MySQL Storage Engines

Archive BDB Blackhole CSV Example Falcon Federated INNODB ISAM Maria Memory MyISAM Merge NDB Cluster Partitions Performance schema Performance

A MySQL installation has six layers (*Chapter 2*, Table 2-1), one of them housing *pluggable database storage engines.* Current community builds typically include the ARCHIVE, CSV, EXAMPLE, FEDERATED, INNODB, MEMORY, MYISAM and MERGE (MRG_MYISAM) engines. ARCHIVE, CSV and FEDERATED were added in version 5.0.3, and ISAM was removed. MySQL 5.5.3 added PERFORMANCE_SCHEMA as an engine. FALCON and MARIA engines were available only in the withdrawn 6.0 series, but MARIA reappeared in MariaDB as the ARIA engine.

Since 5.5.5, INNODB is the default storage engine. SHOW ENGINES lists engines loaded in your installation. A `default_storage_engine` setting in *my.cnf/ini* overrides the default setting for when a `CREATE TABLE` statement specifies no engine. For physical tables requiring FULLTEXT indexing, MYISAM was the only choice until 5.6.4, when INNODB implemented FULLTEXT. Now INNODB is the best choice for entity integrity, referential integrity and FULLTEXT.

Table size limits are mainly due to the operating system rather than the storage engine, *e.g.*, 2GB on Linux 2.3, 2TB on Win NTFS, 4TB on Linux 2.4, 16TB on Solaris 9/10.

Archive engine

Designed for *data warehousing*, the ARCHIVE engine came in with 4.1.3. The server must have been compiled with `--with-archive-storage-engine`. Contrary to the MySQL manual, Windows binaries since 5.1 do not include it. The engine provides compressed keyless data storage with row-level locking in *.frm*, *.arz* and *.arm* files. It accepts SELECT and INSERT, not UPDATE or DELETE or INDEX. SELECT scans the table; OPTIMIZE TABLE packs it. INSERT throughput is 50% greater than with MYISAM.

BDB engine

From 3.23.34 until it was discontinued in 5.1.12, MySQL source and *mysqld-max* binary distributions had the *BerkleleyDb (BDB) engine*. To port BDB tables to another storage engine, back them up with *mysqldump* (Chapter 18), touch up the scripts to use INNODB, and import the scripts into your later MySQL installation.

Each BDB table requires a `PRIMARY KEY`, which is stored with row data. If a `PRIMARY KEY` is not declared, BDB creates a 5-byte auto-increment hidden one. If all `SELECT` columns are in one index, MySQL can execute the query without reading actual rows. Key data uses more space than in MYISAM

tables without PACK_KEYS=0. AUTOCOMMIT, BEGIN WORK, ROLLBACK and COMMIT work as expected.

MySQL does a checkpoint when a new BDB log file is started, and removes log files not needed for current transactions. FLUSH LOGS also checkpoints BDB tables. If a transaction fails with a disk full error, it should roll back. LOCK TABLE works as with other tables. Absent LOCK TABLE, MySQL issues an internal multiple-write lock to ensure proper locking in case another thread issues a table lock. Internal BDB locking is page-level.

For disaster recovery, keep table backups made with the binary log (*Chapter 17*). SELECT COUNT(*) is slow since BDB tables do not maintain exact row counts. The optimiser needs to know approximate row counts, which MySQL provides by maintaining an insert count in a separate segment of each BDB table. Without many DELETEs or ROLLBACKs, these are fairly accurate. Otherwise ANALYZE TABLE or OPTIMIZE TABLE to update row counts. Scanning is slower than with MYISAM tables. There are often holes in the BDB table to allow you to insert new rows in the middle of the key tree, so BDB tables are larger.

Blackhole engine

No, it's not a joke. Added in 5.1, BLACKHOLE accepts DDL but throws away data *after logging it*. Replicate to a BLACKHOLE DB with the desired combination of `replicate-do` and `replicate-ignore` rules, then replicate from that log to your replica. To include BLACKHOLE in a build, configure with `--with-blackhole-storage-engine`. To see if your server has it, issue `SHOW VARIABLES LIKE 'have_blackhole_engine'`.

