

information\_schema\_stats\_expiry can be set to 0 to have INFORMATION\_SCHEMA queries retrieve the latest statistics directly from the storage engine, which is not as fast as retrieving cached statistics.

For more information, see Section 8.2.3, “Optimizing INFORMATION\_SCHEMA Queries” .

INFORMATION\_SCHEMA tables in MySQL 8.0 are closely tied to the data dictionary, resulting in several usage differences. See [Section 14.7, “Data Dictionary Usage Differences”](#_bookmark1) .

**14.6** **Serialized** **Dictionary** **Information** **(SDI)**

In addition to storing metadata about database objects in the data dictionary, MySQL stores it in serialized form. This data is referred to as serialized dictionary information (SDI). InnoDB stores SDI data within its tablespace files. NDBCLUSTER stores SDI data in the NDB dictionary. Other storage engines store SDI data in .sdi files that are created for a given table in the table's database directory. SDI data is generated in a compact JSON format.

Serialized dictionary information (SDI) is present in all InnoDB tablespace files except for temporary tablespace and undo tablespace files. SDI records in an InnoDB tablespace file only describe table and tablespace objects contained within the tablespace.

SDI data is updated by DDL operations on a table or CHECK TABLE FOR UPGRADE. SDI data is not updated when the MySQL server is upgraded to a new release or version.

The presence of SDI data provides metadata redundancy. For example, if the data dictionary becomes unavailable, object metadata can be extracted directly from InnoDB tablespace files using the ibd2sdi tool.

For InnoDB, an SDI record requires a single index page, which is 16KB in size by default. However, SDI data is compressed to reduce the storage footprint.

For partitioned InnoDB tables comprised of multiple tablespaces, SDI data is stored in the tablespace file of the first partition.

The MySQL server uses an internal API that is accessed during DDL operations to create and maintain SDI records.

The IMPORT TABLE statement imports MyISAM tables based on information contained in .sdi files. For more information, see Section 13.2.6, “IMPORT TABLE Statement” .

**14.7** **Data** **Dictionary** **Usage** **Differences**

Use of a data dictionary-enabled MySQL server entails some operational differences compared to a server that does not have a data dictionary:

• Previously, enabling the innodb\_read\_only system variable prevented creating and dropping

tables only for the InnoDB storage engine. As of MySQL 8.0, enabling innodb\_read\_only prevents these operations for all storage engines. Table creation and drop operations for any storage engine modify data dictionary tables in the mysql system database, but those tables use the InnoDB storage engine and cannot be modified when innodb\_read\_only is enabled. The same principle applies to other table operations that require modifying data dictionary tables. Examples:

• ANALYZE TABLE fails because it updates table statistics, which are stored in the data dictionary.

• ALTER TABLE *tbl\_name* ENGINE=*engine\_name* fails because it updates the storage engine designation, which is stored in the data dictionary.

**Note**

Enabling innodb\_read\_only also has important implications for non- data dictionary tables in the mysql system database. For details, see the

description of innodb\_read\_only in Section 15.14, “InnoDB Startup Options and System Variables”

• Previously, tables in the mysql system database were visible to DML and DDL statements. As of MySQL 8.0, data dictionary tables are invisible and cannot be modified or queried directly. However, in most cases there are corresponding INFORMATION\_SCHEMA tables that can be queried instead. This enables the underlying data dictionary tables to be changed as server development proceeds, while maintaining a stable INFORMATION\_SCHEMA interface for application use.

• INFORMATION\_SCHEMA tables in MySQL 8.0 are closely tied to the data dictionary, resulting in several usage differences:

• Previously, INFORMATION\_SCHEMA queries for table statistics in the STATISTICS and TABLES tables retrieved statistics directly from storage engines. As of MySQL 8.0, cached table statistics are used by default. The information\_schema\_stats\_expiry system variable defines the period of time before cached table statistics expire. The default is 86400 seconds (24 hours). (To update the cached values at any time for a given table, use ANALYZE TABLE.) If there are no cached statistics or statistics have expired, statistics are retrieved from storage engines when querying table statistics columns. To always retrieve the latest statistics directly from storage engines, set information\_schema\_stats\_expiry to 0. For more information, see Section 8.2.3, “Optimizing INFORMATION\_SCHEMA Queries” .

• Several INFORMATION\_SCHEMA tables are views on data dictionary tables, which enables the optimizer to use indexes on those underlying tables. Consequently, depending on optimizer choices, the row order of results for INFORMATION\_SCHEMA queries might differ from previous results. If a query result must have specific row ordering characteristics, include an ORDER BY clause.

