
V Skills
Skills for a secure future
Certified MySQL DB
Administrator
Sample Material
V-Skills Certifications
A Government of India
&
Government of NCT Delhi Initiative
V-Skills
Skills for a secure future

1. MYSQL BASICS

MySQL is a relational database management system (RDBMS), and ships with no GUI tools to administer MySQL databases or manage data contained within the databases. Users may use the included command line tools, or use MySQL "front-ends", desktop software and web applications that create and manage MySQL databases, build database structures, back up data, inspect status, and work with data records. The official set of MySQL front-end tools, MySQL Workbench is actively developed by Oracle, and is freely available for use.

1.1. Introduction

MySQL is (as of March 2014) the world's second most widely used open-source relational database management system (RDBMS). It is named after co-founder Michael Widenius's daughter, My. The SQL phrase stands for Structured Query Language.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other 'AMP' stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL.

For proprietary use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, MODx, Joomla, WordPress, phpBB, MyBB, Drupal and other software. MySQL is also used in many high-profile, large-scale websites, including Google (though not for searches), Facebook, Twitter, Flickr, and YouTube.

Many companies use MySQL mainly because it's free, reliable, easy to setup and maintain. I have be using MySQL for many years but have never really gone into depth, mainly because once it has been installed it happily sits working for years with only slight modifications, the reason is that it is a simple database and not to bloated with many features that you don't need, the other feature is that MySQL can have a number of different engine types to suit the application.

MySQL is an open source database which means it's freely available with free redistribution, this means you have full access to the source code. MySQL began as Unireg that was developed by Michael "Monty" Widenius for a swedish company called TcX during the 1980's, the My part is Monty's daughters name. The initial release in 1995 had a SQL interface and a dual license model, a free and an embedded version. David Axmark, Monty and Allen Larrson founded MySQL AB in 1995, it was taken over by Sun Microsystems in 2008 and Sun itself was taken by Oracle in 2010.

MySQL is written in C and C++ and in 2001 MySQL began supporting transactions with the integration of the BDB and InnoDB engines (the default engine), this allowed for safer handling of concurrent write operations, which began the trend of adding features needed by the Enterprise

environments. MySQL supports the following platforms and has both 32-bit and 64-bit versions available

- ✓ Linux
- ✓ Solaris
- ✓ Windows
- ✓ AIX
- ✓ HPUX

Although MySQL comes with no tools, there are a number of graphical tools available the main one being the MySQL workbench. MySQL comes in a number of flavors, commercial customers have a number of different choices depending on your needs.

- ✓ Community Server (free edition)
- ✓ Standard Edition
- ✓ Enterprise Edition
- ✓ Cluster Carrier Grade edition

All the versions have the following features

- ✓ Pluggable Storage Engine Architecture
- ✓ Multiple Storage Engines InnoDB, MyISAM, NDB (MySQL Cluster), Memory, Merge, Archive, CSV, etc
- ✓ Replication
- ✓ Partitioning
- ✓ Stored Procedures, Triggers, Views
- ✓ Information Schema
- ✓ MySQL connectors (ODBC, JDBC, .NET, etc)
- ✓ MySQL Workbench for visual modeling, SQL development and administration.

1.2. Client/Server Basics

The client–server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.

Client and server roles

The client–server characteristic describes the relationship of cooperating programs in an application. The server component provides a function or service to one or many clients, which initiate requests for such services.

Servers are classified by the services they provide. For instance, a web server serves web pages and a file server serves computer files. A shared resource may be any of the server computer's software and electronic components, from programs and data to processors and storage devices. The sharing of resources of a server constitute a service.

Whether a computer is a client, a server, or both, is determined by the nature of the application that requires the service functions. For example, a single computer can run web server and file server software at the same time to serve different data to clients making different kinds of requests. Client software can also communicate with server software within the same computer. Communication between servers, such as to synchronize data, is sometimes called inter-server or server-to-server communication.

Example

When a bank customer accesses online banking services with a web browser (the client), the client initiates a request to the bank's web server. The customer's login credentials may be stored in a database, and the web server accesses the database server as a client. An application server interprets the returned data by applying the bank's business logic, and provides the output to the web server. Finally, the web server returns the result to the client web browser for display.

In each step of this sequence of client–server message exchanges, a computer processes a request and returns data. This is the request-response messaging pattern. When all the requests are met, the sequence is complete and the web browser presents the data to the customer.
This example illustrates a design pattern applicable to the client–server model: separation of concerns.

Client-server is a software architecture model consisting of two parts, client systems and server systems, both communicate over a computer network or on the same computer. A client-server application is a distributed system consisting of both client and server software. The client process always initiates a connection to the server, while the server process always waits for requests from any client. When both the client process and server process are running on the same computer, this is called a single seat setup.

Another type of related software architecture is known as peer-to-peer, because each host or application instance can simultaneously act as both a client and a server (unlike centralized servers of the client-server model) and because each has equivalent responsibilities and status. Peer-to-peer architectures are often abbreviated using the acronym P2P.

The client-server relationship describes the relation between the client and how it makes a service request to the server, and how the server can accept these requests, process them, and return the requested information to the client. The interaction between client and server is often described using sequence diagrams. Sequence diagrams are standardized in the Unified Modeling Language.

Both client-server and P2P architectures are in wide usage today.

The basic type of client-server architecture employs only two types of hosts: clients and servers. This type of architecture is sometimes referred to as two-tier. The two-tier architecture means that the client acts as one tier and server process acts as the other tier.

The client-server architecture has become one of the basic models of network computing. Many types of applications have being written using the client-server model. Standard networked functions such as E-mail exchange, web access and database access, are based on the client-server model. For example, a web browser is a client program at the user computer that may access information at any web server in the world.

Clients Characteristics

- ✓ Always initiates requests to servers.
- ✓ Waits for replies.
- ✓ Receives replies.
- ✓ Usually connects to a small number of servers at one time.
- ✓ Usually interacts directly with end-users using any user interface such as graphical user interface.

Server Characteristics

- ✓ Always wait for a request from one of the clients.
- ✓ Serve clients requests then replies with requested data to the clients.
- ✓ A server may communicate with other servers in order to serve a client request.
- ✓ A server is a source which sends request to client to get needed data of users.

