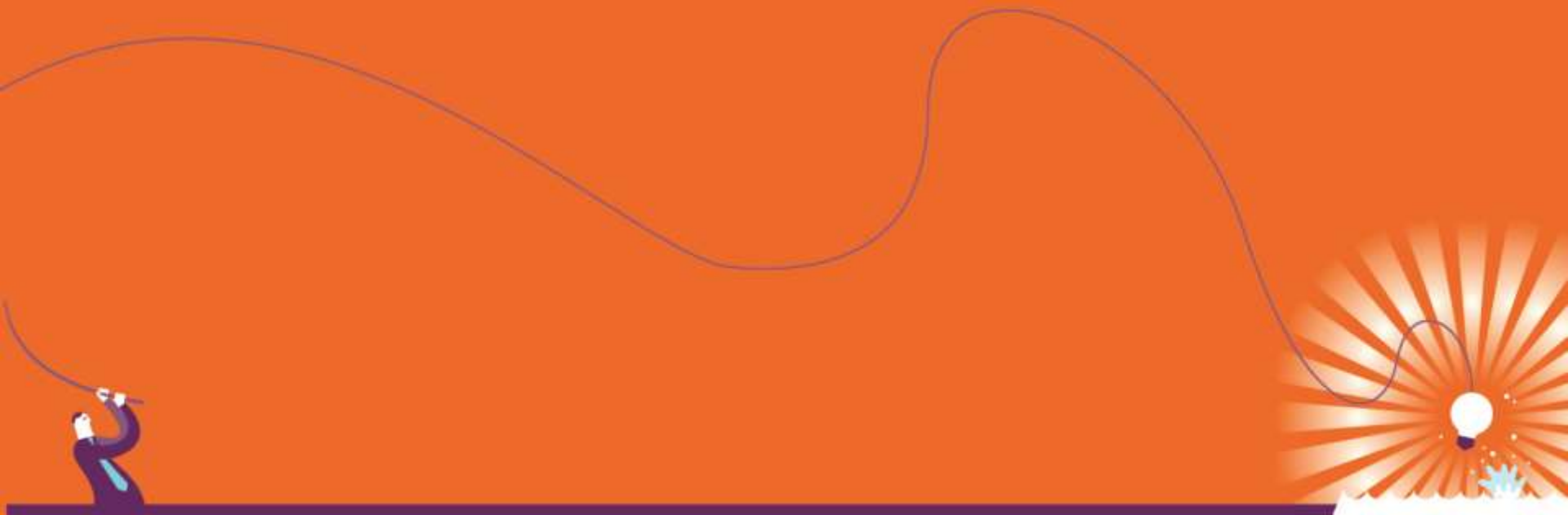
MySQL Architecture & Engines

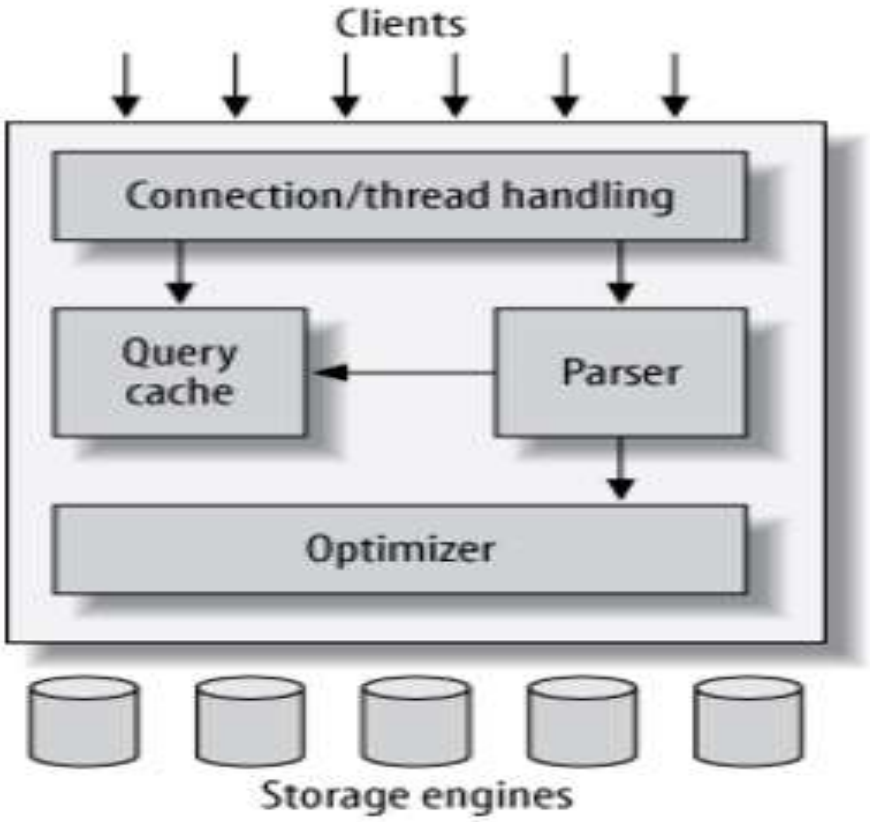
Learning Objectives

- Understand MySQL Architecture
- How MySQL Uses Disk space and Memory
- Storage Engines
  - > MySQL Interaction with Storage Engines
  - > Major Storage engines with details below for each engine
    - Characteristics / Features
    - Storage format
    - Transaction Support
    - Locking
    - Special Features

MySQL Architecture



MySQL Architecture



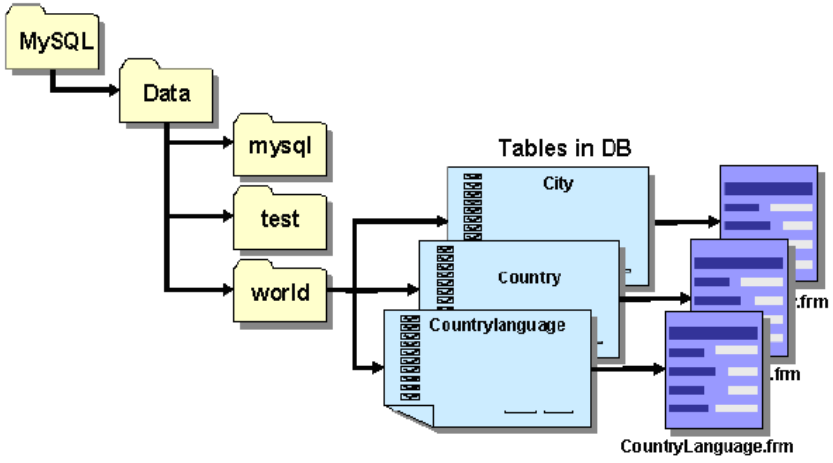
The brains of the MySQL server

Component	Feature
Parsing	Responsible for deconstructing the requested SQL statements
Optimizing	Responsible for finding the optimal execution plan for the query
Executing	Responsible for executing the optimized path for the SQL command passed through the parser and optimizer
Query Cache	The query cache is a fast in-memory store to quickly look up the result set of a particular SELECT statement
Storage Engines	Enables MySQL to use different implementations for storing, retrieving and indexing data

How MySQL uses Disk and Memory

How MySQL Uses Disk Space

- Data directory
- Table and view format files (.frm)
- Server log files and status files
- Trigger storage
- System database (MySQL)



How MySQL Memory

Two different types memory allocation

- per-session (allocated for each connection thread)
  - Session specific
  - Dynamically allocated and deallocated
  - Mostly utilized for handling query results
  - Buffer sizes usually per session
- per-instance (allocated once for the entire server)
  - Allocated only once (per server instance)
  - Shared by the server processes and all of its threads

How MySQL Memory

- Server allocates memory for the following
- Thread caches
- Buffers
- MEMORY tables
- Internal temporary tables
- Client specific buffers