• Queries on INFORMATION\_SCHEMA tables may return column names in a different lettercase

than in earlier MySQL series. Applications should test result set column names in case-insensitive fashion. If that is not feasible, a workaround is to use column aliases in the select list that return column names in the required lettercase. For example:

SELECT TABLE\_SCHEMA AS table\_schema, TABLE\_NAME AS table\_name

FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'users';

• mysqldump and mysqlpump no longer dump the INFORMATION\_SCHEMA database, even if explicitly named on the command line.

• CREATE TABLE *dst\_tbl* LIKE *src\_tbl* requires that *src\_tbl* be a base table and fails if it is an INFORMATION\_SCHEMA table that is a view on data dictionary tables.

• Previously, result set headers of columns selected from INFORMATION\_SCHEMA tables used the capitalization specified in the query. This query produces a result set with a header of table\_name:

SELECT table\_name FROM INFORMATION\_SCHEMA.TABLES;

As of MySQL 8.0, these headers are capitalized; the preceding query produces a result set with a header of TABLE\_NAME. If necessary, a column alias can be used to achieve a different lettercase. For example:

SELECT table\_name AS 'table\_name' FROM INFORMATION\_SCHEMA.TABLES;

• The data directory affects how mysqldump and mysqlpump dump information from the mysql system database:

• Previously, it was possible to dump all tables in the mysql system database. As of MySQL 8.0, mysqldump and mysqlpump dump only non-data dictionary tables in that database.

• Previously, the --routines and --events options were not required to include stored routines and events when using the --all-databases option: The dump included the mysql system database, and therefore also the proc and event tables containing stored routine and event definitions. As of MySQL 8.0, the event and proc tables are not used. Definitions for the corresponding objects are stored in data dictionary tables, but those tables are not dumped. To include stored routines and events in a dump made using --all-databases, use the -- routines and --events options explicitly.

• Previously, the --routines option required the SELECT privilege for the proc table. As of MySQL 8.0, that table is not used; --routines requires the global SELECT privilege instead.

• Previously, it was possible to dump stored routine and event definitions together with their creation and modification timestamps, by dumping the proc and event tables. As of MySQL 8.0, those tables are not used, so it is not possible to dump timestamps.

• Previously, creating a stored routine that contains illegal characters produced a warning. As of MySQL 8.0, this is an error.

**14.8** **Data** **Dictionary** **Limitations**

This section describes temporary limitations introduced with the MySQL data dictionary.

• Manual creation of database directories under the data directory (for example, with mkdir) is unsupported. Manually created database directories are not recognized by the MySQL Server.

• DDL operations take longer due to writing to storage, undo logs, and redo logs instead of .frm files.

Chapter 15 The InnoDB Storage Engine

**Table** **of** **Contents**

[15.1 Introduction InnoDB](#_bookmark2)to [3032](#_bookmark2)