CSV engine

CSV tables came in with 4.1.4. The engine is now compiled into all editions of the MySQL server. A *.frm* file holds the table structure. Data is stored quoted in comma-delimited rows in an indexless *.csv* text file.

Example engine

This is just a stub, an example starting point for developing a new storage engine.

Falcon engine [6.0 only]

Oracle has discontinued the Falcon engine, developed in the 6.0 series for high-volume systems with multi-threaded or multi-core CPUs and lots of memory, optimised for 64 bits. Projected features include Multi-version Concurrency Control (MVCC), flexible locking with smart deadlock prevention, full ACID compliance, a serial log, advanced B-tree indexes, data compression, intelligent disk management, data and index caching, and implicit savepoints.

Cache contents set the relationship between the record cache, which holds rows that are in active use, and the page cache, which holds database metadata, BLOB data and table indexes. Uncommitted row scavenging is implemented. Falcon parameters can be set on the server command-line (See `falcon_*` in *Appendix B*. Performance diagnostics are in the `information_schema` pseudodatabase. Source is no longer available.

The published Falcon roadmap included durable two phase commit in XA Transactions including, LIMIT optimisation, in-line ADD | DROP INDEX, MAC OS support and log file truncation. Before being withdrawn, binary builds were available for 32/64-bit Windows and 32/64-bit Linux systems. It could be built for Linux and OS X systems with `gcc`, `make`, `patch` and `tar`, using `GNU make`, `autoconf`, `automake` and `libtool`, and on Windows with `cygwin`, `gcc` and `make`.

Federated engine

This was added in 5.0.3 to provide access to remote databases. Up to 5.1.25 it was enabled by default if it was in the build; since 5.1.26 and 6.0.6 it is disabled by default if present, and when present, it is enabled with the `-federated` server option; then SHOW VARIABLES and `mysqladmin variables` report `have_federated_engine=YES`. If it is not present, compile MySQL with `--with-federated-storage-engine`.

The FEDERATED engine accesses a table on a remote MySQL server via a local table of identical structure. There are two syntaxes for creating a FEDERATED table. The first syntax defines the remote connection specifically for the table being created:

```
CREATE TABLE ( create_options ) ENGINE=FEDERATED
CONNECTION='scheme://user[:password]@host[:port]:/db/tbl';
```

where `create_options` must match those of the remote table, `scheme` must be `mysql`, `user` and `password` (username and password for the remote server), `port` is the remote access port, `host` is the remote host, `db` is the name of the remote database, which must exist, and `tbl` is the name of the remote table, which also must exist. MySQL writes a *.frm* file with the table definition to the local MySQL data directory, but does *not* create local data and index files. As of 5.1.15, partitioning is disallowed.

For example, if table `t(i INT PRIMARY KEY, j INT)` exists on server `remote` in database `db`, create a local FEDERATED table that connects to it via user USR, PWD with:

```
CREATE TABLE fedt(i INT PRIMARY KEY, j INT) ENGINE=FEDERATED
CONNECTION='mysql://USR:PWD@remote/db/t';
```

The second syntax defines a connection that can be re-used for multiple tables:

```
CREATE SERVER server_name
FOREIGN DATA WRAPPER wrapper_name
OPTIONS (option ...)
```

where `server_name` is the name to use to refer to the remote server, `wrapper_name` must be `mysql`. , and OPTION defines username, password, host name and database. You then use these to create the local FEDERATED table. For our example, you would write:

```
CREATE SERVER remotelink FOREIGN DATA WRAPPER mysql
OPTIONS (USER 'USR', HOST 'remote', DATABASE 'db');

CREATE TABLE fedt(i INT PRIMARY KEY, j INT) ENGINE=FEDERATED
CONNECTION='remotelink.db.t';
```

There is no current support for transactions, some DDL commands (e.g., ALTER TABLE), or HANDLER. Some DDL is supported; you can define a trigger on a FEDERATED table, or on a local table to update another local FEDERATED table which in turn updates its remote table. See *here* for Giuseppe Maxia's useful FEDERATED engine manual.

To read the rest of this and other chapters, *buy a copy of the book*