Advantages

- ✓ In most cases, client-server architecture enables the roles and responsibilities of a computing system to be distributed among several independent computers that are known to each other only through a network, so one of advantages of this model is greater ease of maintenance. For example, it is possible to replace, repair, upgrade, or even relocate a server while its clients remain both unaware and unaffected by that change. This independence from change is also referred to as encapsulation.
- ✓ All the data is stored on the servers, which generally have better security controls than most clients. Servers can better control access and resources, to guarantee that only those clients with the appropriate permissions may access and change data.
- ✓ Since data storage is centralized, updates to that data are much easier to administrators than what would be possible under a P2P architecture. Under a P2P architecture, data updates may need to be distributed and applied to each "peer" in the network, which is both time-consuming and error-prone, as there can be thousands or even millions of peers.
- ✓ Many advanced client-server technologies are already available which were designed to ensure security, user friendly interfaces, and ease of use.
- ✓ It works with multiple different clients of different specifications.

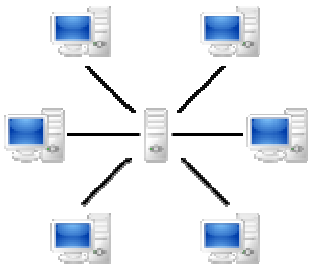
Disadvantages

- ✓ Networks traffic blocking is one of the problems related to the client-server model. As the number of simultaneous client requests to a given server increases, the server can become overloaded. Contrast that to a P2P network, where its bandwidth actually increases as more nodes are added, since the P2P network's overall bandwidth can be roughly computed as the sum of the bandwidths of every node in that network.
- ✓ Comparing client-server model to the P2P model, if one server fail, clients' requests cannot be served but in case of P2P networks, servers are usually distributed among many nodes. Even if

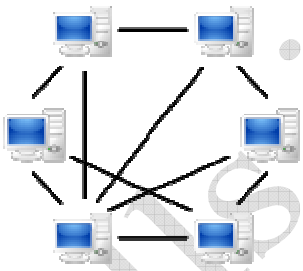
one or more nodes fail, for example if a node failed to download a file the remaining nodes should still have the data needed to complete the download.

Examples includes

- ✓ Specific types of clients include: web browsers, E-mail clients, and online chat clients.
- ✓ Specific types of servers include: web servers, FTP servers, database servers, E-mail servers, file servers, print servers. Most web services are also types of servers.



A one server based network.



A peer-to-peer based network.

Comparison with peer-to-peer architecture

In addition to the client-server model, distributed computing applications often use the peer-to-peer application architecture.

In the client–server model, the server is often designed to be a centralized system that serves many clients. The computing power, memory and storage requirements of a server must be scaled appropriately to the expected work load, i.e. the number of clients connecting simultaneously. Load balancing and failover systems are often employed to scale the server implementation.

In a peer-to-peer (P2P) network, two or more computers (peers) pool their resources and communicate in a decentralized system. Peers are coequal, or equipotent nodes in a non-hierarchical network. Unlike clients in a client–server or client–queue–client network, peers communicate with each other directly. In peer-to-peer networking, an algorithm in the peer-to-peer communications protocol balances load, and even peers with modest resources can help to share the load. If a node becomes unavailable, its shared resources remain available as long as other peers offers it. Ideally, a peer does not need to achieve high availability because other, redundant peers make up for any resource downtime; as the availability and load capacity of peers change, the protocol reroutes requests.

The MySQL database system operates using a client/server architecture. The server is the central program that manages database contents, and client programs connect to the server to retrieve or modify data. MySQL also includes non-client utility programs and scripts. Thus, a complete MySQL installation consists of three general categories of programs

- ✓ MySQL Server - This is the mysqld program that manages databases and tables.

- ✓ Client programs - These are programs that communicate with the server by sending requests to it over a network connection. The client programs included with MySQL distributions are character-based programs and graphical interface programs that display output to your terminal.
- ✓ Non-client utility programs - These are programs that generally are used for special purposes like mysqld_safe is a script for starting up and monitoring the server.

1.3. Database, DBMS and RDBMS

Database

A database is an integrated collection of well defined data and information, centrally controlled in all aspects, created and stored in a typical structure for an organization. In an organization the database could be one or more depending upon the needs and operations of the organization. The data structure and its storage should be such that it facilitates, share ability, availability, resolvability and integrity of the data. The database separates a design of the information system from the data design and its management.

Conventionally, in an information system, the information system, the information is obtained by developing the system and integrating them. This calls for breaking the system into various subsystems and developing the information systems independently. In this approach, each system will have its master files and transaction files. They have to be processed separately at different times, depending upon the needs and schedules. The file layouts and the access methods could be different in different systems. Therefore, the files are updated at different times. This approach does affect the quality of the information across all the systems due to various reasons.

The data in many systems are common, and there is repetition of data storage in various systems. This is called data redundancy. The redundancy of data current and same in all the files, the management is complex in such a situation. The reports generated out of such files show discrepancies in the information. Since the data files are different for different need to be created at different times. Transaction updating is also carried out at different times. It requires the increase of a magnetic media for storage because the systems are developed independently. The redundancy causes lack of integrity and inconsistency of the data in various files. The following are the main causes which gave birth to database

- ✓ The data redundancy and inconsistency: since the files are created for each application differently, the files are likely to have different formats and data design as they are created by different designers and programmers over in period of time. Hence, the same data record may be present in more than on file, the creation, updating, and deletion of which is managed by different programmer. Over a period of time, a situation arises when the data is redundant and inconsistent, due to the changes not being incorporated simultaneously in all the applications and in all the files.
- ✓ Difficulty in access to the data: in conventional system design, the file structure is consistent to the specific information needs. If the information needs change, gaining access to the data present in different files to satisfy the revised need, requires writing the necessary application programs every time. This is difficult and very time consuming at critical times.
- ✓ Security problems: the file systems have a limitation of controlling the access to the record causing in security with respect to the information. Since the application programs are written

time to time and again, it is difficult to enforce a discipline on the security constraints across all the applications.

- ✓ Integrity of the data: in a file system, it is difficult to maintain integrity of the data across the application. The integrity rules are added when the programs are written. If any changes in the rules occur, it is very difficult to ensure that it is affected across the files in all the applications.
- ✓ To eliminate the above problems, database approach is suggested. The database is designed independent of its use for the application. In other words, the user of the database should view the data in the database as on which he can develop his systems separately. Since, the database matching his needs is common, the problems of redundancy and inconsistency are eliminated.

The advantages in the database approach are as follows

- ✓ The application system can be developed independent of the database
- ✓ Any report using the information will not be inconsistent
- ✓ Easy access to the database as per needs
- ✓ The data validation and updating will be once and same for all
- ✓ The data is shared by all the users
- ✓ The data security and privacy can be managed and ensured because data entry in the database occurs once only and is protected by the security measures
- ✓ Since the database is a storage of the structured i9nformation, the queries can be answered fast by using the logic of the data structures

DATABASE MANAGEMENT SYSTEM (DBMS)

The DBMS is software designed to manage and maintain the database of an organization. The main steps are data structuring, defining, integrating, updating and creating. Through these steps, it manipulates the data and provides an environment which is appropriate to use in retrieving and storing the database information.

