**Key** **Advantages** **of** **InnoDB**

• Its DML operations follow the ACID model, with transactions featuring commit, rollback, and crash- recovery capabilities to protect user data. See [Section 15.2, “InnoDB and the ACID Model”](#_bookmark1) .

• Row-level locking and Oracle-style consistent reads increase multi-user concurrency and performance. See Section 15.7, “InnoDB Locking and Transaction Model” .

• InnoDB tables arrange your data on disk to optimize queries based on primary keys. Each InnoDB table has a primary key index called the clustered index that organizes the data to minimize I/O for primary key lookups. See Section 15.6.2.1, “Clustered and Secondary Indexes” .

• To maintain data integrity, InnoDB supports FOREIGN KEY constraints. With foreign keys, inserts, updates, and deletes are checked to ensure they do not result in inconsistencies across related

tables. See Section 13. 1.20.5, “FOREIGN KEY Constraints” .

**Table** **15.1** **InnoDB** **Storage** **Engine** **Features**

|  |  |
| --- | --- |
| **Feature** | **Support** |
| **B-tree** **indexes** | Yes |
| **Backup/point-in-time** **recovery** (Implemented in the server, rather than in the storage engine.) | Yes |
| **Cluster** **database** **support** | No |
| **Clustered** **indexes** | Yes |
| **Compressed** **data** | Yes |
| **Data** **caches** | Yes |
| **Encrypted** **data** | Yes (Implemented in the server via encryption functions; In MySQL 5.7 and later, data-at-rest encryption is supported.) |
| **Foreign** **key** **support** | Yes |
| **Full-text** **search** **indexes** | Yes (Support for FULLTEXT indexes is available in MySQL 5.6 and later.) |
| **Geospatial** **data** **type** **support** | Yes |
| **Geospatial** **indexing** **support** | Yes (Support for geospatial indexing is available in MySQL 5.7 and later.) |
| **Hash** **indexes** | No (InnoDB utilizes hash indexes internally for its Adaptive Hash Index feature.) |
| **Index** **caches** | Yes |
| **Locking** **granularity** | Row |
| **MVCC** | Yes |
| **Replication** **support** (Implemented in the server, rather than in the storage engine.) | Yes |
| **Storage** **limits** | 64TB |
| **T-tree** **indexes** | No |
| **Transactions** | Yes |
| **Update** **statistics** **for** **data** **dictionary** | Yes |

To compare the features of InnoDB with other storage engines provided with MySQL, see the *Storage* *Engine* *Features* table in Chapter 16, *Alternative* *Storage* *Engines*.

**InnoDB** **Enhancements** **and** **New** **Features**

For information about InnoDB enhancements and new features, refer to:

• The InnoDB enhancements list in Section 1.3, “What Is New in MySQL 8.0” .

• The [Release Notes](https://dev.mysql.com/doc/relnotes/mysql/8.0/en/).

**Additional** **InnoDB** **Information** **and** **Resources**

• For InnoDB-related terms and definitions, see the MySQL Glossary.

• For a forum dedicated to the InnoDB storage engine, see [MySQL Forums::InnoDB](http://forums.mysql.com/list.php?22).

• InnoDB is published under the same GNU GPL License Version 2 (of June 1991) as MySQL. For more information on MySQL licensing, see <http://www.mysql.com/company/legal/licensing/>.

**15.1.1** **Benefits** **of** **Using** **InnoDB** **Tables**

InnoDB tables have the following benefits:

• If the server unexpectedly exits because of a hardware or software issue, regardless of what was happening in the database at the time, you don't need to do anything special after restarting the database. InnoDB crash recovery automatically finalizes changes that were committed before the time of the crash, and undoes changes that were in process but not committed, permitting you to restart and continue from where you left off. See Section 15.18.2, “InnoDB Recovery” .

• The InnoDB storage engine maintains its own buffer pool that caches table and index data in main memory as data is accessed. Frequently used data is processed directly from memory. This cache applies to many types of information and speeds up processing. On dedicated database servers, up to 80% of physical memory is often assigned to the buffer pool. See Section 15.5.1, “Buffer Pool” .

• If you split up related data into different tables, you can set up foreign keys that enforce referential integrity. See Section 13.1.20.5, “FOREIGN KEY Constraints” .

• If data becomes corrupted on disk or in memory, a checksum mechanism alerts you to the bogus data before you use it. The innodb\_checksum\_algorithm variable defines the checksum algorithm used by InnoDB.

