Category	Version	Type	Change	Note
SQL_MODE	8.0.11	Incompatible Change	These deprecated compatibility SQL modes have been removed: DB2, MAXDB, MSSQL, MYSQL323, MYSQL40, ORACLE, POSTGRESQL, NO_FIELD_OPTIONS, NO_KEY_OPTIONS, NO_TABLE_OPTIONS. They can no longer be assigned to the sql_mode system variable or used as permitted values for the mysqldump --compatible option.	For MySQL 5.7 applications that use SQL modes removed in MySQL 8.0, statements may fail when replicated from a MySQL 5.7 master to a MySQL 8.0 slave, or may have different effects on master and slave. To avoid such problems, applications that use modes removed in MySQL 8.0 should be revised to avoid them.
AUTH	8.0.11	Deprecate	Using GRANT to create users. Instead, use CREATE USER. Following this practice makes the NO_AUTO_CREATE_USER SQL mode immaterial for GRANT statements, so it too is removed.	Additionally, because IDENTIFIED BY PASSWORD syntax has been removed, the log_builtin_as_identified_by_password system variable is superfluous and has been removed.
AUTH	8.0.11	Deprecate	The old_passwords **system variable** is no longer available.	MySQL 5.7.2 よりも前では、Password カラム内のパスワードハッシュ形式に応じて、 サーバーは暗黙的に mysql_native_password または mysql_old_password プラグインを使用します。 Password 値が空または 4.1 のパスワードハッシュ (41 文字) である場合、サーバーはmysql_native_passwordを使用します。 パスワード値が 4.1 より前のパスワードハッシュ(16 文字) の場合、サーバーは mysql_old_password を使用します https://dev.mysql.com/doc/refman/5.6/ja/account-upgrades.html
Information_schema	8.0.11	Newly Add	The new **INFORMATION_SCHEMA**.KEYWORDS table lists the words considered keywords by MySQL and, for each one, indicates whether it is reserved. T	KEYWORDS テーブルには、MySQL で考慮されるキーワードがリストされ、各キーワードについて予約されているかどうかが示されます予約済キーワードは、識別子として使用される場合の特別な引用符など、一部のコンテキストで特別な処理が必要になることがあります(セクション9.3「キーワードと予約語」 を参照)。 https://dev.mysql.com/doc/refman/8.0/ja/information-schema-keywords-table.html
Logging	8.0.11	Newly Add	Messages written to the **error log** now indicate the subsystem in which the event occurred. Possible subsystem values are InnoDB (the InnoDB storage engine), Repl (the replication subsystem), Server (otherwise).	[root@data]# tail -n 10 MySQL8.err 2022-10-16T23:57:22.549555Z 0 [Warning] [MY-010918] [Server] 'default_authentication_plugin' is deprecated and will be removed in a future release. Please use authentication_policy instead. 2022-10-16T23:57:22.549573Z 0 [System] [MY-010116] [Server] /usr/local/mysql/bin/mysqld (mysqld 8.0.30) starting as process 2282 2022-10-16T23:57:22.609000Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started. 2022-10-16T23:57:23.282852Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended. 2022-10-16T23:57:23.883260Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed. 2022-10-16T23:57:23.883312Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel. 2022-10-16T23:57:23.952189Z 6 [Warning] [MY-013140] [Server] Error in diagnostics area: MY-001681 - Updating 'collation_database' is deprecated. It will be made read-only in a future release. 2022-10-16T23:57:23.952224Z 6 [Warning] [MY-013140] [Server] Error in diagnostics area: MY-001681 - Updating 'default_collation_for_utf8mb4' is deprecated. It will be made read-only in a future release. 2022-10-16T23:57:23.953089Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /tmp/mysqlx.sock 2022-10-16T23:57:23.953176Z 0 [System] [MY-010931] [Server] /usr/local/mysql/bin/mysqld: ready for connections. Version: '8.0.30' socket: '/tmp/mysql.sock' port: 3306 MySQL Community Server - GPL. [root@]#
Performance Schema	8.0.11	Newly Add	A new **Performance Schema** table log_status provides information that enables an online backup tool to copy the required log files without locking those resources for the duration of the copy process. When the log_status table is queried, the server blocks logging and related administrative changes for just long enough to populate the table, then releases the resources. The log_status table informs the online backup which point it should copy up to in the master's binary log and gtid_executed record, and the relay log for each replication channel.	mysql> select * from log_status limit 1\G *************************** 1. row *************************** SERVER_UUID: 5e3c1de7-5cc1-11ec-8b25-0242ac120002 LOCAL: {"gtid_executed": "", "binary_log_file": "binlog.000033", "binary_log_position": 17823} REPLICATION: {"channels": []} STORAGE_ENGINES: {"InnoDB": {"LSN": 134345815, "LSN_checkpoint": 134345815}} 1 row in set (0.00 sec) https://dev.mysql.com/doc/refman/8.0/en/performance-schema-log-status-table.html
Buffer Pool	8.0.11	Newly Add	To improve startup performance on systems with large **buffer pools**, buffer pool initialization is now multithreaded.	
SHOW CREATE TABLE	8.0.11	Newly Add	SHOW CREATE TABLE normally does not show the ROW_FORMAT table option if the row format is the default format. This can cause problems during table import and export operations for transportable tablespaces. MySQL now supports a show_create_table_verbosity system variable that, when enabled, causes SHOW CREATE TABLE to display ROW_FORMAT regardless of whether it is the default format.	mysql> show variables like "show_create_table_verbosity"; +-----------------------------+-------+ | Variable_name | Value | +-----------------------------+-------+ | show_create_table_verbosity | OFF | +-----------------------------+-------+ 1 row in set (0.00 sec) mysql> show create table t1\G *************************** 1. row *************************** Table: t1 Create Table: CREATE TABLE `t1` (`c1` int NOT NULL, PRIMARY KEY (`c1`)) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci 1 row in set (0.03 sec) mysql> set show_create_table_verbosity=ON; Query OK, 0 rows affected (0.00 sec) mysql> show create table t1\G *************************** 1. row *************************** Table: t1 Create Table: CREATE TABLE `t1` (`c1` int NOT NULL, PRIMARY KEY (`c1`)) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci **ROW_FORMAT=DYNAMIC** 1 row in set (0.00 sec) https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_show_create_table_verbosity

ALTER TABLE	8.0.12	Newly Add	InnoDB now supports ALGORITHM=INSTANT for the following ALTER TABLE operations.	Adding a column. This feature is referred to as “Instant ADD COLUMN”. Adding or dropping a virtual column. Adding or dropping a column default value. Modifying the definition of an ENUM or SET column. Changing the index type. Renaming a table. INSTANT: Operations only modify metadata in the data dictionary. An exclusive metadata lock on the table may be taken briefly during the execution phase of the operation. Table data is unaffected, making operations instantaneous. Concurrent DML is permitted. (Introduced in MySQL 8.0.12) Thanks to the Tencent Games DBA team for the contribution. https://dev.mysql.com/doc/refman/8.0/ja/alter-table.html mysql> desc T1; +-------+-------------+------+-----+---------+----------------+ | Field | Type | Null | Key | Default | Extra | +-------+-------------+------+-----+---------+----------------+ | id | int | NO | PRI | NULL | auto_increment | | note | varchar(10) | YES | | NULL | | | who | varchar(100) | YES | | NULL | | +-------+-------------+------+-----+---------+----------------+ 3 rows in set (0.01 sec) mysql> alter table T1 add column instant varchar(100), algorithm=inplace, lock=none; Query OK, 0 rows affected (0.15 sec) Records: 0 Duplicates: 0 Warnings: 0 mysql> desc T1; +---------+-------------+------+-----+---------+----------------+ | Field | Type | Null | Key | Default | Extra | +---------+-------------+------+-----+---------+----------------+ | id | int | NO | PRI | NULL | auto_increment | | note | varchar(10) | YES | | NULL | | | who | varchar(100) | YES | | NULL | | | instant | varchar(100) | YES | | NULL | | +---------+-------------+------+-----+---------+----------------+ 4 rows in set (0.01 sec) mysql>
Account Management	8.0.13	Newly Add	It is now possible to require that attempts to change an account password be verified by specifying the current password to be replaced. This enables DBAs to prevent users from changing a password without proving that they know the current password. It is possible to establish password-verification policy globally using the password_require_current system variable, as well as on a per-account basis using the PASSWORD REQUIRE option of the CREATE USER and ALTER USER statements.	mysql> select Host,user,Password_require_current from mysql.user limit 1; +------+-------+--------------------------+ | Host | user | Password_require_current | +------+-------+--------------------------+ | % | admin | NULL | +------+-------+--------------------------+ 1 row in set (0.00 sec) mysql> show global variables like "password_require_current"; +--------------------------+-------+ | Variable_name | Value | +--------------------------+-------+ | password_require_current | OFF | +--------------------------+-------+ 1 row in set (0.04 sec)
SQL_MODE	8.0.13	Newly Add	The new sql_require_primary_key system variable makes it possible to have statements that create new tables or alter the structure of existing tables enforce the requirement that tables have a primary key. Enabling this variable helps avoid performance problems in row-based replication that can occur when tables have no primary key.	mysql> show global variables like "sql_require_primary_key"; +-------------------------+-------+ | Variable_name | Value | +-------------------------+-------+ | sql_require_primary_key | OFF | +-------------------------+-------+ 1 row in set (0.00 sec) 新しいテーブルを作成するステートメントまたは既存のテーブルの構造を変更するステートメントが、テーブルに主キーがあるという要件を強制するかどうか。sql_require_primary_key は実テーブルと TEMPORARY テーブルの両方に適用され、その値に対する変更はレプリカサーバーにレプリケートされますMySQL 8.0.18 では、レプリケーションに参加できるストレージエンジンにのみ適用されます。 https://dev.mysql.com/doc/refman/8.0/ja/server-system-variables.html
InnoDB; Partitioning:	8.0.13	Incompatible Change	InnoDB; Partitioning: Support for placing table partitions in shared tablespaces was removed. Shared tablespaces include the system tablespace and general tablespaces. For information about identifying partitions in shared tablespaces and moving them to file-per-table tablespaces, see Preparing Your Installation for Upgrade.	2.11.5 アップグレード用のインストールの準備 https://dev.mysql.com/doc/refman/8.0/ja/upgrade-prerequisites.html
TEMPORARY TABLE	8.0.13	Deprecate	CREATE TEMPORARY TABLE での TABLESPACE = innodb_file_per_table 句および TABLESPACE = innodb_temporary 句のサポートは、MySQL 8.0.13 で非推奨になりました。MySQL の将来のバージョンで削除される予定です。	mysql> CREATE TEMPORARY TABLE new_tbl SELECT * FROM t1; Query OK, 0 rows affected (0.02 sec) Records: 0 Duplicates: 0 Warnings: 0 mysql> CREATE TEMPORARY TABLE new_tbl2 TABLESPACE = innodb_file_per_table SELECT * FROM t2 ; ERROR 1478 (HY000): InnoDB: TABLESPACE=innodb_file_per_table option is disallowed for temporary tables with INNODB_STRICT_MODE=ON. This option is deprecated and will be removed in a future release https://dev.mysql.com/doc/refman/8.0/ja/create-temporary-table.html

<table>
<tr><td>character</td><td>8.0.13</td><td>Deprecate</td><td>The utf8mb3 character set is deprecated and will be removed in a future MySQL version. Please use utf8mb4 instead.</td><td>mysql> CREATE TABLE `t_utf8mb4` (
    ->   `c1` int NOT NULL,
  PRI   ->   PRIMARY KEY (`c1`)
    ->) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE `t_utf8mb3` (
    ->   `c1` int NOT NULL,
    ->   PRIMARY KEY (`c1`)
    ->) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
Query OK, 0 rows affected, 1 warning (0.06 sec)

mysql> show warnings;
+---------+------+---+
| Level | Code | Message |
+---------+------+---+
| Warning | 1287 | 'utf8mb3' is deprecated and will be removed in a future release. Please use utf8mb4 instead |
+---------+------+---+
1 row in set (0.00 sec)

mysql> CREATE TABLE `t_utf8` (
 `c1` int NO   ->   `c1` int NOT NULL,
RIMARY K   ->   PRIMARY KEY (`c1`)
    ->) ENGINE=InnoDB DEFAULT CHARSET=utf8;
Query OK, 0 rows affected, 1 warning (0.06 sec)

mysql> show warnings;
+---------+------
+---
------+
| Level | Code | Message
                 |
+---------+------
+---
------+
| Warning | 3719 | 'utf8' is currently an alias for the character set UTF8MB3, but will be an alias for UTF8MB4 in a future release. Please consider using UTF8MB4 in order to be unambiguous. |
+---------+------
+---
------+
1 row in set (0.00 sec)

mysql></td></tr>
<tr><td>INFORMATION_SCHEMA</td><td>8.0.13</td><td>Newly Add</td><td>select * from information_schema.
VIEW_ROUTINE_USAGE;
select * from information_schema.
VIEW_TABLE_USAGE;</td><td>provides information about stored functions used in view definitions.
provides information about tables and views used in view definitions.</td></tr>
<tr><td>Replication</td><td>8.0.13</td><td>Incompatible Change</td><td>Previously, CREATE TEMPORARY TABLE and DROP TEMPORARY TABLE statements were not supported inside transactions, procedures, functions, or triggers when using GTIDs (that is, when the enforce_gtid_consistency system variable is set to ON). It was possible to use these statements with GTIDs enabled, but only outside of any transaction, and only with autocommit=1.

From MySQL 8.0.13, this restriction has been removed when binlog_format is set to ROW or MIXED. With row-based logging in use, CREATE TEMPORARY TABLE and DROP TEMPORARY TABLE statements can now be used inside transactions, procedures, functions, or triggers when GTIDs are enabled. When binlog_format is set to STATEMENT, the restriction remains. Because of this difference in behavior, some additional restrictions now apply to changing the binlog_format setting at runtime:</td><td>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).</td></tr>
</table>

Replication	8.0.13	Incompatible Change	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 @@PERSIST_ONLY.binlog_format) to change the replication format at any time, because this action does not modify the runtime global system variable value, and takes effect only after a server restart.	SET PERSIST および SET PERSIST_ONLY を使用すると、グローバルシステム変数をデータディレクトリ内のmysqld-auto.cnf オプションファイルに永続化できます(セクション13.7.6.1「変数代入の SET 構文」 を参照)。 ただし、すべてのシステム変数を永続化できるわけではありません。または、特定の制限条件下でのみ永続化できるわけではありません。システム変数が永続的または永続的に制限されない理由を次に示します セッションシステム変数は永続化できません。セッション変数はサーバーの起動時に設定できないため、永続化する理由はありません。 グローバルシステム変数には、サーバーホストへの直接アクセス権を持つユーザーのみが設定できるような機密データが含まれる場合があります。 グローバルシステム変数は読取り専用の場合があります(つまり、サーバーによってのみ設定されます。 この場合、サーバーの起動時でも実行時でも、ユーザーが設定することはできません。 グローバルシステム変数は、内部使用のみを目的としている場合があります。 永続的でないシステム変数は、どのような状況でも永続化できません。MySQL 8.0.14 の時点では、永続制限付きシステム変数はSET PERSIST_ONLY で永続化できますが、次の条件を満たすユーザーのみが永続化できます persist_only_admin_x509_subject システム変数は、SSL 証明書の X.509 サブジェクト値に設定されます。 ユーザーは暗号化された接続を使用してサーバーに接続し、指定されたサブジェクト値をSSL 証明書を提供します。 ユーザーには、SET PERSIST_ONLY を使用するための十分な権限があります(セクション5.1.9.1「システム変数権限」 を参照)。 たとえば、protocol_version は読取り専用であり、サーバーによってのみ設定されるため、どのような状況でも永続化できません。一方、bind_address は永続的に制限されるため、前述の条件を満たすユーザーが設定できます。 https://dev.mysql.com/doc/refman/8.0/ja/nonpersistible-system-variables.html
TEMPORARY TABLE	8.0.13	Deprecate	The TempTable storage engine now supports storage of binary large object (BLOB) type columns. This enhancement improves performance for queries that use temporary tables containing BLOB data. Previously, temporary tables that contained BLOB data were stored in the on-disk storage engine defined by internal_tmp_disk_storage_engine.	internal_tmp_disk_storage_engine Prior to MySQL 8.0.16, this variable determines the storage engine used for on-disk internal temporary tables (see Storage Engine for On-Disk Internal Temporary Tables). Permitted values are MYISAM and INNODB (the default). In MySQL 8.0.16 and later, on-disk internal temporary tables always use the InnoDB storage engine; as of MySQL 8.0.16, this variable has been removed and is thus no longer supported. https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_internal_tmp_disk_storage_engine
RENAME TABLE	8.0.13	Newly Add	Previously, executing RENAME TABLE required that there be no tables locked with LOCK TABLES. Now it is possible to rename tables that are locked with a WRITE lock or that are the product of renaming WRITE-locked tables from earlier steps in a multiple-table rename operation.	As of MySQL 8.0.13, you can rename tables locked with a LOCK TABLES statement, provided that they are locked with a WRITE lock or are the product of renaming WRITE-locked tables from earlier steps in a multiple-table rename operation. mysql> LOCK TABLE t1 WRITE; Query OK, 0 rows affected (0.01 sec) mysql> RENAME TABLE t1 TO t2; Query OK, 0 rows affected (0.10 sec) mysql> unlock tables; Query OK, 0 rows affected (0.00 sec) https://dev.mysql.com/doc/refman/8.0/en/rename-table.html --------------------------------- --------------------------------- mysql> set session lock_wait_timeout = 10; Query OK, 0 rows affected (0.01 sec) mysql> show variables like '%lock_wait_timeout%'; +--------------------------+-------+ | Variable_name | Value | +--------------------------+-------+ | innodb_lock_wait_timeout | 50 | | lock_wait_timeout | 10 | +--------------------------+-------+ 2 rows in set (0.01 sec) mysql> LOCK TABLE t1 WRITE; ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction mysql>
PARAMETER	8.0.13	Newly Add	**innodb_fsync_threshold** Specifying a threshold to force smaller, periodic flushes may be beneficial in cases where multiple MySQL instances use the same storage devices. For example, creating a new MySQL instance and its associated data files could cause large surges of disk write activity, impeding the performance of other MySQL instances that use the same storage devices. Configuring a threshold helps avoid such surges in write activity. (Bug #27724600)	デフォルトでは、InnoDB が新しいログファイルやテーブルスペースファイルなどの新しいデータファイルを作成すると、ファイルはディスクにフラッシュされる前にオペレーティングシステムキャッシュに完全に書き込まれるため、大量のディスク書込みアクティビティが一度に発生する可能性があります。オペレーティングシステムキャッシュから定期的にデータを強制的に小さいフラッシュするには、innodb_fsync_threshold 変数を使用してしきい値をバイト単位で定義します。バイトしきい値に達すると、オペレーティングシステムキャッシュの内容がディスクにフラッシュされますデフォルト値の 0 では、デフォルトの動作が強制されます。つまり、ファイルがキャッシュに完全に書き込まれた後にのみ、データがディスクにフラッシュされます。 https://dev.mysql.com/doc/refman/8.0/ja/innodb-parameters.html#sysvar_innodb_fsync_threshold

TEMPORARY TABLE	8.0.13	Improvement	InnoDB: User-created temporary tables and internal temporary tables created by the optimizer are now stored in session temporary tablespaces that are allocated to a session from a pool of temporary tablespaces. When a session disconnects, its temporary tablespaces are truncated and released back to the pool. In previous releases, temporary tables were created in the global temporary tablespace (ibtmp1), which did not return disk space to the operating system after temporary tables were dropped.	The innodb_temp_tablespaces_dir variable defines the location where session temporary tablespaces are created. The default location is the #innodb_temp directory in the data directory. The INNODB_SESSION_TEMP_TABLESPACES table provides metadata about session temporary tablespaces. The **global temporary tablespace (ibtmp1)** now stores rollback segments for changes made to user-created temporary tables. mysql> show variables like 'innodb_temp_tablespaces_dir'; +------------------------------+-----------------+ | Variable_name | Value | +------------------------------+-----------------+ | innodb_temp_tablespaces_dir | ./#innodb_temp/ | +------------------------------+-----------------+ 1 row in set (0.08 sec) mysql> desc information_schema.INNODB_SESSION_TEMP_TABLESPACES; +---------+-----------------+------+-----+---------+-------+ | Field | Type | Null | Key | Default | Extra | +---------+-----------------+------+-----+---------+-------+ | ID | int unsigned | NO | | | | | | SPACE | int unsigned | NO | | | | | | PATH | varchar(4001) | NO | | | | | | SIZE | bigint unsigned | NO | | | | | | STATE | varchar(192) | NO | | | | | | PURPOSE | varchar(192) | NO | | | | | +---------+-----------------+------+-----+---------+-------+ 6 rows in set (0.09 sec) mysql> select * from information_schema.INNODB_SESSION_TEMP_TABLESPACES; +----+------------+----------------------------+-------+----------+-----------+ | ID | SPACE | PATH | SIZE | STATE | PURPOSE | +----+------------+----------------------------+-------+----------+-----------+ | 14 | 4243767290 | ./#innodb_temp/temp_10.ibt | 81920 | ACTIVE | INTRINSIC | | 0 | 4243767281 | ./#innodb_temp/temp_1.ibt | 81920 | INACTIVE | NONE | | 0 | 4243767282 | ./#innodb_temp/temp_2.ibt | 81920 | INACTIVE | NONE | | 0 | 4243767283 | ./#innodb_temp/temp_3.ibt | 81920 | INACTIVE | NONE | | 0 | 4243767284 | ./#innodb_temp/temp_4.ibt | 81920 | INACTIVE | NONE | | 0 | 4243767285 | ./#innodb_temp/temp_5.ibt | 81920 | INACTIVE | NONE | | 0 | 4243767286 | ./#innodb_temp/temp_6.ibt | 81920 | INACTIVE | NONE | | 0 | 4243767287 | ./#innodb_temp/temp_7.ibt | 81920 | INACTIVE | NONE | | 0 | 4243767288 | ./#innodb_temp/temp_8.ibt | 81920 | INACTIVE | NONE | | 0 | 4243767289 | ./#innodb_temp/temp_9.ibt | 81920 | INACTIVE | NONE | +----+------------+----------------------------+-------+----------+-----------+ 10 rows in set (0.01 sec)
ENCRYPTION	8.0.13	Improvement	InnoDB: The InnoDB data-at-rest encryption feature now supports general tablespaces. Previously, only file-per-table tablespaces could be encrypted. To support encryption of general tablespaces, CREATE TABLESPACE and ALTER TABLESPACE syntax was extended to include an ENCRYPTION clause. The INFORMATION_SCHEMA.INNODB_TABLESPACES table now includes an ENCRYPTION column that indicates whether or not a tablespace is encrypted. The stage/innodb/alter tablespace (encryption) Performance Schema stage instrument was added to permit monitoring of general tablespace encryption	mysql> select SPACE,NAME,ROW_FORMAT,ZIP_PAGE_SIZE,SPACE_TYPE,FS_BLOCK_SIZE,SERVER_VERSION,ENCRYPTION,STATE from INFORMATION_SCHEMA.INNODB_TABLESPACES limit 1\G *************************** 1. row *************************** SPACE: 4294967294 NAME: mysql ROW_FORMAT: Any ZIP_PAGE_SIZE: 0 SPACE_TYPE: General FS_BLOCK_SIZE: 4096 SERVER_VERSION: 8.0.27 ENCRYPTION: N STATE: normal 1 row in set (0.00 sec)