The DBMS is a collection of the interrelated files and a set of programs through which the users can access and modify their files. DBMS is software that facilitates flexible management of data. It is generally composed of three sub-systems which are described as follows

- ✓ Database definition: in this sub-system, the complete database is described with help of a special language known as the Data Description Language (DDL). However, in the case of database in different files, one file at a time may be defined as that would give maximum flexibility.
- ✓ Database manipulation: after the database is defined, elements of data can be stored. The stored data may either be retrieved or updated later through data manipulation language (DML). The manipulation sub-system can retrieve the required elements of data in a variety of sequences.
- ✓ Database support: this sub-system performs database utility or service functions that include functions like list files, change file passwords, change file capacities, print file statistics, unlock files, etc.

In DBMS performs a wide variety of functions, which are discussed as follows

- ✓ Data organization: DBMS organizes data items as per DDl. Database administrator decides about the data specification that are most suited top each applications.
- ✓ Data integration: data is inter-related together at the element level and can be manipulated in much combination during execution of a particular application program. DBMS facilitates collection, combination, and retrieval of the required data to the user.
- ✓ Physical/logical level separation: DBMS separates the logical description and relationships of data from the way in which the data is physically stored. It also separates out application, programs and their associated data. This adds data insecurity in view of the data access by different programs that describe data in different ways.
- ✓ Data control: DBMS receives requests for storing data from different programs. It controls how and where data is physically stored. Similarly, it locates and returns requested data to the programs.
- ✓ Data protection: data protection and security is one of the major concerns in a database. DBMS protects the data against access by unauthorized users, physical damage, and operating system failure, simultaneous updating, etc. it also protects and secures the content of a database as well as the relationships of data elements. DBMS is equipped with a facility to backup data and restore it automatically in the case of any system failure. Concurrent access control is ensured by the provision of 'locks'. Other security features implemented in the system include password protection and sophisticated encryption schemes.
- ✓ Data stores: the data are stored in a database in a particular manner. The user of the database needs not to know how the data are actually stored.
- ✓ Data definition and data directory: since the data independence is provided to the user, it requires that all the users understand the data in same manner. Therefore, each data entity is defined in the system and its directory is formed for all the users.
  Interrogation: in interrogation, the data are selected from the database and extracted or copied for processing. For interrogation, it is necessary to identify the data or a part of the data and then through the use of query language the information is processed and printed.
- ✓ Updating: the database needs updating as the value of the data keep changing from time to time. The updates are made by processing the transaction data against the data in the database.
- ✓ Creation: initially, the database is to be created in the manner and the kind as defined in the DBMS. The data is entered in the database by the transaction processing. A special program is written to create the database. The DBMS organizes the data internally in the structure defined in the DBMS.

Types of Database Structure or Data Models

Generally, database system are classified on the basis of one of the three data models, they use in building the conceptual structure or schema of the database. The three models are

- ✓ Hierarchical model
- ✓ Network model
- ✓ Relational model

Hierarchical Model- in the hierarchical structure, the relationships between records are stored in the form of a hierarchy or a tree which has a root. In this model, all records are dependent and arranged in a multiline structure, thus the root may have a number of branches and each branch

may have a number of sub-branches and so on. The lowermost record is known as the 'child' of the next higher level record, whereas the higher level record is called the 'parent' of its child records. Thus in this approach, all the relationships among records are one-to-many. The hierarchical database model (HDBM) is applicable when the data in an organization can be put down in the hierarchical or in terms of the levels, one after another. The data model is equivalent to the tree. A tree has roots, branches and leaves; their equivalent s in the HDBM beings records, notes and field. In the HDBM the data is stored in the hierarchical form recognizing the fact that each of the levels is bounded by the parent-child relations to the earlier level. The typical characteristics of HDBM are

HDBM starts with a root and has several roots

- ✓ A root will have several branches
- ✓ Each branches is connected to one and only one root
- ✓ A branch has several leaves and a set of leaves are connected to one branch

Network database model - the NDBM interconnects the entities of an organization into a network. The NDBM deals with the set of the records components, a part, a subassembly and an assembly are the records. A record located at the tail of the arrow is known as a member record at the head of the arrow is known as an owner. An arrow connecting the owner to a member is a set.

Relational Database Model - in the RDBM, the concept of two dimensional tables is used to show the relations. The relational model is currently more popular among users and developers because of the many interrelated tables, the overall design may get complicated which may lead to slower searches and thus affecting the access time.

RDBMS

Organization needs that MIS would give them a 'competitive strength'. Simply computerizing the back office or the front office operations is no lo0nger sufficient. The need is to handle an on-line operation, mission, control applications and exercise the operational and management control. The need demands a tool to effectively handle both the transaction processing and the decision processing requirements. It also requires the capability of dealing with hundreds of users who are using, and updating a large database. The need further demands the use of multiple databases residing on the hardware platforms situated at different location-nearby sites and remote site. The decision-making is required more in a real time environment where the decision-making process, right from the problem definition to solution, needs to be handled quickly. The business environment is distributed and decentralized requiring a real time resources (hardware, software, data, information) sharing with a complex data flow. All this demands the RDBMS which can serve both the decision support and the transaction processing requirements.

The latest RDBMS systems have two sub-systems or parts. One deals with the data management and transaction processing which is independent of its applications in the information processing. The second part provides a set of tools for developing and utilizing on-line application for the decision support. This is handled by the client-server architecture which separates the data management functions from its application. The data management function is handled by the server and the applications are handled by the client. The server centrally enforces all integrity, security and autonomy rules and the client (user) makes use of the database over the network of heterogeneous hardware.

The latest trend in the information technology is to make the end user computing simple, easy to understand and easy to use. The concept is extended to the system analyst and programmer, where the RDBMS provides the tools, saving, development and processing time. It allows the business rules of the organization, standard transactions and queries to be programmed once and makes them available to all the users and developers as a stored procedure in the data dictionary. These stored procedures can be nested to develop an application. These procedures are both, reusable and sharable and are developed using the standard SQL. The RDBMS is also capable through the interface to handle the data sources from the other database and application tools developed on different operating systems.

