A Look at MySQL's Conceptual Architecture

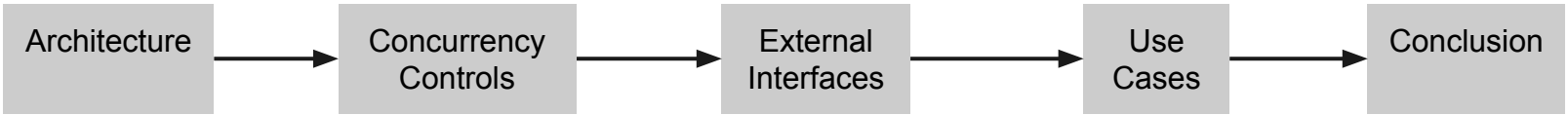
EECS 4314: Advanced Software Engineering

Tabs vs. Spaces

Varsha Ragavendran
Nisha Sharma
Davood Anbarnam
Ayman Abualsunun
Anton Sitkovets
Glib Sitiugin
Kevin Arindaeng

Introduction and Overview

- Provides computations for query processing
- Prominent features - Performance, Speed, and Reliability
- Used in high-profile applications - Facebook, Airbnb, Twitter, etc.



Architecture - Overview

- At High Level, **MySQL** Database System operates via **Client Server Architecture**
- Server:
  - Is the core program that manages database content
  - Handles Bulk of the computation
- Client:
  - Connects to the Server Via Network to read, write or update data
- Non-Client Utility Programs and Services include
  - File retrieval
  - Backup and Restore etc.

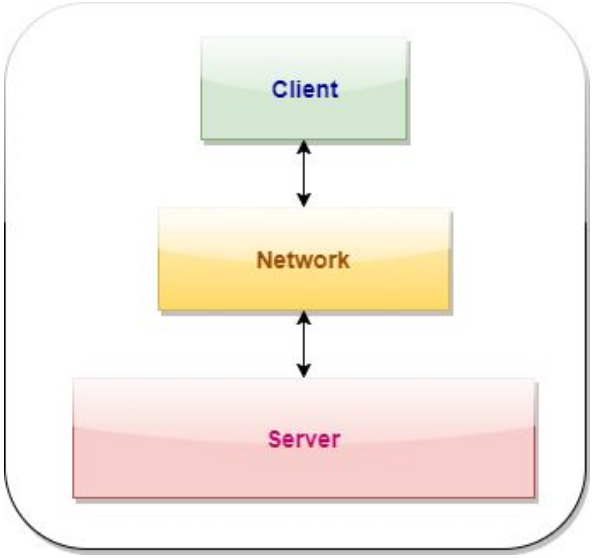


Figure 1. MySQL's Client-Server Architecture

Architecture - Overview (Layered)

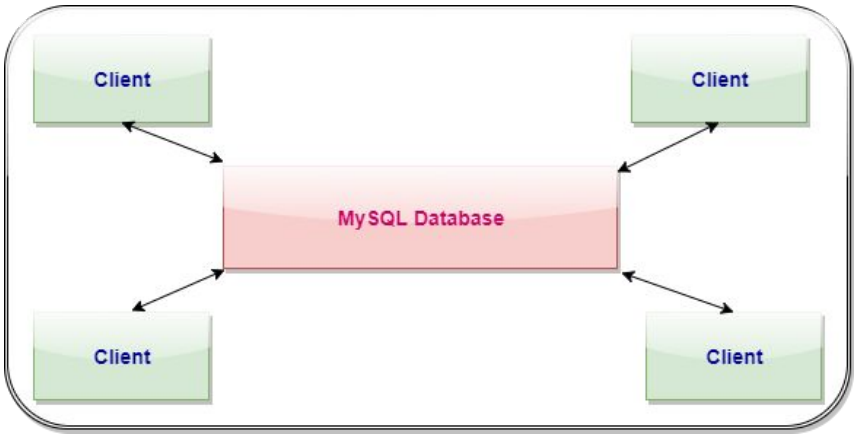
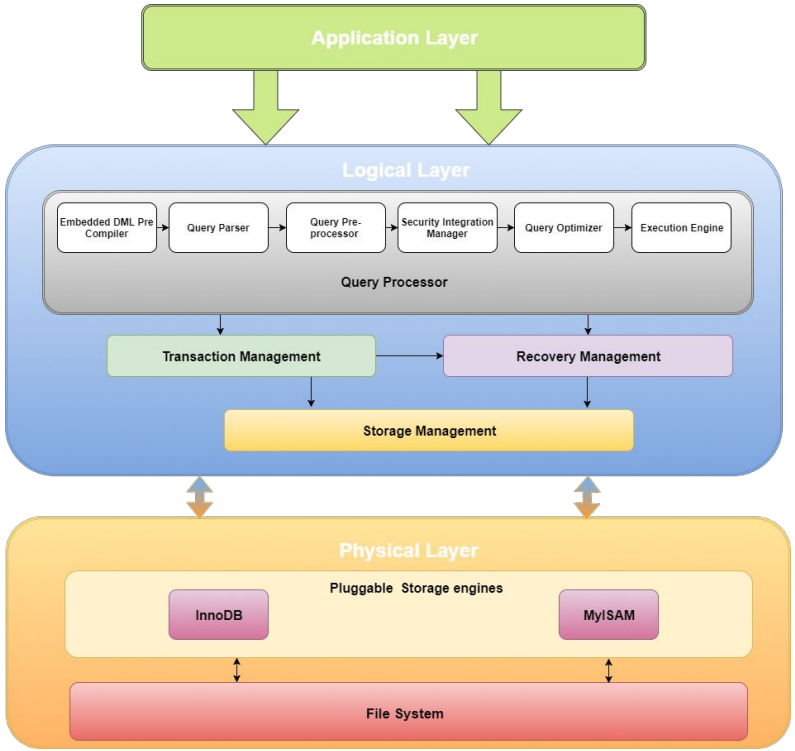


