Assignment 1

Title: Study of open source relational database: MYSQL

Problem Statement: To study open source relational MySQL

Learning Objective: To learn and understand the basic database architecture and various components of it

Learning Outcome: The students will be able to
Understand the basic database architecture and the
various components of it.

S/W packages and hardware apparatus used:
1. MySQL
2. 64-bit Linux based open source OS
3. 8 GB RAM

Concept related theory:
MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons:

MySQL is released under an open-source license. It handles a large subset of the functionality of the most expensive and powerful database packages. MySQL uses a standard form of the well-known SQL data language. MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc. MySQL works very quickly and works well even with large data sets. MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB). MySQL is customizable. The open-source

GPL license allows programmers to modify the MySQL software to fit their own specific environments.

1. Application Layer and Interfaces

MySQL application layer is where the clients and users interact with the MySQL RDBMS. There are three components in this layer. These components illustrate the different kinds of users that can interact with the MySQL RDBMS, which are the administrators, clients and query users. The administrators use the administrative interface and utilities. In MySQL, some of these utilities are MySQL admin which performs tasks like shutting down the server and creating or dropping databases. Clients communicate to the MySQL RDBMS through the interface or utilities. The client interface uses MySQL APIs for various different programming languages such as the C API, DBI API for Perl, PHP API, Java API, Python API, MySQL C++ API and Tcl.

2. Logical Layer

The MySQL documentation gave an indication as to precisely how these modules could be further broken down into subsystems arranged in a layered hierarchy corresponding to the layered architecture in Garlan and Shaw. The following section details these subsystems and the interactions within them.

2.1 Query Processor

The vast majority of interactions in the system occur when a user wishes to view or manipulate the underlying data in storage. These queries, which are specified using a data-manipulation language (ie SQL), are parsed and optimized by a query processor.

2.1.1 Embedded DML Precompiler

When a request is received from a client in the application layer, it is the responsibility of the embedded DML (Data Manipulation Language) precompiler to extract the relevant SQL statements embedded in the client API commands, or to translate the client commands into the corresponding SQL

2.1.2 DDL Compiler

The DDL compiler compiles the commands (which are SQL statements) to interact directly with the database.

2.1.3 Query Parser

It creates a parse tree structure based on the query so that it can be easily understood by the other components later in the pipeline.

2.1.4 Query Preprocessor

It checks the SQL syntax and check the semantics of the MySQL query to determine if the query is valid.

2.1.5 Security/Integration Manager

It checks to see if the client has access to connecting to that particular MySQL database and whether he/she has table and record privileges.

2.1.6 Query Optimizer

It analyzes the processed query to see if it can take advantage of any optimizations that will allow it to process the query more quickly.

2.1.7 Execution Engine

Once the MySQL query optimizer has optimized the MySQL query, the query can then be executed against the database. This is performed by the query execution engine, which then proceeds to execute the SQL statements and access the physical layer of the MySQL database.

2.1.8 Scalability/Evolvability

The layered architecture of the logical layer of the MySQL RDBMS supports the evolvability of the system. If the underlying pipeline of the query processor changes, the other layers in the RDBMS are not affected.

2.2.1 Transaction Manager

The transaction manager is responsible for making sure that q transaction is logged and executed atomically. The transaction manager is also
responsible for resolving any deadlock situations that occur. This situation can occur when two transactions cannot continue because they each have some data that the other needs to proceed. Furthermore, the transaction manager is responsible for issuing the COMMIT and the ROLLBACK SQL commands.

2.2.2 Concurrency-Control Manager

The concurrency-control manager is responsible for making sure that transactions are executed separately and independently. It does so by acquiring locks, from the locking table that is stored in memory, on appropriate pieces of data in the database from the resource manager. Once the lock is acquired, only the operations in one transaction can manipulate the data. If a different transaction
tries to manipulate the same locked data, the concurrency-control manager rejects the request until the first transaction is complete.

2.3 Recovery Management

2.3.1 Log Manager

The log manager is responsible for logging every operation executed in the database. It does so by storing the log on disk through the buffer manager. The operations in the log are stored as MySQL commands.

2.3.2 Recovery Manager

The recovery manager is responsible for restoring the database to its last stable state. It does so by using the log for the database, which is acquired from the buffer manager, and executing each operation in the log.

2.4 Storage Management

All work is done through a number of buffers. The buffers reside in main and virtual memory and are managed by a Buffer Manager. This manager works in conjunction with two other manager entities related to storage: The Resource Manager and the Storage Manager.

2.4.1 Storage Manager

The role of the Storage Manager is to mediate requests between the Buffer Manager and secondary storage.

2.4.2 Buffer Manager

The role of the Buffer Manager is to allocate memory resources for the use of viewing and manipulating data.

2.4.3 Resource Manager

The purpose of the Resource Manager is to accept requests from the execution engine, put them into table requests, and request the tables from the Buffer Manager.

2.5 Evolvability/Scalability

The goals of the Transaction Management subsystem and the Recovery Management subsystem seem to provide non-functional requirements such evolvability and scalability.

CONCLUSION:

Through this assignment, we have learned the basic database architecture and various components associated with it.