The modern RDBMS system operates under the client-server environment as against the traditional master-slave environment. In the traditional DBMS system, a transaction is processed in the database, i.e. the creation, validation and checking the transaction integrity logic. This is done for each transaction separately based on the procedures developed forced transaction for validity and integrity checks. In the modern RDBMS system, the third step of the integrity checking is done through a stored procedure common to all types of transaction.
Modern RDBMS allows high level security by providing various tools to the system administrators, the database owners and the users to grant and revoke permission to the specified users or a group of users on the specified tables, view, columns, stored procedures and commands. In the traditional DBMS system, the data was required to store in different databases In line with the security levels. While in the modern RDBMS system a multiple security is taken care of by one integrated DBMS.

The latest RDBMS allows an online maintenance, rapid recovery and software based fault tolerance. These features ensure the availability of the database round the clock as the database maintenance is possible online when the system is in use. The maintenance activity consists of the following tasks

- ✓ Backup
- ✓ Diagnostic
- ✓ Integrity changes
- ✓ Recovery
- ✓ Design changes
- ✓ Performance tuning

The rapid recovery feature also the system administrator to provide a 'time' to go back for recovery of the data if the system fails due to the power failure or network crash. Based on this, time systems automatically go and collect all the changes and writes the risk.

The modern RDBMS, unlike the traditional DBMS, handles the distributed heterogeneous data sources, software environment and hardware platforms. The system is open RDBMS. The modern business activity using the multiple hardware-software platforms, such a business enterprise has multiple databases residing at various locations. The information needs call for the unification and coordination of these databases. The data would get updated in distributed access to the distributed data.

RDBMS allows communication at the database level and performs in a unified manner as a single entity through the updates, and processing would take place at the respective distributed locations. This is achieved through a software interface across the organization. Since the environment is distributed, it calls for a distributed integrity control and autonomy to perform. The distributed integrity control is achieved through the stored procedures is made to protect the database from a remote external update or processing to maintain the autonomy which would be affected by the unauthorized update from remote locations.

The characteristics of modern RDBMS includes hardware independency, software independence, workability under a client-server architecture, a control feature of integrity, security and autonomy and built-in communication facilities to achieve and open the system features for the MIS. It therefore provides a very efficient and effective tool to a skillful designer, developer and user for handling the information needs of the business enterprise.

1.4. Communication Protocols

Every type of communication, natural or man made, comes in two basic forms - synchronous or asynchronous.

- ✓ Synchronous communication means that for each message, the sender expects to get a reply from the receiver, like a telephone call. There is a back-and-forth exchange, so that each party knows that the other is receiving the message. If there is no response, you can be pretty sure that communication didn't occur.
- ✓ Asynchronous communication means that a message gets sent but the receiver is not expected to reply, like a radio broadcast or a newspaper.

In general, a service is an abstraction of computer resources and a client does not have to be concerned with how the server performs while fulfilling the request and delivering the response. The client only has to understand the response based on the well-known application protocol, i.e. the content and the formatting of the data for the requested service.

Clients and servers exchange messages in a request-response messaging pattern: The client sends a request, and the server returns a response. This exchange of messages is an example of inter-process communication. To communicate, the computers must have a common language, and they must follow rules so that both the client and the server know what to expect. The language and rules of communication are defined in a communications protocol. All client-server protocols operate in the application layer. The application-layer protocol defines the basic patterns of the dialogue. To formalize the data exchange even further, the server may implement an API (such as a web service). The API is an abstraction layer for such resources as databases and custom software. By restricting communication to a specific content format, it facilitates parsing. By abstracting access, it facilitates cross-platform data exchange.

A server may receive requests from many different clients in a very short period of time. Because the computer can perform a limited number of tasks at any moment, it relies on a scheduling system to prioritize incoming requests from clients in order to accommodate them all in turn. To prevent abuse and maximize uptime, the server's software limits how a client can use the server's resources. Even so, a server is not immune from abuse. A denial of service attack exploits a server's

obligation to process requests by bombarding it with requests incessantly. This inhibits the server's ability to responding to legitimate requests.

The Relational Database Management Systems are based on the client-server architecture. There is a database server which handles queries incoming from multiple clients, called also users for simplicity. A single RDBMS handles multiple connections with the users at the same time.

Almost each of RDBMS available these days has its own communication protocol. It is usually provided to programmers as a library of API (Application Programming Interface) functions. A programmer is free to use such a library in his or her program to establish a two way communication between the program and the database. The program, which is a client application, can communicate with a given database system for which the library provides the communication protocol.

For example, an application designed to communicate with Oracle is capable of communication with Oracle only. It cannot be used to connect to another RDBMS, unless this RDBMS provides the same communication protocol as Oracle does.

The protocol serves the purpose of establishing the communication. It allows to send a query, and to retrieve the reply. It is also called the call-level-interface CLI. Concerning RDBMS, the query is sent in a form conforming to the SQL syntax.

Having a separate CLI for each database significantly reduces flexibility. The first attempt to standardize the CLI was made in '90s by the SQLAccess Group (SAG). The SA specification served as the basis of ODBC: Open Database Connectivity specification developed by Microsoft10.1. The interface is pretty complicated, but it became the standard for the database connectivity. It is an API (Application Programming Interface), a set of functions calls based on the SQLAccess Group specification.

ODBC is provided as three components: the ODBC Library, ODBC Driver Manager and ODBC Driver. Applications use the Library to communicate with an abstract data source using ODBC functions. The data source is provided by the Driver Manager along with an ODBC driver. The ODBC driver is a bridge between ODBC and the database.

The application accesses ODBC functions, through ODBC Driver Manager, which dynamically links to the appropriate ODBC driver. The driver translates ODBC request to native format specific for a particular database. An application, based on ODBC, can communicate with any data source which provides the ODBC driver. There are such drivers for most of the database systems these days, the list includes, but it is not limited to: Oracle, Informix, DB2, PostgreSQL, MySQL, Interbase.

MySQL Communication Protocols

Generally, MySQL supports connections between clients and the server using several networking protocols. Below are the main protocols which are used by client to connect with MySQL server.

- ✓ TCP/IP
- ✓ Unix socket file

- ✓ Named pipe
- ✓ Shared memory

TCP/IP connections are supported by any MySQL server unless the server is started with the –skip-networking option. These connections are supported by all type of Operating Systems and also connect from locally or remotely.

Unix socket file connections are supported by all Unix servers and this can be connected from local only. Named-pipe connections are supported only by Windows servers and are disabled by default. To enable named-pipe connections, you have to start the mysql-nt server with the –enable-named-pipe option.

Shared-memory connections are supported by all Windows servers and are disabled by default. To enable shared-memory connections, you have to start the server with the –shared-memory option.