Figure 2. Repository Style Architecture

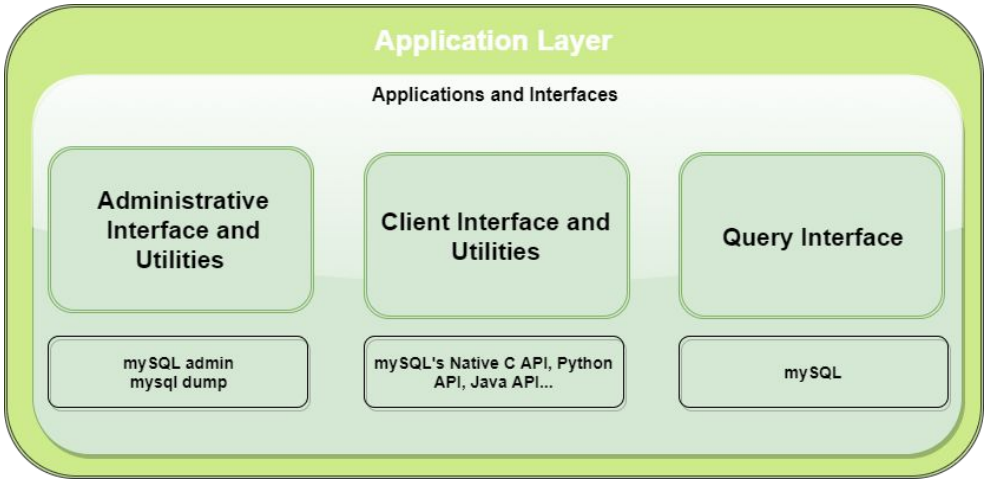
Architecture - Overview (Layered)



MySQL Layered Architecture

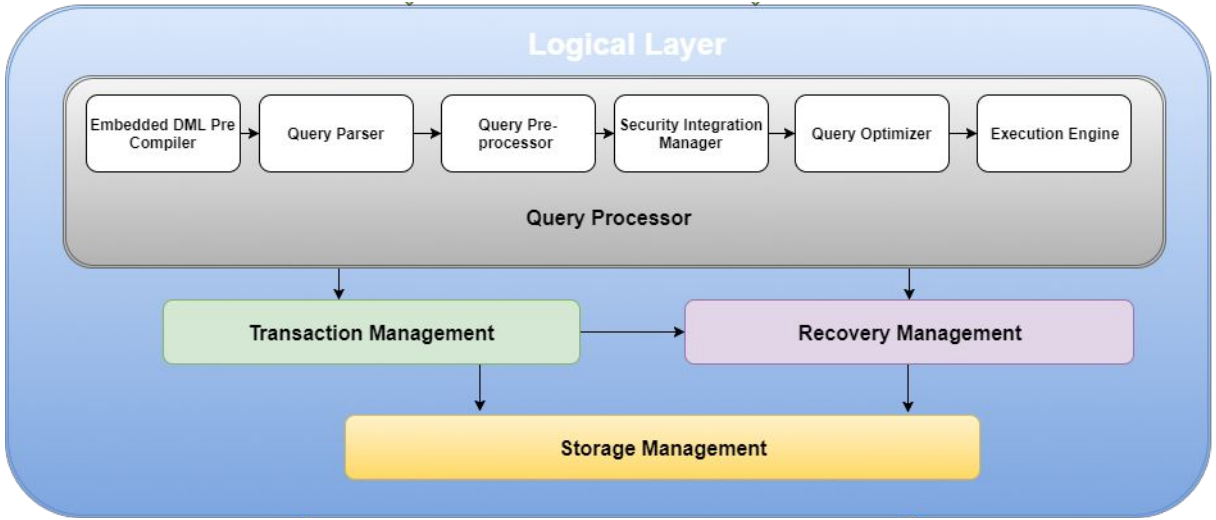
APPLICATION LAYER

- RDBMS interacts with users and clients
- Three components:
  - Administrators
  - Clients
  - Query Users



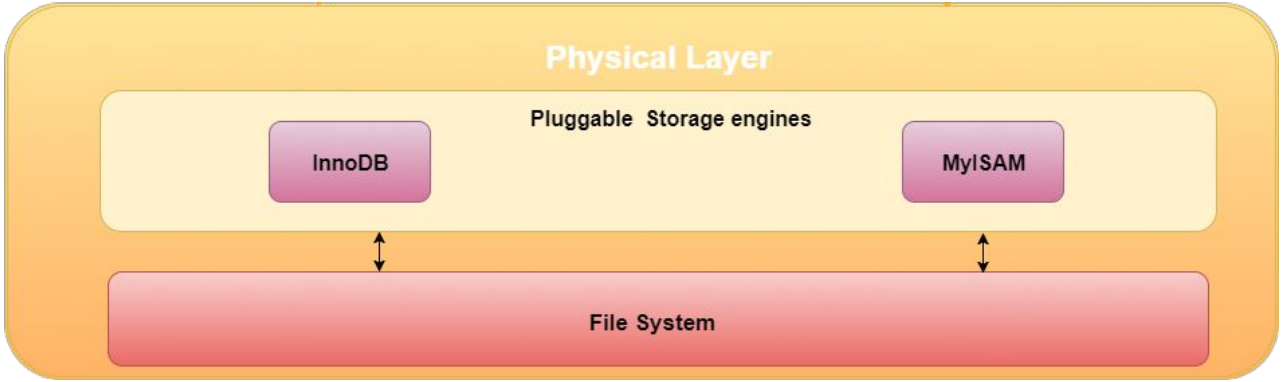
MySQL Layered Architecture (Continued…)

