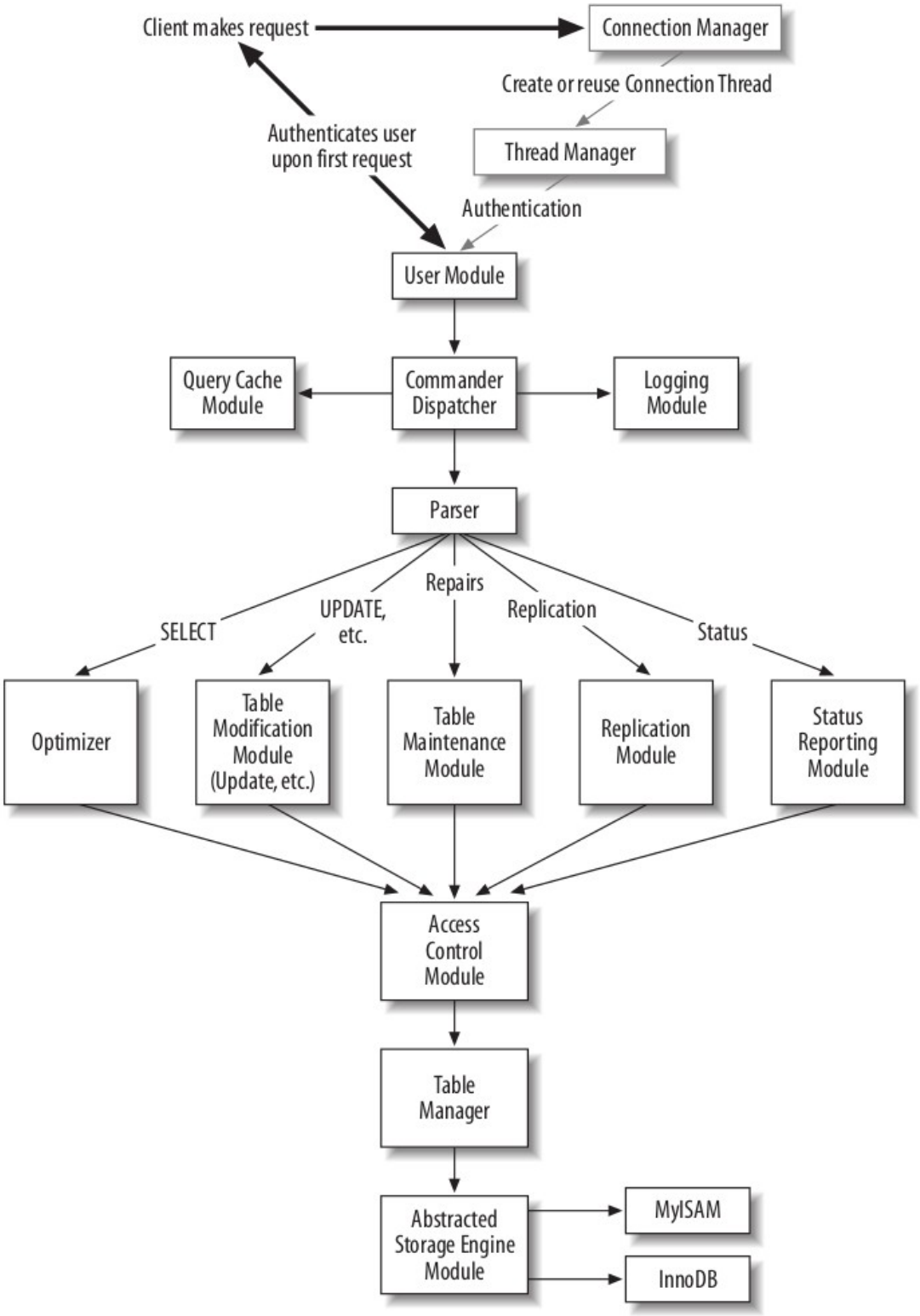
Mysql Architecture

Global view



Core Modules

Server Initialization Module

The Server Initialization Module is responsible for the server initialization on startup.

- init_common_variables( ) in sql/mysqld.cc.
- init_thread_environment( ) in sql/mysqld.cc.
- init_server_components( ) in sql/mysqld.cc.
- grant_init( ) in sql/sql_acl.cc
- init_slave( ) in sql/slave.cc
- get_options( ) in sql/mysqld.cc.

Connection Manager

The Connection Manager listens for incoming connections from clients, and dispatches the requests to the Thread Manager.

handle_connections_sockets( ) in sql/mysqld.cc.

Thread Manager

The Thread Manager is responsible for keeping track of threads and for making sure a thread is allocated to handle the connection from a client.