TEMPORARY TABLE	8.0.13	Improvement	Previously, CREATE TEMPORARY TABLE and DROP TEMPORARY TABLE statements were not supported inside transactions, procedures, functions, or triggers when using GTIDs (that is, when the enforce_gtid_consistency system variable is set to ON). It was possible to use these statements with GTIDs enabled, but only outside of any transaction, and only with autocommit=1. From MySQL 8.0.13, this restriction has been removed when binlog_format is set to ROW or MIXED. With row-based logging in use, CREATE TEMPORARY TABLE and DROP TEMPORARY TABLE statements can now be used inside transactions, procedures, functions, or triggers when GTIDs are enabled. When binlog_format is set to STATEMENT, the restriction remains. Because of this difference in behavior, some additional restrictions now apply to changing the binlog_format setting at runtime: 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 @@PERSIST_ONLY.binlog_format) to change the replication format at any time, because this action does not modify the runtime global system variable value, and takes effect only after a server restart.	mysql> START TRANSACTION;CREATE TEMPORARY TABLE CONFIRM.item_genres(`genre_id` INT PRIMARY KEY, `genre_name` TEXT); Query OK, 0 rows affected (0.00 sec) Query OK, 0 rows affected (0.00 sec) mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_TEMP_TABLE_INFO; +----------+------------+--------+------------+ | TABLE_ID | NAME | N_COLS | SPACE | +----------+------------+--------+------------+ | 1220 | #sql1_e_28 | 5 | 4243767289 | +----------+------------+--------+------------+ 1 row in set (0.00 sec) mysql> ROLLBACK; Query OK, 0 rows affected, 1 warning (0.00 sec) mysql> show warnings; +---------+------+--+ | Level | Code | Message | +---------+------+--+ | Warning | 1751 | The creation of some temporary tables could not be rolled back. | +---------+------+--+ 1 row in set (0.00 sec) mysql> DROP TABLE CONFIRM.item_genres; Query OK, 0 rows affected (0.00 sec) mysql>
AUTH	8.0.14	Newly Add	Previously, each MySQL user account was permitted to have a single password. MySQL now permits an account to have dual passwords, designated as primary and secondary passwords. This capability enables phased password changes to be performed seamlessly in complex multiple-server systems, without downtime. To support dual-password capability, the ALTER USER and SET PASSWORD statements now have a RETAIN CURRENT PASSWORD clause that saves the current password as the secondary password when you assign an account a new primary password. ALTER USER also has a DISCARD OLD PASSWORD clause to discard a secondary password that is no longer needed. See Password Management.	レプリカではない各サーバーで、'password_b'を新しい appuser1 プライマリパスワードとして確立し、現在のパスワードをセカンダリパスワードとして保持します ALTER USER 'appuser1'@'host1.example.com' IDENTIFIED BY 'password_b' RETAIN CURRENT PASSWORD; パスワード変更がシステム全体ですべてのレプリカにレプリケートされるのを待ちます。 'password_a'ではなく'password_b'のパスワードを使用してサーバーに接続するようにappuser1 アカウントを使用する各アプリケーションを変更します。 この時点で、セカンダリパスワードは不要になりました。レプリカではない各サーバーで、セカンダリパスワードを破棄します ALTER USER 'appuser1'@'host1.example.com' DISCARD OLD PASSWORD; https://dev.mysql.com/doc/refman/8.0/ja/password-management.html

AUTH	8.0.14	Newly Add	MySQL Server now permits a TCP/IP port to be configured specifically for administrative connections. This provides an alternative to the single administrative connection that is permitted on the network interfaces used for ordinary connections even when max_connections connections are already established. The administrative network interface has these characteristics: The interface is enabled only if the admin_address system variable is set at startup to indicate the IP address for it. If admin_address is not set, the server maintains no administrative interface. The admin_port system variable specifies the interface TCP/IP port number (default 33062). There is no limit on the number of administrative connections, but connections are permitted only for users who have the SERVICE_CONNECTION_ADMIN privilege. The create_admin_listener_thread system variable enables DBAs to choose at startup whether the administrative interface has its own separate thread. The default is OFF; that is, the manager thread for ordinary connections on the main interface also handles connections for the administrative interface.	管理ネットワークインタフェースでTCP/IP 接続をリスニングする IP アドレス (セクション5.1.12.1「接続インタフェース」 を参照)。 デフォルトの admin_address 値はありません。 この変数が起動時に指定されない場合、サーバーは管理インタフェースを維持しません。サーバーには、通常の (非管理) クライアント TCP/IP 接続を構成するための bind_address システム変数もあります。セクション5.1.12.1「接続インタフェース」を参照してください。 https://dev.mysql.com/doc/refman/8.0/ja/server-system-variables.html#sysvar_admin_address
UNDO	8.0.14	Newly Add	InnoDB: Disabling the innodb_buffer_pool_in_core_file variable reduces the size of core files by excluding InnoDB buffer pool pages. To use this variable, the core_file variable must be enabled and the operating system must support the MADV_DONTDUMP non-POSIX extension to madvise(), which is supported in Linux 3.4 and later. For more information, see Excluding Buffer Pool Pages from Core Files. Thanks to Facebook for the contribution. (Bug #27724476, Bug #90144) InnoDB: By default, undo logs reside in two undo tablespaces that are created when the MySQL instance is initialized. Additional undo tablespaces can be created in a chosen location at runtime using CREATE UNDO TABLESPACE syntax. CREATE UNDO TABLESPACE tablespace_name ADD DATAFILE 'file_name.ibu'; Undo tablespaces created using CREATE UNDO TABLESPACE syntax can be dropped at runtime using DROP UNDO TABLESPACE syntax. DROP UNDO TABLESPACE tablespace_name; ALTER UNDO TABLESPACE syntax can be used to mark an undo tablespace as active or inactive. ALTER UNDO TABLESPACE tablespace_name SET {ACTIVE|INACTIVE}; A STATE column that shows the state of a tablespace was added to the INFORMATION_SCHEMA.INNODB_TABLESPACES table. An undo tablespace must be in an empty state before it can be dropped. The previously deprecated innodb_undo_tablespaces variable is no longer configurable and will be removed in a future MySQL version. For more information, see Undo Tablespaces.	mysql> SELECT TABLESPACE_NAME, FILE_NAME FROM INFORMATION_SCHEMA.FILES -> WHERE FILE_TYPE LIKE 'UNDO LOG'; +-----------------+------------+ | TABLESPACE_NAME | FILE_NAME | +-----------------+------------+ | innodb_undo_001 | ./undo_001 | | innodb_undo_002 | ./undo_002 | +-----------------+------------+ 2 rows in set (0.01 sec) mysql> CREATE UNDO TABLESPACE tablespace_name ADD DATAFILE 'innodb_undo_manual001.ibu'; Query OK, 0 rows affected (0.42 sec) mysql> SELECT TABLESPACE_NAME, FILE_NAME FROM INFORMATION_SCHEMA.FILES WHERE FILE_TYPE LIKE 'UNDO LOG'; +-----------------+----------------------------+ | TABLESPACE_NAME | FILE_NAME | +-----------------+----------------------------+ | innodb_undo_001 | ./undo_001 | | innodb_undo_002 | ./undo_002 | | tablespace_name | ./innodb_undo_manual001.ibu | +-----------------+----------------------------+ 3 rows in set (0.00 sec) mysql> select * from INFORMATION_SCHEMA.INNODB_TABLESPACES where SPACE_TYPE = 'Undo'; +-----------+-----------------+------+------------+-----------+---------------+------------+---------------+-----------+----------------+----------------- +---------------+--------------+-----------+--------+ | SPACE | NAME | FLAG | ROW_FORMAT | PAGE_SIZE | ZIP_PAGE_SIZE | SPACE_TYPE | FS_BLOCK_SIZE | FILE_SIZE | ALLOCATED_SIZE | AUTOEXTEND_SIZE | SERVER_VERSION | SPACE_VERSION | ENCRYPTION | STATE | +-----------+-----------------+------+------------+-----------+---------------+------------+---------------+-----------+----------------+----------------- +---------------+--------------+-----------+--------+ | 4294967279 | innodb_undo_001 | 0 | Undo | 16384 | 0 | Undo | 4096 | 16777216 | 6213632 | 0 | 8.0.27 | 1 | N | active | | 4294967278 | innodb_undo_002 | 0 | Undo | 16384 | 0 | Undo | 4096 | 16777216 | 9117696 | 0 | 8.0.27 | 1 | N | active | | 4294967277 | tablespace_name | 0 | Undo | 16384 | 0 | Undo | 4096 | 16777216 | 16777216 | 0 | 8.0.27 | 1 | N | active | +-----------+-----------------+------+------------+-----------+---------------+------------+---------------+-----------+----------------+----------------- +---------------+--------------+-----------+--------+ 3 rows in set (0.00 sec)

INNODB	8.0.14	Newly Add	InnoDB: InnoDB now supports parallel clustered index reads, which can improve CHECK TABLE performance. This feature does not apply to secondary index scans. The innodb_parallel_read_threads session variable must be set to a value greater than 1 for parallel clustered index reads to occur. The default value is 4. The actual number of threads used to perform a parallel clustered index read is determined by the innodb_parallel_read_threads setting or the number of index subtrees to scan, whichever is smaller.	mysql> show variables like 'innodb_parallel_read_threads'; +------------------------------+-------+ | Variable_name | Value | +------------------------------+-------+ | innodb_parallel_read_threads | 4 | +------------------------------+-------+ 1 row in set (0.00 sec) パラレルクラスタインデックス読取りに使用できるスレッドの数を定義します。パーティションのパラレルスキャンは、MySQL 8.0.17 でサポートされています。パラレル読取りスレッドを使用すると、CHECK TABLE のパフォーマンスを向上できます。InnoDB は、CHECK TABLE 操作中にクラスタ化されたインデックスを2 回読み取ります。 2 番目の読取りはパラレルで実行できます。この機能は、セカンダリインデックススキャンには適用されません。 パラレルクラスタインデックス読取りを実行するにはinnodb_parallel_read_threads セッション変数を 1 より大きい値に設定する必要があります。パラレルクラスタインデックス読取りの実行に使用されるスレッドの実際の数は、innodb_parallel_read_threads 設定またはスキャンするインデックスサブツリーの数(いずれか小さい方) によって決まります。 スキャン中にバッファープールに読み取られたページは、空きバッファープールページが必要なときにすぐに破棄できるように、バッファープールLRU リストの末尾に保持されます。 MySQL 8.0.17 では、パラレル読取りスレッドの最大数(256) は、すべてのクライアント接続のスレッドの合計数です。スレッド制限に達すると、接続は単一スレッドの使用にフォールバックします。 https://dev.mysql.com/doc/refman/8.0/ja/innodb-parameters.html#sysvar_innodb_parallel_read_threads
INNODB	8.0.14	Deprecate	The previously deprecated innodb_undo_tablespaces variable is no longer configurable and will be removed in a future MySQL version.	mysql> show variables like 'innodb_undo_tablespaces'; +-------------------------+-------+ | Variable_name | Value | +-------------------------+-------+ | innodb_undo_tablespaces | 2 | +-------------------------+-------+ 1 row in set (0.00 sec)
ALTER TABLE	8.0.14	Newly Add	ALTER TABLE now can be used to change a column character set in place (without a table rebuild), when these conditions apply: The column data type is CHAR, VARCHAR, a TEXT type, or ENUM. The character set change is from utf8mb3 to utf8mb4, or any character set to binary. There is no index on the column.	mysql> show create table T1\G *************************** 1. row *************************** Table: T1 Create Table: CREATE TABLE `T1` (`id` int NOT NULL AUTO_INCREMENT COMMENT 'PGの場合はSEQUENCE', `note` varchar(10) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL DEFAULT '-' COMMENT '8.0.27', `who` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci DEFAULT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci 1 row in set (0.00 sec) mysql> insert into T1(note,who) values('10xutf8mb3','Size 100 with utf8mb4'); Query OK, 1 row affected (0.02 sec) mysql> select *,version() from T1; +----+------------+----------------------+-----------+ | id | note | who | version() | +----+------------+----------------------+-----------+ | 1 | 10xutf8mb3 | Size 100 with utf8mb4 | 8.0.27 | +----+------------+----------------------+-----------+ 1 row in set (0.00 sec) mysql> ALTER TABLE T1 MODIFY note varchar(10) CHARACTER SET utf8mb4, ALGORITHM=INPLACE, LOCK=NONE; Query OK, 0 rows affected (0.10 sec) Records: 0 Duplicates: 0 Warnings: 0 mysql> show create table T1\G *************************** 1. row *************************** Table: T1 Create Table: CREATE TABLE `T1` (`id` int NOT NULL AUTO_INCREMENT COMMENT 'PGの場合はSEQUENCE', `note` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci DEFAULT NULL, `who` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci DEFAULT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci 1 row in set (0.00 sec) mysql> ALTER TABLE T1 MODIFY note varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci, ALGORITHM=INPLACE, LOCK=NONE; Query OK, 0 rows affected (0.03 sec) Records: 0 Duplicates: 0 Warnings: 0 mysql> show create table T1\G *************************** 1. row *************************** Table: T1 Create Table: CREATE TABLE `T1` (`id` int NOT NULL AUTO_INCREMENT COMMENT 'PGの場合はSEQUENCE', `note` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci DEFAULT NULL, `who` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci DEFAULT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci 1 row in set (0.00 sec) mysql>

Account Management	8.0.16	Newly Add	Previously, it was not possible to grant privileges that apply globally except for certain schemas. This is now possible if the new partial_revokes system variable is enabled. For example, the following statements enable an account to select from or insert into any table except those in the mysql system schema:	mysql> CREATE USER u1@'%' identified by 'Password2022'; Query OK, 0 rows affected (0.02 sec) mysql> SELECT user,host from mysql.user where user = 'u1'; +------+------+ | user | host | +------+------+ | u1 | % | +------+------+ 1 row in set (0.00 sec) mysql> GRANT SELECT, INSERT ON *.* TO u1@'%'; Query OK, 0 rows affected (0.01 sec) mysql> REVOKE SELECT, INSERT ON mysql.* FROM u1@'%'; ERROR 1141 (42000): There is no such grant defined for user 'u1' on host '%' mysql> SET PERSIST partial_revokes = ON; Query OK, 0 rows affected (0.00 sec) mysql> REVOKE SELECT, INSERT ON mysql.* FROM u1@'%'; Query OK, 0 rows affected (0.02 sec) mysql> show grants for u1@'%'\G *************************** 1. row *************************** Grants for u1@%: GRANT SELECT, INSERT ON *.* TO `u1`@`%` *************************** 2. row *************************** Grants for u1@%: REVOKE SELECT, INSERT ON `mysql`.* FROM `u1`@`%` 2 rows in set (0.00 sec)
TEMPORARY TABLE	8.0.16	Deprecate	The TempTable storage engine now always uses InnoDB to manage internal temporary tables on disk, and the choice of storage engine employed for this task is no longer user-configurable. The internal_tmp_disk_storage_engine system variable has been removed. (Bug #91377, Bug #28234637)	mysql> show variables like 'internal_tmp_disk_storage_engine'; Empty set (0.00 sec) オンディスク内部一時テーブルのストレージエンジン MySQL 8.0.16 以降、サーバーは常に InnoDB ストレージエンジンを使用してディスク上の内部一時テーブルを管理します。 MySQL 8.0.15 以前では、internal_tmp_disk_storage_engine 変数を使用して、ディスク上の内部一時テーブルに使用されるストレージエンジンを定義していましたこの変数は MySQL 8.0.16 で削除され、この目的に使用されるストレージエンジンはユーザーが構成できなくなりました。 MySQL 8.0.15 以前: 共通テーブル式 (CTE) の場合、ディスク上の内部一時テーブルに使用されるストレージエンジンをMyISAM にすることはできません。internal_tmp_disk_storage_engine=MYISAM の場合、ディスク上の一時テーブルを使用してCTE を実体化しようとすると、エラーが発生します。 MySQL 8.0.15 以前: internal_tmp_disk_storage_engine=INNODB を使用している場合、InnoDB row or column limits を超えるディスク上の内部一時テーブルを生成するクエリーは、「行サイズが大きすぎます」または「カラムが多すぎます」エラーを返します。回避策は、internal_tmp_disk_storage_engine を MYISAM に設定することです。 https://dev.mysql.com/doc/refman/8.0/ja/internal-temporary-tables.html
DATA DICTIONARY	8.0.16	Newly Add	**mysql_upgrade is deprecated because it is no longer necessary.** The --no-dd-upgrade server option is deprecated because the --upgrade option supersedes it. Previously, after installation of a new version of MySQL, the MySQL server automatically upgraded the data dictionary tables at the next startup, after which the DBA was expected to invoke mysql_upgrade manually to upgrade the system tables in the mysql schema, as well as objects in other schemas such as the sys schema and user schemas. The server now performs the tasks previously handled by mysql_upgrade. After installation of a new MySQL version, the server now automatically performs all necessary upgrade tasks at the next startup and is not dependent on the DBA invoking mysql_upgrade. In addition, the server updates the contents of the help tables (something mysql_upgrade did not do). A new --upgrade server option provides control over how the server performs automatic data dictionary and server upgrade operations. For more information, see Upgrading MySQL.	This option controls whether and how the server performs an automatic upgrade at startup. Automatic upgrade involves two steps: Step 1: Data dictionary upgrade. This step upgrades: The data dictionary tables in the mysql schema. If the actual data dictionary version is lower than the current expected version, the server upgrades the data dictionary. If it cannot, or is prevented from doing so, the server cannot run. The Performance Schema and INFORMATION_SCHEMA. Step 2: Server upgrade. This step comprises all other upgrade tasks. If the existing installation data has a lower MySQL version than the server expects, it must be upgraded: The system tables in the mysql schema (the remaining non-data dictionary tables). The sys schema. User schemas. For details about upgrade steps 1 and 2, see Section 2.11.3, “What the MySQL Upgrade Process Upgrades”.

PLUGIN	8.0.16	Newly Add	MySQL now includes a ddl_rewriter plugin that modifies CREATE TABLE statements received by the server before it parses and executes them. The plugin removes ENCRYPTION, DATA DIRECTORY, and INDEX DIRECTORY clauses, which may be helpful when restoring tables from SQL dump files created from databases that are encrypted or that have their tables stored outside the data directory. For example, the plugin may enable restoring such dump files into an unencrypted instance or in an environment where the paths outside the data directory are not accessible. When installed, ddl_rewriter exposes the Performance Schema memory/rewriter/ddl_rewriter instrument for tracking plugin memory use. For more information, see The ddl_rewriter Plugin.	mysql> SELECT PLUGIN_NAME, PLUGIN_STATUS, PLUGIN_TYPE FROM INFORMATION_SCHEMA.PLUGINS WHERE PLUGIN_NAME LIKE 'ddl%'; Empty set (0.00 sec) mysql> install plugin ddl_rewriter soname 'ddl_rewriter.so'; Query OK, 0 rows affected (0.01 sec) mysql> SELECT PLUGIN_NAME, PLUGIN_STATUS, PLUGIN_TYPE FROM INFORMATION_SCHEMA.PLUGINS WHERE PLUGIN_NAME LIKE 'ddl%'; +--------------+---------------+-------------+ | PLUGIN_NAME | PLUGIN_STATUS | PLUGIN_TYPE | +--------------+---------------+-------------+ | ddl_rewriter | ACTIVE | AUDIT | +--------------+---------------+-------------+ 1 row in set (0.01 sec) mysql> use POC Database changed mysql> CREATE TABLE `T_ENCRYPTION` (-> `id` int NOT NULL AUTO_INCREMENT COMMENT 'PGの場合はSEQUENCE', -> `note` varchar(10) DEFAULT NULL, -> `who` varchar(100) DEFAULT NULL, -> PRIMARY KEY (`id`) ->) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci ENCRYPTION='Y'; Query OK, 0 rows affected, 1 warning (0.04 sec) mysql> show warnings\G *************************** 1. row *************************** Level: Note Code: 1105 Message: Query 'CREATE TABLE `T_ENCRYPTION` (`id` int NOT NULL AUTO_INCREMENT COMMENT 'PGの場合はSEQUENCE', `note` varchar(10) DEFAULT NULL, `who` varchar(100) DEFAULT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci ENCRYPTION='Y'' rewritten to 'CREATE TABLE `T_ENCRYPTION` (`id` int NOT NULL AUTO_INCREMENT COMMENT 'PGの場合はSEQUENCE', `note` varchar(10) DEFAULT NULL, `who` varchar(100) DEFAULT NULL, PRIMARY KEY (`id`)) ENGINE=I 1 row in set (0.00 sec) mysql> UNINSTALL PLUGIN ddl_rewriter; Query OK, 0 rows affected, 1 warning (0.01 sec) mysql> show warnings\G *************************** 1. row *************************** Level: Warning Code: 1620 Message: Plugin is busy and will be uninstalled on shutdown 1 row in set (0.00 sec) mysql> SELECT PLUGIN_NAME, PLUGIN_STATUS, PLUGIN_TYPE FROM INFORMATION_SCHEMA.PLUGINS WHERE PLUGIN_NAME LIKE 'ddl%'; +--------------+---------------+-------------+ | PLUGIN_NAME | PLUGIN_STATUS | PLUGIN_TYPE | +--------------+---------------+-------------+ | ddl_rewriter | DELETED | AUDIT | +--------------+---------------+-------------+ 1 row in set (0.00 sec) mysql>
ALTER INSTANCE	8.0.16 8.0.21	Newly Add	The ALTER INSTANCE statement supports a RELOAD TLS action that reconfigures the TLS context from the current values of the system variables that define the context.	ALTER INSTANCE instance_action instance_action: { | {ENABLE|DISABLE} INNODB REDO_LOG | ROTATE INNODB MASTER KEY | ROTATE BINLOG MASTER KEY | RELOAD TLS [FOR CHANNEL {mysql_main | mysql_admin}] [NO ROLLBACK ON ERROR] } ALTER INSTANCE {ENABLE | DISABLE} INNODB REDO_LOG このアクションは、InnoDB redo ロギングを有効または無効にします。redo ロギングはデフォルトで有効になっています。この機能は、新しい MySQL インスタンスへのデータのロードのみを目的としています。ステートメントはバイナリログに書き込まれません。MySQL 8.0.21 で導入されました。 ALTER INSTANCE RELOAD TLS このアクションは、コンテキストを定義するシステム変数の現在の値からLS コンテキストを再構成します。また、アクティブなコンテキスト値を反映するステータス変数も更新されます。このアクションには、CONNECTION_ADMIN 権限が必要です。 TLS コンテキストの再構成の詳細（コンテキスト関連のシステム変数やステータス変数など は、サーバー側のランタイム構成および暗号化された接続の監視を参照してください。 デフォルトでは、ステートメントはメイン接続インタフェースのLS コンテキストをリロードします。（MySQL 8.0.21 で使用可能な）FOR CHANNEL 句が指定されている場合、ステートメントは指定されたチャネルのLS コンテキストをリロード: メイン接続インタフェースの場合はmysql_main、管理接続インタフェースの場合はmysql_admin。 様々なインタフェースの詳細は、セクション5.1.12.1「接続インタフェース」 を参照してください。 更新された TLS コンテキストプロパティーは、パフォーマンススキーマtls_channel_status テーブルで公開されます。セクション27.12.19.11「tls_channel_status テーブル」を参照してください。 https://dev.mysql.com/doc/refman/8.0/ja/alter-instance.html