Logical Layer:



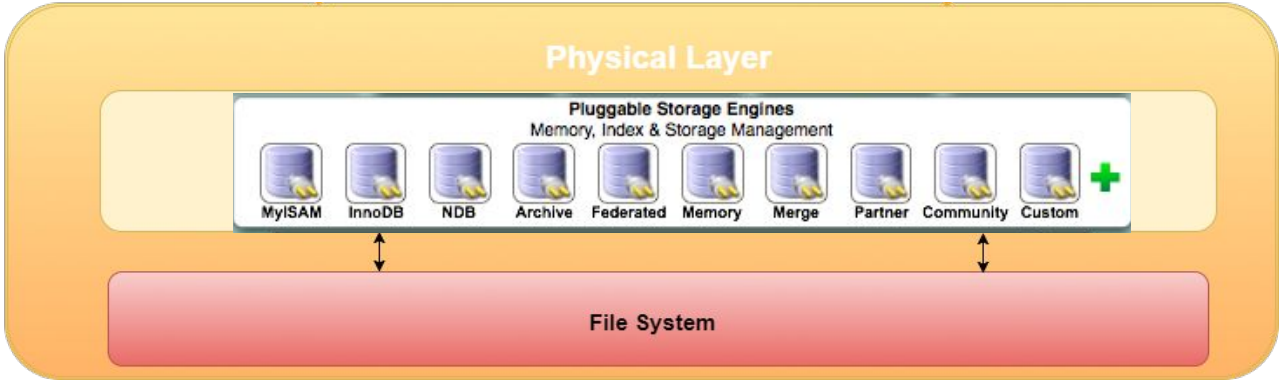
MySQL Layered Architecture (Continued…)

Physical Layer:



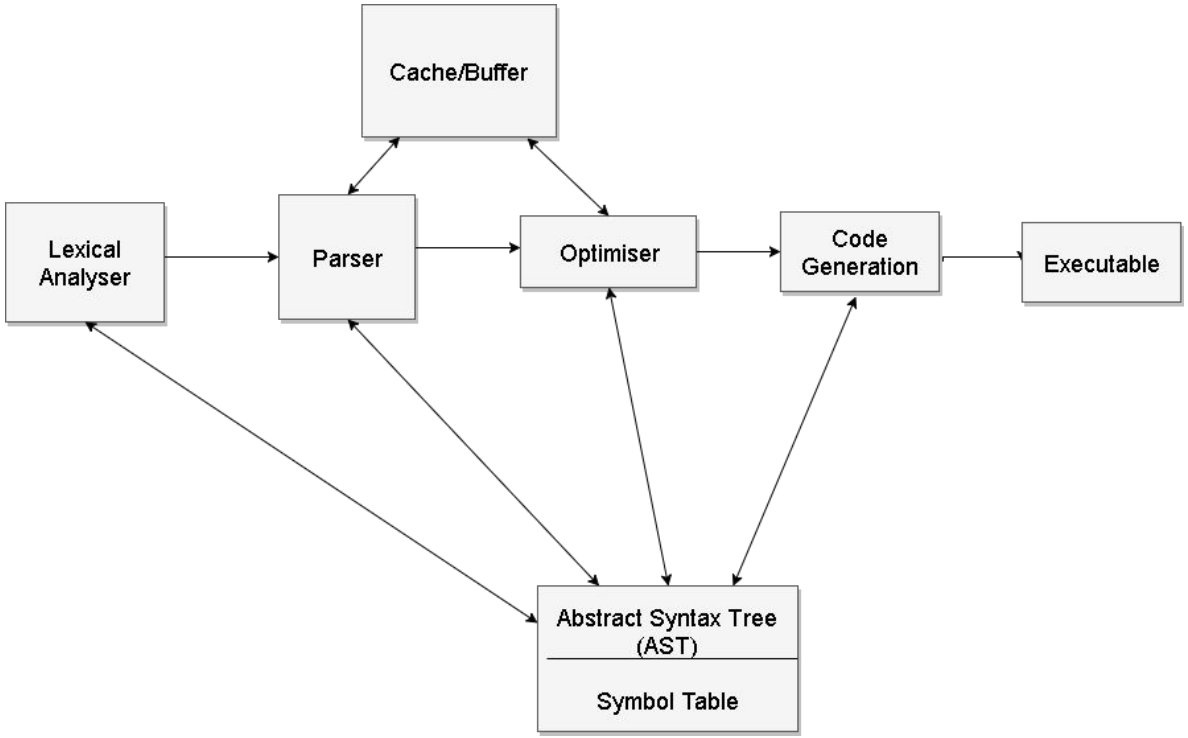
MySQL Layered Architecture (Continued…)

Physical Layer:



