting for the variable.

Servers running MySQL 8.0.2 or earlier do not recognize the log events used for JSON partial updates. For this reason, when replicating to such a server from a server running MySQL 8.0.3 or later, `binlog_row_value_options` must be disabled on the source by setting this variable to `''` (empty string). See the description of this variable for more information.

17.5.1.18 Replication and LIMIT

Statement-based replication of `LIMIT` clauses in `DELETE`, `UPDATE`, and `INSERT ... SELECT` statements is unsafe since the order of the rows affected is not defined. (Such statements can be replicated correctly with statement-based replication only if they also contain an `ORDER BY` clause.) When such a statement is encountered:

- When using `STATEMENT` mode, a warning that the statement is not safe for statement-based replication is now issued.

  When using `STATEMENT` mode, warnings are issued for DML statements containing `LIMIT` even when they also have an `ORDER BY` clause (and so are made deterministic). This is a known issue. (Bug #42851)

- When using `MIXED` mode, the statement is now automatically replicated using row-based mode.

17.5.1.19 Replication and LOAD DATA

`LOAD DATA` is considered unsafe for statement-based logging (see Section 17.2.1.3, “Determination of Safe and Unsafe Statements in Binary Logging”). When `binlog_format=MIXED` is set, the statement is logged in row-based format. When `binlog_format=STATEMENT` is set, note that `LOAD DATA` does not generate a warning, unlike other unsafe statements.

If you use `LOAD DATA` with `binlog_format=STATEMENT`, each replica on which the changes are to be applied creates a temporary file containing the data. The replica then uses a `LOAD DATA` statement to apply the changes. This temporary file is not encrypted, even if binary log encryption is active on the source, If encryption is required, use row-based or mixed binary logging format instead, for which replicas do not create the temporary file.

If a `PRIVILEGE_CHECKS_USER` account has been used to help secure the replication channel (see Section 17.3.3, “Replication Privilege Checks”), it is strongly recommended that you log `LOAD DATA` operations using row-based binary logging (`binlog_format=ROW`). If `REQUIRE_ROW_FORMAT` is set for the channel, row-based binary logging is required. With this logging format, the `FILE` privilege is not needed to execute the event, so do not give the `PRIVILEGE_CHECKS_USER` account this privilege. If you need to recover from a replication error involving a `LOAD DATA INFILE` operation logged in statement format, and the replicated event is trusted, you could grant the `FILE` privilege to the `PRIVILEGE_CHECKS_USER` account temporarily, removing it after the replicated event has been applied.

When `mysqlbinlog` reads log events for `LOAD DATA` statements logged in statement-based format, a generated local file is created in a temporary directory. These temporary files are not automatically removed by `mysqlbinlog` or any other MySQL program. If you do use `LOAD DATA` statements with statement-based binary logging, you should delete the temporary files yourself after you no longer need the statement log. For more information, see Section 4.6.9, “mysqlbinlog — Utility for Processing Binary Log Files”.

17.5.1.20 Replication and max_allowed_packet