15.1.1 Benefits of Using InnoDB Tables 3034

15.1.2 Best Practices for InnoDB Tables 3035

15.1.3 Verifying that InnoDB is the Default Storage Engine 3035

15.1.4 Testing and Benchmarking with InnoDB 3036

15.2 InnoDB and the ACID Model 3036

15.3 InnoDB Multi-Versioning 3037

15.4 InnoDB Architecture 3039

15.5 InnoDB In-Memory Structures 3039

15.5.1 Buffer Pool 3040

15.5.2 Change Buffer 3045

15.5.3 Adaptive Hash Index 3049

15.5.4 Log Buffer 3050

15.6 InnoDB On-Disk Structures 3050

15.6.1 Tables 3050

15.6.2 Indexes 3074

15.6.3 Tablespaces 3081

15.6.4 Doublewrite Buffer 3104

15.6.5 Redo Log 3105

15.6.6 Undo Logs 3112

15.7 InnoDB Locking and Transaction Model 3113

15.7.1 InnoDB Locking 3114

15.7.2 InnoDB Transaction Model 3118

15.7.3 Locks Set by Different SQL Statements in InnoDB 3127

15.7.4 Phantom Rows 3130

15.7.5 Deadlocks in InnoDB 3131

15.7.6 Transaction Scheduling 3136

15.8 InnoDB Configuration 3137

15.8.1 InnoDB Startup Configuration 3137

15.8.2 Configuring InnoDB for Read-Only Operation 3143

15.8.3 InnoDB Buffer Pool Configuration 3145

15.8.4 Configuring Thread Concurrency for InnoDB 3159

15.8.5 Configuring the Number of Background InnoDB I/O Threads 3160

15.8.6 Using Asynchronous I/O on Linux 3161

15.8.7 Configuring InnoDB I/O Capacity 3161

15.8.8 Configuring Spin Lock Polling 3163

15.8.9 Purge Configuration 3164

15.8.10 Configuring Optimizer Statistics for InnoDB 3165

15.8.11 Configuring the Merge Threshold for Index Pages 3176

15.8.12 Enabling Automatic Configuration for a Dedicated MySQL Server 3178

15.9 InnoDB Table and Page Compression 3181

15.9.1 InnoDB Table Compression 3181

15.9.2 InnoDB Page Compression 3195

15.10 InnoDB Row Formats 3198

15.11 InnoDB Disk I/O and File Space Management 3204

15.11.1 InnoDB Disk I/O 3205

15.11.2 File Space Management 3205

15.11.3 InnoDB Checkpoints 3207

15.11.4 Defragmenting a Table 3207

15.11.5 Reclaiming Disk Space with TRUNCATE TABLE 3208

15.12 InnoDB and Online DDL 3208

15.12.1 Online DDL Operations 3209

15.12.2 Online DDL Performance and Concurrency 3224

15.12.3 Online DDL Space Requirements 3227

15.12.4 Online DDL Memory Management 3228

15.12.5 Configuring Parallel Threads for Online DDL Operations 3228

15.12.6 Simplifying DDL Statements with Online DDL 3229

15.12.7 Online DDL Failure Conditions 3229

15.12.8 Online DDL Limitations 3230

15.13 InnoDB Data-at-Rest Encryption 3230

15.14 InnoDB Startup Options and System Variables 3239

15.15 InnoDB INFORMATION\_SCHEMA Tables 3329

15.15.1 InnoDB INFORMATION\_SCHEMA Tables about Compression 3330

15.15.2 InnoDB INFORMATION\_SCHEMA Transaction and Locking Information 3331

15.15.3 InnoDB INFORMATION\_SCHEMA Schema Object Tables 3338

15.15.4 InnoDB INFORMATION\_SCHEMA FULLTEXT Index Tables 3343

15.15.5 InnoDB INFORMATION\_SCHEMA Buffer Pool Tables 3346

15.15.6 InnoDB INFORMATION\_SCHEMA Metrics Table 3350

15.15.7 InnoDB INFORMATION\_SCHEMA Temporary Table Info Table 3360

15.15.8 Retrieving InnoDB Tablespace Metadata from INFORMATION\_SCHEMA.FILES 3360

15.16 InnoDB Integration with MySQL Performance Schema 3362

15.16.1 Monitoring ALTER TABLE Progress for InnoDB Tables Using Performance Schema 3363

15.16.2 Monitoring InnoDB Mutex Waits Using Performance Schema 3365

15.17 InnoDB Monitors 3369

15.17.1 InnoDB Monitor Types 3369

15.17.2 Enabling InnoDB Monitors 3369

15.17.3 InnoDB Standard Monitor and Lock Monitor Output 3371

15.18 InnoDB Backup and Recovery 3375

15.18.1 InnoDB Backup 3375

15.18.2 InnoDB Recovery 3376

15.19 InnoDB and MySQL Replication 3378

15.20 InnoDB memcached Plugin 3380

15.20.1 Benefits of the InnoDB memcached Plugin 3381

15.20.2 InnoDB memcached Architecture 3382

15.20.3 Setting Up the InnoDB memcached Plugin 3383

15.20.4 InnoDB memcached Multiple get and Range Query Support 3388

15.20.5 Security Considerations for the InnoDB memcached Plugin 3391

15.20.6 Writing Applications for the InnoDB memcached Plugin 3392

15.20.7 The InnoDB memcached Plugin and Replication 3404

15.20.8 InnoDB memcached Plugin Internals 3408

15.20.9 Troubleshooting the InnoDB memcached Plugin 3412

15.21 InnoDB Troubleshooting 3414

15.21.1 Troubleshooting InnoDB I/O Problems 3415

15.21.2 Troubleshooting Recovery Failures 3415

15.21.3 Forcing InnoDB Recovery 3416

15.21.4 Troubleshooting InnoDB Data Dictionary Operations 3417

15.21.5 InnoDB Error Handling 3418

15.22 InnoDB Limits 3419

15.23 InnoDB Restrictions and Limitations 3420

**15.1** **Introduction** **to** **InnoDB**

InnoDB is a general-purpose storage engine that balances high reliability and high performance. In MySQL 8.0, InnoDB is the default MySQL storage engine. Unless you have configured a different default storage engine, issuing a CREATE TABLE statement without an ENGINE clause creates an InnoDB table.