- create_new_thread( ) sql/mysqld.cc.
- start_cached_thread( ) sql/mysqld.cc.

Connection Thread

The Connection Thread is the heart of the work of processing client requests on an established connection.

handle_one_connection( ) in sql/sql_parse.cc.

User Authentication Module

The User Authentication Module authenticates the connecting user and initializes the structures and variables containing the information on his level of privileges.

- check_connection( ) in sql/sql_parse.cc(entry point).
- acl_check_host( ) in sql/sql_acl.cc
- create_random_string( ) in sql/password.cc
- check_user( ) in sql/sql_parse.cc

- acl_getroot( ) in sql/sql_acl.cc

Access Control Module

The Access Control Module verifies that the client user has sufficient privileges to perform the requested operation.

- check_access( ) in sql/sql_parse.cc.
- check_grant( ) in sql/sql_acl.cc
- check_table_access( ) in sql/sql_parse.cc
- check_grant_column( ) in sql/sql_acl.cc
- acl_get( )in sql/sql_acl.cc

Parser

The Parser is responsible for parsing queries and generating a parse tree.

- mysql_parse( ) in sql/sql_parse.cc(entry point)
- yyparse( ) in sql/sql_yacc.cc
- sql/gen_lex_hash.cc
- sql/lex.h
- sql/lex_symbol.h
- sql/lex_hash.h (generated file)
- sql/sql_lex.h
- sql/sql_lex.cc
- The group of files under sql/ with names starting in item_ and extensions of .h or .cc

Command Dispatcher

The Command Dispatcher is responsible for directing requests to the lower-level modules that will know how to resolve them.

- do_command( ) in sql/sql_parse.cc
- dispatch_command( ) in sql/sql_parse.cc

Query Cache Module

The Query Cache Module caches query results, and tries to short-circuit the execution of queries by delivering the cached result whenever possible.

- Query_cache::store_query( ) in sql/sql_cache.cc
- Query_cache::send_result_to_client( ) in sql/sql_cache.cc

Optimizer

The Optimizer is responsible for creating the best strategy to answer the query, and executing it to deliver the result to the client.

- mysql_select( ) in sql/sql_select.cc(entry point).
- JOIN::prepare( ) in sql/sql_select.cc
- JOIN::optimize( ) in sql/sql_select.cc
- JOIN::exec( ) in sql/sql_select.cc
- make_join_statistics( ) in sql/sql_select.cc
- find_best_combination( ) in sql/sql_select.cc
- optimize_cond( ) in sql/sql_select.cc
- SQL_SELECT::test_quick_select( ) in sql/opt_range.cc.

Table Manager

The Table Manager is responsible for creating, reading, and modifying the table definition files (.frm extension), maintaining a cache of table descriptors called table cache, and managing table-level locks.

- openfrm( ) in sql/table.cc
- mysql_create_frm( ) in sql/unireg.cc
- open_table( ) in sql/sql_base.cc
- open_tables( ) in sql/sql_base.cc
- open_ltable( ) in sql/sql_base.cc
- mysql_lock_table( ) in sql/lock.cc

Table Modification Modules

This collection of modules is responsible for operations such as creating, deleting,renaming, dropping, updating, or inserting into a table.

- mysql_update( ) and mysql_multi_update( ) in sql/sql_update.cc
- mysql_insert( ) in sql/sql_insert.cc
- mysql_create_table( ) in sql/sql_table.cc
- mysql_alter_table( ) in sql/sql_table.cc
- mysql_rm_table( ) in sql/sql_table.cc
- mysql_delete( ) in sql/sql_delete.cc

Table Maintenance Module

The Table Maintenance Module is responsible for table maintenance operations such as check, repair, back up, restore, optimize (defragment), and analyze (update key distribution statistics).

- mysql_admin_table( ) in sql/sql_table.cc
- mysql_check_table( ) in sql/sql_table.cc
- mysql_repair_table( ) in sql/sql_table.cc
- mysql_backup_table( ) in sql/sql_table.cc
- mysql_restore_table( ) in sql/sql_table.cc
- mysql_optimize_table( ) in sql/sql_table.cc
- mysql_analyze_table( ) in sql/sql_table.cc

Status Reporting Module

The Status Reporting Module is responsible for answering queries about server configuration settings, performance tracking variables, table structure information, replication progress, condition of the table cache, and other things. It handles queries that begin with SHOW.