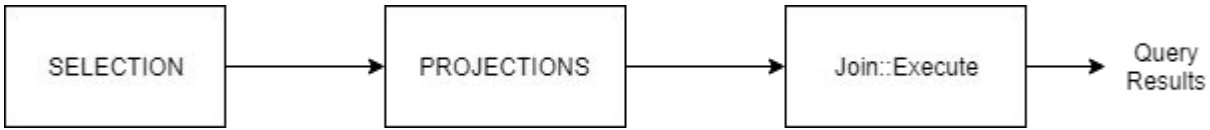
Architecture - Compilation

- Lex/YACC
- Syntax
- Semantics
- Optimising



Architecture - Optimizer

- 3 steps, SELECTION > PROJECTIONS > JOIN::EXECUTE

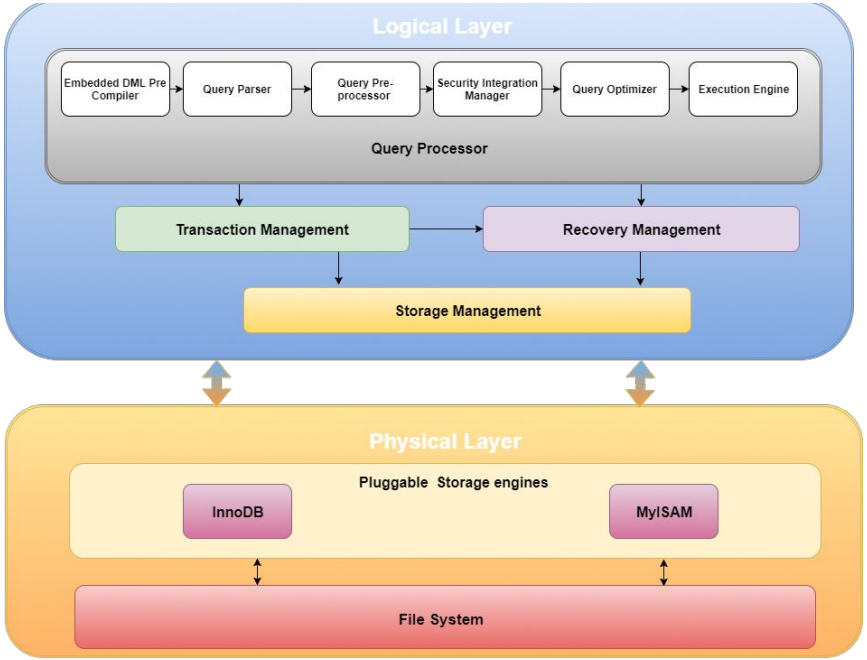


Architecture - Query Cache & Buffers

- Query Cache: Cache for frequently used queries, which caches the query and the results, as well as purging old data on updated data.
- Table Cache: Memory cache for table metadata to reduce time for opening, reading and closing tables. Each thread has its own list of table cache structures allowing them to have their own views of tables.
- Buffer Pool: Stores table and index data.
- Record Cache: Cache for sequential reading of tables. Consists of a read ahead buffer that takes one block of data at a time.
- Key Cache: Cache for frequently used index data. Implemented using a warm system to classify how frequently an index has been accessed over time (LRU).
- Privilege Cache: Caches user privilege within system.

Architecture - Storage Engines

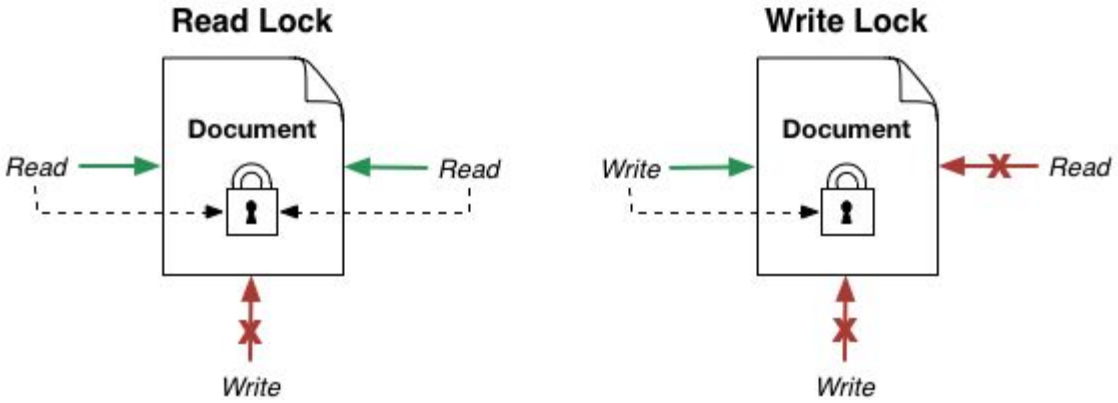
- Responsible for the storage and retrieval of all stored data
- Pluggability feature
- Two commonly used engines - InnoDB and MyISAM



Concurrency Control: Lock Types

- **Read locks:** mutually nonblocking locks.
  Many clients may read from a resource at the same time and not interfere with each other.

- **Write locks**: exclusive locks.
  Write locks block both read locks and other write locks.



Concurrency Control: Lock Granularity

- **Table Lock**: the most basic and low-overhead locking strategy available in MySQL.

- Advantages:
  - Low memory requirements
  - Fast if you often do GROUP BY operations on a large part of the data
  - Fast if must scan the entire table frequently

Concurrency Control: Lock Granularity

- **Row locks** offer more concurrency than table locks, but also carry a heavier overhead.

- Advantages:
  - Fewer lock conflicts

- InnoDB storage engine supports both table-level locking and row-level locking, while MyISAM only supports table locking.
- InnoDB is more efficient when there is heavy write load.

Concurrency Control: Lock Granularity

- **Row locks** offer more concurrency than table locks, but also carry a heavier overhead.

- Advantages:
  - With row-level locking there are fewer lock conflicts when different sessions access different rows. As a result, it is possible to lock a single row for a long time.

- Answer: InnoDB

Architecture - Transaction Processing System

1. Transactions are ATOMIC, all or nothing.
2. Required to pass the ACID test.
3. Different isolation levels.
4. Transactions logging.