From the client perspective, a client run on the same host as the server can use any of the connection protocols that the server supports but If the client will run on a different host, connections always use TCP/IP. MySQL communication protocols are implemented by various libraries and program drivers. Client programs included with MySQL distributions like mysql, mysqladmin etc establish connections to the server using the native C client library.

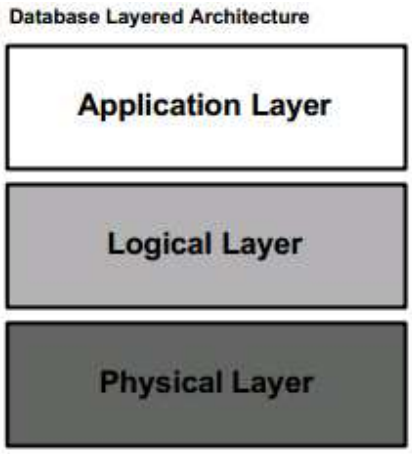
The different connection methods are not all equally efficient. In many Windows configurations, communication via named pipes is much slower than using TCP/IP. You should use named pipes only when you choose to disable TCP/IP (using the –skip-networking parameter) or when you can confirm that named pipes actually are faster for your particular setup.

On Unix, a Unix socket file connection provides better performance than a TCP/IP connection. On any platform, an ODBC connection made via MySQL Connector/ODBC is slower than a connection established directly using the native C client library. This is because ODBC is layered on top of the C library, which adds overhead. On any platform, a JDBC connection made via MySQL Connector/J is likely to be roughly about the same speed as a connection established using the native C client library.

1.5. MySQL Architecture

RDBMS Architecture

It was found in all of the consulted resources that all database systems, can be viewed as a Garlan & Shaw layered architecture at the highest level of abstraction. It has three main components as outlined in the following diagram



Application Layer

The application layer represents the interface for all the users of the system; it essentially provides the means by which the outside world can interact with the database server. In general, it has been found that users can be categorized into four main groups

- ✓ Sophisticated users interact with the system without using any form of application; they form their requests directly with the use of a database query language.
- ✓ Specialized users are application programmers who write specialized database applications that do not fit into the traditional data-processing framework.
- ✓ Naïve users are unsophisticated users who interact with the system by invoking one of the permanent application programs that have been previously written.
- ✓ Database Administrators have complete control over the entire database system. They have a wide variety of responsibilities, including schema definition, the granting of access authorization, as well as the specification of integrity constraints.

All database systems provide extensive services for each of these groups; it is then left to the logical layer to appropriately process the varying requests.

Logical Layer

The core functionality of the RDBMS is represented in the logical layer of the architecture; it is in this portion of the system that there are a wide variety of vendor specific implementations. However, upon reading a number of general database management texts, it was found that general core logical functionality can indeed be abstracted. The following diagram is a high level abstraction of the modules found in any robust RDBMS.

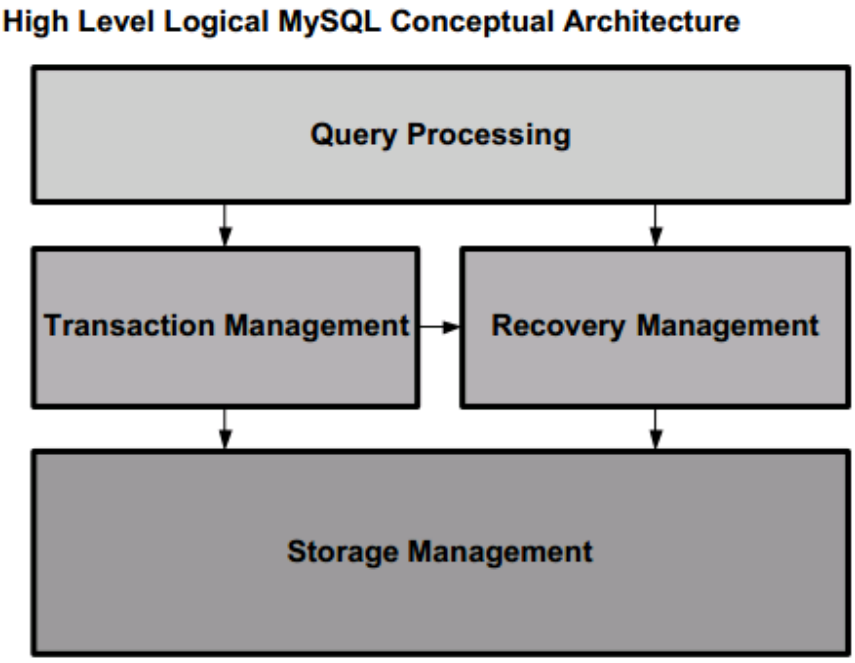


Figure 2: General High Level RDBMS Logical Modules

It can be observed that there are four main modules present in the RDBMS; their implementations and interactions have been summarized based on general documentation from a variety of sources. Specific implementations and interactions between the subsystems vary greatly from vendor to vendor.

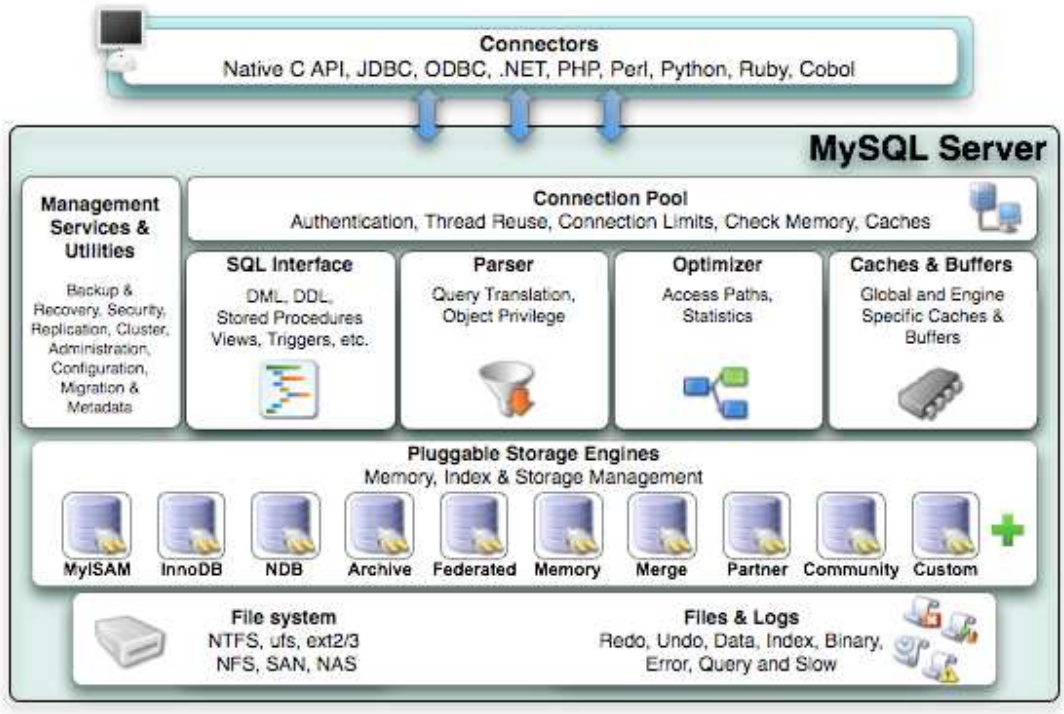
Physical Layer

The RDBMS is responsible for the storage of a variety of information, which is kept in secondary storage and accessed via the storage manager. The main types of data kept in the system are:

