- `STATEMENT` causes logging to be statement based.
- `ROW` causes logging to be row based. This is the default.
- `MIXED` causes logging to use mixed format.

Setting the binary logging format does not activate binary logging for the server. The setting only takes effect when binary logging is enabled on the server, which is the case when the `log_bin` system variable is set to `ON`. From MySQL 8.0, binary logging is enabled by default, and is only disabled if you specify the `--skip-log-bin` or `--disable-log-bin` option at startup.

The logging format also can be switched at runtime, although note that there are a number of situations in which you cannot do this, as discussed later in this section. Set the global value of the `binlog_format` system variable to specify the format for clients that connect subsequent to the change:

```
mysql> SET GLOBAL binlog_format = 'STATEMENT';
mysql> SET GLOBAL binlog_format = 'ROW';
mysql> SET GLOBAL binlog_format = 'MIXED';
```

An individual client can control the logging format for its own statements by setting the session value of `binlog_format`:

```
mysql> SET SESSION binlog_format = 'STATEMENT';
mysql> SET SESSION binlog_format = 'ROW';
mysql> SET SESSION binlog_format = 'MIXED';
```

Changing the global `binlog_format` value requires privileges sufficient to set global system variables. Changing the session `binlog_format` value requires privileges sufficient to set restricted session system variables. See Section 5.1.9.1, “System Variable Privileges”.

There are several reasons why a client might want to set binary logging on a per-session basis:

- A session that makes many small changes to the database might want to use row-based logging.
- A session that performs updates that match many rows in the `WHERE` clause might want to use statement-based logging because it is more efficient to log a few statements than many rows.
- Some statements require a lot of execution time on the source, but result in just a few rows being modified. It might therefore be beneficial to replicate them using row-based logging.

There are exceptions when you cannot switch the replication format at runtime:

- The replication format cannot be changed from within a stored function or a trigger.
- If the `NDB` storage engine is enabled.
- If a session has open temporary tables, the replication format cannot be changed for the session (`SET @@SESSION.binlog_format`).
- If any replication channel has open temporary tables, the replication format cannot be changed globally (`SET @@GLOBAL.binlog_format` or `SET @@PERSIST.binlog_format`).
- If any replication channel applier thread is currently running, the replication format cannot be changed globally (`SET @@GLOBAL.binlog_format` or `SET @@PERSIST.binlog_format`).

Trying to switch the replication format in any of these cases (or attempting to set the current replication format) results in an error. You can, however, use `PERSIST_ONLY` (`SET`