External Interfaces

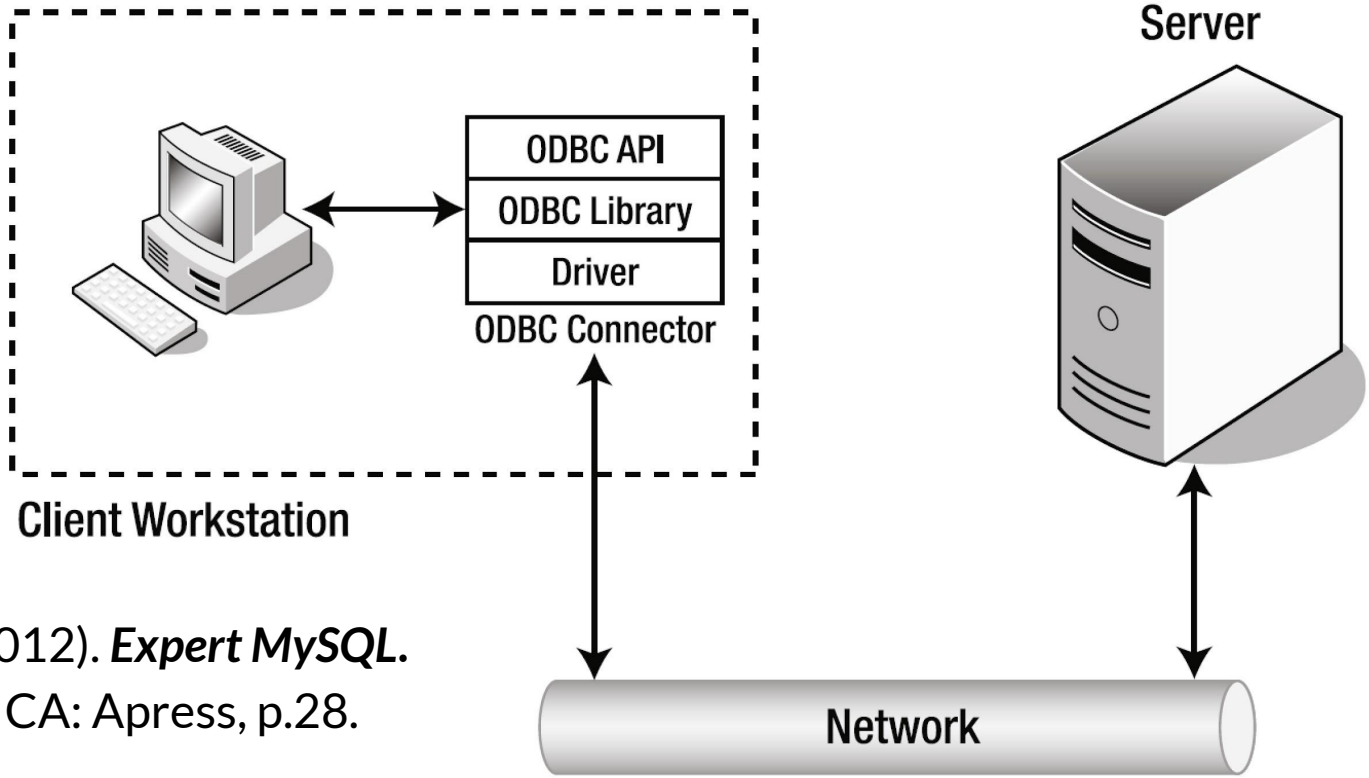
- **Connectors and APIs**
  - Connector/J, Connector/C++, X DevAPI

- Graphical User Interfaces
  - MySQL Workbench, Adminer, DBEdit

Connectors and APIs

- Common way to interact with MySQL Server, along with MySQL Shell
- Supports general connectors with APIs for many languages (Java, C, C++, .NET)
- Most connectors are based on the **Open Database Connectivity (ODBC) model**.