- ✓ Data files, which store the user data in the database
- ✓ Data dictionary, which stores metadata about the structure of the database
- ✓ Indices, which provide fast access to data items that hold particular values
- ✓ Statistical Data, which store statistical information about the data in the database; it is used by the query processor to select efficient ways to execute a query.
- ✓ Log Information, used to keep track of executed queries such that the recovery manager can use the information to successfully recover the database in the case of a system crash.

MySQL Architecture

MySQL is based on tiered architecture, consisting of both subsystems and support components that interact with each other to read, parse and execute queries and to cache and return query results. MySQL architecture consists of five primary subsystems that work together to respond to a request made to MySQL database server.



Query Engine

The SQL Interface - The SQL interface provides the mechanisms to receive commands and transmit results to the user. The MySQL SQL interface was built to the ANSI SQL standard and accepts the same basic SQL statements as most ANSI-compliant database servers. Although many of the SQL commands supported in MySQL have options that are not ANSI standard, the MySQL developers have stayed very close to the ANSI SQL standard.

Connections to the database server are received from the network communication pathways and a thread is created for each. The threaded process is the heart of the executable pathway in the MySQL server. MySQL is built as a true multithreaded application whereby each thread executes independently of the other threads (except for certain helper threads). The incoming SQL command is stored in a class structure and the results are transmitted to the client by writing the results out to the network communication protocols. Once a thread has been created, the MySQL server attempts to parse the SQL command and store the parts in the internal data structure.

Parser - When a client issues a query, a new thread is created and the SQL statement is forwarded to the parser for syntactic validation (or rejection due to errors). The MySQL parser is implemented using a large Lex-YACC script that is compiled with Bison. The parser constructs a query structure used to represent the query statement (SQL) in memory as a tree structure (also called an abstract syntax tree) that can be used to execute the query.

Query Optimizer - The MySQL query optimizer subsystem is considered by some to be misnamed. The optimizer used is a SELECT-PROJECT-JOIN strategy that attempts to restructure the query by first doing any restrictions (SELECT) to narrow the number of tuples to

work with, then performs the projections to reduce the number of attributes (fields) in the resulting tuples, and finally evaluates any join conditions. While not considered a member of the extremely complicated query optimizer category, the SELECT-PROJECT-JOIN strategy falls into the category of heuristic optimizers. In this case, the heuristics (rules) are simply

- ✓ Horizontally eliminate extra data by evaluating the expressions in the WHERE (HAVING) clause.
- ✓ Vertically eliminate extra data by limiting the data to the attributes specified in the attribute list. The exception is the storage of the attributes used in the join clause that may not be kept in the final query.
- ✓ Evaluate join expressions.

This results in a strategy that ensures a known-good access method to retrieve data in an efficient manner. Despite critical reviews, the SELECT-PROJECT-JOIN strategy has proven effective at executing the typical queries found in transaction processing.

The first step in the optimizer is to check for the existence of tables and access control by the user. If there are errors, the appropriate error message is returned and control returns to the thread manager, or listener. Once the correct tables have been identified, they are opened and the appropriate locks are applied for concurrency control.

Once all of the maintenance and setup tasks are complete, the optimizer uses the internal query structure and evaluates the WHERE conditions (a restrict operation) of the query. Results are returned as temporary tables to prepare for the next step. If UNION operators are present, the optimizer executes the SELECT portions of all statements in a loop before continuing.

The next step in the optimizer is to execute the projections. These are executed in a similar manner as the restrict portions, again storing the intermediate results as temporary tables and saving only those attributes specified in the column specification in the SELECT statement.

Lastly, the structure is analyzed for any JOIN conditions that are built using the join class, and then the join::optimize() method is called. At this stage the query is optimized by evaluating the expressions and eliminating any conditions that result in dead branches or always true or always false conditions (as well as many other similar optimizations). The optimizer is attempting to eliminate any known-bad conditions in the query before executing the join. This is done because joins are the most expensive and time consuming of all of the relational operators. The join optimization step is performed for all queries that have a WHERE or HAVING clause regardless of whether there are any join conditions. This enables developers to concentrate all of the expression evaluation code in one place. Once the join optimization is complete, the optimizer uses a series of conditional statements to route the query to the appropriate library method for execution.

Query Execution - Execution of the query is handled by a set of library methods designed to implement a particular query. For example, the mysql_insert() method is designed to insert data. Likewise, there is a mysql_select() method designed to find and return data matching the WHERE clause. This library of execution methods is located in a variety of source code files under a file of a similar name (e.g., sql_insert.cc or sql_select.cc). All of these methods have as a parameter a

thread object that permits the method to access the internal query structure and eases execution. Results from each of the execution methods are returned using the network communication pathways library. The query execution library methods are clearly implemented using the interpretative model of query execution

Query Cache - While not its own subsystem, the query cache should be considered a vital part of the query optimization and execution subsystem. The query cache is a marvelous invention that caches not only the query structure but also the query results themselves. This enables the system to check for frequently used queries and shortcut the entire query optimization and execution stages altogether. This is another of the technologies that is unique to MySQL. Other database system cache queries, but no others cache the actual results. As you can appreciate, the query cache must also allow for situations where the results are "dirty" in the sense that something has changed since the last time the query was run (e.g., an INSERT, UPDATE, or DELETE was run against the base table) and that the cached queries may need to be occasionally purged.

Buffer Manager/Cache and Buffers

The caching and buffers subsystem is responsible for ensuring that the most frequently used data (or structures, as you will see) are available in the most efficient manner possible. In other words, the data must be resident or ready to read at all times. The caches dramatically increase the response time for requests for that data because the data is in memory and thus no additional disk access is necessary to retrieve it. The cache subsystem was created to encapsulate all of the caching and buffering into a loosely coupled set of library functions. Although you will find the caches implemented in several different source code files, they are considered part of the same subsystem.

