For beginners, a useful startup option is `--safe-updates` (or `--i-am-a-dummy`, which has the same effect). Safe-updates mode is helpful for cases when you might have issued an `UPDATE` or `DELETE` statement but forgotten the `WHERE` clause indicating which rows to modify. Normally, such statements update or delete all rows in the table. With `--safe-updates`, you can modify rows only by specifying the key values that identify them, or a `LIMIT` clause, or both. This helps prevent accidents. Safe-updates mode also restricts `SELECT` statements that produce (or are estimated to produce) very large result sets.

The `--safe-updates` option causes `mysql` to execute the following statement when it connects to the MySQL server, to set the session values of the `sql_safe_updates`, `sql_select_limit`, and `max_join_size` system variables:

```
SET sql_safe_updates=1, sql_select_limit=1000, max_join_size=1000000;
```

The `SET` statement affects statement processing as follows:

- Enabling `sql_safe_updates` causes `UPDATE` and `DELETE` statements to produce an error if they do not specify a key constraint in the `WHERE` clause, or provide a `LIMIT` clause, or both. For example:

  ```
  UPDATE tbl_name SET not_key_column=val WHERE key_column=val;

  UPDATE tbl_name SET not_key_column=val LIMIT 1;
  ```

- Setting `sql_select_limit` to 1,000 causes the server to limit all `SELECT` result sets to 1,000 rows unless the statement includes a `LIMIT` clause.

- Setting `max_join_size` to 1,000,000 causes multiple-table `SELECT` statements to produce an error if the server estimates it must examine more than 1,000,000 row combinations.

To specify result set limits different from 1,000 and 1,000,000, you can override the defaults by using the `--select-limit` and `--max-join-size` options when you invoke `mysql`:

```
mysql --safe-updates --select-limit=500 --max-join-size=10000
```

It is possible for `UPDATE` and `DELETE` statements to produce an error in safe-updates mode even with a key specified in the `WHERE` clause, if the optimizer decides not to use the index on the key column:

- Range access on the index cannot be used if memory usage exceeds that permitted by the `range_optimizer_max_mem_size` system variable. The optimizer then falls back to a table scan. See Limiting Memory Use for Range Optimization.

- If key comparisons require type conversion, the index may not be used (see Section 8.3.1, “How MySQL Uses Indexes”). Suppose that an indexed string column `c1` is compared to a numeric value using `WHERE c1 = 2222`. For such comparisons, the string value is converted to a number and the operands are compared numerically (see Section 12.3, “Type Conversion in Expression Evaluation”), preventing use of the index. If safe-updates mode is enabled, an error occurs.

As of MySQL 8.0.13, safe-updates mode also includes these behaviors:

- `EXPLAIN` with `UPDATE` and `DELETE` statements does not produce safe-updates errors. This enables use of `EXPLAIN` plus `SHOW WARNINGS` to see why an index is not used, which can be helpful in cases such as when a `range_optimizer_max_mem_size` violation or type conversion occurs and the optimizer does not use an index even though a key column was specified in the `WHERE` clause.

- When a safe-updates error occurs, the error message includes the first diagnostic that was produced, to provide information about the reason for failure. For example, the message may indicate that the `range_optimizer_max_mem_size` value was exceeded or type conversion occurred, either of which can preclude use of an index.