ODBC Low-level architecture

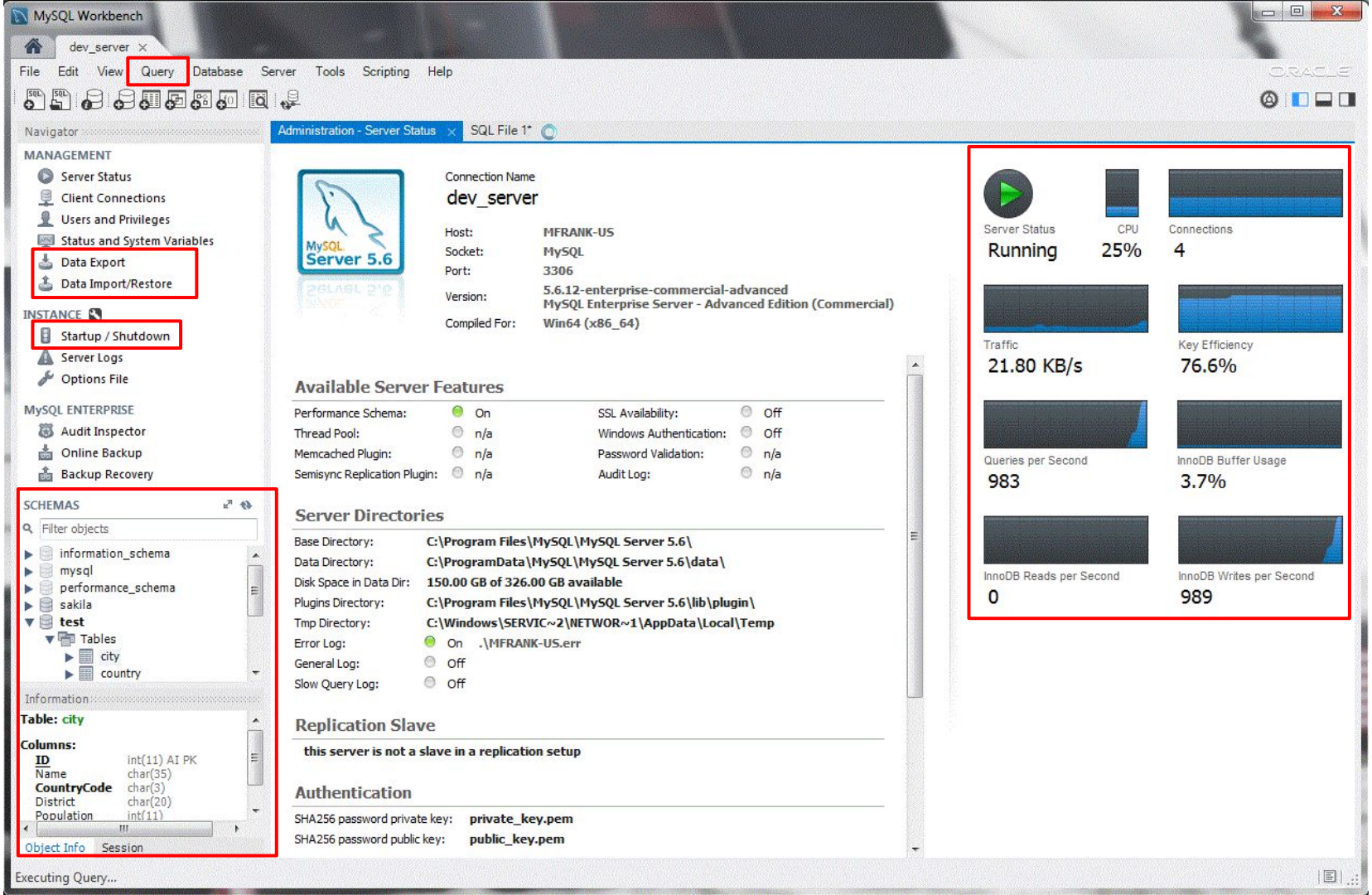


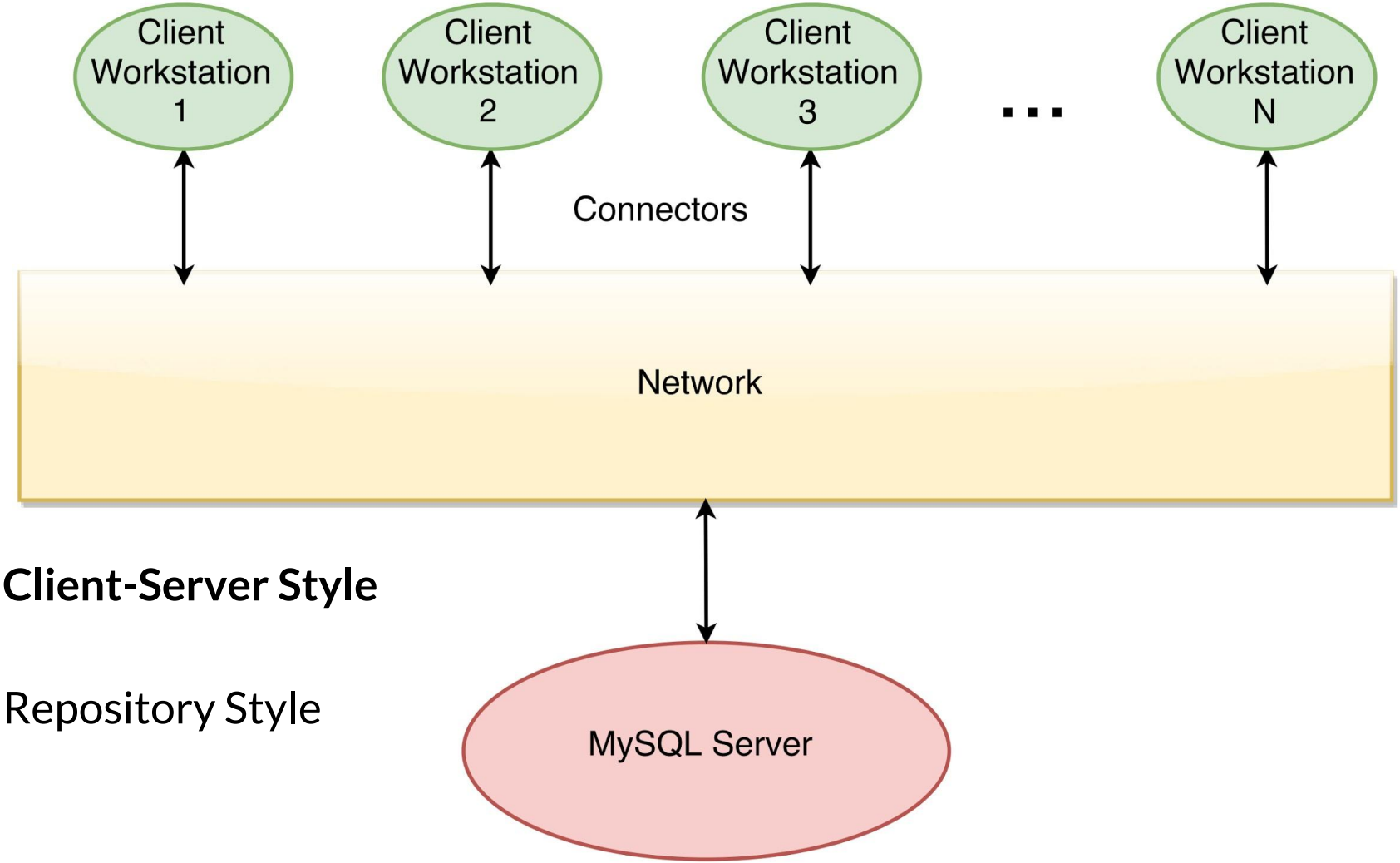
Bell, C. (2012). ***Expert MySQL.*** Berkeley, CA: Apress, p.28.