- mysqld_list_processes( ) in sql/sql_show.cc
- mysqld_show( ) in sql/sql_show.cc
- mysqld_show_create( ) in sql/sql_show.cc
- mysqld_show_fields( ) in sql/sql_show.cc
- mysqld_show_open_tables( ) in sql/sql_show.cc
- mysqld_show_warnings( ) in sql/sql_show.cc
- show_master_info( ) in sql/slave.cc
- show_binlog_info( ) in sql/sql_repl.cc

Logging Module

The Logging Module is responsible for maintaining higher-level (logical) logs. A storage engine may additionally maintain its own lower-level (physical or logical) logs for its own purposes, but the Logging Module would not be concerned with those; the storage engine itself takes charge. The logical logs at this point include the binary update log (used mostly for replication, otherwise), command log (used mostly for server monitoring and application debugging), and slow query log (used for tracking down poorly optimized queries).

Both the Replication Master and Replication Slave modules rely heavily on this functionality of the Logging Module.

Replication Master Module

The Replication Master Module is responsible for the replication functionality on the master. The most common operation for this module is to deliver a continuous feed of replication log events to the slave upon request.

mysql_binlog_send( ) in sql/sql_repl.cc

Replication Slave Module

The Replication Slave Module is responsible for the replication functionality of the slave. The role of the slave is to retrieve updates from the master, and apply them on the slave.

- handle_slave_io( ) in sql/slave/cc
- handle_slave_sql( ) in sql/slave/cc

Interaction of the core models

Server Initialization Module

When the server is started on the command line, the Initialization Module takes control. It parses the configuration file and the command-line arguments, allocates global memory buffers, initializes global variables and structures, loads the access control tables, and performs a number of other initialization tasks.

Connection Manager:

Once the initialization job is complete, the Initialization Module passes control to the Connection Manager, which starts listening for connections from clients in a loop.

Thread Manager

When a client connects to the database server, the Connection Manager performs a number of low-level network protocol tasks and then passes control to the Thread Manager, which in turn supplies a thread to handle the connection (which from now on will be referred to as the Connection Thread). The Connection Thread might be created a new, or retrieved from the thread cache and called to active duty.

User Authentication Module

Once the Connection Thread receives control, it first invokes the User Authentication Module. The credentials of the connecting user are verified, and the client may now issue requests.

Command Dispatcher

The Connection Thread passes the request data to the Command Dispatcher. Some requests, known in the MySQL code terminology as commands, can be accommodated by the Command Dispatcher directly, while more complex ones need to be redirected to another module. A typical command may request the server to run a query, change the active database, report the status, send a continuous dump of the replication updates, close the connection, or perform some other operation.
In MySQL server terminology, there are two types of client requests:

- A query is anything that has to go through the parser.
- A command is a request that can be executed without the need to invoke the parser.

We will use the term query in the context of MySQL internals. Thus, not only a SELECT but also a DELETE or INSERT in our terminology would be called a query. What we would call a query is sometimes called an SQL statement.

Logging Module

If full query logging is enabled, the Command Dispatcher will ask the Logging Module to log the query or the command to the plain-text log prior to the dispatch. Thus in the full logging configuration all queries will be logged, even the ones that are not syntactically correct and will never be executed, immediately returning an error.

Query Cache Module

The Command Dispatcher forwards queries to the Parser through the Query Cache Module. The Query Cache Module checks whether the query is of the type that can be cached, and if there exists a previously computed cached result that is still valid. In the case of a hit, the execution is short-circuited at this point, the cached result is returned to the user, and the Connection Thread receives control and is now ready to process another command. If the Query Cache Module reports a miss, the query goes to the Parser, which will make a decision on how to transfer control based on the query type.

Models between Parser and Access Control Model

One can identify the following modules that could continue from that point: the Optimizer,the Table Modification Module, the Table Maintenance Module, the Replication Module, and the Status Reporting Module.

1.Optimizer

Select queries are forwarded to the Optimizer;

2.Table Modification Modules

Updates, inserts, deletes, and table-creation and schema-altering queries go to the respective Table Modification Modules;

There also exist a number of Table Modification Modules: Delete Module, Create Module, Update Module, Insert Module, and Alter Module.

3.Table Maintenance module

queries that check, repair, update key statistics, or defragment the table go to the Table Maintenance module;

4.Replication Module

queries related to replication go to the Replication Module;

5.Status Reporting Module

status requests go to the Status Reporting Module.

Access Control Module

At this point, each of the modules that will receive control from the Parser passes the list of tables involved in the query to the Access Control Module

Table Manager

And then, upon success, to the Table Manager, which opens the tables and acquires the necessary locks.

Now the table operation module is ready to proceed with its specific task and will issue a number of requests to the Abstracted Storage Engine Module for low-level operations such as inserting or updating a record, retrieving the records based on a key value, or performing an operation on the table level, such as repairing it or updating the index statistics.