MySQL Storage Engines

1. InnoDB

Definition

This is the default storage engine for MySQL 5.5 and higher. It provides transaction-safe (ACID compliant) tables, supports FOREIGN KEY referential-integrity constraints. It supports commit, rollback, and crash-recovery capabilities to protect data. It also support row-level locking. It's "consistent nonlocking reads" increases performance when used in a multiuser environment. It stores data in clustered indexes which reduces I/O for queries based on primary keys.

InnoDB is a storage engine for MySQL that balances high reliability and high performance. As of MySQL 5.5 and later, it is the default storage engine.

Use Cases

?

Advantages

* InnoDB has maximum performance when processing large data volumes.
* Its DML operations (add, update and delete data) is ACID (atomic, consistent, isolated and durable) model compatible, with transactions featuring commit, rollback, and crash-recovery capabilities to protect user data.
* Row-level locking (locks are placed on single records (rows)) system increase multi-user concurrency and performance. All InnoDB locks held by a transaction are released when the transaction is committed or aborted.
* InnoDB tables arrange your data on disk to optimize queries based on primary keys.
* InnoDB supports FOREIGN KEY constraints to maintain data integrity. Therefore inserts, updates, and deletes are all checked to ensure they do not result in inconsistencies across different tables.
* It is possible to mix InnoDB tables with tables from other MySQL storage engines within the same statement. For example, you can use a join operation to combine data from InnoDB and MEMORY tables in a single query.

Disadvantages/Limitation

* InnoDB permits a foreign key to reference any index column or group of columns. However, in the referenced table, there must be an index where the referenced columns are listed as the first columns in the same order.
* InnoDB does not currently support foreign keys for tables with user-defined partitioning. This means that no user-partitioned InnoDB table may contain foreign key references or columns referenced by foreign keys.
* InnoDB allows a foreign key constraint to reference a non-unique key. This is an InnoDB extension to standard SQL.
* Maximum 1017 columns are allowed in a table (raised in MySQL 5.6.9 from the earlier limit of 1000).
* Maximum 64 secondary indexes are allowed in a table. Secondary indexes is a type of InnoDB index that represents a subset of table columns.
* By default, an index key for a single-column index can be up to 767 bytes. The same length limit applies to any index key prefix.
* The InnoDB internal maximum key length is 3500 bytes, but MySQL itself restricts this to 3072 bytes (combined index key in a multi-column index).
* The maximum row length except for variable-length columns (VARBINARY, VARCHAR, BLOB and TEXT), is about 8000 bytes for the default page size of 16KB.
* Internally InnoDB supports row sizes larger than 65,535 bytes, but MySQL itself imposes a row-size limit of 65,535 for the combined size of all columns.
* The maximum table space size is four billion database pages (64TB) and the minimum table space size is slightly larger than 10MB.

**Table 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 |

1. MyISAM

Definition

This storage engine, manages non transactional tables, provides high-speed storage and retrieval, supports full text searching.

MyISAM storage engine is based on the older ISAM storage engine (not available now) but has many useful extensions.

These tables have a small footprint. [Table-level locking](https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_table_lock) limits the performance in read/write workloads, so it is often used in read-only or read-mostly workloads in Web and data warehousing configurations.

MyISAM is based on the older (and no longer available) ISAM storage engine but has many useful extensions.

Use Cases

Characteristics

* Up to 63-bit file length large files are supported on file systems and operating systems that support large files.
* (232)2(1.844E+19) rows are allowed in a MyISAM table.
* Maximum 64 number of indexes and 16 number of columns per index are allowed.
* The maximum key length is 1000 bytes.
* Internal handling of one AUTO\_INCREMENT column per table is supported.
* You can put the data file and index file in different directories on different physical devices to get more speed with the DATA DIRECTORY and INDEX DIRECTORY table options to CREATE TABLE
* BLOB and TEXT columns can be indexed.
* NULL values are permitted in indexed columns. This takes 0 to 1 bytes per key.
* Each character column can have a different character set.
* Support for a true VARCHAR type; a VARCHAR column starts with a length stored in one or two bytes.
* Tables with VARCHAR columns may have fixed or dynamic row length.
* The sum of the lengths of the VARCHAR and CHAR columns in a table may be up to 64KB.
* Arbitrary length UNIQUE constraints.

Disadvantages/Corruption Events

MyISAM table format is very reliable, but in some occasion you can get corrupted tables if any of the following events occur:

* The mysqld (Known as MySQL Server) process is killed in the middle of a write.
* Hardware failures.
* An unexpected computer shutdown occurs.
* Using an external program to modify a table
* A software bug in the MySQL or MyISAM code.

**Table MyISAM 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** | No |
| **Compressed data** | Yes (Compressed MyISAM tables are supported only when using the compressed row format. Tables using the compressed row format with MyISAM are read only.) |
| **Data caches** | No |
| **Encrypted data** | Yes (Implemented in the server via encryption functions.) |
| **Foreign key support** | No |
| **Full-text search indexes** | Yes |
| **Geospatial data type support** | Yes |
| **Geospatial indexing support** | Yes |
| **Hash indexes** | No |
| **Index caches** | Yes |
| **Locking granularity** | Table |
| **MVCC** | No |
| **Replication support** (Implemented in the server, rather than in the storage engine.) | Yes |
| **Storage limits** | 256TB |
| **T-tree indexes** | No |
| **Transactions** | No |
| **Update statistics for data dictionary** | Yes |

1. Memory

Definition

Provides in-memory tables, formerly known as HEAP. It sores all data in RAM for faster access than storing data on disks. Useful for quick looks up of reference and other identical data.