A number of caches are implemented in this subsystem. Most of the cache mechanisms use the same or similar concept of storing data as structures in a linked list. The caches are implemented in different portions of the code to tailor the implementation to the type of data that is being cached. Let's look at each of the caches.

Table Cache - The table cache was created to minimize the overhead in opening, reading, and closing tables (the .FRM files on disk). For this reason, the table cache is designed to store metadata about the tables in memory. This makes it much faster for a thread to read the schema of the table without having to reopen the file every time. Each thread has its own list of table cache structures. This permits the threads to maintain their own views of the tables so that if one thread is altering the schema of a table (but has not committed the changes) another thread may use that table with the original schema. The structure used is a simple one that includes all of the metadata information for a table. The structures are stored in a linked list in memory and associated with each thread.

Record Cache - The record cache was created to enhance sequential reads from the storage engines. Thus the record cache is usually only used during table scans. It works like a read-ahead buffer by retrieving a block of data at a time, thus resulting in fewer disk accesses during the scan. Fewer disk accesses generally equates to improved performance. Interestingly, the record cache is also used in writing data sequentially by writing the new (or altered) data to the cache first and then writing the cache to disk when full. In this way write performance is improved as well. This sequential behavior (called locality of reference) is the main reason the record cache is most often used with the MyISAM storage engine, although it is not limited to MyISAM. The record cache is

implemented in an agnostic manner that doesn't interfere with the code used to access the storage engine API. Developers don't have to do anything to take advantage of the record cache as it is implemented within the layers of the API.

Key Cache - The key cache is a buffer for frequently used index data. In this case, it is a block of data for the index file (B-tree) and is used exclusively for MyISAM tables (the .MYI files on disk). The indexes themselves are stored as linked lists within the key cache structure. A key cache is created when a MyISAM table is opened for the first time. The key cache is accessed on every index read. If an index is found in the cache, it is read from there; otherwise, a new index block must be read from disk and placed into the cache.

However, the cache has a limited size and is tunable by changing the key_cache_block_size configuration variable. Thus not all blocks of the index file will fit into memory. So how does the system keep track of which blocks have been used? The cache implements a monitoring system to keep track of how frequent the index blocks are used. The key cache has been implemented to keep track of how "warm" the index blocks are.

Warm in this case refers to how many times the index block has been accessed over time. Values for warm include BLOCK_COLD, BLOCK_WARM, and BLOCK_HOT. As the blocks cool off and new blocks become warm, the cold blocks are purged and the warm blocks added. This strategy is a least recently used (LRU) page-replacement strategy—the same algorithm used for virtual memory management and disk buffering in operating systems—that has been proven to be remarkably efficient even in the face of much more sophisticated page-replacement algorithms. In a similar way, the key cache keeps track of the index blocks that have changed (called getting "dirty").

When a dirty block is purged, its data is written back to the index file on disk before being replaced. Conversely, when a clean block is purged it is simply removed from memory.

Privilege Cache - The privilege cache is used to store grant data on a user account. This data is stored in the same manner as an access control list (ACL), which lists all of the privileges a user has for an object in the system. The privilege cache is implemented as a structure stored in a first in, last out (FILO) hash table. Data for the cache is gathered when the grant tables are read during user authentication and initialization. It is important to store this data in memory as it saves a lot of time reading the grant tables.

Hostname Cache - The hostname cache is another of the helper caches, like the privilege cache. It too is implemented as a stack of a structure. It contains the hostnames of all the connections to the server. It may seem surprising, but this data is frequently requested and therefore in high demand and a candidate for a dedicated cache.

Miscellaneous - A number of other small cache mechanisms are implemented throughout the MySQL source code. One example is the join buffer cache used during complex join operations. For example, some join operations require comparing one tuple to all the tuples in the second table. A cache in this case can store the tuples read so that the join can be implemented without having to reread the second table into memory multiple times.

The Storage Manager

The storage manager interfaces with the operating system to write data to the disk efficiently. Because the storage functions reside in a separate subsystem, the MYSQL engine operates at a level of abstraction away from the operating system. The storage manager writes to disk all of the data in the user tables. indexes, and logs as well as the internal system data.

The Transaction Manager

The function of the Transaction manager is to facilitate concurrency in data access. This subsystem provides a locking facility to ensure that multiple simultaneous users access the data in consistent way. Without corrupting or damaging the data. Transaction control takes place via the lock manager subcomponent, which places and release locks on various objects being used in transaction.

Recovery Manager

The Recovery Manager's job is to keep copies of data for retrieval later, in case of loss of data. It also logs commands that modify the data and other significant events inside the database
So far, only InnoDB and BDB table handlers.

Subsystem Interaction and Control Flow - The Query engine requests the data to be read or written to buffer manager to satisfy a users query. It depend on the transaction manager for locking of the data so that the concurrency is ensured. To perform table creation and drop operations, the query engine accesses the storage manager directly, bypassing the buffer manager, to create or delete the files in the file system.

The buffer manager caches data from the storage manager for efficient retrieval by the query engine. It depends on the transaction manager to check the locking status of data before it performs any modification action. The Transaction manager depends on the Query cache and storage manager to place locks on data in memory and in the file system.

The recovery manager uses the Storage manager to store command/event logs and backups of the data in the file system. It depends on the transaction manager to obtain locks on the log files being written. The recovery manager also needs to use the buffer manager during recovery from crashes. The storage manager depends on the operating system file system for persistent storage and retrieval of data. It depends on the transaction manager to obtain locking status information.

1.6. Storage Engine Tiers

MySQL provides support for thirteen different storage engines which act as varying table type handlers. Most people who use MySQL on a regular basis already know about the two most common storage engines, MyISAM and InnoDB. Most of the time, the default storage engine as defined by the store_engine option in the MySQL config file is typically MyISAM, and this is usually what most people go with. In fact, many people do not even take the time to select a storage-engine, and just use the default. You can also assign a storage engine to a specific table with the following syntax: CREATE TABLE tablename (column1, column2, [etc...]) ENGINE = [storage_engine].