Figure 2-1. *Client Application/database Server Communication*

Graphical User Interfaces

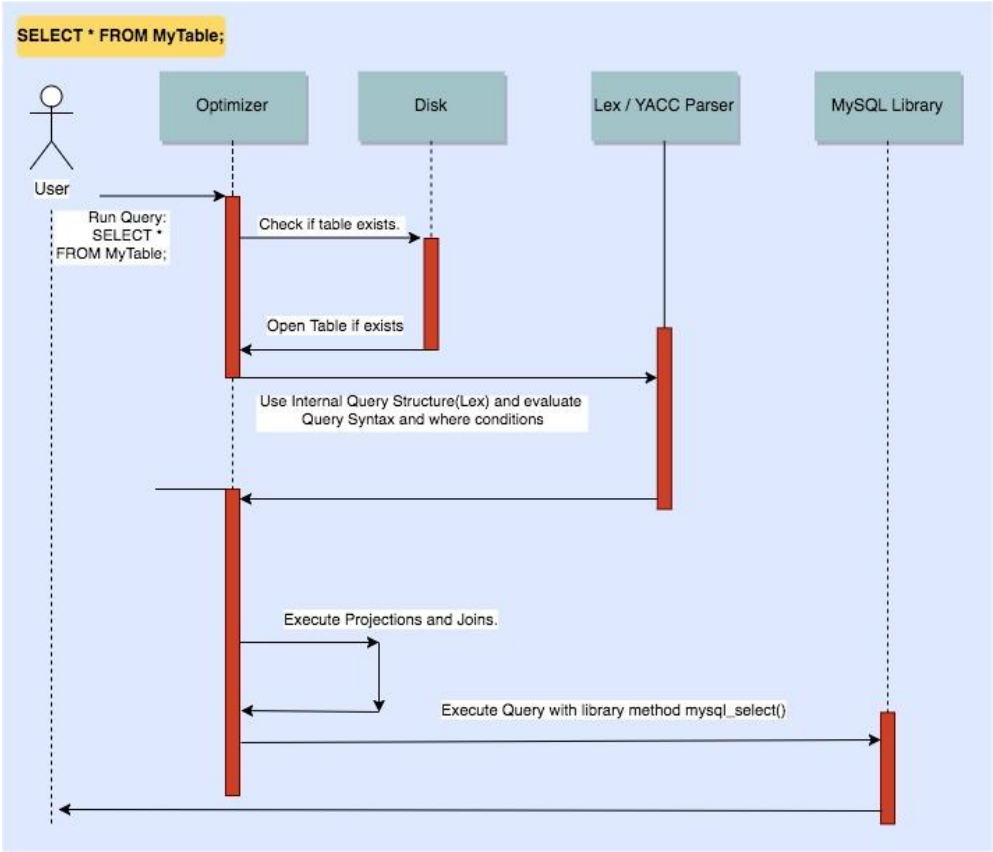
- Alternative to traditional connectors and APIs
- Allow you to view and interact with a user's database graphically
- Official integrated environment is MySQL Workbench. Many other 3rd party options exist.