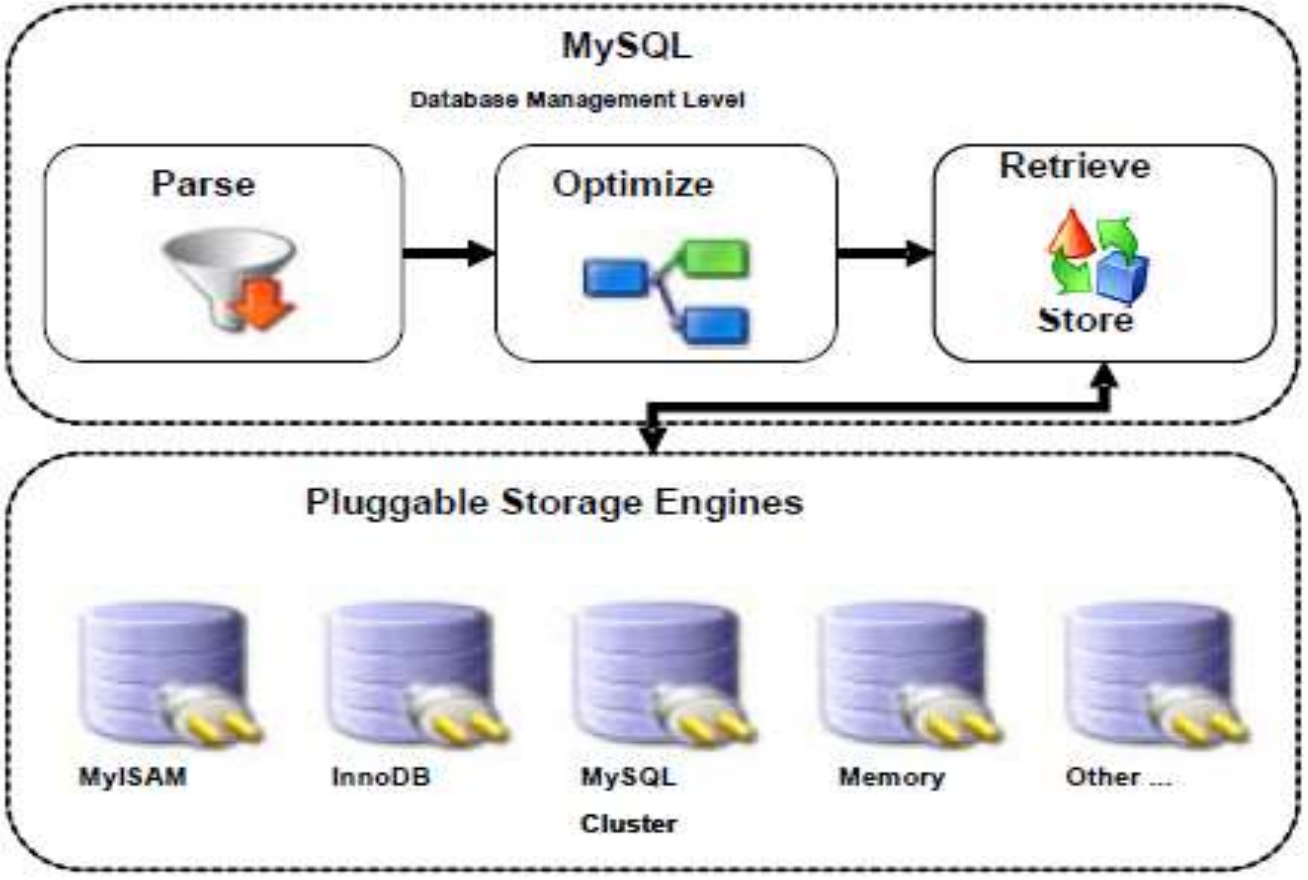
MySQL Engines

Storage Engines

A storage engine is a software module that a database management system uses to create, read, update data from a database

- Client sends requests to the server as SQL
- Two-tier processing
  - >Upper tier includes SQL parser and optimizer
  - >Lower tier comprises a set of storage engines
- SQL tier not dependent on storage engine
  - >Engine setting does not effect processing
  - >Some Exceptions

MySQL Interaction with Storage Engines



What makes Storage Engine different

- Storage medium
- Transactional capabilities
- Locking
- Backup and recovery
- Optimization
- Special features
  - Full-text search
  - Referential integrity
  - Spatial data handling

Available Storage Engines

Developed by MySQL

- MyISAM
- Falcon
- NDBCluster
- MEMORY
- ARCHIVE
- FEDERATED
- BLACKHOLE
- CSV

Third party storage engines

- InnoDB
- InfoBright
- Nitro
- PBXT

Engines

- View Available Storage Engines
  >SHOW ENGINES

- Setting the Storage Engine
  >Specify engine using CREATE TABLE
  >CREATE TABLE t (i INT) ENGINE = InnoDB;

- Uses system default if not set
  >--default-storage-engine
  >@@storage_engine

- Change storage engine using **ALTER TABLE**
  >ALTER TABLE t ENGINE = MEMORY;

The MyISAM Storage Engine

The MyISAM storage engine was the default storage engine from MySQL 3.23 until it was replaced by InnoDB in MariaDB and MySQL 5.5.