• When you design a database with appropriate primary key columns for each table, operations involving those columns are automatically optimized. It is very fast to reference the primary key columns in WHERE clauses, ORDER BY clauses, GROUP BY clauses, and join operations. See

Section 15.6.2. 1, “Clustered and Secondary Indexes” .

• Inserts, updates, and deletes are optimized by an automatic mechanism called change buffering. InnoDB not only allows concurrent read and write access to the same table, it caches changed data to streamline disk I/O. See Section 15.5.2, “Change Buffer” .

• Performance benefits are not limited to large tables with long-running queries. When the same rows are accessed over and over from a table, the Adaptive Hash Index takes over to make these lookups even faster, as if they came out of a hash table. See Section 15.5.3, “Adaptive Hash Index” .

• You can compress tables and associated indexes. See Section 15.9, “InnoDB Table and Page Compression” .

• You can encrypt your data. See Section 15.13, “InnoDB Data-at-Rest Encryption” .

• You can create and drop indexes and perform other DDL operations with much less impact on performance and availability. See Section 15.12.1, “Online DDL Operations” .

• Truncating a file-per-table tablespace is very fast and can free up disk space for the operating system to reuse rather than only InnoDB. See Section 15.6.3.2, “File-Per-Table Tablespaces” .

• The storage layout for table data is more efficient for BLOB and long text fields, with the DYNAMIC row format. See Section 15.10, “InnoDB Row Formats” .

• You can monitor the internal workings of the storage engine by querying INFORMATION\_SCHEMA tables. See Section 15.15, “InnoDB INFORMATION\_SCHEMA Tables” .

• You can monitor the performance details of the storage engine by querying Performance Schema tables. See Section 15.16, “InnoDB Integration with MySQL Performance Schema” .

• You can mix InnoDB tables with tables from other MySQL storage engines, even within the same statement. For example, you can use a join operation to combine data from InnoDB and MEMORY

tables in a single query.

• InnoDB has been designed for CPU efficiency and maximum performance when processing large data volumes.

• InnoDB tables can handle large quantities of data, even on operating systems where file size is limited to 2GB.

For InnoDB-specific tuning techniques you can apply to your MySQL server and application code, see Section 8.5, “Optimizing for InnoDB Tables” .

**15.1.2** **Best** **Practices** **for** **InnoDB** **Tables**

This section describes best practices when using InnoDB tables.

• Specify a primary key for every table using the most frequently queried column or columns, or an auto-increment value if there is no obvious primary key.

• Use joins wherever data is pulled from multiple tables based on identical ID values from those tables. For fast join performance, define foreign keys on the join columns, and declare those columns with the same data type in each table. Adding foreign keys ensures that referenced columns are indexed, which can improve performance. Foreign keys also propagate deletes and updates to all affected tables, and prevent insertion of data in a child table if the corresponding IDs are not present in the parent table.

• Turn off autocommit. Committing hundreds of times a second puts a cap on performance (limited by the write speed of your storage device).

• Group sets of related DML operations into transactions by bracketing them with START TRANSACTION and COMMIT statements. While you don't want to commit too often, you also don't want to issue huge batches of INSERT, UPDATE, or DELETE statements that run for hours without committing.

• Do not use LOCK TABLES statements. InnoDB can handle multiple sessions all reading and writing to the same table at once without sacrificing reliability or high performance. To get exclusive write access to a set of rows, use the SELECT ... FOR UPDATE syntax to lock just the rows you intend to update.

• Enable the innodb\_file\_per\_table variable or use general tablespaces to put the data and indexes for tables into separate files instead of the system tablespace. The innodb\_file\_per\_table variable is enabled by default.

• Evaluate whether your data and access patterns benefit from the InnoDB table or page compression features. You can compress InnoDB tables without sacrificing read/write capability.

• Run the server with the --sql\_mode=NO\_ENGINE\_SUBSTITUTION option to prevent tables from being created with storage engines that you do not want to use.

**15.1.3** **Verifying** **that** **InnoDB** **is** **the** **Default** **Storage** **Engine**

Issue the SHOW ENGINES statement to view the available MySQL storage engines. Look for DEFAULT in the SUPPORT column.

mysql> SHOW ENGINES; Alternatively, query the Information Schema ENGINES table. mysql> SELECT \* FROM INFORMATION\_SCHEMA.ENGINES;

**15.1.4** **Testing** **and** **Benchmarking** **with** **InnoDB**

If InnoDB is not the default storage engine, you can determine if your database server and applications work correctly with InnoDB by restarting the server with --default-storage-engine=InnoDB defined on the command line or with default-storage-engine=innodb defined in the [mysqld] section of the MySQL server option file.