INFORMATION_SCHEMA	8.0.16	Newly Add	Previously, MySQL permitted a limited form of CHECK constraint syntax, but parsed and ignored it. MySQL now implements the core features of table and column CHECK constraints, for all storage engines. Constraints are defined using CREATE TABLE and ALTER TABLE statements. The new INFORMATION_SCHEMA.CHECK_CONSTRAINTS table provides information about CHECK constraints defined on tables. For more information, see CHECK Constraints. (Bug #11744849, Bug #3464, Bug #3465, Bug #11746042, Bug #22759)	mysql> CREATE TABLE T_CHECK (-> `c1` int(11) DEFAULT NULL, -> `c2` int(11) DEFAULT NULL, -> `c3` int(11) DEFAULT NULL, -> CONSTRAINT `c1_nonzero` CHECK ((`c1` <> 0)), -> CONSTRAINT `c2_positive` CHECK ((`c2` > 0)), -> CONSTRAINT `t1_chk_1` CHECK ((`c1` <> `c2`)), -> CONSTRAINT `t1_chk_2` CHECK ((`c1` > 10)), -> CONSTRAINT `t1_chk_3` CHECK ((`c3` < 100)), -> CONSTRAINT `t1_chk_4` CHECK ((`c1` > `c3`)) ->) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci; Query OK, 0 rows affected, 3 warnings (0.05 sec) mysql> select * from INFORMATION_SCHEMA.CHECK_CONSTRAINTS; +--------------------+-------------------+-----------------+---------------+ | CONSTRAINT_CATALOG | CONSTRAINT_SCHEMA | CONSTRAINT_NAME | CHECK_CLAUSE | +--------------------+-------------------+-----------------+---------------+ | def | POC | c1_nonzero | (`c1` <> 0) | | def | POC | c2_positive | (`c2` > 0) | | def | POC | t1_chk_1 | (`c1` <> `c2`) | | def | POC | t1_chk_2 | (`c1` > 10) | | def | POC | t1_chk_3 | (`c3` < 100) | | def | POC | t1_chk_4 | (`c1` > `c3`) | +--------------------+-------------------+-----------------+---------------+ 6 rows in set (0.01 sec) mysql>
FUNCTION	8.0.16	Newly Add	FORMAT_BYTES(): Converts a byte count to a value with units. Similar to sys.format_bytes(). FORMAT_PICO_TIME(): Converts a time in picoseconds to a value with units. Similar to sys.format_time(). PS_THREAD_ID(): Returns the Performance Schema thread ID for a given thread. Similar to sys.ps_thread_id() invoked with a non-NULL argument. PS_CURRENT_THREAD_ID(): Returns the Performance Schema thread ID for the current thread. Shortcut for sys.ps_thread_id() invoked with a NULL argument.	mysql> select sys.format_bytes(1024),FORMAT_BYTES(1024); +-----------------------+--------------------+ | sys.format_bytes(1024) | FORMAT_BYTES(1024) | +-----------------------+--------------------+ | 1.00 KiB | 1.00 KiB | +-----------------------+--------------------+ 1 row in set (0.00 sec) mysql> select sys.format_bytes(1024000),FORMAT_BYTES(1024000); +--------------------------+----------------------+ | sys.format_bytes(1024000) | FORMAT_BYTES(1024000) | +--------------------------+----------------------+ | 1000.00 KiB | 1000.00 KiB | +--------------------------+----------------------+ 1 row in set (0.00 sec) mysql>
TEMPORARY TABLE	8.0.16	Newly Add	InnoDB: When the amount of memory occupied by the TempTable storage engine exceeds the limit defined by the temptable_max_ram variable, the TempTable storage engine allocates space for internal in-memory temporary tables as memory-mapped temporary files. This behavior is now controlled by the temptable_use_mmap variable, which can be disabled to have the TempTable storage engine use InnoDB on-disk internal temporary tables instead. For more information, see Internal Temporary Table Use in MySQL. (Bug #28944457)	mysql> show variables like 'temptable_use_mmap'; +--------------------+-------+ | Variable_name | Value | +--------------------+-------+ | temptable_use_mmap | ON | +--------------------+-------+ 1 row in set (0.00 sec) mysql> show variables like 'temptable_max_ram'; +-------------------+------------+ | Variable_name | Value | +-------------------+------------+ | temptable_max_ram | 1073741824 | +-------------------+------------+ 1 row in set (0.00 sec) mysql> select 1073741824/1024/1024; +----------------------+ | 1073741824/1024/1024 | +----------------------+ | 1024.00000000 | +----------------------+ 1 row in set (0.00 sec) mysql>
FUNCTION	8.0.16	Deprecate	The SQL_CALC_FOUND_ROWS query modifier and accompanying FOUND_ROWS() function are now deprecated and will be removed in a future MySQL version. As a replacement, considering executing your query with LIMIT, and then a second query with COUNT(*) and without LIMIT to determine whether there are additional rows. For example, instead of these queries: SELECT SQL_CALC_FOUND_ROWS * FROM tbl_name WHERE id > 100 LIMIT 10; SELECT FOUND_ROWS();	mysql> SELECT FOUND_ROWS(); +--------------+ | FOUND_ROWS() | +--------------+ | 3 | +--------------+ 1 row in set, 1 warning (0.00 sec) mysql> show warnings; +---------+------+---+ | Level | Code | Message | +---------+------+---+ | Warning | 1287 | FOUND_ROWS() is deprecated and will be removed in a future release. Consider using COUNT(*) instead. | +---------+------+---+ 1 row in set (0.00 sec)

JSON	8.0.16	Newly Add	InnoDB; JSON: InnoDB now supports multi-valued indexes on JSON arrays. A multi-valued index is an index in which multiple index records can point to the same data record. This can be useful for indexing JSON documents such as {"user":"Bob","zipcode":[94477,94536]} in which, if we wish to search all zip codes, it is necessary to have two index records for each zip code in the document. We can create such an index on the zipcode array using a CREATE INDEX statement such as this one:	mysql> select * from T_JSON where 94477 MEMBER OF(doc->'$.zipcode'); +----+---+ \| id \| doc \| +----+---+ \| 1 \| {"user": "Bob", "zipcode": [94477, 94536]} \| +----+---+ 1 row in set (0.00 sec) mysql> explain select * from T_JSON where 94477 MEMBER OF(doc->'$.zipcode'); +----+-------------+--------+------------+------+---------------+----------+---------+------+------+----------+-------------+ | id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra | +----+-------------+--------+------------+------+---------------+----------+---------+------+------+----------+-------------+ | 1 | SIMPLE | T_JSON | NULL | ref | idx_zips | idx_zips | 9 | const | 1 | 100.00 | Using where | +----+-------------+--------+------------+------+---------------+----------+---------+------+------+----------+-------------+ 1 row in set, 1 warning (0.00 sec) mysql> select * from T_JSON where JSON_CONTAINS(doc->'$.zipcode', CAST('[94477, 94536]' AS JSON)); +----+---+ \| id \| doc \| +----+---+ \| 1 \| {"user": "Bob", "zipcode": [94477, 94536]} \| +----+---+ 1 row in set (0.00 sec) mysql> select * from T_JSON where JSON_OVERLAPS(doc->'$.zipcode', CAST('[94477, 94536]' AS JSON)); +----+---+ | id | doc | +----+---+ | 1 | {"user": "Bob", "zipcode": [94477, 94536]} | +----+---+ 1 row in set (0.01 sec)
VARIABLES	8.0.17	Newly Add	In MySQL 8.0, the lower_case_table_names variable can only be configured when the MySQL server is initialized. Because a MySQL server installation on Debian and Ubuntu performed using APT initializes the MySQL server for you, there was no opportunity to enable lower_case_table_names. To work around this issue, you can now use the debconf-set-selection utility to enable lower_case_table_names (set lower_case_table_names=1) prior to installing MySQL using APT. To enable lower_case_table_names prior to installing MySQL using APT, execute the following command: shell> sudo debconf-set-selections <<< "mysql-server mysql-server/lowercase-table-names select Enabled	mysql> show variables like 'lower_case_table_%'; +------------------------+-------+ | Variable_name | Value | +------------------------+-------+ | lower_case_table_names | 0 | +------------------------+-------+ 1 row in set (0.01 sec) shinya@DESKTOP-8BDL7KA:/usr/bin$ ls -l debconf* -rwxr-xr-x 1 root root 2859 Aug 3 2019 debconf -rwxr-xr-x 1 root root 11541 Aug 3 2019 debconf-apt-progress -rwxr-xr-x 1 root root 608 Aug 3 2019 debconf-communicate -rwxr-xr-x 1 root root 1719 Aug 3 2019 debconf-copydb -rwxr-xr-x 1 root root 647 Aug 3 2019 debconf-escape -rwxr-xr-x 1 root root 2935 Aug 3 2019 debconf-set-selections -rwxr-xr-x 1 root root 1827 Aug 3 2019 debconf-show shell> sudo debconf-set-selections <<< "mysql-server mysql-server/lowercase-table-names select Enabled
AUTH	8.0.17	CHANGED	The umask for files created using SELECT ... INTO OUTFILE or SELECT ... INTO DUMPFILE was changed from 0666 to 0640. The LOAD_FILE() function no longer requires files to be world-readable, just readable by the server. (Bug #24513720)	0666 to 0640
Replication	8.0.17	CHANGED	The mysqldump option --set-gtid-purged controls whether or not a SET @@GLOBAL.gtid_purged statement is added to the mysqldump output. The statement updates the value of gtid_purged on a server where the dump file is reloaded, to add the GTID set from the source server's gtid_executed system variable. A new choice --set-gtid-purged=COMMENTED is now available. When this value is set, if GTIDs are enabled on the server you are backing up, SET @@GLOBAL.gtid_purged is added to the output (unless gtid_executed is empty), but it is commented out. This means that the value of gtid_executed is available in the output, but no action is taken automatically when the dump file is reloaded. With COMMENTED, you can control the use of the gtid_executed set manually or through automation. For example, you might prefer to do this if you are migrating data to another server that already has different active databases. Thanks to Facebook for this contribution. (Bug #94332, Bug #29357665)	SET @@GLOBAL.gtid_purged statement is added to the mysqldump output. 補足：UUIDはMacアドレスとタイムスタンプからUUIDを生成している。 root@localhost [mysql]> show variables like 'gtid_purged'\G *************************** 1. row *************************** Variable_name: gtid_purged Value: 50fca08e-5a35-11e8-a4f2-06db798d79c8:1-6625843 1 row in set (0.01 sec) gtid_purged システム変数 (@@GLOBAL.gtid_purged) 内の GTID のセットには、サーバー上でコミットされたが、サーバー上のバイナリログファイルには存在しないすべてのトランザクションのGTID が含まれています。gtid_purged は、gtid_executed のサブセットです。 GTID の次のカテゴリが gtid_purged にあります: - レプリカでバイナリロギングを無効にしてコミットされたレプリケートされたトランザクションのGTID。 - 現在パージされているバイナリログファイルに書き込まれたトランザクションのGTID。 - ステートメント SET @@GLOBAL.gtid_purged によってセットに明示的に追加されたGTID。

FUNCTION	8.0.17	Newly Add	MySQL now supports explicit casts to DOUBLE, FLOAT, and REAL using either of the functions CAST() or CONVERT(). For more information, see Cast Functions and Operators. (Bug #30524, Bug #11747058)	mysql> create table t2 as select Cast(0 as Double); Query OK, 1 row affected (0.06 sec) Records: 1 Duplicates: 0 Warnings: 0 mysql> desc t2; +-------------------+--------+------+-----+---------+-------+ | Field | Type | Null | Key | Default | Extra | +-------------------+--------+------+-----+---------+-------+ | Cast(0 as Double) | double | NO | | 0 | | +-------------------+--------+------+-----+---------+-------+ 1 row in set (0.00 sec) mysql> create table t as select 0.00 as rowd; Query OK, 1 row affected (0.04 sec) Records: 1 Duplicates: 0 Warnings: 0 mysql> desc t; +-------+--------------+------+-----+---------+-------+ | Field | Type | Null | Key | Default | Extra | +-------+--------------+------+-----+---------+-------+ | rowd | decimal(3,2) | NO | | 0.00 | | +-------+--------------+------+-----+---------+-------+ 1 row in set (0.01 sec)
TOOL	8.0.17	Newly Add	MySQL now provides a clone plugin that permits cloning InnoDB data locally or from a remote MySQL server instance. A local cloning operation stores cloned data on the same server or node where the MySQL instance runs. A remote cloning operation transfers cloned data over the network from a donor MySQL server instance to the recipient server or node where the cloning operation was initiated.	5.6.7 クローンプラグイン https://dev.mysql.com/doc/refman/8.0/ja/clone-plugin.html
ACCount Management	8.0.18	Newly Add	The CREATE USER, ALTER USER, and SET PASSWORD statements now have the capability of generating random passwords for user accounts, as an alternative to requiring explicit administrator-specified literal passwords. See Password Management.	mysql> CREATE USER 'RANDOM_PASSWORD_USER_TEST'@'%' IDENTIFIED BY RANDOM PASSWORD; +---------------------------+------+----------------------+-------------+ | user | host | generated password | auth_factor | +---------------------------+------+----------------------+-------------+ | RANDOM_PASSWORD_USER_TEST | % | HwAO9}FQgwK3c<M}}CK3 | 1 | +---------------------------+------+----------------------+-------------+ 1 row in set (0.05 sec)
VARIABLES	8.0.18	Deprecate	Use of the MYSQL_PWD environment variable to specify a MySQL password is considered insecure because its value may be visible to other system users. MYSQL_PWD is now deprecated and will be removed in a future MySQL version.	MYSQL_PWD: mysqld に接続する際のデフォルトのパスワード。これの使用はセキュアではありません。表のあとにある注釈を参照してください。 https://dev.mysql.com/doc/refman/8.0/ja/environment-variables.html
HASH JOIN	8.0.18	Newly Add	Hash joins have been implemented as a way of executing inner equi-joins in MySQL. For example, a query such as this one can be executed as a hash join beginning with this release: Multi-table joins using equi-joins can also take advantage of this optimization. A hash join requires no index for execution. In most cases, a hash join is more efficient than the block-nested loop algorithm previously used for equi-joins without indexes. By default, beginning with this release, a hash join is used whenever a join includes at least one equi-join condition, and no indexes can be applied to the join condition. This preference can be overridden by setting the hash_join optimizer switch to off, or by using the NO_HASH_JOIN optimizer hint. In addition, you can control the amount of memory used by a hash join by setting join_buffer_size. A join whose memory requirement exceeds this amount is executed on disk; an on-disk hash join uses a number of disk files and may not be executable if this number exceeds open_files_limit.	HASH_JOIN, NO_HASH_JOINヒント: 指定したテーブルに対するハッシュ結合の使用を有効または無効にしますMySQL 8.0.18 のみ。MySQL 8.0.19 以降では無効です)。 mysql> explain select * from T_CHECK INNER JOIN T_CHECK2 ON T_CHECK.c1 = T_CHECK2.c1; +----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+--+ | id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra | +----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+--+ | 1 | SIMPLE | T_CHECK | NULL | ALL | NULL | NULL | NULL | NULL | 3 | 100.00 | NULL | | 1 | SIMPLE | T_CHECK2 | NULL | ALL | NULL | NULL | NULL | NULL | 3 | 33.33 | Using where; Using join buffer (hash join) | +----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+--+ 2 rows in set, 1 warning (0.01 sec) mysql> explain select /*+ NO_HASH_JOIN(T_CHECK,T_CHECK2) */ * from T_CHECK INNER JOIN T_CHECK2 ON T_CHECK.c1 = T_CHECK2.c1; +----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+--+ | id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra | +----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+--+ | 1 | SIMPLE | T_CHECK | NULL | ALL | NULL | NULL | NULL | NULL | 3 | 100.00 | NULL | | 1 | SIMPLE | T_CHECK2 | NULL | ALL | NULL | NULL | NULL | NULL | 3 | 33.33 | Using where; Using join buffer (hash join) | +----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+--+ 2 rows in set, 1 warning (0.00 sec) mysql> EXPLAIN FORMAT=TREE select * from T_CHECK INNER JOIN T_CHECK2 ON T_CHECK.c1 = T_CHECK2.c1\G *************************** 1. row *************************** EXPLAIN: -> Inner hash join (T_CHECK2.c1 = T_CHECK.c1) (cost=1.70 rows=3) -> Table scan on T_CHECK2 (cost=0.12 rows=3) -> Hash -> Table scan on T_CHECK (cost=0.55 rows=3) 1 row in set (0.00 sec) mysql> EXPLAIN ANALYZE select * from T_CHECK INNER JOIN T_CHECK2 ON T_CHECK.c1 = T_CHECK2.c1\G *************************** 1. row *************************** EXPLAIN: -> Inner hash join (T_CHECK2.c1 = T_CHECK.c1) (cost=1.70 rows=3) (actual time=0.496..0.504 rows=3 loops=1) -> Table scan on T_CHECK2 (cost=0.12 rows=3) (actual time=0.078..0.084 rows=3 loops=1) -> Hash -> Table scan on T_CHECK (cost=0.55 rows=3) (actual time=0.295..0.338 rows=3 loops=1) 1 row in set (0.00 sec) mysql>

SYS	8.0.18	Newly Add	The sys.schema_unused_indexes view now filters out unique indexes. Thanks to Gillian Gunson for the contribution. (Bug #24798995, Bug #83257)	mysql> select * from sys.schema_unused_indexes limit 10; +--------------------+----------------+--------------------------------+ | object_schema | object_name | index_name | +--------------------+----------------+--------------------------------+ | performance_schema | cond_instances | NAME | | performance_schema | data_lock_waits | BLOCKING_THREAD_ID | | performance_schema | data_lock_waits | REQUESTING_ENGINE_LOCK_ID | | performance_schema | data_lock_waits | BLOCKING_ENGINE_LOCK_ID | | performance_schema | data_lock_waits | REQUESTING_ENGINE_TRANSACTION_ID | | performance_schema | data_lock_waits | BLOCKING_ENGINE_TRANSACTION_ID | | performance_schema | data_lock_waits | REQUESTING_THREAD_ID | | performance_schema | data_locks | ENGINE_TRANSACTION_ID | | performance_schema | data_locks | OBJECT_SCHEMA | | performance_schema | data_locks | THREAD_ID | +--------------------+----------------+--------------------------------+ 10 rows in set (0.04 sec)
SYS	8.0.18	Newly Add	The sys.ps_is_consumer_enabled() function now produces an error rather than returning NULL if the argument is an unknown non-NULL consumer name. (Bug #24760317)	指定されたパフォーマンススキーマコンシューマが有効かどうかを示すES または NO を返します。引数が NULL の場合は NULL を返します。 引数が有効なコンシューマ名でない場合は、エラーが発生します。(MySQL 8.0.18 より前では、引数が有効なコンシューマ名でない場合、この関数はNULL を戻します。) この関数はコンシューマ階層を考慮しているため、依存するすべてのコンシューマも有効になっていないかぎり、コンシューマは有効とみなされません。 mysql> SELECT sys.ps_is_consumer_enabled('thread_instrumentation'); +--+ | sys.ps_is_consumer_enabled('thread_instrumentation') | +--+ | YES | +--+ 1 row in set (0.00 sec) mysql> SELECT sys.ps_is_consumer_enabled('events_stages_current'); +---+ | sys.ps_is_consumer_enabled('events_stages_current') | +---+ | NO | +---+ 1 row in set (0.00 sec) mysql> SELECT * FROM performance_schema.setup_consumers; +--------------------------------+---------+ | NAME | ENABLED | +--------------------------------+---------+ | events_stages_current | NO | | events_stages_history | NO | | events_stages_history_long | NO | | events_statements_cpu | NO | | events_statements_current | YES | | events_statements_history | YES | | events_statements_history_long | NO | | events_transactions_current | YES | | events_transactions_history | YES | | events_transactions_history_long | NO | | events_waits_current | NO | | events_waits_history | NO | | events_waits_history_long | NO | | global_instrumentation | YES | | thread_instrumentation | YES | | statements_digest | YES | +--------------------------------+---------+ 16 rows in set (0.00 sec) mysql> SELECT sys.ps_is_consumer_enabled('events_statements_current'); +---+ | sys.ps_is_consumer_enabled('events_statements_current') | +---+ | YES | +---+ 1 row in set (0.00 sec) mysql>

AUTH	8.0.19	Newly Add	MySQL now enables administrators to configure user accounts such that too many consecutive login failures due to incorrect passwords cause temporary account locking. The required number of failures and the lock time are configurable per account, using the FAILED_LOGIN_ATTEMPTS and PASSWORD_LOCK_TIME options of the CREATE USER and ALTER USER statements. See Password Management. (Bug #27733694, Bug #90169)	mysql> CREATE USER 'non-official-user'@'%' IDENTIFIED BY RANDOM PASSWORD FAILED_LOGIN_ATTEMPTS 3 PASSWORD_LOCK_TIME 1; +-------------------+------+----------------------+-------------+ | user | host | generated password | auth_factor | +-------------------+------+----------------------+-------------+ | non-official-user | % | mhn2sP-[k93v/<NneI0A | 1 | +-------------------+------+----------------------+-------------+ 1 row in set (0.03 sec) shinya@DESKTOP-8BDL7KA:~/git/rdbms-docker/mysql$ mysql -h 127.0.0.1 -P 13306 -u non-official-user -p"mhn2sP-[k93v/<NneI0A" mysql: [Warning] Using a password on the command line interface can be insecure. Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL connection id is 29 Server version: 8.0.31 MySQL Community Server - GPL Copyright (c) 2000, 2022, Oracle and/or its affiliates. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners. Type 'help;' or '\h' for help. Type '\c' to clear the current input statement. mysql> select user(); +------------------------------+ | user() | +------------------------------+ | non-official-user@172.18.0.1 | +------------------------------+ 1 row in set (0.00 sec) mysql> \q Bye shinya@DESKTOP-8BDL7KA:~/git/rdbms-docker/mysql$ mysql -h 127.0.0.1 -P 13306 -u non-official-user -p"mhn2sP-[k93v/<NneI0A?" mysql: [Warning] Using a password on the command line interface can be insecure. ERROR 1045 (28000): Access denied for user 'non-official-user'@'172.18.0.1' (using password: YES) shinya@DESKTOP-8BDL7KA:~/git/rdbms-docker/mysql$ mysql -h 127.0.0.1 -P 13306 -u non-official-user -p"mhn2sP-[k93v/<NneI0A?" mysql: [Warning] Using a password on the command line interface can be insecure. ERROR 1045 (28000): Access denied for user 'non-official-user'@'172.18.0.1' (using password: YES) shinya@DESKTOP-8BDL7KA:~/git/rdbms-docker/mysql$ mysql -h 127.0.0.1 -P 13306 -u non-official-user -p"mhn2sP-[k93v/<NneI0A?" mysql: [Warning] Using a password on the command line interface can be insecure. ERROR 3955 (HY000): Access denied for user 'non-official-user'@'172.18.0.1'. Account is blocked for 1 day(s) (1 day(s) remaining) due to 3 consecutive failed logins. shinya@DESKTOP-8BDL7KA:~/git/rdbms-docker/mysql$
VARIABLES	8.0.19	Newly Add	Innodb_system_rows_read, Innodb_system_rows_inserted, Innodb_system_rows_deleted status variables were added for counting row operations on InnoDB tables that belong to system-created schemas. The new status variables are similar to the existing Innodb_rows_read, Innodb_rows_inserted, Innodb_rows_deleted status variables, which count operations on InnoDB tables that belong to both user-created and system-created schemas.	mysql> show status like 'Innodb_system_%'; +-----------------------------+--------+ | Variable_name | Value | +-----------------------------+--------+ | Innodb_system_rows_deleted | 22292 | | Innodb_system_rows_inserted | 26494 | | Innodb_system_rows_read | 129879 | | Innodb_system_rows_updated | 9092 | +-----------------------------+--------+ 4 rows in set (0.00 sec)

HASH JOIN	8.0.19	Deprecate	Setting the hash_join optimizer switch (see optimizer_switch system variable) no longer has any effect. The same applies with respect to the HASH_JOIN and NO_HASH_JOIN optimizer hints. Both the optimizer switch and the optimizer hint are now deprecated, and subject to removal in a future release of MySQL. mysql> show variables like 'optimizer_switch'\G *************************** 1. row *************************** Variable_name: optimizer_switch Value: index_merge=on, index_merge_union=on,index_merge_sort_union=on, index_merge_intersection=on, engine_condition_pushdown=on, index_condition_pushdown=on,mrr=on, mrr_cost_based=on,block_nested_loop=on, batched_key_access=off,materialization=on, semijoin=on,loosescan=on,firstmatch=on, duplicateweedout=on, subquery_materialization_cost_based=on, use_index_extensions=on, condition_fanout_filter=on,derived_merge=on, use_invisible_indexes=off,skip_scan=on, hash_join=on,subquery_to_derived=off, prefer_ordering_index=on, hypergraph_optimizer=off, derived_condition_pushdown=on 1 row in set (0.01 sec) mysql>	mysql> explain format=tree select * from T_CHECK INNER JOIN T_CHECK2 ON T_CHECK.c1 = T_CHECK2.c1\G *************************** 1. row *************************** EXPLAIN: -> Inner hash join (T_CHECK2.c1 = T_CHECK.c1) (cost=1.70 rows=3) -> Table scan on T_CHECK2 (cost=0.12 rows=3) -> Hash -> Table scan on T_CHECK (cost=0.55 rows=3) 1 row in set (0.01 sec) mysql> explain analyze select * from T_CHECK INNER JOIN T_CHECK2 ON T_CHECK.c1 = T_CHECK2.c1\G *************************** 1. row *************************** EXPLAIN: -> Inner hash join (T_CHECK2.c1 = T_CHECK.c1) (cost=1.70 rows=3) (actual time=0.356..0.399 rows=3 loops=1) -> Table scan on T_CHECK2 (cost=0.12 rows=3) (actual time=0.130..0.170 rows=3 loops=1) -> Hash -> Table scan on T_CHECK (cost=0.55 rows=3) (actual time=0.129..0.174 rows=3 loops=1) 1 row in set (0.00 sec) mysql> explain select /*+ NO_HASH_JOIN(T_CHECK,T_CHECK2) */ * from T_CHECK INNER JOIN T_CHECK2 ON T_CHECK.c1 = T_CHECK2.c1\G *************************** 1. row *************************** id: 1 select_type: SIMPLE table: T_CHECK partitions: NULL type: ALL possible_keys: NULL key: NULL key_len: NULL ref: NULL rows: 3 filtered: 100.00 Extra: NULL *************************** 2. row *************************** id: 1 select_type: SIMPLE table: T_CHECK2 partitions: NULL type: ALL possible_keys: NULL key: NULL key_len: NULL ref: NULL rows: 3 filtered: 33.33 Extra: Using where; Using join buffer (hash join) 2 rows in set, 1 warning (0.00 sec) mysql>
STATEMENT	8.0.19	Newly Add	Important Change: MySQL now supports explicit table clauses and table value constructors according to the SQL standard. These have now been implemented, respectively, as the TABLE statement and the VALUES statement, each described in brief here:	MySQL 8.0.19 で導入された DML ステートメントで、指定されたテーブルの行とカラムを返します mysql> TABLE T_CHECK; +------+------+------+ | c1 | c2 | c3 | +------+------+------+ | 1000 | 2000 | 10 | | 1001 | 2001 | 11 | | 1002 | 2002 | 12 | +------+------+------+ 3 rows in set (0.00 sec) mysql> select * from T_CHECK; +------+------+------+ | c1 | c2 | c3 | +------+------+------+ | 1000 | 2000 | 10 | | 1001 | 2001 | 11 | | 1002 | 2002 | 12 | +------+------+------+ 3 rows in set (0.00 sec) mysql> select * from T_CHECK where c3 >=11; +------+------+------+ | c1 | c2 | c3 | +------+------+------+ | 1001 | 2001 | 11 | | 1002 | 2002 | 12 | +------+------+------+ 2 rows in set (0.01 sec) mysql> TABLE T_CHECK where c3 >=11; ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'where c3 >=11' at line 1 mysql>