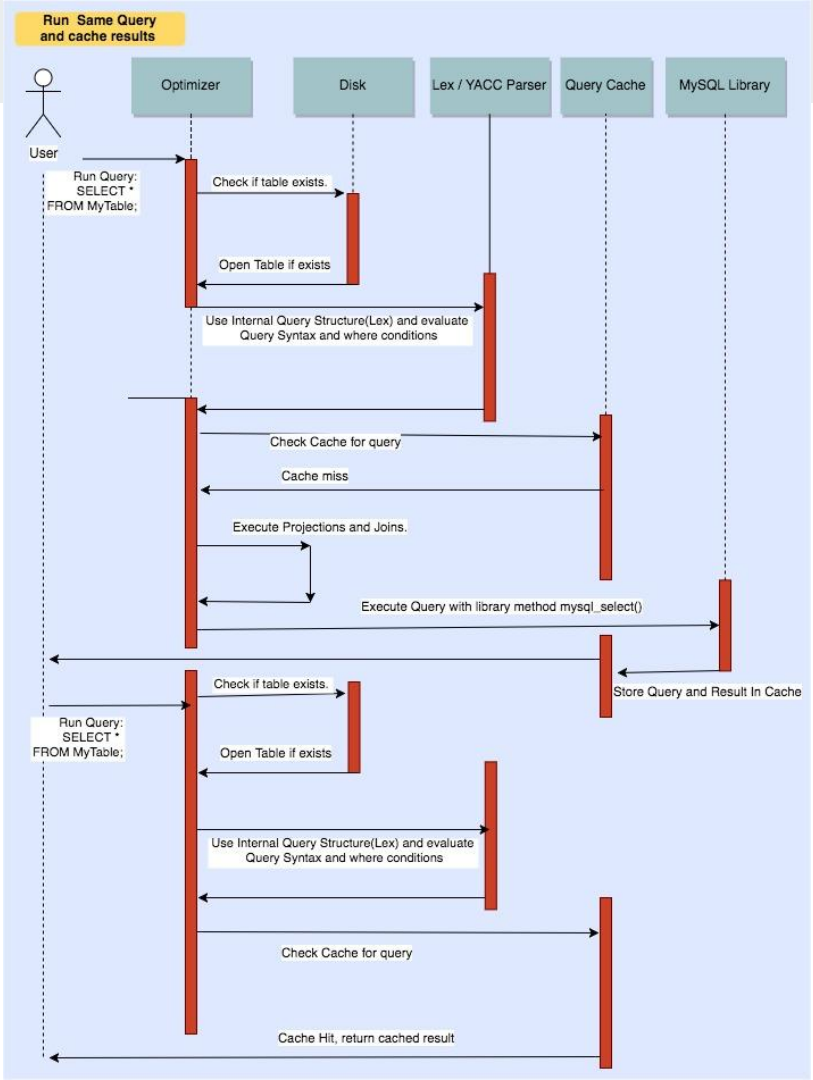

MySQL Workbench
dev_server
File Edit View Query Database Server Tools Scripting Help
Navigator
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
MySQL ENTERPRISE
Audit Inspector
Online Backup
Backup Recovery
SCHEMAS
Filter objects
information_schema
mysql
performance_schema
sakila
test
Tables
city
country
Information
Table: city
Columns:
ID int(11) AI PK
Name char(35)
CountryCode char(3)
District char(20)
Population int(11)
Object Info
Session
Administration - Server Status
SQL File 1*
MySQL Server 5.6
Connection Name
dev_server
Host: MFRANK-US
Socket: MySQL
Port: 3306
Version: 5.6.12-enterprise-commercial-advanced
MySQL Enterprise Server - Advanced Edition (Commercial)
Compiled For: Win64 (x86_64)
Available Server Features
Performance Schema: On
Thread Pool: n/a
Memcached Plugin: n/a
Semisync Replication Plugin: n/a
SSL Availability: Off
Windows Authentication: Off
Password Validation: n/a
Audit Log: n/a
Server Directories
Base Directory: C:\Program Files\MySQL\MySQL Server 5.6\
Data Directory: C:\ProgramData\MySQL\MySQL Server 5.6\data\
Disk Space in Data Dir: 150.00 GB of 326.00 GB available
Plugins Directory: C:\Program Files\MySQL\MySQL Server 5.6\lib\plugin\
Tmp Directory: C:\Windows\SERVIC~2\NETWOR~1\AppData\Local\Temp
Error Log: On .\MFRANK-US.err
General Log: Off
Slow Query Log: Off
Replication Slave
this server is not a slave in a replication setup
Authentication
SHA256 password private key: private_key.pem
SHA256 password public key: public_key.pem
Server Status
Running
CPU
25%
Connections
4
Traffic
21.80 KB/s
Key Efficiency
76.6%
Queries per Second
983
InnoDB Buffer Usage
3.7%
InnoDB Reads per Second
0
InnoDB Writes per Second
989
Executing Query...



- **Client-Server Style**
- Repository Style

Use Cases

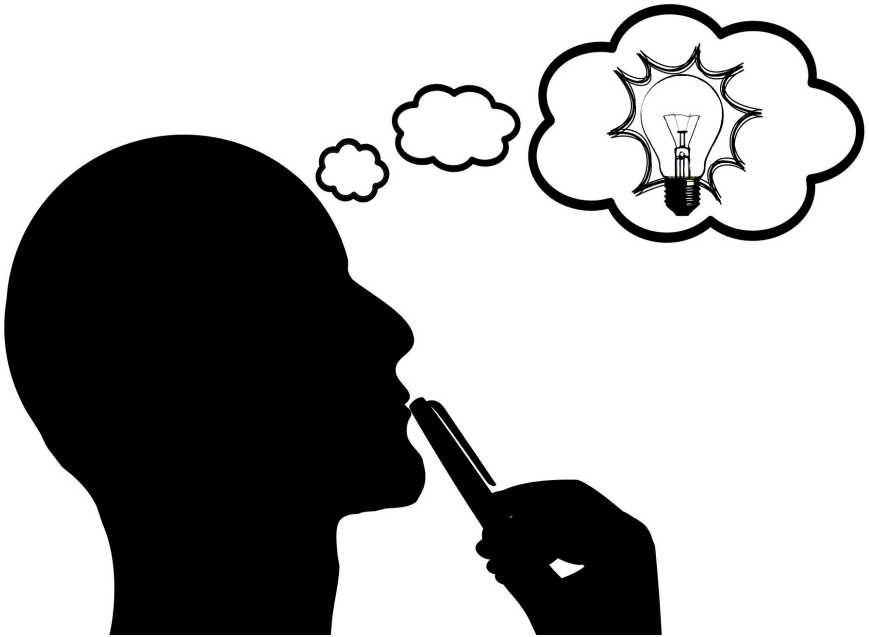

SELECT * FROM MyTable;
Optimizer
Disk
Lex / YACC Parser
MySQL Library
User
Run Query:
SELECT *
FROM MyTable;
Check if table exists.
Open Table if exists
Use Internal Query Structure(Lex) and evaluate
Query Syntax and where conditions
Execute Projections and Joins.
Execute Query with library method mysql_select()


Run Same Query and cache results
Optimizer
Disk
Lex / YACC Parser
Query Cache
MySQL Library
User
Run Query: SELECT * FROM MyTable;
Check if table exists.
Open Table if exists
Use Internal Query Structure(Lex) and evaluate Query Syntax and where conditions
Check Cache for query
Cache miss
Execute Projections and Joins.
Execute Query with library method mysql_select()
Store Query and Result In Cache
Check if table exists.
Run Query: SELECT * FROM MyTable;
Open Table if exists
Use Internal Query Structure(Lex) and evaluate Query Syntax and where conditions
Check Cache for query
Cache Hit, return cached result

Conclusions

- Division of Responsibility
- Storage Engines
- Architecture Burrito

Lessons Learned



Question Period