Since changing the default storage engine only affects newly created tables, run your application installation and setup steps to confirm that everything installs properly, then exercise the application features to make sure the data loading, editing, and querying features work. If a table relies on a feature that is specific to another storage engine, you receive an error. In this case, add the ENGINE=*other\_engine\_name* clause to the CREATE TABLE statement to avoid the error.

If you did not make a deliberate decision about the storage engine, and you want to preview how certain tables work when created using InnoDB, issue the command ALTER TABLE table\_name ENGINE=InnoDB; for each table. Alternatively, to run test queries and other statements without disturbing the original table, make a copy:

CREATE TABLE ... ENGINE=InnoDB AS SELECT \* FROM *other\_engine\_table*;

To assess performance with a full application under a realistic workload, install the latest MySQL server and run benchmarks.

Test the full application lifecycle, from installation, through heavy usage, and server restart. Kill the server process while the database is busy to simulate a power failure, and verify that the data is recovered successfully when you restart the server.

Test any replication configurations, especially if you use different MySQL versions and options on the source server and replicas.

**15.2** **InnoDB** **and** **the** **ACID** **Model**

The ACID model is a set of database design principles that emphasize aspects of reliability that are important for business data and mission-critical applications. MySQL includes components such as the InnoDB storage engine that adhere closely to the ACID model so that data is not corrupted and results are not distorted by exceptional conditions such as software crashes and hardware malfunctions. When you rely on ACID-compliant features, you do not need to reinvent the wheel of consistency checking and crash recovery mechanisms. In cases where you have additional software safeguards, ultra- reliable hardware, or an application that can tolerate a small amount of data loss or inconsistency, you can adjust MySQL settings to trade some of the ACID reliability for greater performance or throughput.

The following sections discuss how MySQL features, in particular the InnoDB storage engine, interact with the categories of the ACID model:

• **A**: atomicity.

• **C**: consistency.

• **I:**: isolation.

• **D**: durability.

**Atomicity**

The **atomicity** aspect of the ACID model mainly involves InnoDB transactions. Related MySQL features include:

• The autocommit setting.

• The COMMIT statement.

• The ROLLBACK statement.

**Consistency**

The **consistency** aspect of the ACID model mainly involves internal InnoDB processing to protect data from crashes. Related MySQL features include:

• The InnoDB doublewrite buffer. See Section 15.6.4, “Doublewrite Buffer” .

• InnoDB crash recovery. See InnoDB Crash Recovery.

**Isolation**

The **isolation** aspect of the ACID model mainly involves InnoDB transactions, in particular the isolation level that applies to each transaction. Related MySQL features include:

• The autocommit setting.

• Transaction isolation levels and the SET TRANSACTION statement. See Section 15.7.2.1, “Transaction Isolation Levels” .

• The low-level details of InnoDB locking. Details can be viewed in the INFORMATION\_SCHEMA tables (see Section 15. 15.2, “InnoDB INFORMATION\_SCHEMA Transaction and Locking Information” ) and

Performance Schema data\_locks and data\_lock\_waits tables.

**Durability**

The **durability** aspect of the ACID model involves MySQL software features interacting with your particular hardware configuration. Because of the many possibilities depending on the capabilities of your CPU, network, and storage devices, this aspect is the most complicated to provide concrete guidelines for. (And those guidelines might take the form of “buy new hardware” .) Related MySQL features include:

• The InnoDB doublewrite buffer. See Section 15.6.4, “Doublewrite Buffer” .

• The innodb\_flush\_log\_at\_trx\_commit variable.

• The sync\_binlog variable.

• The innodb\_file\_per\_table variable.

• The write buffer in a storage device, such as a disk drive, SSD, or RAID array.

• A battery-backed cache in a storage device.

• The operating system used to run MySQL, in particular its support for the fsync() system call.

• An uninterruptible power supply (UPS) protecting the electrical power to all computer servers and storage devices that run MySQL servers and store MySQL data.

• Your backup strategy, such as frequency and types of backups, and backup retention periods.

• For distributed or hosted data applications, the particular characteristics of the data centers where the hardware for the MySQL servers is located, and network connections between the data centers.

**15.3** **InnoDB** **Multi-Versioning**