MySQL Configuration Settings

From the start, MySQL has had many configuration variables, and the set has been growing quickly. In the early years, MySQL AB and Sun Microsystems had more pressing issues than build a comprehensive multi-version settings lookup site, so we developed an all-in-one system variables summary table. As MySQL and MariaDB versions multiply, and as they become more sophisicated, the growth rate in settings and complexity itself has been growing. Also our bealeagured summary table..

MySQL now has a usable and comprehensive online lookup table for its options and variables, ***here*.** For each active MySQL version—now 5.5, 5.6, 5.7, 8.0—it has a link to its primary documentation, and shows when each setting was introduced, whether it's local and/or global, whether it's dynamic, if it's been deprecated then when, and if it's been removed then when. As of version 8.0/14, the table has 1,649 entries—a blizzard of customisations, efficiencies, conveniences, interdependencies, backward incompatibilities and error opportunities. To help with incompatibilities, it also has links to version change lists. Mercifully, all this allows us to retire our summary table.

If you are using just one version of MySQL, you may want to restrict your view to just that version. Or you may wish to see settings for just a particular topic. Or if you're using a MariaDB fork of MySQL, you'll want to be aware of MariaDB-MySQL differences in configuration settings. Here are links to some useful lookup summary grids …

Topic	MySQL 5.5	MySQL 5.6	MySQL 5.7	MySQL 8.0
Server options	page	page	page	page
Server options for InnoDB	page	page	page	page
Server options for MyISAM	page	page	page	page
Server options for encryption	page	page	page	page
Server options for security	page	page	page	page
Server options for the binary log	page	page	page	page
Server options for replication	page	page	page	page
Server system variables	page	page	page	page
Differences with MariaDB 5.5	page	page		
Differences with MariaDB 10.0		page		
Differences with MariaDB 10.1		page	page	
Differences with MariaDB 10.2		page	page	
Differences with MariaDB 10.3			page	page
Differences with mariaDB 10.4				page

Fetching and setting these variables and options

Some options are Server Command Options, some are Dynamic Variables, some are *both* but *beware, they are written differently: with dashes* on the command line and in option files, *with underscores* when SELECTed or SET..

Some options and vriables can be set only in the server connection string; some can be set only by recompiling the server.
Since 8.0, find where a current setting came from by querying `performance_schema.variable_info`.

Running *mysqladmin -uUSR -pPWD variables* lists most setttings and options. More than half are dynamic.

From 5.1.22 to 5.7.6, an `information_schema` query retrieves the same variables; since 5.7.6, query `performance_schema` instead:

```
SELECT variable_name, variable_value
FROM performance_schema.{global|session}_variables
[WHERE variable_name LIKE '…'] [ORDER BY variable_name];
```

GLOBAL variable settings go away when the server shuts down, LOCAL (SESSION) settings with the connection—both unless, since 8.0, set with PERSIST (see below). The usual syntax for *retrieving* SELECTable variable values is …

SELECT @@[GLOBAL|LOCAL|SESSION].variable_name;

but in some cases, for example `autocommit`, the leading @@ is optional.

Some system variables are dynamic (can be set with SQL), some not. There are two syntaxes for *SETting* dynamic system variables:

SET [GLOBAL | SESSION | LOCAL | PERSIST] mysql_option=value[,...]
SET @@[GLOBAL.|SESSION|.LOCAL.|PERSIST.]mysql_option=value[,...]

Numeric settngs ending with K, M, G, T, P, or E indicate a power of 1024 from 1…6. SET ONE_SHOT has been removed.

As of 8.0, SET…PERSIST persists the setting across boots by writing it to *<datadir>/mysqld-auto.cnf*, SET…PERSIST_ONLY sets the variable only for subsequent boots, and RESET PERSIST undoes PERSIST.

To set a MySQL session variable for execution of just one statement, embed `/*+ SET_VAR(varname=val)*/` in the SELECT list; slaves ignore it.

sql_mode

Added in version 3.23.41, `sql_mode` may be set on the server command line, for example …

```
--sql-mode=ansi
```

where its name has a dash, not an underscore, or similarly in *my.cnf/ini* without the '—' prefix, or with `SET/SELECT` syntax where it is `sql_mode`. Enclose multiple parameters in quotes. Before 5.6 the default is '', in 5.6 `NO_ENGINE_ SUBSTITUTION`, since 5.7.5 `ONLY_FULL_GROUP_BY,NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES`.

Setting	***Meaning***
`ALLOW_INVALID_DATES`	In STRICT mode, for DATE and DATETIME columns, check only that 1<=month<=12 and day<=1<=31, so permit dates like `2005-02-31`. Since 5.0.2. In 5.0.1 and before, this was default behaviour.

`ANSI`	Equivalent to
	`SET GLOBAL sql_mode = "REAL AS FLOAT,PIPES AS CONCAT,ANSI QUOTES,IGNORE SPACE";`
	`SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;`
	Before 5.03 and since 5.7.5, ANSI also includes `ONLY_FULL_GROUP_BY`.

`ANSI_QUOTES`	Double-quote character becomes an identifier, like the backtick `, *not a string quote character*, so double quotes cannot be used to quote literal string values.