STATEMENT	8.0.19	Newly Add	VALUES consists of the VALUES keyword followed by a series of row constructors (ROW()), separated by commas. It can be used to supply row values in an SQL-compliant fashion to an INSERT statement or REPLACE statement. For example, the following two statements are equivalent:	mysql> SELECT a,b,c FROM (VALUES ROW(1,2,3), ROW(4,5,6)) AS t(a,b,c); +---+---+---+ | a | b | c | +---+---+---+ | 1 | 2 | 3 | | 4 | 5 | 6 | +---+---+---+ 2 rows in set (0.00 sec) mysql> SELECT a,c FROM (VALUES ROW(1,2,3), ROW(4,5,6)) AS t(a,b,c); +---+---+ | a | c | +---+---+ | 1 | 3 | | 4 | 6 | +---+---+ 2 rows in set (0.00 sec) mysql> VALUES ROW(1,-2,3), ROW(5,7,9), ROW(4,6,8); +----------+----------+----------+ | column_0 | column_1 | column_2 | +----------+----------+----------+ | 1 | -2 | 3 | | 5 | 7 | 9 | | 4 | 6 | 8 | +----------+----------+----------+ 3 rows in set (0.00 sec) mysql> https://dev.mysql.com/doc/refman/8.0/en/values.html
STATEMENT	8.0.19	Newly Add	Previously, it was not possible to use LIMIT in the recursive SELECT part of a recursive common table expression (CTE). LIMIT is now supported in such cases, along with an optional OFFSET clause. An example of such a recursive CTE is shown here:	mysql> WITH RECURSIVE cte AS (-> SELECT CAST("x" AS CHAR(100)) AS a FROM DUAL -> UNION ALL -> SELECT CONCAT("x",cte.a) FROM cte -> WHERE LENGTH(cte.a) < 20 -> LIMIT 3 OFFSET 2 ->) -> SELECT * FROM cte; +-------+ | a | +-------+ | xxx | | xxxx | | xxxxx | +-------+ 3 rows in set (0.00 sec) mysql> mysql> WITH RECURSIVE cte AS (-> SELECT a,b,c FROM (VALUES ROW(1,2,3), ROW(4,5,6), ROW(7,8,9)) AS t(a,b,c) -> LIMIT 1 OFFSET 2 ->) -> select * from cte; +---+---+---+ | a | b | c | +---+---+---+ | 7 | 8 | 9 | +---+---+---+ 1 row in set (0.00 sec)
FUNCTION	8.0.19	Deprecate	sys.format_bytes(), sys.format_time(), and sys.ps_thread_id() will be removed in a future MySQL version, so applications that use them should be adjusted to use the built-in functions instead, keeping in mind some minor differences between the sys functions and the built-in functions.	mysql> select FORMAT_PICO_TIME(188732396662000),FORMAT_BYTES(512); +-----------------------------------+------------------+ | FORMAT_PICO_TIME(188732396662000) | FORMAT_BYTES(512) | +-----------------------------------+------------------+ | 3.15 min | 512 bytes | +-----------------------------------+------------------+ 1 row in set (0.00 sec)
Shell	8.0.19	Newly Add	From MySQL 8.0.19, compression is supported for messages sent over X Protocol connections. Connections can be compressed if the server and the client agree on a compression algorithm to use.	mysqlx_compression_algorithms system variable to include only the ones you permit.

PARTITION	8.0.19	CHANGED	Historically, delimiter strings have been uppercase (#P# and #SP#) on case-sensitive file systems such as Linux, and lowercase (#p# and #sp#) on case-insensitive file systems such as Windows. To avoid issues when migrating data directories between case-sensitive and case-insensitive file systems, delimiter strings are now lowercase on all file systems. Uppercase delimiter strings are no longer used. Additionally, partition tablespace names and file names generated based on user-specified partition or subpartition names, which can be specified in uppercase or lowercase, are now generated (and stored internally) in lowercase regardless of the lower_case_table_names setting to ensure case-insensitivity. For example, if a table partition is created with the name PART_1, the tablespace name and file name are generated in lowercase:	mysql> SELECT FILE_NAME FROM INFORMATION_SCHEMA.FILES E FILE_NAME LIKE -> WHERE FILE_NAME LIKE '%#p#%' OR FILE_NAME LIKE '%#sp#%'; +-----------------------------+ | FILE_NAME | +-----------------------------+ | ./POC/T_RANGE#p#p202112.ibd | | ./POC/T_RANGE#p#p202201.ibd | | ./POC/T_RANGE#p#p202202.ibd | | ./POC/T_RANGE#p#p202203.ibd | | ./POC/T_RANGE#p#p999999.ibd | +-----------------------------+ 5 rows in set (0.01 sec)
STATEMENT	8.0.20	Deprecate	The use of VALUES() to access new row values in INSERT ... ON DUPLICATE KEY UPDATE statements is now deprecated, and is subject to removal in a future MySQL release. Instead, you should use aliases for the new row and its columns as implemented in MySQL 8.0.19 and later.	INSERT INTO t1 (a,b,c) VALUES (1,2,3),(4,5,6) ON DUPLICATE KEY UPDATE c=VALUES(a)+VALUES(b); 新しい行とカラムを参照するためのVALUES() の使用は、MySQL 8.0.20 以降非推奨になり、将来のバージョンのMySQL で削除される予定です。かわりに、このセクションの次のいくつかの段落で説明するように、行およびカラムのエイリアスを使用します。 https://dev.mysql.com/doc/refman/8.0/ja/insert-on-duplicate.html
HINT	8.0.20	Newly Add	This release implements several new index-level optimizer hints, which function much like existing index hints that employ SQL keywords such as FORCE INDEX and IGNORE INDEX. These are intended to replace the equivalent index hints, which will be deprecated in a future MySQL release (and eventually removed). The new hints are listed here, along with a brief description of each:	mysql> SELECT id,name FROM members USE INDEX FOR ORDER BY (idx_uq_members_name) ORDER BY name; +----+------+ | id | name | +----+------+ | 4 | Mr.S | | 1 | Mr.T | | 2 | Mr.U | | 3 | Mr.V | | 5 | Mr.Z | +----+------+ 5 rows in set (0.00 sec) mysql> explain SELECT id,name FROM members USE INDEX FOR ORDER BY (idx_uq_members_name) ORDER BY name; +----+-------------+---------+------------+-------+---------------+--------------------+---------+------+------+----------+-------------+ | id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra | +----+-------------+---------+------------+-------+---------------+--------------------+---------+------+------+----------+-------------+ | 1 | SIMPLE | members | NULL | index | NULL | idx_uq_members_name | 1022 | NULL | 4 | 100.00 | Using index | +----+-------------+---------+------------+-------+---------------+--------------------+---------+------+------+----------+-------------+ 1 row in set, 1 warning (0.00 sec) mysql> SELECT /*+ ORDER_INDEX(members idx_uq_members_name) */ id,name FROM members ORDER BY name; +----+------+ | id | name | +----+------+ | 4 | Mr.S | | 1 | Mr.T | | 2 | Mr.U | | 3 | Mr.V | | 5 | Mr.Z | +----+------+ 5 rows in set (0.00 sec) mysql> explain SELECT /*+ ORDER_INDEX(members idx_uq_members_name) */ id,name FROM members ORDER BY name; +----+-------------+---------+------------+-------+---------------+--------------------+---------+------+------+----------+-------------+ | id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra | +----+-------------+---------+------------+-------+---------------+--------------------+---------+------+------+----------+-------------+ | 1 | SIMPLE | members | NULL | index | NULL | idx_uq_members_name | 1022 | NULL | 4 | 100.00 | Using index | +----+-------------+---------+------------+-------+---------------+--------------------+---------+------+------+----------+-------------+ 1 row in set, 1 warning (0.00 sec) mysql> https://dev.mysql.com/doc/refman/8.0/en/optimizer-hints.html#optimizer-hints-index-level
STATEMENT	8.0.20	Newly Add	Previously, the INTO clause for SELECT statements could appear at either of two positions: INTO now can appear in a third position, at the end of SELECT statements:	SELECT * INTO OUTFILE 'file_name' FROM table_name; SELECT * FROM table_name INTO OUTFILE 'file_name' FOR UPDATE; SELECT * FROM table_name FOR UPDATE INTO OUTFILE 'file_name'; mysql> select * INTO OUTFILE '/var/lib/mysql-files/T1-0.sql' from T1; Query OK, 1 row affected (0.00 sec) mysql> select * from T1 INTO OUTFILE '/var/lib/mysql-files/T1-1.sql'; Query OK, 1 row affected (0.00 sec) mysql> select * from T1 INTO OUTFILE '/var/lib/mysql-files/T1-2.sql' FOR UPDATE; Query OK, 1 row affected, 1 warning (0.00 sec)

InnoDB	8.0.20	Newly Add	A TRX_SCHEDULE_WEIGHT column was added to the INFORMATION_SCHEMA.INNODB_TRX table, which permits querying transaction scheduling weights assigned by the CATS algorithm. The following INNODB_METRICS counters were added for monitoring code-level transaction scheduling events: lock_rec_release_attempts The number of attempts to release record locks. lock_rec_grant_attempts The number of attempts to grant record locks. lock_schedule_refreshes The number of times the wait-for graph was analyzed to update transaction schedule weights.	mysql> select * from INFORMATION_SCHEMA.INNODB_TRX; Empty set (0.01 sec) mysql> desc INFORMATION_SCHEMA.INNODB_TRX; +----------------------------+----------------+------+-----+---------+-------+ | Field | Type | Null | Key | Default | Extra | +----------------------------+----------------+------+-----+---------+-------+ | trx_id | bigint unsigned | NO | | | | | trx_state | varchar(13) | NO | | | | | trx_started | datetime | NO | | | | | trx_requested_lock_id | varchar(105) | YES | | | | | trx_wait_started | datetime | YES | | | | | trx_weight | bigint unsigned | NO | | | | | trx_mysql_thread_id | bigint unsigned | NO | | | | | trx_query | varchar(1024) | YES | | | | | trx_operation_state | varchar(64) | YES | | | | | trx_tables_in_use | bigint unsigned | NO | | | | | trx_tables_locked | bigint unsigned | NO | | | | | trx_lock_structs | bigint unsigned | NO | | | | | trx_lock_memory_bytes | bigint unsigned | NO | | | | | trx_rows_locked | bigint unsigned | NO | | | | | trx_rows_modified | bigint unsigned | NO | | | | | trx_concurrency_tickets | bigint unsigned | NO | | | | | trx_isolation_level | varchar(16) | NO | | | | | trx_unique_checks | int | NO | | | | | trx_foreign_key_checks | int | NO | | | | | trx_last_foreign_key_error | varchar(256) | YES | | | | | trx_adaptive_hash_latched | int | NO | | | | | trx_adaptive_hash_timeout | bigint unsigned | NO | | | | | trx_is_read_only | int | NO | | | | | trx_autocommit_non_locking | int | NO | | | | | trx_schedule_weight | bigint unsigned | YES | | | | +----------------------------+----------------+------+-----+---------+-------+
InnoDB	8.0.20	CHANGED	InnoDB: The storage area for the doublewrite buffer was moved from the system tablespace to doublewrite files. Moving the doublewrite buffer storage area out of the system tablespace reduces write latency, increases throughput, and provides flexibility with respect to placement of doublewrite buffer pages. The following system variables were introduced for advanced doublewrite buffer configuration: innodb_doublewrite_dir Defines the doublewrite buffer file directory. innodb_doublewrite_files Defines the number of doublewrite files. innodb_doublewrite_pages Defines the maximum number of doublewrite pages per thread for a batch write. innodb_doublewrite_batch_size Defines the number of doublewrite pages to write in a batch.	MySQL 8.0.20 より前は、二重書込みバッファ記憶域はInnoDB システムテーブルスペースにありました。MySQL 8.0.20 では、二重書込みバッファ記憶域は二重書込みファイルにあります。 https://dev.mysql.com/doc/refman/8.0/ja/innodb-doublewrite-buffer.html
AUTH	8.0.21	Newly Add	You can now set per-user comments and attributes when creating or updating MySQL user accounts. A user comment consists of arbitrary text passed as the argument to a COMMENT clause used with a CREATE USER or ALTER USER statement. A user attribute consists of data in the form of a JSON object passed as the argument to an ATTRIBUTE clause used with either of these two statements. The attribute can contain any valid key-value pairs in JSON object notation. For example, the first of the following two statements creates a user account bill@localhost with the comment text This is Bill's user account. The second statement adds a user attribute to this account, using the key email, with the value bill@example.com.	mysql> select user,host,User_attributes from mysql.user where User_attributes is not null; +------------------+-----------+---+ | user | host | User_attributes | +------------------+-----------+---+ | admin | % | {"metadata": {"description": "Admin Account for WSL on docker"}} | | non-official-user | % | {"Password_locking": {"failed_login_attempts": 3, "password_lock_time_days": 1}} | | bill | localhost | {"metadata": {"comment": "This is Bill's user account"}} | +------------------+-----------+---+ 3 rows in set (0.01 sec)

AUTH	8.0.21	Newly Add	There are new configuration parameters that apply specifically to the administrative interface. The ALTER INSTANCE RELOAD TLS statement is extended with a FOR CHANNEL clause that enables specifying the channel (interface) for which to reload the TLS context. The new Performance Schema tls_channel_status table exposes TLS context properties for the main and administrative interfaces. For backward compatibility, the administrative interface uses the same TLS context as the main interface unless some nondefault TLS parameter value is configured for the administrative interface.	mysql> select * from performance_schema.tls_channel_status; +-------------+--------------------------------+--------------------------+ | CHANNEL | PROPERTY | VALUE | +-------------+--------------------------------+--------------------------+ | mysql_main | Enabled | Yes | | mysql_main | Ssl_accept_renegotiates | 0 | | mysql_main | Ssl_accepts | 25 | | mysql_main | Ssl_callback_cache_hits | 0 | | mysql_main | Ssl_client_connects | 0 | | mysql_main | Ssl_connect_renegotiates | 0 | | mysql_main | Ssl_ctx_verify_depth | -1 | | mysql_main | Ssl_ctx_verify_mode | 5 | | mysql_main | Current_tls_ca | ca.pem | | mysql_main | Current_tls_capath | | | mysql_main | Current_tls_cert | server-cert.pem | | mysql_main | Current_tls_cipher | | | mysql_main | Current_tls_ciphersuites | | | mysql_main | Current_tls_crl | | | mysql_main | Current_tls_crlpath | | | mysql_main | Current_tls_key | server-key.pem | | mysql_main | Current_tls_version | TLSv1.2,TLSv1.3 | | mysql_main | Ssl_finished_accepts | 25 | | mysql_main | Ssl_finished_connects | 0 | | mysql_main | Ssl_server_not_after | Nov 1 07:23:33 2031 GMT | | mysql_main | Ssl_server_not_before | Nov 3 07:23:33 2021 GMT | | mysql_main | Ssl_session_cache_hits | 0 | | mysql_main | Ssl_session_cache_misses | 0 | | mysql_main | Ssl_session_cache_mode | SERVER | | mysql_main | Ssl_session_cache_overflows | 0 | | mysql_main | Ssl_session_cache_size | 128 | | mysql_main | Ssl_session_cache_timeouts | 0 | | mysql_main | Ssl_used_session_cache_entries | 0 | | mysql_main | Ssl_session_cache_timeout | 300 | | mysql_admin | Enabled | No | | mysql_admin | Ssl_accept_renegotiates | 0 | | mysql_admin | Ssl_accepts | 0 | | mysql_admin | Ssl_callback_cache_hits | 0 | | mysql_admin | Ssl_client_connects | 0 | | mysql_admin | Ssl_connect_renegotiates | 0 | | mysql_admin | Ssl_ctx_verify_depth | 0 | | mysql_admin | Ssl_ctx_verify_mode | 0 | | mysql_admin | Current_tls_ca | | | mysql_admin | Current_tls_capath | | | mysql_admin | Current_tls_cert | | | mysql_admin | Current_tls_cipher | | | mysql_admin | Current_tls_ciphersuites | | | mysql_admin | Current_tls_crl | | | mysql_admin | Current_tls_crlpath | | | mysql_admin | Current_tls_key | | | mysql_admin | Current_tls_version | TLSv1.2,TLSv1.3 | | mysql_admin | Ssl_finished_accepts | 0 | | mysql_admin | Ssl_finished_connects | 0 | | mysql_admin | Ssl_server_not_after | | | mysql_admin | Ssl_server_not_before | | | mysql_admin | Ssl_session_cache_hits | 0 | | mysql_admin | Ssl_session_cache_misses | 0 | | mysql_admin | Ssl_session_cache_mode | NONE | | mysql_admin | Ssl_session_cache_overflows | 0 | | mysql_admin | Ssl_session_cache_size | 0 | | mysql_admin | Ssl_session_cache_timeouts | 0 | | mysql_admin | Ssl_used_session_cache_entries | 0 | | mysql_admin | Ssl_session_cache_timeout | 0 | +-------------+--------------------------------+--------------------------+ 58 rows in set (0.04 sec)
PARTITION	8.0.21	Deprecate	When one or more columns using index prefixes are specified as part of the partitioning key, a warning is now generated for each such column. In addition, when a CREATE TABLE or ALTER TABLE statement is rejected because all columns specified in the proposed partitioning key employ index prefixes, the error message returned now makes clear the reason the statement did not succeed.	■ ﾊﾟｰﾃｨｼｮﾝ作成のvalidationの様なので別途確認 Partitioning: Columns with index prefixes are not supported as part of a table's partitioning key; previously such columns were simply omitted by the server when referenced in creating, altering, or upgrading a table that was partitioned by key, with no indication that this omission had taken place, except when the proposed partitioning function used only columns with prefixes, in which case the statement failed with an error message that did not identify the actual source of the problem. mysql> show create table T_RANGE\G *************************** 1. row *************************** Table: T_RANGE Create Table: CREATE TABLE `T_RANGE` (`id` int NOT NULL, `note` varchar(1024) NOT NULL DEFAULT '', `updated_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci /*!50100 PARTITION BY RANGE (unix_timestamp(`updated_date`)) (PARTITION p202112 VALUES LESS THAN (1638316800) ENGINE = InnoDB, PARTITION p202201 VALUES LESS THAN (1640995200) ENGINE = InnoDB, PARTITION p202202 VALUES LESS THAN (1643673600) ENGINE = InnoDB, PARTITION p202203 VALUES LESS THAN (1646092800) ENGINE = InnoDB, PARTITION p999999 VALUES LESS THAN MAXVALUE ENGINE = InnoDB) */ 1 row in set (0.00 sec) mysql> alter table T_RANGE add index idx_note(note(10)); Query OK, 0 rows affected (0.17 sec) Records: 0 Duplicates: 0 Warnings: 0

JSON	8.0.21	Newly Add	Added the JSON_VALUE() function, which simplifies creating indexes on JSON columns. A call to JSON_VALUE(json_doc, path RETURNING type) is equivalent to calling CAST(JSON_UNQUOTE(JSON_EXTRACT(json_doc, path)) AS type), where json_doc is a JSON document, path is a JSON path expression pointing to a single value within the document, and type is a data type compatible with CAST(). RETURNING type is optional; if no return type is specified, JSON_VALUE() returns VARCHAR(512). JSON_VALUE() also supports ON EMPTY and ON ERROR clauses similar to those used with JSON_TABLE().	mysql> CREATE DATABASE `POC8021` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci */ /*!80016 DEFAULT ENCRYPTION='N' */ ; Query OK, 1 row affected (0.02 sec) mysql> use POC8021 Database changed mysql> CREATE TABLE inventory(items JSON, INDEX i1 ((JSON_VALUE(items, '$.name' RETURNING CHAR(50)))), INDEX i2 ((JSON_VALUE(items, '$.price' RETURNING DECIMAL(5,2)))), INDEX i3 ((JSON_VALUE(items, '$.quantity' RETURNING UNSIGNED)))); Query OK, 0 rows affected (0.07 sec) mysql> insert into inventory(items) values('{"name": "hat", "price": "22.95", "quantity": "17"}'); Query OK, 1 row affected (0.03 sec) mysql> select * from inventory; +--+ \| items \| +--+ \| {"name": "hat", "price": "22.95", "quantity": "17"} \| +--+ 1 row in set (0.00 sec) mysql> SELECT items->"$.price" FROM inventory WHERE JSON_VALUE -> WHERE JSON_VALUE(items, '$.name' RETURNING CHAR(50)) = "hat"; +-----------------+ \| items->"$.price" | +-----------------+ | "22.95" | +-----------------+ 1 row in set (0.01 sec) mysql> explain SELECT items->"$.price" FROM inventory WHERE JSON_VALUE(items, '$.name' RETURNING CHAR(50)) = "hat"; +----+-------------+-----------+------------+------+---------------+------+---------+-------+------+----------+-------+ | id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra | +----+-------------+-----------+------------+------+---------------+------+---------+-------+------+----------+-------+ | 1 | SIMPLE | inventory | NULL | ref | i1 | i1 | 203 | const | 1 | 100.00 | NULL | +----+-------------+-----------+------------+------+---------------+------+---------+-------+------+----------+-------+ 1 row in set, 1 warning (0.00 sec) mysql>
OPTIMIZER	8.0.21	Newly Add	MySQL attempts to use an ordered index for any ORDER BY or GROUP BY query that has a LIMIT clause, overriding any other choices made by the optimizer, whenever it determines that this would result in faster execution. Because the algorithm for making this determination makes certain assumptions about data distribution and other conditions, it may not always be completely correct, and it is possible in some cases that choosing a different optimization for such queries can provide better performance. To handle such occurrences, it is now possible to disable this optimization by setting the optimizer_switch system variable's prefer_ordering_index flag to off.	mysql> show variables like 'optimizer_switch'\G *************************** 1. row *************************** Variable_name: optimizer_switch Value: index_merge=on,index_merge_union=on,index_merge_sort_union=on,index_merge_intersection=on,engine_condition_pushdown=on,index_condition_pushdown=on, mrr=on,mrr_cost_based=on,block_nested_loop=on,batched_key_access=off,materialization=on,semijoin=on,loosescan=on,firstmatch=on,duplicateweedout=on, subquery_materialization_cost_based=on,use_index_extensions=on,condition_fanout_filter=on,derived_merge=on,use_invisible_indexes=off,skip_scan=on,hash_join=on, subquery_to_derived=off,**prefer_ordering_index**=on,hypergraph_optimizer=off,derived_condition_pushdown=on 1 row in set (0.00 sec) mysql>