The MEMORY storage engine creates tables that are stored in memory. Because the data can be crashed due to hardware or power issues, you can only use these tables as temporary work areas or read-only caches for data pulled from other tables. When the MySQL server halts or restarts, the data in MEMORY tables is lost.

The MEMORY storage engine (formerly known as HEAP) creates special-purpose tables with contents that are stored in memory. Because the data is vulnerable to crashes, hardware issues, or power outages, only use these tables as temporary work areas or read-only caches for data pulled from other tables.

**When to Use MEMORY storage engine:**

* Operations involving transient, non-critical data such as session management or caching.
* In-memory storage for fast access and low latency. Data volume can fit entirely in memory without causing the operating system to swap out virtual memory pages.
* By default, an index key for a single-column index can be up to 767 bytes. The same length limit applies to any index key prefix.
* The InnoDB internal maximum key length is 3500 bytes, but MySQL itself restricts this to 3072 bytes (combined index key in a multi-column index).
* The maximum row length except for variable-length columns (VARBINARY, VARCHAR, BLOB and TEXT), is about 8000 bytes for the default page size of 16KB.
* Internally InnoDB supports row sizes larger than 65,535 bytes, but MySQL itself imposes a row-size limit of 65,535 for the combined size of all columns.
* The maximum tablespace size is four billion database pages (64TB) and the minimum tablespace size is slightly larger than 10MB.

**Performance Characteristics**

MEMORY performance is constrained by contention resulting from single-thread execution and table lock overhead when processing updates. This limits scalability when load increases, particularly for statement mixes that include writes.

Despite the in-memory processing for MEMORY tables, they are not necessarily faster than [InnoDB](https://dev.mysql.com/doc/refman/8.0/en/innodb-storage-engine.html) tables on a busy server, for general-purpose queries, or under a read/write workload. In particular, the table locking involved with performing updates can slow down concurrent usage of MEMORY tables from multiple sessions.

Depending on the kinds of queries performed on a MEMORY table, you might create indexes as either the default hash data structure (for looking up single values based on a unique key), or a general-purpose B-tree data structure (for all kinds of queries involving equality, inequality, or range operators such as less than or greater than). The following sections illustrate the syntax for creating both kinds of indexes. A common performance issue is using the default hash indexes in workloads where B-tree indexes are more efficient.

**Characteristics of MEMORY Tables**

The MEMORY storage engine does not create any files on disk. The table definition is stored in the MySQL data dictionary.

MEMORY tables have the following characteristics:

* Space for MEMORY tables is allocated in small blocks. Tables use 100% dynamic hashing for inserts. No overflow area or extra key space is needed. No extra space is needed for free lists. Deleted rows are put in a linked list and are reused when you insert new data into the table. MEMORY tables also have none of the problems commonly associated with deletes plus inserts in hashed tables.
* MEMORY tables use a fixed-length row-storage format. Variable-length types such as [VARCHAR](https://dev.mysql.com/doc/refman/8.0/en/char.html) are stored using a fixed length.
* MEMORY tables cannot contain [BLOB](https://dev.mysql.com/doc/refman/8.0/en/blob.html) or [TEXT](https://dev.mysql.com/doc/refman/8.0/en/blob.html) columns.
* MEMORY includes support for AUTO\_INCREMENT columns.
* Non-TEMPORARY MEMORY tables are shared among all clients, just like any other non-TEMPORARY table.

**Table MEMORY 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** | No |
| **Compressed data** | No |
| **Data caches** | N/A |
| **Encrypted data** | Yes (Implemented in the server via encryption functions.) |
| **Foreign key support** | No |
| **Full-text search indexes** | No |
| **Geospatial data type support** | No |
| **Geospatial indexing support** | No |
| **Hash indexes** | Yes |
| **Index caches** | N/A |
| **Locking granularity** | Table |
| **MVCC** | No |
| **Replication support** (Implemented in the server, rather than in the storage engine.) | Limited (See the discussion later in this section.) |
| **Storage limits** | RAM |
| **T-tree indexes** | No |
| **Transactions** | No |
| **Update statistics for data dictionary** | Yes |

1. CSV

Definition

Stores data in Comma Separated Value format in a text file.

The CSV storage engine stores data in text files using comma-separated values format.

The CSV storage engine is always compiled into the MySQL server.

To examine the source for the CSV engine, look in the storage/csv directory of a MySQL source distribution.

When you create a CSV table, the server creates a plain text data file having a name that begins with the table name and has a .CSV extension. When you store data into the table, the storage engine saves it into the data file in comma-separated values format.

Creating a CSV table also creates a corresponding metafile that stores the state of the table and the number of rows that exist in the table. The name of this file is the same as the name of the table with the extension CSM.

The CSV storage engine stores data in text files using comma-separated values format and the CSV storage engine is always compiled into the MySQL server. The server creates a table format file (.frm extension) and a data file (.csv extension) in the database directory when you create a CSV table. Both .frm and .csv files name begins with the table name. The data file is a plain text file and the storage engine saves data in comma-separated values format.

Use Cases

Advantages

Disadvantages/Limitation

**CSV Limitations :**

* Does not support indexing.
* Does not support partitioning.
* All columns must have the NOT NULL attribute in a CSV table.