Setting	*Meaning*
DB2	Equivalent to PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTION. *Removed 8.0.*
ERROR_FOR_DIVISION_BY_ZERO	Throw error in strict mode, else warning, on division by zero in an INSERT or UPDATE. Otherwise server returns NULL. If used with IGNORE the server generates a warning. Deprecated, s*ince 5.6.17, does nothing.*
HIGH_NOT_PRECEDENCE	Read NOT x BETWEEN a AND b as (NOT x) BETWEEN a AND b rather than NOT (x BETWEEN a AND b). Since 5.0.2
IGNORE_SPACE	No limit on number of spaces between a function name and the first left parenthesis (. Therefore all function names become reserved words, so to access a database, table or column whose name is reserved, the name must be quoted
MAXDB	Compatibility with MAXDB, equivalent to PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTIONS, NO_AUTO_CREATE_USER. *Removed 8.0.*.
MSSQL	Compatibility with MSSQL (since 4.1.1), same as DB2. *Removed 8.0.*
MYSQL323	Compatibility with MySQL 3.23, equivalent to NO_FIELD_OPTIONS, HIGH_NOT_PRECEDENCE. *Removed 8.0.*
MYSQL40	Compatibility with MySQL 4.0, equivalent to NO_FIELD_OPTIONS, HIGH_NOT_PRESENCE *Removed 8.0.*
NO_AUTO_CREATE_USER	GRANT is not to automatically create user. DEPRECATED in 5.7.6.
NO_AUTO_VALUE_ON_ZERO	Assigning auto_increment value of 0 does not insert the next sequential value. Not recommended. (Since 4.1.1)
NO_BACKSLASH_ESCAPES	'\' does not escape
NO_DIR_IN_CREATE	Omit directory options from the output of SHOW CREATE TABLE. (Since 4.1.1)
NO_ENGINE_SUBSTITUTION	Do not substitute engine when requested storage engine is not available
NO_FIELD_OPTIONS	Omit column options from the output of SHOW CREATE TABLE. *Removed 8.0.*
NO_KEY_OPTIONS	Omit key options from the output of SHOW CREATE TABLE. *Removed 8.0.*
NO_TABLE_OPTIONS	Omit table options from the output of SHOW CREATE TABLE. *Removed 8.0.*
NO_UNSIGNED_SUBTRACTION	Undoes default handling of UNSIGNED ints in subtractions, where such subtractions always yield UNSIGNED results.
NO_ZERO_DATE	Reject the date `'0000-00-00" unless using IGNORE. Deprecated, s*ince 5.6.17 does nothing.*
NO_ZERO_IN_DATE	Reject dates with month=0 or day=0. If IGNORE is used, set date to '0000-00-00'. Deprecated, s*ince 5.6.17 does nothing.*
ONLY_FULL_GROUP_BY	GROUP BY columns or expressions must appear in the SELECT list. As of 5.1.1, includes HAVING columns.
ORACLE	Compatibility with Oracle, equivalent to DB2. *Removed 8.0.*
PAD_CHAR_TO_FULL_LENGTH	Pad CHAR values to full length. Since 5.0.51, 5.1.20. Deprecated 8.0.13.
PIPES_AS_CONCAT	|| is a string concatenation operator, not a synonym for OR
POSTGRESQL	Compatibility with PostgreSql, Equivalent to PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTIONS. *Removed 8.0.*
REAL_AS_FLOAT	REAL means FLOAT, not DOUBLE
STRICT_ALL_TABLES	All engines: (1) reject all invalid data including missing values for columns without default values; in non-transactional tables, pre-eror updates are not not rolled back (since 5.02). (2) creating VARCHAR and VARBINARY columns > 65,535 chars throws an error. Since 5.6.17 enforces ERROR_FOR_DIVISION_BY_ZERO, NO_ZERO_DATE, NO_ZERO_IN_DATE
STRICT_TRANS_TABLES (default since 5.7.5)	Transactional engines: (1) reject all invalid data including missing values for columns without default values; other engines: convert invalid data to closest possible valid data and issue warning. (2) creating VARCHAR and VARBINARY columns > 65,535 chars throws error. Since 5.6.17 includes ERROR_FOR_DIVISION_BY_ZERO, NO_ZERO_DATE, NO_ZERO_IN_DATE
TRADITIONAL	Equivalent to ERROR_FOR_DIVISION_BY_ZERO, NO_ZERO_IN_DATE, NO_ZERO_DATE, STRICT_ALL_TABLES, STRICT_TRANS_TABLES. In 5.4.4 NO_ENGINE_SUBSTITUTION was added.
""	Unset all sql_mode settings

sql_mode=ansi

Starting the server with `--ansi` or `--sql-mode=ANSI`, or (since version 4.1.1) issuing the SQL command

```
SET GLOBAL | LOCAL sql_mode = ANSI;
```

has the same effect as running the server with

```
--sql-mode="REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE"
--transaction-isolation=SERIALIZABLE
```

or (since version 4.1) issuing these commands:

```
SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET GLOBAL sql_mode =
"REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE";
```

as described in Table B2. Before 5.03, `sql_mode=ansi` also (incorrectly) enforced ONLY_FULL_GROUP_BY.