Characteristics

- Represented by three files (`.frm`, `.MYD` and `.MYI`)
- Most Flexible AUTO_INCREMENT
- Compressed, read-only tables save space
- Table-level locking
- Portable storage format
- Specify number of rows for a table
- Disable updating of non-unique indexes and enable the indexes
- Tables take up very little space

MyISAM Row Storage Formats

- Fixed-row format
- Dynamic-row format
- Compressed format

Compressing MyISAM Tables

- Tables do not get compressed by default
- Compress tables with the `myisampack` utility
- Only make sense for some applications
  - Tables are read-only
  - Must decompress before **UPDATE**, **INSERT**, **DELETE**, then recompress
- `myisampack` consists of true compression and optimizations
- Must use `myisamchk` to update indexes afterward

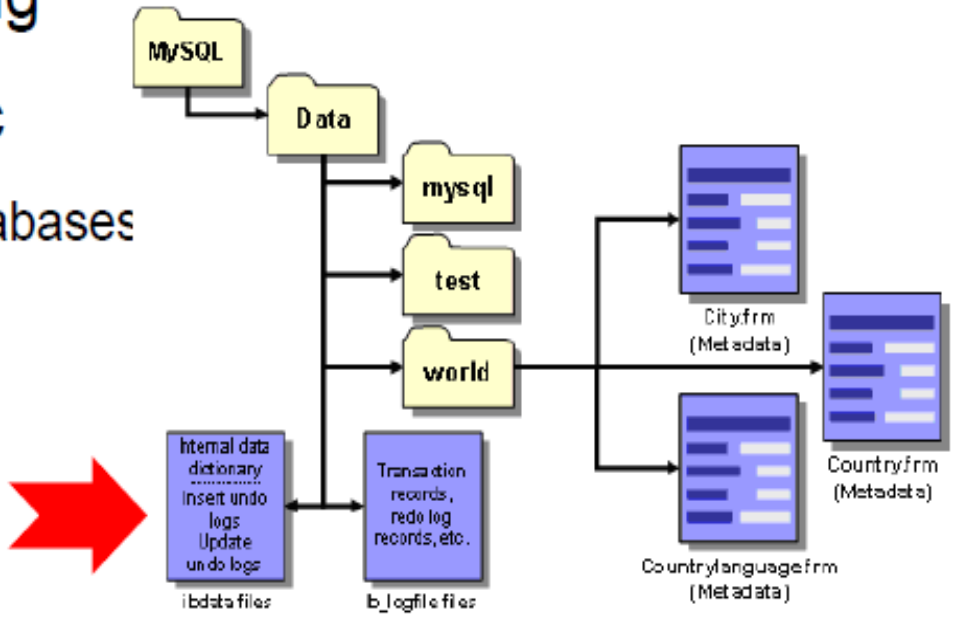
The InnoDB Storage Engine

Characteristics

- Represented by `.frm` file
- Data and Index files are in the InnoDB tablespace
- A set of log files used for recording transaction activity
- Supports **COMMIT**, **SAVEPOINT** and **ROLLBACK**
- Full **ACID** compliance
- Auto-recovery after a crash
- Multi-versioning (MVCC) and row-level locking
- Supports foreign keys and referential integrity

InnoDB Tablespace

- Single logical storage area
- Can consist of one or multiple files
- Each file can be a regular or raw partition
- Can be auto-extending
- Not database specific
  - Used for all InnoDB databases



Multiple Tablespace

- If you do not want to use the shared tablespace
  - Use the `--innodb_file_per_table`
  - Sets up a `.ibd` file in the database for each InnoDB table
  - Placed in database directory
  - Acts as the tables' own tablespace file
  - Does not affect access to tables created in shared tablespace