OPTIMIZER	8.0.21	Newly Add	Multi-Range Read Flags mrr (default on) Controls the Multi-Range Read strategy. mrr_cost_based (default on) Controls use of cost-based MRR if mrr=on. For more information, see Section 8.2.1.11, “Multi-Range Read Optimization”. Semijoin Flags duplicateweedout (default on) Controls the semijoin Duplicate Weedout strategy. firstmatch (default on) Controls the semijoin FirstMatch strategy. loosescan (default on) Controls the semijoin LooseScan strategy (not to be confused with Loose Index Scan for GROUP BY). semijoin (default on) Controls all semijoin strategies. In MySQL 8.0.17 and later, this also applies to the antijoin optimization. The semijoin, firstmatch, loosescan, and duplicateweedout flags enable control over semijoin strategies. The semijoin flag controls whether semijoins are used. If it is set to on, the firstmatch and loosescan flags enable finer control over the permitted semijoin strategies. If the duplicateweedout semijoin strategy is disabled, it is not used unless all other applicable strategies are also disabled. If semijoin and materialization are both on, semijoins also use materialization where applicable. These flags are on by default. For more information, see Section 8.2.2.1, “Optimizing IN and EXISTS Subquery Predicates with Semijoin Transformations”. Skip Scan Flags skip_scan (default on) Controls use of Skip Scan access method. For more information, see Skip Scan Range Access Method. Subquery Materialization Flags materialization (default on) Controls materialization (including semijoin materialization). subquery_materialization_cost_based (default on) Use cost-based materialization choice. The materialization flag controls whether subquery materialization is used. If semijoin and materialization are both on, semijoins also use materialization where applicable. These flags are on by default. The subquery_materialization_cost_based flag enables control over the choice between subquery materialization and IN-to-EXISTS subquery transformation. If the flag is on (the default), the optimizer performs a cost-based choice between subquery materialization and IN-to-EXISTS subquery transformation if either method could be used. If the flag is off, the optimizer chooses subquery materialization over IN-to-EXISTS subquery transformation. For more information, see Section 8.2.2, “Optimizing Subqueries, Derived Tables, View	mysql> show variables like 'optimizer_switch'\G *************************** 1. row *************************** Variable_name: optimizer_switch Value: index_merge=on,index_merge_union=on,index_merge_sort_union=on,index_merge_intersection=on,engine_condition_pushdown=on,index_condition_pushdown=on, mrr=on,mrr_cost_based=on,block_nested_loop=on,batched_key_access=off,materialization=on,semijoin=on,loosescan=on,firstmatch=on,duplicateweedout=on, subquery_materialization_cost_based=on,use_index_extensions=on,condition_fanout_filter=on,derived_merge=on,use_invisible_indexes=off,skip_scan=on,hash_join=on, subquery_to_derived=off,prefer_ordering_index=on,hypergraph_optimizer=off,derived_condition_pushdown=on 1 row in set (0.00 sec) mysql>

OPTIMIZER	8.0.21	Newly Add	Prior to MySQL 8.0.22, the subquery could not contain a GROUP BY clause. This optimization can also be applied to a table subquery which is the argument to IN, NOT IN, EXISTS, or NOT EXISTS, that does not contain a GROUP BY. The default value for this flag is off, since, in most cases, enabling this optimization does not produce any noticeable improvement in performance (and in many cases can even make queries run more slowly), but you can enable the optimization by setting the subquery_to_derived flag to on. It is primarily intended for use in testing.	mysql> show variables like 'optimizer_switch'\G *************************** 1. row *************************** Variable_name: optimizer_switch Value: index_merge=on,index_merge_union=on,index_merge_sort_union=on,index_merge_intersection=on,engine_condition_pushdown=on,index_condition_pushdown=on, mrr=on,mrr_cost_based=on,block_nested_loop=on,batched_key_access=off,materialization=on,semijoin=on,loosescan=on,firstmatch=on,duplicateweedout=on, subquery_materialization_cost_based=on,use_index_extensions=on,condition_fanout_filter=on,derived_merge=on,use_invisible_indexes=off,skip_scan=on,hash_join=on, **subquery_to_derived=off**,prefer_ordering_index=on,hypergraph_optimizer=off,derived_condition_pushdown=on 1 row in set (0.00 sec) mysql>
Replication	8.0.21	Newly Add	variable binlog_checksum, which defaults to the setting CRC32. Previously, Group Replication did not support the presence of checksums in the binary log, so binlog_checksum had to be set to NONE when configuring a server instance that would become a group member. This requirement is now removed, and the default can be used. The setting for binlog_checksum does not have to be the same for all members of a group.	CHECK SUM (Group Replication)
REDO	8.0.21	Newly Add	InnoDB: Redo logging can now be enabled and disabled using ALTER INSTANCE {ENABLE|DISABLE} INNODB REDO_LOG syntax. This functionality is intended for loading data into a new MySQL instance. Disabling redo logging helps speed up data loading by avoiding redo log writes. The new INNODB_REDO_LOG_ENABLE privilege permits enabling and disabling redo logging. The new Innodb_redo_log_enabled status variable permits monitoring redo logging status.	mysql> ALTER INSTANCE DISABLE INNODB REDO_LOG; Query OK, 0 rows affected (0.01 sec) mysql> ALTER INSTANCE ENABLE INNODB REDO_LOG; Query OK, 0 rows affected (0.03 sec) mysql>
InnoDB	8.0.21	Newly Add	At startup, InnoDB validates the paths of known tablespace files against tablespace file paths stored in the data dictionary in case tablespace files have been moved to a different location. The new innodb_validate_tablespace_paths variable permits disabling tablespace path validation. This feature is intended for environments where tablespaces files are not moved. Disabling tablespace path validation improves startup time on systems with a large number of tablespace files.	mysql> show variables like 'innodb_validate_tablespace_paths'; +----------------------------------+-------+ | Variable_name | Value | +----------------------------------+-------+ | innodb_validate_tablespace_paths | ON | +----------------------------------+-------+ 1 row in set (0.04 sec)
TRUNCATE	8.0.21	Change	Truncating an InnoDB table that resides in a file-per-table tablespace **drops the existing tablespace and creates a new one**. As of MySQL 8.0.21, InnoDB creates the new tablespace in the default location and writes a warning to the error log if the tablespace was created with an earlier version and the current tablespace directory is unknown. To have TRUNCATE TABLE create the tablespace in its current location, add the directory to the innodb_directories setting before running TRUNCATE TABLE.	mysql> show variables like 'innodb_directories'; +--------------------+-------+ | Variable_name | Value | +--------------------+-------+ | innodb_directories | | +--------------------+-------+ 1 row in set (0.00 sec) mysql>
Docker	8.0.21	Newly Add	MySQL Server Docker containers now support server restart within a client session (which happens, for example, when the RESTART statement is executed by a client or during the configuration of an InnoDB Cluster instance). To enable this important feature, containers should be started with the **docker run option --restart** set to the value on-failure. See Starting a MySQL Server Instance for details. (Bug #30750730)	mysql> restart; ERROR 3707 (HY000): Restart server failed (mysqld is not managed by supervisor process).
EXPLAIN	8.0.21	Change	EXPLAIN ANALYZE now supports the FORMAT option. Currently, TREE is the only supported format. (Bug #30315224)	mysql> EXPLAIN ANALYZE FORMAT=TREE SELECT a, COUNT(*) FROM t1 GROUP BY a WITH ROLLUP\G *************************** 1. row *************************** EXPLAIN: -> Group aggregate with rollup: count(0) (cost=1.05 rows=4) (actual time=0.056..0.057 rows=5 loops=1) -> Sort: a (cost=0.65 rows=4) (actual time=0.051..0.052 rows=4 loops=1) -> Table scan on t1 (cost=0.65 rows=4) (actual time=0.029..0.038 rows=4 loops=1) 1 row in set (0.00 sec) mysql>

Replication	8.0.21	Change	On storage engines that **support atomic DDL, the CREATE TABLE ... SELECT statement is now logged as one transaction** in the binary log when row-based replication is in use. Previously, it was logged as two transactions, one to create the table, and the other to insert data. **With this change, CREATE TABLE ... SELECT statements are now safe for row-based replication and permitted for use with GTID-based replication.** For more information, see Atomic Data Definition Statement Support. (Bug #11756034, Bug #47899)	mysql> create table T3 select * from t1; Query OK, 4 rows affected (0.16 sec) Records: 4 Duplicates: 0 Warnings: 0 mysql>
Replication	8.0.22	Change	Deprecation and Removal Notes From MySQL 8.0.22, the group_replication_ip_whitelist system variable is deprecated, and the system variable **group_replication_ip_allowlist** has been added to replace it. The system variable works in the same way as before, only the terminology has changed.	mysql> show variables like 'group%'; +------------------------------+----------+ | Variable_name | Value | +------------------------------+----------+ | group_concat_max_len | 1024 | | group_replication_consistency | EVENTUAL | +------------------------------+----------+ 2 rows in set (0.02 sec) mysql>
Replication	8.0.22	Change	From MySQL 8.0.22, the statements START SLAVE, STOP SLAVE, SHOW SLAVE STATUS, SHOW SLAVE HOSTS and RESET SLAVE are deprecated. The following aliases should be used instead:	Instead of START SLAVE use START REPLICA Instead of STOP SLAVE use STOP REPLICA Instead of SHOW SLAVE STATUS use SHOW REPLICA STATUS Instead of SHOW SLAVE HOSTS use SHOW REPLICAS Instead of RESET SLAVE use RESET REPLICA
Memcached	8.0.22	Deprecate	The InnoDB memcached plugin is deprecated and support for it will be removed in a future MySQL version.	memcached
INFORMATION_SCHEMA	8.0.22	Deprecate	The INFORMATION_SCHEMA.TABLESPACES table is unused. It is now deprecated and will be removed in a future MySQL version. Other INFORMATION_SCHEMA tables may provide related information, as described in The INFORMATION_SCHEMA TABLESPACES Table.	mysql> select * from INFORMATION_SCHEMA.TABLESPACES; Empty set, 1 warning (0.00 sec) mysql> show warnings; +---------+------+---+ | Level | Code | Message | +---------+------+---+ | Warning | 1681 | 'INFORMATION_SCHEMA.TABLESPACES' is deprecated and will be removed in a future release. | +---------+------+---+ 1 row in set (0.00 sec)
TEMPORARY TABLE	8.0.22	Change	The filesort algorithm now supports sorting a join on multiple tables, and not just a single table. (Bug #31310238, Bug #31559978, Bug #31563876)	mysql> explain select gid,name,category,type,geohash from tourism_and_busstop order by geohash; +----+-------------+--------------------+------------+------+---------------+------+---------+------+-------+----------+----------------+ | id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra | +----+-------------+--------------------+------------+------+---------------+------+---------+------+-------+----------+----------------+ | 1 | SIMPLE | tourism_and_busstop | NULL | ALL | NULL | NULL | NULL | NULL | 29950 | 100.00 | Using filesort | +----+-------------+--------------------+------------+------+---------------+------+---------+------+-------+----------+----------------+ 1 row in set, 1 warning (0.01 sec)
DUMP	8.0.22	Add	Added support for periodic synchronization when writing to files with SELECT INTO DUMPFILE and SELECT INTO OUTFILE statements. This feature can be enabled by setting the select_into_disk_sync system variable to ON; the size of the write buffer cn be set using the server system variable select_into_buffer_size; the default buffer size is 131072 (217) bytes. An optional delay following synchronization to disk can also be set using the select_into_disk_sync_delay system variable; the default behaviour is not to allow any delay (that is, a delay time of 0 milliseconds).	mysql> show variables like 'select_into_%'; +----------------------------+--------+ | Variable_name | Value | +----------------------------+--------+ | select_into_buffer_size | 131072 | | select_into_disk_sync | OFF | | select_into_disk_sync_delay | 0 | +----------------------------+--------+ 3 rows in set (0.01 sec)
OPTIMIZER	8.0.22	Add	To enable derived condition pushdown, the optimizer_switch system variable's derived_condition_pushdown flag (added in this release) must be set to on. This is the default setting. If this optimization is disabled by the optimizer switch setting, you can enable it for a specific query using the DERIVED_CONDITION_PUSHDOWN optimizer hint (also added in this release). Use the NO_DERIVED_CONDITION_PUSHDOWN optimizer hint to disable the optimization for a given query.	mysql> SELECT @@optimizer_switch LIKE '%derived_condition_pushdown=off%'; +---+ | @@optimizer_switch LIKE '%derived_condition_pushdown=off%' | +---+ | 0 | +---+ 1 row in set (0.01 sec) mysql> SELECT @@optimizer_switch LIKE '%derived_condition_pushdown=on%'; +--+ | @@optimizer_switch LIKE '%derived_condition_pushdown=on%' | +--+ | 1 | +--+ 1 row in set (0.00 sec) mysql>

Performance Schema	8.0.22	Add	An alternative SHOW PROCESSLIST implementation is now available based on the new Performance Schema processlist table. This implementation queries active thread data from the Performance Schema rather than the thread manager and does not require a mutex To enable the alternative implementation, enable the performance_schema_show_processlist system variable. The alternative implementation of SHOW PROCESSLIST also applies to the mysqladmin processlist command. The alternative implementation does not apply to the INFORMATION_SCHEMA PROCESSLIST table or the COM_PROCESS_INFO command of the MySQL client/server protocol. To ensure that the default and alternative implementations yield the same information, certain configuration requirements must be met; see The processlist Table.	mysql> show processlist; +----+-----------------+-----------------+---------+---------+--------+-----------------------+------------------+ | Id | User | Host | db | Command | Time | State | Info | +----+-----------------+-----------------+---------+---------+--------+-----------------------+------------------+ | 7 | event_scheduler | localhost | NULL | Daemon | 127500 | Waiting on empty queue | NULL | | 35 | root | 172.18.0.1:51622 | POC8021 | Query | 0 | init | show processlist | +----+-----------------+-----------------+---------+---------+--------+-----------------------+------------------+ 2 rows in set (0.00 sec) mysql> select * from performance_schema.processlist; +----+-----------------+-----------------+---------+---------+--------+-----------------------+---+------------------+ | ID | USER | HOST | DB | COMMAND | TIME | STATE | INFO | EXECUTION_ENGINE | +----+-----------------+-----------------+---------+---------+--------+-----------------------+---+------------------+ | 7 | event_scheduler | localhost | NULL | Daemon | 127514 | Waiting on empty queue | NULL | PRIMARY | | 35 | root | 172.18.0.1:51622 | POC8021 | Query | 0 | executing | select * from performance_schema.processlist | PRIMARY | +----+-----------------+-----------------+---------+---------+--------+-----------------------+---+------------------+ 2 rows in set (0.03 sec) *************************** 1. row *************************** thd_id: 13 conn_id: NULL user: innodb/page_flush_coordinator_thread db: NULL command: NULL state: NULL time: 127593 current_statement: NULL execution_engine: PRIMARY statement_latency: NULL progress: NULL lock_latency: NULL cpu_latency: NULL rows_examined: NULL rows_sent: NULL rows_affected: NULL tmp_tables: NULL tmp_disk_tables: NULL full_scan: NO last_statement: NULL last_statement_latency: NULL current_memory: 1.93 KiB last_wait: NULL last_wait_latency: NULL source: NULL trx_latency: NULL trx_state: NULL trx_autocommit: NULL pid: NULL program_name: NULL 1 row in set (0.04 sec) mysql> mysql> show variables like 'performance_schema_show_processlist'; +---------------------------------------+-------+ | Variable_name | Value | +---------------------------------------+-------+ | performance_schema_show_processlist | OFF | +---------------------------------------+-------+ 1 row in set (0.01 sec) mysql>
Logging	8.0.22	Add	An SQL interface to the most recent events written to the MySQL server error log is now available by means of queries on the new Performance Schema **error_log** table. This table has a fixed size, with old events automatically discarded as necessary to make room for new ones. The table is populated if error log configuration includes a log sink component that supports this capability (currently the traditional-format log_sink_internal and JSON-format log_sink_json sinks). Several new status variables provide information about error_log table operation. See The error_log Table.	mysql> select * from performance_schema.error_log limit 2 \G *************************** 1. row *************************** LOGGED: 2022-10-27 16:09:42.730414 THREAD_ID: 0 PRIO: Warning ERROR_CODE: MY-011068 SUBSYSTEM: Server DATA: The syntax '--skip-host-cache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0 instead. *************************** 2. row *************************** LOGGED: 2022-10-27 16:09:42.730423 THREAD_ID: 0 PRIO: Warning ERROR_CODE: MY-010918 SUBSYSTEM: Server DATA: 'default_authentication_plugin' is deprecated and will be removed in a future release. Please use authentication_policy instead. 2 rows in set (0.01 sec) mysql>

CAST	8.0.22	CHANGE	It is now possible to cast values of other types to YEAR, using either the CAST() function or the CONVERT() function. These functions now support YEAR values of one or two digits in the range 0-99, and four-digit values in the range 1901-2155. Integer 0 is converted to Year 0; a string consisting of one or more zeroes (following possible truncation) is converted to the year 2000. Casting adds 2000 to values in the range 1-69 inclusive, and 1900 to values in the range 70-99 inclusive. Strings beginning with one, two, or four digits followed by at least one non-digit character (and possibly other digit or non-digit characters) are truncated prior to conversion to YEAR; in such cases, the server emits a truncation warning. Floating-point values are rounded prior to conversion; CAST(1944.5 AS YEAR) returns 1945 due to rounding, and CAST ("1944.5" AS YEAR) returns 1944 (with a warning) due to truncation. DATE, DATETIME, and TIMESTAMP are cast to the YEAR portion of the value. A TIME value is cast to the current year. Not specifying the value to be cast as a TIME value may yield a different result from what is expected; CAST ("13:47" AS YEAR) returns 2013 due to truncation of the string value, and CAST(TIME "13:47" AS YEAR) returns 2020 as of the year of this release.	mysql> select CAST("1944.5" AS YEAR); +------------------------+ | CAST("1944.5" AS YEAR) | +------------------------+ | 1944 | +------------------------+ 1 row in set, 1 warning (0.00 sec) mysql> select CAST(now() AS YEAR); +---------------------+ | CAST(now() AS YEAR) | +---------------------+ | 2022 | +---------------------+ 1 row in set (0.00 sec) mysql> select CAST("13:47" AS YEAR); +------------------------+ | CAST("13:47" AS YEAR) | +------------------------+ | 2013 | +------------------------+ 1 row in set, 1 warning (0.00 sec) mysql> select CAST(TIME "13:47" AS YEAR); +----------------------------+ | CAST(TIME "13:47" AS YEAR) | +----------------------------+ | 2022 | +----------------------------+ 1 row in set (0.00 sec) mysql>
CAST	8.0.22	CHANGE	When selecting a TIMESTAMP column value, it is now **possible to convert it from the system time zone to a UTC DATETIME** when retrieving it, using the AT TIME ZONE operator which is implemented for the CAST() function in this release.	mysql> TABLE ex; +---------------------+ | ts | +---------------------+ | 2022-10-29 03:54:07 | | 2020-08-01 05:44:30 | +---------------------+ 2 rows in set (0.00 sec) mysql> SELECT ts, CAST(ts AT TIME ZONE 'UTC' AS DATETIME) AS ut FROM ex; +---------------------+---------------------+ | ts | ut | +---------------------+---------------------+ | 2022-10-29 03:54:07 | 2022-10-29 03:54:07 | | 2020-08-01 05:44:30 | 2020-08-01 05:44:30 | +---------------------+---------------------+ 2 rows in set (0.01 sec)
InnoDB	8.0.22	Newly Add	The new innodb_extend_and_initialize variable permits configuring **how InnoDB allocates space to file-per-table and general tablespaces on Linux.** By default, when an operation requires additional space in a tablespace, InnoDB allocates pages to the tablespace and physically writes NULLs to those pages. This behavior affects performance if new pages are allocated frequently. As of MySQL 8.0.22, you can disable innodb_extend_and_initialize on Linux systems to avoid physically writing NULLs to newly allocated tablespace pages. When innodb_extend_and_initialize is disabled, space is allocated using posix_fallocate() calls, which reserve space without physically writing NULLs.	mysql> show variables like 'innodb_extend_and_initialize'; +-------------------------------+-------+ | Variable_name | Value | +-------------------------------+-------+ | innodb_extend_and_initialize | ON | +-------------------------------+-------+ 1 row in set (0.01 sec) mysql>

Auth	8.0.23	Newly Add	Granting the RELOAD privilege enables a user to perform a wide variety of operations. In some cases, it may be desirable for a user to be able to perform only some of these operations. To enable DBAs to avoid granting RELOAD and tailor user privileges more closely to the operations permitted, these new privileges of more limited scope are available: FLUSH_OPTIMIZER_COSTS: Enables use of the FLUSH OPTIMIZER_COSTS statement. FLUSH_STATUS: Enables use of the FLUSH STATUS statement. FLUSH_TABLES: Enables use of the FLUSH TABLES statement. FLUSH_USER_RESOURCES: Enables use of the FLUSH USER_RESOURCES statement.	mysql> FLUSH OPTIMIZER_COSTS; Query OK, 0 rows affected (0.02 sec) mysql> select * from mysql.server_cost; +------------------------------+------------+---------------------+---------+---------------+ | cost_name | cost_value | last_update | comment | default_value | +------------------------------+------------+---------------------+---------+---------------+ | disk_temptable_create_cost | NULL | 2021-11-03 07:23:34 | NULL | 20 | | disk_temptable_row_cost | NULL | 2021-11-03 07:23:34 | NULL | 0.5 | | key_compare_cost | NULL | 2021-11-03 07:23:34 | NULL | 0.05 | | memory_temptable_create_cost | NULL | 2021-11-03 07:23:34 | NULL | 1 | | memory_temptable_row_cost | NULL | 2021-11-03 07:23:34 | NULL | 0.1 | | row_evaluate_cost | NULL | 2021-11-03 07:23:34 | NULL | 0.1 | +------------------------------+------------+---------------------+---------+---------------+ 6 rows in set (0.01 sec) mysql> select * from mysql.engine_cost; +-------------+-------------+------------------------+------------+---------------------+---------+---------------+ | engine_name | device_type | cost_name | cost_value | last_update | comment | default_value | +-------------+-------------+------------------------+------------+---------------------+---------+---------------+ | default | 0 | io_block_read_cost | NULL | 2021-11-03 07:23:34 | NULL | 1 | | default | 0 | memory_block_read_cost | NULL | 2021-11-03 07:23:34 | NULL | 0.25 | +-------------+-------------+------------------------+------------+---------------------+---------+---------------+ 2 rows in set (0.00 sec) mysql>
Replication	8.0.23	Change	From MySQL 8.0.23, the statement CHANGE MASTER TO is deprecated. The alias CHANGE REPLICATION SOURCE TO should be used instead. The parameters for the statement also have aliases that replace the term MASTER with the term SOURCE. For example, MASTER_HOST and MASTER_PORT can now be entered as SOURCE_HOST and SOURCE_PORT. The START REPLICA | SLAVE statement's parameters MASTER_LOG_POS and MASTER_LOG_FILE now have aliases SOURCE_LOG_POS and SOURCE_LOG_FILE. The statements work in the same way as before, only the terminology used for each statement has changed. A deprecation warning is issued if the old versions are used. A new status variable, Com_change_replication_source, has been added as an alias for the Com_change_master status variable. Both the old and new version of the statement update both the old and new version of the status variable.	root@localhost [mysql]> show replica status\G *************************** 1. row *************************** Replica_IO_State: Waiting for source to send event Source_Host: xxx.xxx.xxx.xxx Source_Port: 3306 Connect_Retry: 60 Source_Log_File: ip-xxx.xxx.xxx.xxx.001189 Read_Source_Log_Pos: 25307285 Relay_Log_File: xxx.xxx.xxx.xxx-relay-bin.000285 Relay_Log_Pos: 25306702 Relay_Source_Log_File: ip-xxx.xxx.xxx.xxx-bin.001189 Replica_IO_Running: Yes Replica_SQL_Running: Yes Replicate_Do_DB: Replicate_Ignore_DB: Replicate_Do_Table: Replicate_Ignore_Table: APP.sessions Replicate_Wild_Do_Table: Replicate_Wild_Ignore_Table: Last_Errno: 0 Last_Error: Skip_Counter: 0 Exec_Source_Log_Pos: 25306468 Relay_Log_Space: 25307835 Until_Condition: None
Replication	8.0.23	Deprecate	The use of the system variables master_info_repository and relay_log_info_repository is now deprecated, and a warning message is issued if you attempt to set them or read their values. The system variables will be removed in a future MySQL version. These system variables were used to specify whether the replica's connection metadata repository and applier metadata repository were written to an InnoDB table in the mysql system database, or to a file in the data directory. The FILE setting was already deprecated in a previous release, and tables are the default for the replication metadata repositories in MySQL 8.0.	mysql> show variables like 'master_info_repository'; +------------------------+-------+ | Variable_name | Value | +------------------------+-------+ | master_info_repository | TABLE | +------------------------+-------+ 1 row in set (0.01 sec) mysql> show variables like 'relay_log_info_repository'; +---------------------------+-------+ | Variable_name | Value | +---------------------------+-------+ | relay_log_info_repository | TABLE | +---------------------------+-------+ 1 row in set (0.01 sec)

Auth	8.0.23	Deprecate	Flushing the host cache can be done using any of these methods: Execute a TRUNCATE TABLE statement that truncates the Performance Schema host_cache table. This requires the DROP privilege for the table. Execute a FLUSH HOSTS statement. This requires the RELOAD privilege. Execute a mysqladmin flush-hosts command. This requires the RELOAD privilege. Although those methods are equivalent in effect, granting the RELOAD privilege enables a number of other operations in addition to host cache flushing, which is undesirable from a security standpoint. Granting the DROP privilege for the host_cache table is preferable because it has a more limited scope. Therefore, the FLUSH HOSTS statement is deprecated and will be removed in a future MySQL version. Instead, truncate the host_cache table.	mysql> FLUSH HOSTS; Query OK, 0 rows affected, 1 warning (0.01 sec) mysql> show warnings; +---------+------+--+ | Level | Code | Message | +---------+------+--+ | Warning | 1287 | 'FLUSH HOSTS' is deprecated and will be removed in a future release. Please use TRUNCATE TABLE performance_schema.host_cache instead | +---------+------+--+ 1 row in set (0.00 sec) mysql> desc performance_schema.host_cache; +---+------------------+------+-----+---------+-------+ | Field | Type | Null | Key | Default | Extra | +---+------------------+------+-----+---------+-------+ | IP | varchar(64) | NO | PRI | NULL | | | HOST | varchar(255) | YES | MUL | NULL | | | HOST_VALIDATED | enum('YES','NO') | NO | | NULL | | | SUM_CONNECT_ERRORS | bigint | NO | | NULL | | | COUNT_HOST_BLOCKED_ERRORS | bigint | NO | | NULL | | | COUNT_NAMEINFO_TRANSIENT_ERRORS | bigint | NO | | NULL | | | COUNT_NAMEINFO_PERMANENT_ERRORS | bigint | NO | | NULL | | | COUNT_FORMAT_ERRORS | bigint | NO | | NULL | | | COUNT_ADDRINFO_TRANSIENT_ERRORS | bigint | NO | | NULL | | | COUNT_ADDRINFO_PERMANENT_ERRORS | bigint | NO | | NULL | | | COUNT_FCRDNS_ERRORS | bigint | NO | | NULL | | | COUNT_HOST_ACL_ERRORS | bigint | NO | | NULL | | | COUNT_NO_AUTH_PLUGIN_ERRORS | bigint | NO | | NULL | | | COUNT_AUTH_PLUGIN_ERRORS | bigint | NO | | NULL | | | COUNT_HANDSHAKE_ERRORS | bigint | NO | | NULL | | | COUNT_PROXY_USER_ERRORS | bigint | NO | | NULL | | | COUNT_PROXY_USER_ACL_ERRORS | bigint | NO | | NULL | | | COUNT_AUTHENTICATION_ERRORS | bigint | NO | | NULL | | | COUNT_SSL_ERRORS | bigint | NO | | NULL | | | COUNT_MAX_USER_CONNECTIONS_ERRORS | bigint | NO | | NULL | | | COUNT_MAX_USER_CONNECTIONS_PER_HOUR_ERRORS | bigint | NO | | NULL | | | COUNT_DEFAULT_DATABASE_ERRORS | bigint | NO | | NULL | | | COUNT_INIT_CONNECT_ERRORS | bigint | NO | | NULL | | | COUNT_LOCAL_ERRORS | bigint | NO | | NULL | | | COUNT_UNKNOWN_ERRORS | bigint | NO | | NULL | | | FIRST_SEEN | timestamp | NO | | NULL | | | LAST_SEEN | timestamp | NO | | NULL | | | FIRST_ERROR_SEEN | timestamp | YES | | NULL | | | LAST_ERROR_SEEN | timestamp | YES | | NULL | | +---+------------------+------+-----+---------+-------+ 29 rows in set (0.01 sec)
Performance	8.0.23	Change	Functionality Added or Changed InnoDB: Performance was improved for the following operations: Dropping a large tablespace on a MySQL instance with a large buffer pool (>32GBs). Dropping a tablespace with a significant number of pages referenced from the adaptive hash index. Truncating temporary tablespaces. The pages of dropped or truncated tablespaces and associated AHI entries are now removed from the buffer pool passively as pages are encountered during normal operations. Previously, dropping or truncating tablespaces initiated a full list scan to remove pages from the buffer pool immediately, which negatively impacted performance. (Bug #31008942, Bug #98869)	mysql> show variables like '%hash%'; +---------------------------------+-------+ | Variable_name | Value | +---------------------------------+-------+ | innodb_adaptive_hash_index | ON | | innodb_adaptive_hash_index_parts | 8 | +---------------------------------+-------+ 2 rows in set (0.00 sec)

InnoDB	8.0.23	Change	InnoDB: The new AUTOEXTEND_SIZE option defines the amount by which InnoDB extends the size of a tablespace when it becomes full, making it possible to extend tablespace size in larger increments. Allocating space in larger increments helps to avoid fragmentation and facilitates ingestion of large amounts of data. The AUTOEXTEND_SIZE option is supported with the CREATE TABLE, ALTER TABLE, CREATE TABLESPACE, and ALTER TABLESPACE statements. For more information, see Tablespace AUTOEXTEND_SIZE Configuration.	mysql> use POC8023 Database changed mysql> CREATE TABLE t1 (c1 INT) AUTOEXTEND_SIZE = 4M; Query OK, 0 rows affected (0.10 sec) mysql> ALTER TABLE t1 AUTOEXTEND_SIZE = 8M; Query OK, 0 rows affected (0.03 sec) Records: 0 Duplicates: 0 Warnings: 0 mysql> CREATE TABLESPACE ts1 AUTOEXTEND_SIZE = 4M; Query OK, 0 rows affected (0.05 sec) mysql> ALTER TABLESPACE ts1 AUTOEXTEND_SIZE = 8M; Query OK, 0 rows affected (0.01 sec) mysql> show create table t1\G *************************** 1. row *************************** Table: t1 Create Table: CREATE TABLE `t1` (`c1` int DEFAULT NULL) /*!80023 AUTOEXTEND_SIZE=8388608 */ ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci 1 row in set (0.01 sec) https://dev.mysql.com/doc/refman/8.0/ja/innodb-tablespace-autoextend-size.html
Replication	8.0.23	Newly Add	For a multithreaded replica (where slave_parallel_workers is greater than 0), setting slave_preserve_commit_order=1 ensures that transactions are executed and committed on the replica in the same order as they appear in the replica's relay log. Each executing worker thread waits until all previous transactions are committed before committing. If a worker thread fails to execute a transaction because a possible deadlock was detected, or because the transaction's execution time exceeded a relevant wait timeout, it automatically retries the number of times specified by **slave_transaction_retries** before stopping with an error. Transactions with a non-temporary error are not retried. The replication applier on a multithreaded replica has always handled data access deadlocks that were identified by the storage engines involved. However, some other types of lock were not detected by the replication applier, such as locks involving access control lists (ACLs) or metadata locking (for example, FLUSH TABLES WITH READ LOCK statements). This could lead to three-actor deadlocks with the commit order locking, which could not be resolved by the replication applier, and caused replication to hang indefinitely. From MySQL 8.0.23, deadlock handling on multithreaded replicas that preserve the commit order has been enhanced to mitigate these types of deadlocks. The deadlocks are not specifically resolved by the replication applier, but the applier is aware of them and initiates automatic retries for the transaction, rather than hanging. If the retries are exhausted, **replication stops in a controlled manner so that the deadlock can be resolved manually.** (Bug #107574, Bug #34291887)	From MySQL 8.0.23, deadlock handling on multithreaded replicas that preserve the commit order has been enhanced to mitigate these types of deadlocks. The deadlocks are not specifically resolved by the replication applier, but the applier is aware of them and initiates automatic retries for the transaction, rather than hanging +------------------------------+----------------------+ | Variable_name | Value | +------------------------------+----------------------+ | slave_allow_batching | ON | | slave_checkpoint_group | 512 | | slave_checkpoint_period | 300 | | slave_compressed_protocol | OFF | | slave_exec_mode | STRICT | | slave_load_tmpdir | /tmp | | slave_max_allowed_packet | 1073741824 | | slave_net_timeout | 60 | | slave_parallel_type | LOGICAL_CLOCK | | slave_parallel_workers | 4 | | slave_pending_jobs_size_max | 134217728 | | slave_preserve_commit_order | ON | | slave_rows_search_algorithms | INDEX_SCAN,HASH_SCAN | | slave_skip_errors | OFF | | slave_sql_verify_checksum | ON | | **slave_transaction_retries** | 10 | | slave_type_conversions | | +------------------------------+----------------------+ 17 rows in set (0.00 sec) https://bugs.mysql.com/bug.php?id=107574 https://dev.mysql.com/worklog/task/?id=13574
Performance	8.0.23	Change	InnoDB: Performance was improved for the following operations: Dropping a large tablespace on a MySQL instance with a large buffer pool (>32GBs). Dropping a tablespace with a significant number of pages referenced from the adaptive hash index. Truncating temporary tablespaces. The pages of dropped or truncated tablespaces and associated AHI entries are **now removed from the buffer pool passively** as pages are encountered during normal operations. Previously, dropping or truncating tablespaces initiated a full list scan to remove pages from the buffer pool immediately, which negatively impacted performance. (Bug #31008942, Bug #98869)	root@localhost [world]> show variables like '%adaptive%'; +----------------------------------+--------+ | Variable_name | Value | +----------------------------------+--------+ | innodb_adaptive_flushing | ON | | innodb_adaptive_flushing_lwm | 10 | | innodb_adaptive_hash_index | ON | | innodb_adaptive_hash_index_parts | 8 | | innodb_adaptive_max_sleep_delay | 150000 | +----------------------------------+--------+

OPTIMIZER	8.0.24	Newly Add	The MySQL query optimizer can now apply the derived table optimization to correlated scalar subqueries, whenever the **subquery_to_derived** flag of the optimizer_switch variable is enabled. This is done by applying an extra grouping and then an outer join on the lifted predicate. For example, a query such as SELECT * FROM t1 WHERE (SELECT a FROM t2 WHERE t2.a=t1.a) > 0 can be rewritten as SELECT t1.* FROM t1 LEFT OUTER JOIN (SELECT a, COUNT(*) AS ct FROM t2 GROUP BY a) AS derived ON t1.a = derived.a WHERE derived.a > 0.	サブクエリー変換フラグ MySQL 8.0.21 以降、オプティマイザは多くの場合、SELECT, WHERE, JOIN または HAVING 句のスカラーサブクエリーを導出テーブルの左外部結合に変換できます。 (導出テーブルの NULL 値可能性によっては、内部結合にさらに簡略化される場合があります)。これは、次の条件を満たすサブクエリーに対して実行できます • サブクエリーでは、RAND() などの非決定的関数は使用されません。 • サブクエリーは、MIN() または MAX() を使用するようにリライトできるANY または ALL サブクエリーではありません。 • 親クエリーはユーザー変数を設定しません。リライトすると実行順序に影響する可能性があるため、同じクエリーで変数に複数回アクセスすると、予期しない結果が発生する可能性があります。 • サブクエリーは相関付けしないでください。つまり、外部クエリーのテーブルからカラムを参照したり、外部クエリーで評価される集計を含むことはできません。 MySQL 8.0.22 より前は、サブクエリーにGROUP BY 句を含めることはできませんでした。 この最適化は、GROUP BY を含まない IN, NOT IN, EXISTS または NOT EXISTS の引数であるテーブルサブクエリーにも適用できます。 このフラグのデフォルト値はoff ですが、ほとんどの場合、この最適化を有効にしてもパフォーマンスが著しく向上することはありません (多くの場合、クエリーの実行速度が遅くなることもあります)。ただし、subquery_to_derived フラグを on に設定することで最適化を有効にできます。主にテストで使用することを目的としています。 mysql> CREATE TABLE T_subquery_to_derived(a INT); Query OK, 0 rows affected (0.03 sec) mysql> CREATE TABLE T_subquery_to_derived2(a INT); Query OK, 0 rows affected (0.05 sec) mysql> INSERT INTO T_subquery_to_derived VALUES ROW(1), ROW(2), ROW(3), ROW(4); Query OK, 4 rows affected (0.02 sec) Records: 4 Duplicates: 0 Warnings: 0 mysql> INSERT INTO T_subquery_to_derived2 VALUES ROW(1), ROW(2); Query OK, 2 rows affected (0.02 sec) Records: 2 Duplicates: 0 Warnings: 0 mysql> SELECT * FROM T_subquery_to_derived WHERE T_subquery_to_derived.a > (SELECT COUNT(a) FROM T_subquery_to_derived2); +------+ | a | +------+ | 3 | | 4 | +------+ 2 rows in set (0.00 sec) mysql> explain SELECT * FROM T_subquery_to_derived WHERE T_subquery_to_derived.a > (SELECT COUNT(a) FROM T_subquery_to_derived2); +----+-------------+-----------------------+------------+------+---------------+------+---------+------+------+----------+-------------+ | id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra | +----+-------------+-----------------------+------------+------+---------------+------+---------+------+------+----------+-------------+ | 1 | PRIMARY | T_subquery_to_derived | NULL | ALL | NULL | NULL | NULL | NULL | 4 | 33.33 | Using where | | 2 | SUBQUERY | T_subquery_to_derived2 | NULL | ALL | NULL | NULL | NULL | NULL | 2 | 100.00 | NULL | +----+-------------+-----------------------+------------+------+---------------+------+---------+------+------+----------+-------------+ 2 rows in set, 1 warning (0.00 sec) mysql> SET @@optimizer_switch='subquery_to_derived=on'; Query OK, 0 rows affected (0.00 sec) mysql> SELECT @@optimizer_switch LIKE '%subquery_to_derived=off%'; +---+ | @@optimizer_switch LIKE '%subquery_to_derived=off%' | +---+ | 0 | +---+ 1 row in set (0.00 sec) mysql> explain SELECT * FROM T_subquery_to_derived WHERE T_subquery_to_derived.a > (SELECT COUNT(a) FROM T_subquery_to_derived2); +----+-------------+-----------------------+------------+------+---------------+------+---------+------+------+----------+---+ | id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra | +----+-------------+-----------------------+------------+------+---------------+------+---------+------+------+----------+---+ | 1 | PRIMARY | <derived2> | NULL | ALL | NULL | NULL | NULL | NULL | 1 | 100.00 | NULL | | 1 | PRIMARY | T_subquery_to_derived | NULL | ALL | NULL | NULL | NULL | NULL | 4 | 33.33 | Using where; Using join buffer (hash join) | | 2 | DERIVED | T_subquery_to_derived2 | NULL | ALL | NULL | NULL | NULL | NULL | 2 | 100.00 | NULL | +----+-------------+-----------------------+------------+------+---------------+------+---------+------+------+----------+---+ 3 rows in set, 1 warning (0.00 sec) mysql> mysql> SELECT @@optimizer_switch LIKE '%subquery_to_derived=on%'; +--+ | @@optimizer_switch LIKE '%subquery_to_derived=on%' | +--+ | 0 | +--+ 1 row in set (0.00 sec) mysql> SELECT @@optimizer_switch LIKE '%subquery_to_derived=off%';

InnoDB	8.0.24	CHANE	InnoDB: The AUTOEXTEND_SIZE maximum setting was increased from 64M to 4GB. The AUTOEXTEND_SIZE option, introduced in MySQL 8.0.23, defines the amount by which InnoDB extends the size of a tablespace when it becomes full. The option is supported with the CREATE TABLE, ALTER TABLE, CREATE TABLESPACE, and ALTER TABLESPACE statements. For more information, see Tablespace AUTOEXTEND_SIZE Configuration. (Bug #32438606)	テーブルスペースが一杯になったときにInnoDB がテーブルスペースのサイズを拡張する量を定義します。MySQL 8.0.23 で導入されました。 設定は 4MB の倍数である必要があります。デフォルト設定は 0 で、暗黙的なデフォルト動作に従ってテーブルスペースが拡張されます。 mysql> SELECT @@GLOBAL.innodb_page_size; +---------------------------+ | @@GLOBAL.innodb_page_size | +---------------------------+ | 16384 | +---------------------------+ 1 row in set (0.00 sec) mysql> CREATE TABLE t1 (c1 INT) AUTOEXTEND_SIZE = 4M; Query OK, 0 rows affected (0.06 sec) mysql> CREATE TABLE t2 (c1 INT) AUTOEXTEND_SIZE = 16M; Query OK, 0 rows affected (0.05 sec) mysql> SELECT NAME, AUTOEXTEND_SIZE FROM INFORMATION_SCHEMA.INNODB_TABLESPACES WHERE NAME LIKE 'POC8024%'; +------------+-----------------+ | NAME | AUTOEXTEND_SIZE | +------------+-----------------+ | POC8024/t1 | 4194304 | | POC8024/t2 | 16777216 | +------------+-----------------+ 2 rows in set (0.00 sec) mysql> CREATE TABLESPACE TS1 ADD DATAFILE 'TS1.ibd' FILE_BLOCK_SIZE=16K AUTOEXTEND_SIZE=64M ENGINE=INNODB; Query OK, 0 rows affected (0.10 sec) mysql> CREATE TABLE t3 (c1 INT PRIMARY KEY) TABLESPACE TS1; Query OK, 0 rows affected (0.03 sec) mysql> SELECT NAME, AUTOEXTEND_SIZE FROM INFORMATION_SCHEMA.INNODB_TABLESPACES WHERE NAME LIKE 'POC8024%'; +------------+-----------------+ | NAME | AUTOEXTEND_SIZE | +------------+-----------------+ | POC8024/t1 | 4194304 | | POC8024/t2 | 16777216 | +------------+-----------------+ 2 rows in set (0.00 sec) mysql>

TLS	8.0.26	Deprecate	The TLSv1 and TLSv1.1 connection protocols now are deprecated and support for them is subject to removal in a future MySQL version. (For background, refer to the IETF memo Deprecating TLSv1.0 and TLSv1.1.) It is recommended that connections be made using the more-secure TLSv1.2 and TLSv1.3 protocols. TLSv1.3 requires that both the MySQL server and the client application be compiled with OpenSSL 1.1.1 or higher.	mysql> SHOW GLOBAL VARIABLES LIKE 'tls_version'; +---------------+-----------------+ | Variable_name | Value | +---------------+-----------------+ | tls_version | TLSv1.2,TLSv1.3 | +---------------+-----------------+ 1 row in set (0.01 sec) mysql> SHOW SESSION STATUS LIKE 'Ssl_cipher_list'\G *************************** 1. row *************************** Variable_name: Ssl_cipher_list Value: TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_128_CCM_SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:DHE-RSA-AES128-SHA256:DHE-DSS-AES128-SHA256:DHE-DSS-AES256-GCM-SHA384:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-DSS-AES128-SHA:DHE-RSA-AES128-SHA:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES256-SHA:CAMELLIA256-SHA:CAMELLIA128-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA 1 row in set (0.00 sec) mysql> SELECT * FROM performance_schema.session_status WHERE VARIABLE_NAME IN ('Ssl_version','Ssl_cipher'); +---------------+------------------------+ | VARIABLE_NAME | VARIABLE_VALUE | +---------------+------------------------+ | Ssl_cipher | TLS_AES_256_GCM_SHA384 | | Ssl_version | TLSv1.3 | +---------------+------------------------+ 2 rows in set (0.00 sec) mysql> show variables like 'tls%'; +-----------------+-----------------+ | Variable_name | Value | +-----------------+-----------------+ | tls_ciphersuites | | | tls_version | TLSv1.2,TLSv1.3 | +-----------------+-----------------+ 2 rows in set (0.00 sec) mysql> ALTER INSTANCE RELOAD TLS; Query OK, 0 rows affected (0.01 sec) システム全体のホスト構成を変更して、追加のTLS プロトコルを許可します。手順については、オペレーティングシステムのドキュメントを参照してくださいたとえば、TLS プロトコルを TLSv1.2 以上に制限するために、システムに次の行を含むetc/ssl/openssl.cnf ファイルがあるとします: [system_default_sect] MinProtocol = TLSv1.2 値を低いプロトコルバージョンまたはNone に変更すると、システムがより許可されます。この回避策には、低い (安全性の低い) プロトコルを許可するというデメリットがあり、セキュリティに悪影響を与える可能性がありま す。 https://dev.mysql.com/doc/refman/8.0/ja/encrypted-connection-protocols-ciphers.html tls_version の値を変更するには、サーバーの起動時に設定します。たとえば、TLSv1.1 または TLSv1.2 プロトコルを使用するが、セキュアでないTLSv1 プロトコルを使用する接続を禁止するには、サーバーmy.cnf ファイルで次の行を使用します: [mysqld] tls_version=TLSv1.1,TLSv1.2 さらに限定的にして TLSv1.2 接続のみを許可するには、次のようにtls_version を設定します: [mysqld] tls_version=TLSv1.2
EVENT	8.0.26	CHANGE	If the Event Scheduler is enabled, enabling the super_read_only system variable prevents it from updating event “last executed” timestamps in the events data dictionary table. This causes the Event Scheduler to stop the next time it tries to execute a scheduled event, after writing a message to the server error log.	mysql> show variables like 'super_read_only'; +-----------------+-------+ | Variable_name | Value | +-----------------+-------+ | super_read_only | OFF | +-----------------+-------+ 1 row in set (0.00 sec) mysql> show variables like 'event%'; +-----------------+-------+ | Variable_name | Value | +-----------------+-------+ | event_scheduler | ON | +-----------------+-------+ 1 row in set (0.00 sec) mysql>

VARIABLES	8.0.26	Newly Add	InnoDB: The new innodb_segment_reserve_factor system variable permits configuring the percentage of tablespace file segment pages that are reserved as empty pages. For more information, see Configuring the Percentage of Reserved File Segment Pages.Thanks to Facebook for the contribution. (Bug #32312743, Bug #102044) Configuring the Percentage of Reserved File Segment Pages The innodb_segment_reserve_factor variable, introduced in MySQL 8.0.26, is an advanced feature that permits defining the percentage of tablespace file segment pages reserved as empty pages. **A percentage of pages are reserved for future growth so that pages in the B-tree can be allocated contiguously**. The ability to modify the percentage of reserved pages permits fine-tuning InnoDB to address issues of data fragmentation or inefficient use of storage space. The setting is applicable to file-per-table and general tablespaces. The innodb_segment_reserve_factor default setting is 12.5 percent, which is the same percentage of pages reserved in previous MySQL releases. The innodb_segment_reserve_factor variable is dynamic and can be configured using a SET statement. For example: mysql> SET GLOBAL innodb_segment_reserve_factor=10;	mysql> show variables like 'innodb_segment_reserve_factor'; +-------------------------------+-----------+ | Variable_name | Value | +-------------------------------+-----------+ | innodb_segment_reserve_factor | 12.500000 | +-------------------------------+-----------+ 1 row in set (0.00 sec)
VARIABLES	8.0.26	Newly Add	On platforms that support fdatasync() system calls, the new innodb_use_fdatasync variable permits using fdatasync() instead of fsync() for operating system flushes. An fdatasync() system call does not flush changes to file metadata unless required for subsequent data retrieval, providing a potential performance benefit. The innodb_use_fdatasync variable can be set dynamically using a SET statement.	mysql> show variables like 'innodb_use_fdatasync'; +----------------------+-------+ | Variable_name | Value | +----------------------+-------+ | innodb_use_fdatasync | OFF | +----------------------+-------+ 1 row in set (0.00 sec)
VARIABLES	8.0.27	Deprecate	Important Change: **The default_authentication_plugin variable is deprecated as of MySQL 8.0.27; expect support for it to be removed in a future version of MySQL.** The default_authentication_plugin variable is still used in MySQL 8.0.27, but in conjunction with and at a lower precedence than the new authentication_policy system variable, which is introduced in MySQL 8.0.27 with the multifactor authentication feature. For details, see The Default Authentication Plugin. (Bug #27515356)	mysql> show variables like 'default_authentication_plugin'; +-------------------------------+-----------------------+ | Variable_name | Value | +-------------------------------+-----------------------+ | default_authentication_plugin | mysql_native_password | +-------------------------------+-----------------------+ 1 row in set (0.01 sec) mysql>
BLACKHOLE	8.0.27	Change	The BLACKHOLE storage engine maximum key length has been increased from **1000 to 3072 bytes (the same as InnoDB)**. Thanks to Adam Cable for the contribution. (Bug #32788749, Bug #103371)	The BLACKHOLE storage engine acts as a “black hole” that accepts data but throws it away and does not store it. Retrievals always return an empty result. mysql> CREATE TABLE test(i INT, c CHAR(10)) ENGINE = BLACKHOLE; Query OK, 0 rows affected (0.02 sec) mysql> INSERT INTO test VALUES(1,'record one'),(2,'record two'); Query OK, 2 rows affected (0.02 sec) Records: 2 Duplicates: 0 Warnings: 0 mysql> SELECT * FROM test; Empty set (0.00 sec) mysql>
EXPLAIN	8.0.27	Change	EXPLAIN FORMAT=TREE now shows **more precise information than displayed previously** about scans generated by the range optimizer. In particular, sub-iterators are now displayed explicitly, and are properly timed with EXPLAIN ANALYZE; index range scans now show the actual ranges being scanned. Descriptions in the output are also more user-friendly than before; for example, index_for_group_by shown for a query using DISTINCT is replaced by index skip scan for deduplication.	mysql> explain format=tree select * from tourism_and_busstop where category = '史跡' limit 10\G *************************** 1. row *************************** EXPLAIN: -> Limit: 10 row(s) (cost=3218.03 rows=10) -> Filter: (tourism_and_busstop.category = '史跡') (cost=3218.03 rows=2995) -> Table scan on tourism_and_busstop (cost=3218.03 rows=29950) 1 row in set (0.00 sec)