MyISAM versus InnoDB

MyISAM	Innodb
Not ACID-compliant and non-transactional	ACID-compliant and hence fully transactional with ROLLBACK and COMMIT and support for Foreign Keys
MySQL 5.0 default engine	Rackspace Cloud default engine
Offers compression	Offers compression
Requires full repair and rebuild of indexes and tables	Provides automatic recovery from crashes via the replay of logs
Changed database pages written to disk instantly	Dirty pages converted from random to sequential before commit and flush to disk
No ordering in storage of data	Row data stored in pages in PK order
Table-level locking	Row-level locking

The MEMORY Storage Engine(1/2)

- Uses tables stored in memory
- Characteristics
  - Represented by `.frm` file
  - Table data and indexes stored in memory
  - Storage very fast
  - Fixed-length rows
  - Table contents do not survive restart
  - Can limit file size using `--max-heap-table-size`
  - Table-level locking
  - Cannot contain **TEXT** or **BLOB**
- Formerly called HEAP engine

The MEMORY Storage Engine(2/2)

- MEMORY indexing algorithms
  - **HASH**
  - **BTREE**
- MEMORY best practices
  - Minimize the size that a MEMORY table grows to
    - `max_heap_table_size` variable
    - `MAX_ROWS` within **CREATE TABLE** or **ALTER TABLE**
  - Populate data to MEMORY tables on start up
    - `--init-file` startup option
    - Execute **LOAD DATA INFILE** or **INSERT INTO...SELECT**

ARCHIVE Storage Engine

- For Storing large volumes of data in a compressed format allowing for a very small footprint.
- Characteristics
  - Represented by `.frm` file
  - Data and metadata files: `.ARZ` and . `.ARM`
  - Does *not* support indexes (but supports `ORDER BY`)
  - Supports `INSERT` and `SELECT`, but not `DELETE`, `REPLACE`, or `UPDATE`
  - Supports all data types (including BLOBs) except spatial types.
  - Uses row-level locking
  - Supports `AUTO_INCREMENT` columns

ARCHIVE Storage

- Rows are compressed as they are inserted
  - **INSERT** Statement
    - An **INSERT** statement pushes rows into a compression buffer, and that buffer flushes as necessary
    - The insertion into the buffer is protected by a lock
  - Bulk Insert
    - A bulk insert is visible only after it completes
    - If other inserts occur at the same time, the bulk insert can be partially seen

ARCHIVE Retrieving and Archiving

- Retrieval
  - Rows are uncompressed on demand; there is no row cache
  - A **SELECT** operation performs a complete table scan
    - When a **SELECT** occurs, it finds out how many rows are currently available and reads that number of rows
  - **SELECT** is performed as a consistent read
- Archival issues
  - Reduces the size of most other storage engine files by up to 70%
  - Extremely effective for reducing the size of data that is no longer being modified and is strictly historical in nature

CSV Storage Engine

- Specialized storage engine
  - stores all the data associated with it in comma-separated file format
- Associated files
  - Located in the data directory under a directory with the name of the database
  - **table_name.frm** - Table format file
  - **table_name.csv** – Plain text data file with the data in comma-separated values format
    - The table data can be copied from the database directly
    - The table may be opened in a spreadsheet format
  - No indexes are allowed
- Used for the log tables (in the mysql database)
  - general_log
  - slow_log

Choosing Appropriate Storage Engines

- Ask yourself what types of queries you will use
- Choose engine with locking level needed for expected workload
  - MyISAM: for many reads, few writes (table locking)
  - InnoDB: mixed reads / writes, many concurrent users
  - InnoDB row-level and multi-versioning for good concurrency
- Special considerations for MyISAM
  - Fixed-length columns to achieve fixed row format
  - Variable-length columns
  - Can use compressed read-only tables

Choosing Appropriate Storage Engines

- Special considerations for InnoDB
  - **CHAR** takes more space than **VARCHAR**
- **MEMORY** engine
  - for caching often queried data
  - only when you can afford to lose the data (or have it backed up)
  - Can regenerate from disk based tables
  - Session variables are an example

To reduce memory usage, do not configure unneeded storage engines into the server.

Thank You!
Your Questions Please