Performance_Schema	8.0.27	Newly Add	To assist monitoring and troubleshooting, the Performance Schema instrumentation is now used to export names of instrumented threads to the operating system. This enables utilities that display thread names, such as debuggers and the Unix ps command, to display distinct mysqld thread names rather than “mysqld”. This feature is supported only on Linux, macOS, and Windows. For more information, see The setup_instruments Table.	[root@ip-172-30-2-244 ec2-user]# ps -C mysqld H -o "pid tid cmd comm" PID TID CMD COMMAND 3677 3677 /usr/local/mysql/bin/mysqld mysqld 3677 3680 /usr/local/mysql/bin/mysqld ib_io_ibuf 3677 3681 /usr/local/mysql/bin/mysqld ib_io_log 3677 3682 /usr/local/mysql/bin/mysqld ib_io_rd-1 3677 3683 /usr/local/mysql/bin/mysqld ib_io_rd-2 3677 3684 /usr/local/mysql/bin/mysqld ib_io_rd-3 3677 3685 /usr/local/mysql/bin/mysqld ib_io_rd-4 3677 3686 /usr/local/mysql/bin/mysqld ib_io_wr-1 3677 3687 /usr/local/mysql/bin/mysqld ib_io_wr-2 3677 3688 /usr/local/mysql/bin/mysqld ib_io_wr-3 3677 3689 /usr/local/mysql/bin/mysqld ib_io_wr-4 3677 3691 /usr/local/mysql/bin/mysqld ib_pg_flush_co 3677 3693 /usr/local/mysql/bin/mysqld ib_log_checkpt 3677 3694 /usr/local/mysql/bin/mysqld ib_log_fl_notif 3677 3695 /usr/local/mysql/bin/mysqld ib_log_flush 3677 3696 /usr/local/mysql/bin/mysqld ib_log_wr_notif 3677 3697 /usr/local/mysql/bin/mysqld ib_log_writer 3677 3698 /usr/local/mysql/bin/mysqld ib_log_files_g 3677 3699 /usr/local/mysql/bin/mysqld ib_srv_lock_to 3677 3700 /usr/local/mysql/bin/mysqld ib_srv_err_mon 3677 3701 /usr/local/mysql/bin/mysqld ib_srv_mon 3677 3702 /usr/local/mysql/bin/mysqld ib_buf_resize 3677 3703 /usr/local/mysql/bin/mysqld ib_src_main 3677 3704 /usr/local/mysql/bin/mysqld ib_dict_stats 3677 3705 /usr/local/mysql/bin/mysqld ib_fts_opt 3677 3708 /usr/local/mysql/bin/mysqld xpl_worker-2 3677 3709 /usr/local/mysql/bin/mysqld xpl_worker-1 3677 3710 /usr/local/mysql/bin/mysqld xpl_accept-1 3677 3714 /usr/local/mysql/bin/mysqld ib_buf_dump 3677 3715 /usr/local/mysql/bin/mysqld ib_clone_gtid 3677 3716 /usr/local/mysql/bin/mysqld ib_srv_purge 3677 3717 /usr/local/mysql/bin/mysqld ib_srv_wkr-1 3677 3718 /usr/local/mysql/bin/mysqld ib_srv_wkr-2 3677 3719 /usr/local/mysql/bin/mysqld ib_srv_wkr-3 3677 3720 /usr/local/mysql/bin/mysqld evt_sched 3677 3721 /usr/local/mysql/bin/mysqld sig_handler 3677 3722 /usr/local/mysql/bin/mysqld xpl_accept-3 3677 3723 /usr/local/mysql/bin/mysqld xpl_accept-2 3677 3724 /usr/local/mysql/bin/mysqld gtid_zip [root@ip-172-30-2-244 ec2-user]# ^C https://dev.mysql.com/doc/refman/8.0/en/performance-schema-setup-instruments-table.html
Replication	8.0.27	Change	Replication: **Multithreading is now enabled by default for replica servers.** A multithreaded applier has a number of applier threads that execute transactions in parallel. This behavior can avoid many cases of unwanted replication lag that can cause temporary divergence between the source and replicas. The following default server settings are used to produce the multithreading behavior: replica_parallel_workers=4. This setting enables multithreading and creates four applier threads on the replica, plus a coordinator thread to manage them. If you are using multiple replication channels, each channel has this number of threads. Four applier threads provide a base level of parallelism, and you can change the setting to specify up to 1024 applier threads. replica_preserve_commit_order=1. This setting ensures that transactions are externalized on the replica in the same order as they appear in the replica's relay log, so the replica never enters a state that the master was not in, and there are no gaps in the sequence of transactions that have been executed from the relay log. replica_parallel_type=LOGICAL_CLOCK. This setting specifies that transactions that are part of the same binary log group commit on a replication source server are applied in parallel on a replica. It is required when replica_preserve_commit_order=1 is set.	root@localhost [(none)]> show variables like 'repli%'; +---+---------------+ | Variable_name | Value | +---+---------------+ | replica_allow_batching | ON | | replica_checkpoint_group | 512 | | replica_checkpoint_period | 300 | | replica_compressed_protocol | OFF | | replica_exec_mode | STRICT | | replica_load_tmpdir | /tmp | | replica_max_allowed_packet | 1073741824 | | replica_net_timeout | 60 | | replica_parallel_type | LOGICAL_CLOCK | | replica_parallel_workers | 4 | | replica_pending_jobs_size_max | 134217728 | | replica_preserve_commit_order | ON | | replica_skip_errors | OFF | | replica_sql_verify_checksum | ON | | replica_transaction_retries | 10 | | replica_type_conversions | | | replication_optimize_for_static_plugin_config | OFF | | replication_sender_observe_commit_only | OFF | +---+---------------+ 18 rows in set (0.00 sec)
Docker	8.0.27	Change	A default time zone can now be set for a server by using the server option **--default-time-zone** while starting a MySQL Server Docker container. Before, the container failed to start if the option was used.	--default-time-zone

VARIABLES	8.0.27	Newly Add	For online DDL operations, storage is usually the bottleneck. To address this issue, CPU utilization and index building has been improved. Indexes can now be built simultaneously instead of serially. Memory management has also been tightened to respect memory The number of parallel threads that can be used to scan clustered index is defined by the innodb_parallel_read_threads variable. The default setting is 4. The maximum setting is 256, which is the maximum number for all sessions. The actual number of threads that scan the clustered index is the number defined by the innodb_parallel_read_threads setting or the number of index subtrees to scan, whichever is smaller. If the thread limit is reached, sessions fall back to using a single thread. The number of parallel threads that sort and load data is controlled by the innodb_ddl_threads variable, introduced in MySQL 8.0.27. The default setting is 4. Prior to MySQL 8.0.27, sort and load operations are single-threaded. The following limitations apply: Parallel threads are not supported for building indexes that include virtual columns. Parallel threads are not supported for full-text index creation. Parallel threads are not supported for spatial index creation. Parallel scan is not supported on tables defined with virtual columns. Parallel scan is not supported on tables defined with a full-text index. Parallel scan is not supported on tables defined with a spatial index.	root@localhost [(none)]> show variables like 'innodb_ddl_%'; +-----------------------+---------+ | Variable_name | Value | +-----------------------+---------+ | innodb_ddl_buffer_size | 1048576 | | innodb_ddl_threads | 4 | +-----------------------+---------+ 2 rows in set (0.00 sec) https://dev.mysql.com/doc/refman/8.0/en/online-ddl-parallel-thread-configuration.html
character	8.0.28	Deprecate	CONVERT(string USING charset) did not compute the correct maximum length for its return value, which should be the same as that calculated for CAST(string AS charset). This meant that some conversions of strings from BINARY to nonbinary character sets which should have been rejected as invalid returned values instead. Prior to upgrading, applications that may rely on the previous CONVERT() behavior should be checked and updated as necessary. In particular, for indexes on generated columns using CONVERT() with invalid values, you should correct such values, drop the index, then re-create it before upgrading to this release. In some cases, it may be simpler to rebuild the table using ALTER TABLE table FORCE, rather than dropping and re-creating the index. See SQL Changes, for more information. (Bug #33199145)	The BINARY operator is deprecated as of MySQL 8.0.27, and you should expect its removal in a future version of MySQL. Use CAST(... AS BINARY) instead. root@localhost [(none)]> SELECT BINARY 'a' = 'A','a' = 'A', 'a' = 'a',BINARY 'a' = 'a'; +-----------------+-----------+-----------+-----------------+ | BINARY 'a' = 'A' | 'a' = 'A' | 'a' = 'a' | BINARY 'a' = 'a' | +-----------------+-----------+-----------+-----------------+ | 0 | 1 | 1 | 1 | +-----------------+-----------+-----------+-----------------+ 1 row in set, 2 warnings (0.00 sec) root@localhost [(none)]> root@localhost [(none)]> select CONVERT("test string" USING utf8mb4),cast("test string" AS CHAR CHARACTER SET utf8mb4); +-------------------------------------+---+ | CONVERT("test string" USING utf8mb4) | cast("test string" AS CHAR CHARACTER SET utf8mb4) | +-------------------------------------+---+ | test string | test string | +-------------------------------------+---+ 1 row in set (0.00 sec) root@localhost [(none)]>

Buffer Pool	8.0.28	Change	Sorts of some column types, including JSON and TEXT, sometimes exhausted the sort buffer if its size was not at least 15 times that of the largest row in the sort. **Now the sort buffer need only be only 15 times as large as the largest sort key.** (Bug #103325, Bug #105532, Bug #32738705, Bug #33501541)	root@localhost [(none)]> show variables like '%buffer%'; +-----------------------------------+----------------+ | Variable_name | Value | +-----------------------------------+----------------+ | bulk_insert_buffer_size | 8388608 | | innodb_buffer_pool_chunk_size | 134217728 | | innodb_buffer_pool_dump_at_shutdown | ON | | innodb_buffer_pool_dump_now | OFF | | innodb_buffer_pool_dump_pct | 25 | | innodb_buffer_pool_filename | ib_buffer_pool | | innodb_buffer_pool_in_core_file | ON | | innodb_buffer_pool_instances | 1 | | innodb_buffer_pool_load_abort | OFF | | innodb_buffer_pool_load_at_startup | ON | | innodb_buffer_pool_load_now | OFF | | innodb_buffer_pool_size | 134217728 | | innodb_change_buffer_max_size | 25 | | innodb_change_buffering | all | | innodb_ddl_buffer_size | 1048576 | | innodb_log_buffer_size | 16777216 | | innodb_sort_buffer_size | 1048576 | | join_buffer_size | 131072 | | key_buffer_size | 16777216 | | myisam_sort_buffer_size | 8388608 | | net_buffer_length | 16384 | | preload_buffer_size | 32768 | | read_buffer_size | 131072 | | read_rnd_buffer_size | 262144 | | select_into_buffer_size | 131072 | | sort_buffer_size | 2097152 | | sql_buffer_result | OFF | +-----------------------------------+----------------+ 27 rows in set (0.00 sec) root@localhost [(none)]>
SSL	8.0.28	Change	Support for the TLSv1 and TLSv1.1 connection protocols is removed as of MySQL 8.0.28. The protocols were deprecated from MySQL 8.0.26. For background, refer to the IETF memo Deprecating TLSv1.0 and TLSv1.1. Make connections using the more-secure TLSv1.2 and TLSv1.3 protocols. TLSv1.3 requires that both the MySQL Server software and the client application were compiled with OpenSSL 1.1.1 or higher.	root@localhost [(none)]> SHOW GLOBAL VARIABLES LIKE 'tls_version'; +---------------+-----------------+ | Variable_name | Value | +---------------+-----------------+ | tls_version | TLSv1.2,TLSv1.3 | +---------------+-----------------+ 1 row in set (0.00 sec) root@localhost [(none)]>
DDL	8.0.28	Change	InnoDB: InnoDB now supports ALTER TABLE ... RENAME COLUMN operations using ALGORITHM=INSTANT. Operations that support ALGORITHM=INSTANT only modify metadata in the data dictionary. Table data is unaffected, making the operations instantaneous. If not specified explicitly, ALGORITHM=INSTANT is used by default by DDL operations that support it.	mysql> desc T1; +---------+-------------+------+-----+---------+----------------+ | Field | Type | Null | Key | Default | Extra | +---------+-------------+------+-----+---------+----------------+ | id | int | NO | PRI | NULL | auto_increment | | note | varchar(10) | YES | | NULL | | | who | varchar(100) | YES | | NULL | | | instant | varchar(100) | YES | | NULL | | +---------+-------------+------+-----+---------+----------------+ 4 rows in set (0.01 sec) mysql> alter table T1 rename column note to memo,ALGORITHM=INSTANT, LOCK=NONE; ERROR 1221 (HY000): Incorrect usage of ALGORITHM=INSTANT and LOCK=NONE/SHARED/EXCLUSIVE mysql> alter table T1 rename column note to memo,ALGORITHM=INSTANT; Query OK, 0 rows affected (0.03 sec) Records: 0 Duplicates: 0 Warnings: 0 mysql> desc T1; +---------+-------------+------+-----+---------+----------------+ | Field | Type | Null | Key | Default | Extra | +---------+-------------+------+-----+---------+----------------+ | id | int | NO | PRI | NULL | auto_increment | | memo | varchar(10) | YES | | NULL | | | who | varchar(100) | YES | | NULL | | | instant | varchar(100) | YES | | NULL | | +---------+-------------+------+-----+---------+----------------+ 4 rows in set (0.00 sec) mysql> alter table T1 rename column memo to footnote,ALGORITHM=INPLACE, LOCK=NONE; Query OK, 0 rows affected (0.03 sec) Records: 0 Duplicates: 0 Warnings: 0 mysql>
TEMPORARY TABLE	8.0.28	Change	The **tmp_table_size variable now defines the maximum size of individual in-memory internal temporary tables** created by the TempTable storage engine. An appropriate size limit prevents individual queries from consuming an inordinate amount global TempTable resources. See Internal Temporary Table Storage Engine.	mysql> show variables like 'tmp%'; +----------------+---------+ | Variable_name | Value | +----------------+---------+ | tmp_table_size | 8388608 | | tmpdir | /tmp | +----------------+---------+ 2 rows in set (0.00 sec) mysql>

VARIABLES	8.0.28	Change	The innodb_open_files variable, which defines the number of files InnoDB can have open at one time, can now be set at runtime using a SELECT innodb_set_open_files_limit(N) statement. The statement executes a stored procedure that sets the new limit. To prevent non-LRU manged files from consuming the entire innodb_open_files limit, non-LRU managed files are now limited to 90 percent of the innodb_open_files limit, which reserves 10 percent of the innodb_open_files limit for LRU managed files. The innodb_open_files limit now includes temporary tablespace files, which were not counted toward the limit previously.	mysql> show variables like 'open_file%'; +------------------+---------+ | Variable_name | Value | +------------------+---------+ | open_files_limit | 1048576 | +------------------+---------+ 1 row in set (0.00 sec) mysql> SELECT innodb_set_open_files_limit(1048576*2); +--+ | innodb_set_open_files_limit(1048576*2) | +--+ | 2097152 | +--+ 1 row in set (0.00 sec)
TIME	8.0.28	Change	The functions FROM_UNIXTIME(), UNIX_TIMESTAMP(), and CONVERT_TZ() **now handle 64-bit values on platforms** that support them, including 64-bit versions of Linux, MacOS, and Windows. On compatible platforms, FROM_UNIXTIME() now accepts a maximum argument of 32536771199.999999 seconds, corresponding to '3001-01-18 23:59:59.999999' UTC (including the optional fraction of up to 6 digits). If the argument is larger than this, the function returns NULL.	mysql> select UNIX_TIMESTAMP(now()); +----------------------+ | UNIX_TIMESTAMP(now()) | +----------------------+ | 1668326458 | +----------------------+ 1 row in set (0.00 sec) mysql> select UNIX_TIMESTAMP('2038-01-19 03:14:07.999999'); +--+ | UNIX_TIMESTAMP('2038-01-19 03:14:07.999999') | +--+ | 2147483647.999999 | +--+ 1 row in set (0.00 sec) mysql>
Monitoring	8.0.28	Newly Add	This release introduces monitoring and limiting of memory allocation on a global and per-user basis. You can now observe the total memory consumed by all user connections by checking the value of the Global_connection_memory status variable, which must be enabled by setting global_connection_memory_tracking = 1. The total includes memory used by system processes, or by the MySQL root user, although these users are not subject to disconnection due to memory usage.	mysql> show variables like "global_connection_memory_tracking"; +----------------------------------+-------+ | Variable_name | Value | +----------------------------------+-------+ | global_connection_memory_tracking | OFF | +----------------------------------+-------+ 1 row in set (0.00 sec) mysql> show status like "Global_connection_memory"; +--------------------------+-------+ | Variable_name | Value | +--------------------------+-------+ | Global_connection_memory | 0 | +--------------------------+-------+ 1 row in set (0.00 sec) mysql> set global_connection_memory_tracking = 1; Query OK, 0 rows affected (0.00 sec) mysql> show status like "Global_connection_memory"; +--------------------------+---------+ | Variable_name | Value | +--------------------------+---------+ | Global_connection_memory | 1122912 | +--------------------------+---------+ 1 row in set (0.00 sec) mysql> **Memory used by the InnoDB buffer pool is not included in the total.** You can control indirectly how often the status variable is updated by adjusting connection_memory_chunk_size; Global_connection_memory is updated only when the total memory usage varies by more than this amount. You can specify limits on resource consumption per user connection by setting connection_memory_limit; any user whose memory usage exceeds this amount cannot issue additional queries. You can also impose a global memory limit by setting global_connection_memory_limit. Whenever Global_connection_memory exceeds the global limit, no regular users can issue new queries requiring memory usage. System users such as MySQL root are not bound by these limits.

character	8.0.29	Change	Important Note: The server now uses utf8mb3 rather than utf8 in the following cases: In the output of SHOW SQL statements (SHOW CREATE TABLE, SHOW CREATE VIEW, SHOW CREATE DATABASE) When reporting invalid strings. (Bug #33385252, Bug #33395007) The server now uses utf8mb3 in place of the alias utf8 for character set names when populating data dictionary tables from built-in character sets. This affects the display of character set and related information in the MySQL Information Schema tables listed here: CHARACTER_SETS COLLATIONS COLUMNS COLLATION_CHARACTER_SET_APPLICABILITY PARAMETERS ROUTINES SCHEMATA This change also affects the output of the SQL SHOW CHARACTER SET, SHOW COLLATION, SHOW CREATE DATABASE, and SHOW CREATE TABLE statements. (Bug #30624990)	root@localhost [mysql]> show create table plugin\G *************************** 1. row *************************** Table: plugin Create Table: CREATE TABLE `plugin` (`name` varchar(64) NOT NULL DEFAULT '', `dl` varchar(128) NOT NULL DEFAULT '', PRIMARY KEY (`name`)) /*!50100 TABLESPACE `mysql` */ ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 STATS_PERSISTENT=0 ROW_FORMAT=DYNAMIC COMMENT='MySQL plugins' 1 row in set (0.00 sec) root@localhost [mysql]>
TIMESTAMP	8.0.29	Deprecate	Important Change: Previously, MySQL allowed arbitrary delimiters and an arbitrary number of them in TIME, DATE, DATETIME, and TIMESTAMP literals, as well as an arbitrary number of whitespaces before, after, and between the date and time values in DATETIME and TIMESTAMP literals. This behavior is now deprecated, and you should expect it to be removed in a future version of MySQL.	mysql> SELECT DATE"2020/02/20"; +------------------+ | DATE"2020/02/20" | +------------------+ | 2020-02-20 | +------------------+ 1 row in set, 1 warning (0.00 sec) mysql> show warnings; +---------+------+--+ | Level | Code | Message | +---------+------+--+ | Warning | 4095 | Delimiter '/' in position 4 in datetime value '2020/02/20' at row 1 is deprecated. Prefer the standard '-'. | +---------+------+--+ 1 row in set (0.00 sec) mysql> SELECT DATE"2020-02-20"; +------------------+ | DATE"2020-02-20" | +------------------+ | 2020-02-20 | +------------------+ 1 row in set (0.00 sec)
REPLICATION	8.0.29	CHANGE	From MySQL 8.0.26, use replica_parallel_type in place of slave_parallel_type, which is deprecated from that release. In releases before MySQL 8.0.26, use slave_parallel_type. For multithreaded replicas (replicas on which replica_parallel_workers or slave_parallel_workers is set to a value greater than 0), replica_parallel_type specifies the policy used to decide which transactions are allowed to execute in parallel on the replica. replica_parallel_type is deprecated beginning with MySQL 8.0.29, as is support for parallelization of transactions using database partitioning. Expect support for these to be removed in a future release, and for **LOGICAL_CLOCK to be used exclusively thereafter.** Beginning with MySQL 8.0.30, setting this variable to 0 is deprecated, and doing so raises a warning; 0 as a permitted value for replica_parallel_workers is subject to removal in a future MySQL release; set it to 1 instead, which has the same effect.	mysql> SELECT @@replica_parallel_type,@@slave_parallel_type,@@replica_parallel_workers,@@slave_parallel_workers; SHOW WARNINGS\G +-------------------------+----------------------+---------------------------+---------------------------+ | @@replica_parallel_type | @@slave_parallel_type | @@replica_parallel_workers | @@slave_parallel_workers | +-------------------------+----------------------+---------------------------+---------------------------+ | LOGICAL_CLOCK | LOGICAL_CLOCK | 4 | 4 | +-------------------------+----------------------+---------------------------+---------------------------+ 1 row in set, 3 warnings (0.00 sec) *************************** 1. row *************************** Level: Warning Code: 1287 Message: '@@replica_parallel_type' is deprecated and will be removed in a future release. *************************** 2. row *************************** Level: Warning Code: 1287 Message: '@@slave_parallel_type' is deprecated and will be removed in a future release. Please use replica_parallel_type instead. *************************** 3. row *************************** Level: Warning Code: 1287 Message: '@@slave_parallel_workers' is deprecated and will be removed in a future release. Please use replica_parallel_workers instead. 3 rows in set (0.01 sec) mysql>
STORAGE ENGINE	8.0.29	Deprecate	The myisam_repair_threads system variable and myisamchk --parallel-recover option are deprecated; expect support for both to be removed in a future release of MySQL.	mysql> show variables like 'myisam_repair_threads'; Empty set (0.02 sec) mysql>

Buffer Pool	8.0.29	Deprecate	The server system variables query_prealloc_size and transaction_prealloc_size are now deprecated, and setting either or both of these no longer has any effect in the MySQL server. Expect them to be removed in a future MySQL release.	mysql> select @@query_prealloc_size,@@transaction_prealloc_size; +-----------------------+-----------------------------+ | @@query_prealloc_size | @@transaction_prealloc_size | +-----------------------+-----------------------------+ | 8192 | 4096 | +-----------------------+-----------------------------+ 1 row in set, 2 warnings (0.00 sec) mysql> show warnings; +---------+------+---+ | Level | Code | Message | +---------+------+---+ | Warning | 1287 | '@@query_prealloc_size' is deprecated and will be removed in a future release. | | Warning | 1287 | '@@transaction_prealloc_size' is deprecated and will be removed in a future release. | +---------+------+---+ 2 rows in set (0.00 sec) mysql>
NULL	8.0.29	BugFix	Aggregate functions based on expressions comparing values with a NULL were not ignoring the NULL correctly. (Bug #33624777, Bug #105762) https://bugs.mysql.com/bug.php?id=105762	mysql> create table t1 (a double, b double); Query OK, 0 rows affected (0.04 sec) mysql> insert into t1 values(1, NULL); Query OK, 1 row affected (0.01 sec) mysql> insert into t1 values(1, 1); Query OK, 1 row affected (0.01 sec) mysql> select sum(least(a, b)) from t1; +------------------+ | sum(least(a, b)) | +------------------+ | 1 | +------------------+ 1 row in set (0.01 sec) mysql>
DDL	8.0.29	CHANGE	An IF NOT EXISTS option is now supported for the statements CREATE FUNCTION, CREATE PROCEDURE, and CREATE TRIGGER.	CREATE PROCEDURE IF NOT EXISTS, CREATE FUNCTION IF NOT EXISTS, and CREATE TRIGGER IF NOT EXISTS are implemented in MySQL 8.0.29.
XA	8.0.29	CHANGE	Group replication in some scenarios faced problems because it was not possible to commit an XA transaction prepared on another connection. To address such issues, MySQL now supports detached XA transactions; once prepared, an XA transaction is no longer connected to the current session. This happens as part of executing XA PREPARE. The prepared XA transaction can be committed or rolled back by another connection, and the current session can then initiate another XA or local transaction without waiting for the prepared XA transaction to complete.	Check for XA Transaction Manual
DDL	8.0.29	Newly Add	InnoDB: InnoDB now supports ALTER TABLE ... DROP COLUMN operations using ALGORITHM=INSTANT. Operations that support ALGORITHM=INSTANT only modify metadata in the data dictionary. Table data is unaffected, making the operations instantaneous. If not specified explicitly, ALGORITHM=INSTANT is used by default by DDL operations that support it. Prior to MySQL 8.0.29, an instantly added column could only be added as the last column of the table. From MySQL 8.0.29, an instantly added column can be added to any position in the table.	mysql> desc t1; +-------+--------+------+-----+---------+-------+ | Field | Type | Null | Key | Default | Extra | +-------+--------+------+-----+---------+-------+ | a | double | YES | | NULL | | | b | double | YES | | NULL | | +-------+--------+------+-----+---------+-------+ 2 rows in set (0.00 sec) mysql> alter table t1 drop column b, ALGORITHM=INSTANT; Query OK, 0 rows affected (0.03 sec) Records: 0 Duplicates: 0 Warnings: 0 mysql> desc t1; +-------+--------+------+-----+---------+-------+ | Field | Type | Null | Key | Default | Extra | +-------+--------+------+-----+---------+-------+ | a | double | YES | | NULL | | +-------+--------+------+-----+---------+-------+ 1 row in set (0.01 sec) mysql>
DDL	8.0.29	Change	INSTANT is the default algorithm as of MySQL 8.0.29, and INPLACE before that.	Instant -> Inplace -> Copy

DDL	8.0.29	Change	Prior to MySQL 8.0.29, an instantly added column could only be added as the last column of the table. From MySQL 8.0.29, an instantly added column can be added to any position in the table.	mysql> desc T1; +----------+--------------+------+-----+---------+----------------+ | Field | Type | Null | Key | Default | Extra | +----------+--------------+------+-----+---------+----------------+ | id | int | NO | PRI | NULL | auto_increment | | footnote | varchar(10) | YES | | NULL | | | who | varchar(100) | YES | | NULL | | | instant | varchar(100) | YES | | NULL | | +----------+--------------+------+-----+---------+----------------+ 4 rows in set (0.00 sec) mysql> alter table T1 add column INSTANT0829 varchar(20) after who, ALGORITHM=INSTANT; Query OK, 0 rows affected (0.04 sec) Records: 0 Duplicates: 0 Warnings: 0 mysql> desc T1; +-------------+--------------+------+-----+---------+----------------+ | Field | Type | Null | Key | Default | Extra | +-------------+--------------+------+-----+---------+----------------+ | id | int | NO | PRI | NULL | auto_increment | | footnote | varchar(10) | YES | | NULL | | | who | varchar(100) | YES | | NULL | | | INSTANT0829 | varchar(20) | YES | | NULL | | | instant | varchar(100) | YES | | NULL | | +-------------+--------------+------+-----+---------+----------------+ 5 rows in set (0.00 sec) mysql>
DDL	8.0.29	Newly Add	Instantly added or dropped columns create a new row version. Up to 64 row versions are permitted. A new TOTAL_ROW_VERSIONS column was added to the INFORMATION_SCHEMA.INNODB_TABLES table to track the number of row versions.	mysql> select * from INFORMATION_SCHEMA.INNODB_TABLES where TOTAL_ROW_VERSIONS <> 0; +----------+-----------+------+--------+-------+------------+---------------+------------+-------------+--------------------+ | TABLE_ID | NAME | FLAG | N_COLS | SPACE | ROW_FORMAT | ZIP_PAGE_SIZE | SPACE_TYPE | INSTANT_COLS | TOTAL_ROW_VERSIONS | +----------+-----------+------+--------+-------+------------+---------------+------------+-------------+--------------------+ | 1266 | POC8029/t1 | 33 | 4 | 96 | Dynamic | 0 | Single | 0 | 1 | | 1255 | POC/T1 | 33 | 8 | 84 | Dynamic | 0 | Single | 0 | 1 | +----------+-----------+------+--------+-------+------------+---------------+------------+-------------+--------------------+ 2 rows in set (0.01 sec) mysql>
BINLOG	8.0.29	Change	Automatic purging of binary log files by the server is now controlled using the binlog_expire_logs_auto_purge system variable introduced in this release. By default, automatic purging is enabled (binlog_expire_logs_auto_purge set to ON); to disable it, set the value of this variable to OFF.	mysql> show variables like 'binlog_expire_logs_auto_purge'; +-------------------------------+-------+ | Variable_name | Value | +-------------------------------+-------+ | binlog_expire_logs_auto_purge | ON | +-------------------------------+-------+ 1 row in set (0.00 sec) mysql>
BINLOG	8.0.29	Change	The interval to wait before purging is controlled by binlog_expire_logs_seconds and expire_logs_days. Setting both of these system variables to 0 stops automatic purging from taking place, **even when binlog_expire_logs_auto_purge is set to ON.**	mysql> select @@binlog_expire_logs_seconds,@@expire_logs_days; +-------------------------------+--------------------+ | @@binlog_expire_logs_seconds | @@expire_logs_days | +-------------------------------+--------------------+ | 2592000 | 0 | +-------------------------------+--------------------+ 1 row in set, 1 warning (0.00 sec) mysql>
character	8.0.30	Change	Important Change: A previous change renamed character sets having deprecated names prefixed with utf8_ to use utf8mb3_ instead. In this release, we rename the utf8_ collations as well, using the utf8mb3_ prefix; this is to make the collation names consistent with those of the character sets, not to rely any longer on the deprecated collation names, and to clarify the distinction between utf8mb3 and utf8mb4.	mysql> show collation like 'utf8%'; +-----------------------------+---------+-----+---------+----------+---------+---------------+ | Collation | Charset | Id | Default | Compiled | Sortlen | Pad_attribute | +-----------------------------+---------+-----+---------+----------+---------+---------------+ | utf8mb3_bin | utf8mb3 | 83 | | Yes | 1 | PAD SPACE | | utf8mb3_croatian_ci | utf8mb3 | 213 | | Yes | 8 | PAD SPACE | | utf8mb3_czech_ci | utf8mb3 | 202 | | Yes | 8 | PAD SPACE | | utf8mb3_danish_ci | utf8mb3 | 203 | | Yes | 8 | PAD SPACE | | utf8mb3_esperanto_ci | utf8mb3 | 209 | | Yes | 8 | PAD SPACE | | utf8mb3_estonian_ci | utf8mb3 | 198 | | Yes | 8 | PAD SPACE | | utf8mb3_general_ci | utf8mb3 | 33 | Yes | Yes | 1 | PAD SPACE | | utf8mb3_general_mysql500_ci | utf8mb3 | 223 | | Yes | 1 | PAD SPACE | | utf8mb3_german2_ci | utf8mb3 | 212 | | Yes | 8 | PAD SPACE |

REPLICATION	8.0.30	Deprecate	Setting the replica_parallel_workers system variable (or the equivalent server option --replica-parallel-workers) to 0 is now deprecated, and doing so now raises a warning. To achieve the same result (that is, use single threading) without the warning, set replica_parallel_workers=1 instead.	mysql> select @@replica_parallel_workers; +----------------------------+ | @@replica_parallel_workers | +----------------------------+ | 4 | +----------------------------+ 1 row in set (0.00 sec) mysql> set global replica_parallel_workers = 0; Query OK, 0 rows affected, 1 warning (0.01 sec) mysql> show warnings; +---------+------+--+ | Level | Code | Message | +---------+------+--+ | Warning | 1287 | '0' is deprecated and will be removed in a future release. Please use 1 instead | +---------+------+--+ 1 row in set (0.00 sec) mysql> select @@replica_parallel_workers; +----------------------------+ | @@replica_parallel_workers | +----------------------------+ | 0 | +----------------------------+ 1 row in set (0.00 sec) mysql> set global replica_parallel_workers = 4; Query OK, 0 rows affected (0.00 sec) mysql> select @@replica_parallel_workers; +----------------------------+ | @@replica_parallel_workers | +----------------------------+ | 4 | +----------------------------+ 1 row in set (0.00 sec)
Cache	8.0.30	Deprecate	The --skip-host-cache server option is now deprecated, and subject to removal in a future release. Use of --skip-host-cache is similar to setting the host_cache_size system variable to 0, but host_cache_size is more flexible because it can also be used to resize, enable, or disable the host cache at runtime, not just at server startup.	mysql> select @@host_cache_size; +--------------------+ | @@host_cache_size | +--------------------+ | 228 | +--------------------+ 1 row in set (0.00 sec)

PK	8.0.30	Newly Add	Generated Invisible Primary Keys (GIPKs) MySQL 8.0.30 now supports GIPK mode, which causes a generated invisible primary key (GIPK) to be added to any InnoDB table that is created without an explicit primary key. This enhancement applies to InnoDB tables only. The definition of the generated key column added to an InnoDB table by GIPK mode is is shown here: my_row_id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT INVISIBLE PRIMARY KEY The name of the generated primary key is always **my_row_id**; you cannot, while GIPK mode is in effect, use this as a column name in a CREATE TABLE statement that creates a new InnoDB table unless it includes an explicit primary key. You cannot alter a generated invisible primary key while GIPKs are in effect, with one exception: You can toggle the visibility of the GIPK using ALTER TABLE tbl CHANGE COLUMN my_row_id SET VISIBLE and ALTER TABLE tbl CHANGE COLUMN my_row_id SET INVISIBLE. You can exclude generated invisible primary keys from the output of mysqldump using the --skip-generated-invisible-primary-key option added in this release. mysqlpump also now supports a --skip-generated-invisible-primary-key option which excludes GIPKs from its output.	mysql> SELECT @@sql_generate_invisible_primary_key; +---------------------------------------+ | @@sql_generate_invisible_primary_key | +---------------------------------------+ | 0 | +---------------------------------------+ 1 row in set (0.00 sec) mysql> CREATE TABLE auto_0 (c1 VARCHAR(50), c2 INT); Query OK, 0 rows affected (0.06 sec) mysql> show create table auto_0\G *************************** 1. row *************************** Table: auto_0 Create Table: CREATE TABLE `auto_0` (`c1` varchar(50) COLLATE utf8mb4_general_ci DEFAULT NULL, `c2` int DEFAULT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci 1 row in set (0.00 sec) mysql> SET sql_generate_invisible_primary_key=ON; Query OK, 0 rows affected (0.00 sec) mysql> SELECT @@sql_generate_invisible_primary_key; +---------------------------------------+ | @@sql_generate_invisible_primary_key | +---------------------------------------+ | 1 | +---------------------------------------+ 1 row in set (0.00 sec) mysql> CREATE TABLE auto_1 (c1 VARCHAR(50), c2 INT); Query OK, 0 rows affected (0.05 sec) mysql> show create table auto_1\G *************************** 1. row *************************** Table: auto_1 Create Table: CREATE TABLE `auto_1` (`my_row_id` bigint unsigned NOT NULL AUTO_INCREMENT /*!80023 INVISIBLE */, `c1` varchar(50) COLLATE utf8mb4_general_ci DEFAULT NULL, `c2` int DEFAULT NULL, PRIMARY KEY (`my_row_id`)) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci 1 row in set (0.00 sec) mysql> desc auto_1; +-----------+-----------------+------+-----+---------+--------------------------+ | Field | Type | Null | Key | Default | Extra | +-----------+-----------------+------+-----+---------+--------------------------+ | my_row_id | bigint unsigned | NO | PRI | NULL | auto_increment INVISIBLE | | c1 | varchar(50) | YES | | NULL | | | c2 | int | YES | | NULL | | +-----------+-----------------+------+-----+---------+--------------------------+ 3 rows in set (0.01 sec) mysql> select * from auto_1; Empty set (0.00 sec) mysql> insert into auto_1(c1,c2) values("invisible pk",2022); Query OK, 1 row affected (0.02 sec) mysql> select * from auto_1; +--------------+------+ | c1 | c2 | +--------------+------+ | invisible pk | 2022 | +--------------+------+ 1 row in set (0.00 sec) mysql> select my_row_id,c1,c2 from auto_1; +-----------+--------------+------+ | my_row_id | c1 | c2 | +-----------+--------------+------+ | 1 | invisible pk | 2022 | +-----------+--------------+------+ 1 row in set (0.00 sec) mysql> #### Default ONなのでOFFにするとSHOW CREATE TABLE等から見えなくなる。 mysql> show global variables like 'show_gipk_in_create_table_and_information_schema'; +---+-------+ | Variable_name | Value | +---+-------+ | show_gipk_in_create_table_and_information_schema | OFF | +---+-------+ 1 row in set (0.00 sec) mysql> use POC8030 Reading table information for completion of table and column names You can turn off this feature to get a quicker startup with -A Database changed mysql> desc auto_1;

SSL	8.0.30	Newly Add	It is now possible to compile the MySQL server package (mysqld + libmysql + client tools) using OpenSSL 3.0 on supported platforms, which should not change the behavior of the server or client programs. For additional information, see https://wiki.openssl.org/index.php/OpenSSL_3.0.	Suppot OpenSSL3.0
double write buffer	8.0.30	Newly Add	InnoDB: The innodb_doublewrite system variable, which enables or disables the doublewrite buffer, has two new settings, DETECT_ONLY and DETECT_AND_RECOVER. With the DETECT_ONLY setting, database page content is not written to the doublewrite buffer, and recovery does not use the doublewrite buffer to fix incomplete page writes. This lightweight setting is intended for detecting incomplete page writes only. The DETECT_AND_RECOVER setting is equivalent to the existing ON setting. For more information, see Doublewrite Buffer.	mysql> show global variables like 'innodb_doublewrite'; +--------------------+-------+ | Variable_name | Value | +--------------------+-------+ | innodb_doublewrite | ON | +--------------------+-------+ 1 row in set (0.00 sec) ON OFF DETECT_AND_RECOVER (Same as on) DETECT_ONLY (This lightweight setting is intended for detecting incomplete page writes only.)
double write buffer	8.0.30	Change	MySQL 8.0.30 onwards supports dynamic changes to the innodb_doublewrite setting that enables the doublewrite buffer, between ON, DETECT_AND_RECOVER, and DETECT_ONLY. MySQL does not support dynamic changes between a setting that enables the doublewrite buffer and OFF or vice versa. If the doublewrite buffer is located on a Fusion-io device that supports atomic writes, the doublewrite buffer is automatically disabled and data file writes are performed using Fusion-io atomic writes instead. However, be aware that the innodb_doublewrite setting is global. When the doublewrite buffer is disabled, it is disabled for all data files including those that do not reside on Fusion-io hardware. This feature is only supported on Fusion-io hardware and is only enabled for Fusion-io NVMFS on Linux. To take full advantage of this feature, an innodb_flush_method setting of O_DIRECT is recommended.	mysql> show global variables like 'innodb_doublewrite'; +--------------------+-------+ | Variable_name | Value | +--------------------+-------+ | innodb_doublewrite | ON | +--------------------+-------+ 1 row in set (0.00 sec) mysql> set global innodb_doublewrite = "DETECT_AND_RECOVER"; Query OK, 0 rows affected (0.00 sec) mysql> set global innodb_doublewrite = "DETECT_ONLY"; Query OK, 0 rows affected (0.01 sec) mysql> set global innodb_doublewrite = "OFF"; ERROR 1210 (HY000): Incorrect arguments to InnoDB: cannot change doublewrite mode to OFF if doublewrite is enabled. Please shutdown and change value to OFF mysql>
REDO LOG	8.0.30	Change and Deplicated	InnoDB: InnoDB now supports dynamic configuration of redo log capacity. The **innodb_redo_log_capacity** system variable can be set at runtime to increase or decrease the total amount of disk space occupied by redo log files. With this change, the number of redo log files and their default location has also changed. From MySQL 8.0.30, InnoDB maintains **32 redo log files in the #innodb_redo directory in the data directory**. Previously, InnoDB created two redo log files in the data directory by default, and the number and size of redo log files were controlled by the **innodb_log_files_in_group and innodb_log_file_size variables. These two variables are now deprecated.**	mysql> show variables like 'innodb_redo_%'; +------------------------------+-----------+ | Variable_name | Value | +------------------------------+-----------+ | innodb_redo_log_archive_dirs | | | innodb_redo_log_capacity | 104857600 | | innodb_redo_log_encrypt | OFF | +------------------------------+-----------+ 3 rows in set (0.01 sec) innodb_log_files_in_group * innodb_log_file_size = innodb_redo_log_capacity
Upgrade	8.0.30	BugFix	The changed tables are mysql.db, mysql.tables_priv, mysql.columns_priv and mysql.procs_priv. When you upgrade to MySQL 8.0.30 or later, these tables are modified in the second step of the MySQL upgrade process. Use the --upgrade=FORCE option when performing logical upgrades using a backup or export utility such as mysqldump or mysqlpump, which ensures that the table structures are checked and rebuilt with the new column order. (Bug #33644645, Bug #33637244)	MEMO: --upgrade=FORCE option when performing logical upgrades using a backup or export utility such as mysqldump or mysqlpump, which ensures that the table structures are checked and rebuilt with the new column order.
myisam_repair_threads	8.0.30	Deprecate	The myisam_repair_threads system variable and myisamchk --parallel-recover option were removed. (Bug #31052408)	Removed

Keyword	8.0.31	Change	Important Change: Previously, MySQL supported the use of “full” as the name of a table, column, view, stored procedure, or stored function, as well as for the alias of a table, view, or column. Beginning with this release, using “full” (regardless of letter case) in this fashion as an unquoted identifier is now deprecated, and raises a warning. This is to align more closely with the SQL standard, in which FULL is reserved as a keyword.	**Warningにならないので別途確認** mysql> CREATE TABLE full (c1 INT, c2 INT); Query OK, 0 rows affected (0.10 sec) mysql> CREATE TABLE FULL(c1 INT, c2 INT); Query OK, 0 rows affected (0.07 sec) mysql> show tables; +-------------------+ | Tables_in_POC8030 | +-------------------+ | FULL | | auto_0 | | auto_1 | | full | +-------------------+ 4 rows in set (0.00 sec) mysql>
OPTIMIZER	8.0.31	Newly Add	It is now possible to set a column histogram to a user-specified JSON value. This can be useful when sampling leaves out important values. This also means that a secondary (replica) MySQL server can assume the work of sampling the data and building the histogram, which can then be used on the primary (source) without impacting its performance.	mysql> analyze table T_Dummy update histogram on dummy_id; +-------------+----------+----------+--+ | Table | Op | Msg_type | Msg_text | +-------------+----------+----------+--+ | POC.T_Dummy | histogram | status | Histogram statistics created for column 'dummy_id'. | +-------------+----------+----------+--+ 1 row in set (4.58 sec) mysql> TABLE information_schema.column_statistics\G *************************** 1. row *************************** SCHEMA_NAME: POC TABLE_NAME: T_Dummy COLUMN_NAME: dummy_id HISTOGRAM: {"buckets": [[0, 0.06119], [1, 0.18675], [2, 0.31347], [3, 0.43734], [4, 0.55997], [5, 0.68637], [6, 0.81219], [7, 0.93801], [8, 1.0]], "data-type": "int", "null-values": 0.0, "collation-id": 8, "last-updated": "2022-12-26 09:13:44.166603", "sampling-rate": 1.0, "histogram-type": "singleton", "number-of-buckets-specified": 100} 1 row in set (0.00 sec) mysql>
VARIABLES	8.0.31	Newly Add	InnoDB: Two new status variables are provided for monitoring online buffer pool resizing operations. The Innodb_buffer_pool_resize_status_code status variable reports a status code indicating the stage of an online buffer pool resizing operation. The Innodb_buffer_pool_resize_status_progress status variable reports a percentage value indicating the progress of each stage.	mysql> show global status like 'Innodb_buffer_pool_resize_status_code'; +---------------------------------------+-------+ | Variable_name | Value | +---------------------------------------+-------+ | Innodb_buffer_pool_resize_status_code | 0 | +---------------------------------------+-------+ 1 row in set (0.01 sec) mysql> show global status like 'Innodb_buffer_pool_resize_status_progress'; +---+-------+ | Variable_name | Value | +---+-------+ | Innodb_buffer_pool_resize_status_progress | 0 | +---+-------+ 1 row in set (0.01 sec) mysql>
THREADS	8.0.31	Change	InnoDB: InnoDB now supports parallel index builds, which improves index build performance. In particular, loading sorted index entries into a B-tree is now multithreaded. Previously, this action was performed by a single thread.	mysql> show global variables like '%threads%'; +------------------------------+-------+ | Variable_name | Value | +------------------------------+-------+ | innodb_ddl_threads | 4 | | innodb_log_writer_threads | ON | | innodb_parallel_read_threads | 4 | | innodb_purge_threads | 4 | | innodb_read_io_threads | 4 | | innodb_write_io_threads | 4 | | max_delayed_threads | 20 | | max_insert_delayed_threads | 20 | | mysqlx_min_worker_threads | 2 | +------------------------------+-------+ 9 rows in set (0.01 sec) mysql>

Replication	8.0.31	Change	Replication: When replication filtering is in use, a replica no longer raises replication errors related to privilege checks or require_row_format validation for events which are filtered out. Previously, all privileges were checked in the applier, with some being checked before applying filters and others not until after; with this release, privilege checks are now deferred until after all replication filters have been applied. In addition, prior to this release, checks for require_row_format equal to 1 took place on both the receiver and the applier; now the applier alone performs this check, before any filters are evaluated. In addition, prior to this release, checks for require_row_format equal to 1 took place on both the receiver and the applier; now the applier alone performs this check, before any filters are evaluated.	この変数は、レプリケーションおよびmysqlbinlog による内部サーバー用です。セッションで実行される DML イベントを、 行ベースのバイナリロギング形式でエンコードされたイベントのみに制限し、一時テーブルを作成することはできませ/制限を尊重しないクエリは失敗します。 root@localhost [mysql]> show global variables like '%require_row_format%'; Empty set (0.28 sec) root@localhost [mysql]> show variables like '%require_row_format%'; +--------------------+-------+ | Variable_name | Value | +--------------------+-------+ | require_row_format | OFF | +--------------------+-------+ 1 row in set (0.02 sec)