Storage Engine

For those of you who do not know, a storage engine is what stores, handles, and retrieves information from a table. There is no "perfect" or "recommended" storage engine to use, but for most applications the default MyISAM is fine. In MySQL there are 10 different storage engines, though all of them may not be available to you. To get a list of supported storage engines on your MySQL Server, you can do

```
mysql -uroot -p
Password:
mysql> show engines;
```

This should provide a list of supported storage engines for your server. The standard engines supported by MySQL are:

- ✓ MyISAM
- ✓ InnoDB
- ✓ MERGE
- ✓ MEMORY (HEAP)
- ✓ BDB (BerkeleyDB)
- ✓ EXAMPLE
- ✓ ARCHIVE
- ✓ CSV
- ✓ BLACKHOLE
- ✓ ISAM
- ✓ FEDERATED

The benefits to selecting a storage engine comes down to the added benefits of speed and functionality. For example, if you store a lot of log data you might want to use the ARCHIVE storage engine which only supports INSERT and SELECT. All storage engines can be selected per server, per database and per table, giving you fine-grained control over your schema. MySQL storage engines is a major reason why MySQL has gained such popularity over the years, as opposed to other single-storage-engine supporting databases.

- ✓ MyISAM - As stated earlier, MyISAM is the typical default storage engine for MySQL servers. The MyISAM type is actually a branch, or child, of the ISAM engine. If you are ok with not having TRANSACTION, and with having table-level locking as opposed to row-level locking, MyISAM is typically the best for performance and functionality. The MyISAM type is great for sites that have a very low INSERT/UPDATE rate and a very high SELECT rate. Other options might want to be considered with very high INSERT/UPDATE queries due to the restrictions of table-level locking causing a decrease in performance. The max number of rows supported for MyISAM is ~4.29E+09 with up to 64 indexs per table. Fields of TEXT/BLOB can be fully indexed for cases such as searching.
- ✓ InnoDB - Compared to MyISAM, InnoDB provides many more feature to increase performance. There is some additional time spent during initial setup, as opposed to MyISAM, but the benefits far outweigh the time spent. One major difference is the ability to do row-level locking, as opposed to table-level locking, to increase performance time. This

allows parallel INSERT/UPDATE/DELETE queries to be ran on the same table, unlike MyISAM where each query has to wait its turn to run. Additionally, InnoDB provides foreign key functionality. This allows you to ensure that dependent data in table one is present before inserting data into table two. Likewise, it prevents data in table one from being deleted if there is data in table two that depends on it. InnoDB also provides caching for data and indexes in memory, as well as on disk, which provides a large increase in performance gain. For those low on RAM this might not be the ideal solution, but with memory as cheap as it is this is becoming less of an issue.

- ✓ MERGE - The MERGE storage engine was added in MySQL 3.23.25. It enables users to have a collection of identical MyISAM tables to be handled by a single table. There are constraints to this type, such as all tables needing to have the same definition, but the usefulness here quickly becomes apparently. If you store sales transactions on your site in a daily table, and want to pull a report for the month, this would allow you to execute a query against a single table to pull all sales for the month.
- ✓ MEMORY (HEAP) - The HEAP storage engine, also referred to as MEMORY, provides in-memory tables. MySQL Server will retain the format of the tables in order to quickly create a "trash" table and access the information on the fly for better processing, it is not great for long term usage due to data integrity. Similar to InnoDB, this is not for the light of RAM.
- ✓ BDB (BerkeleyDB) - The BDB handles transaction-safe tables and uses a hash based storage system. This allows for some of the quickest reading of data, especially when paired with unique keys. There are, however, many downfalls to the BDB system, including the speed on un-index rows, and this makes the BDB engine a less than perfect engine choice. Because of this, many people tend to overlook the BDB engine. I feel, however, that it does have a place in database design when the right situation calls for it.

- ✓ EXAMPLE - This storage engine was added in MySQL 4.1.3. It is a "stub" engine that serves no real purpose, except to programmers. EXAMPLE provides the ability to create tables, but no information can be inserted or retrieved.
- ✓ ARCHIVE - The ARCHIVE storage engine was added in MySQL 4.1.3 as well, and is used for storing large amounts of data without indexes in a very small footprint. This engine will only support INSERT and SELECT, and all information is compresses. This makes it the perfect storage engine for logs, point of sale transactions, accounting, etc. While this may seem very useful, keep in mind that when reading data the entire table must be de-compressed and read before data can be returned. Therefore, this is the ideal engine for storage that is only called on a low-rate basis.
- ✓ CSV - CSV, added in MySQL 4.1.4, stores data in text files using comma-separated values. As such this is not an ideal engine for large data storage, tables requiring indexing, etc. The best use-case for this is transferring data to a spreadsheet for later use.
- ✓ BLACKHOLE - Though seemingly pointless at first, the BLACKHOLE engine, which does not allow for any data to be stored or retrieved, is good for testing database structures, indexes, and queries. You can still run INSERTS against a BLACKHOLE table, with the knowledge that all inserts will vanish into the void.
- ✓ ISAM - The original storage engine was ISAM, which managed nontransactional tables. This engine has since been replaced by MyISAM, and while MySQL Engineers recommend that it should no longer be used, it does have its place in this article for historical purposes. If you feel that you need to use ISAM, just remember that MyISAM is backwards compatible and should provide you with everything and more.

- ✓ FEDERATED - The FEDERATED storage engine allows you to manage data from a remote MySQL server without using cluster or replication technology. The local federated table stores no data. When you query data from a local federated table, the data is pull automatically from the remote federated tables.

1.7. The SQL Parser

A parser is a program, usually part of a compiler, that receives input in the form of sequential source program instructions, interactive online commands, markup tags, or some other defined interface and breaks them up into parts (for example, the nouns (objects), verbs (methods), and their attributes or options) that can then be managed by other programming (for example, other components in a compiler).

Parsing or syntactic analysis is the process of analysing a string of symbols, either in natural language or in computer languages, conforming to the rules of a formal grammar. The term parsing comes from Latin pars (orationis), meaning part (of speech)

There is only one parser in the SQL layer in the MySQL server. This parser is capable of understanding every SQL statement, including statements related to Stored Programs. The parser is implemented as an ascendant parser, using bison. The source code is located in the file sql/sql_yacc.yy. The parts of the parser dedicated more specially to Stored Programs are starting at the following rules:

- ✓ CREATE PROCEDURE : see rule sp_tail,
- ✓ CREATE FUNCTION : see rule sp_tail,
- ✓ CREATE TRIGGER : see rule trigger_tail,
- ✓ CREATE EVENT : see rule event_tail.

In every case, the parser reads the SQL text stream that represents the code as input, and creates an internal representation of the Stored Program as output, with one C++ object of type sp_head. A limiting consequence of this approach is that a stored program does not support nesting: it is impossible to embed one CREATE PROCEDURE into another, since the parser currently may only support one sp_head object at a time. Conceptually, there are many different layers involved during parsing

- ✓ Lexical analysis (making words or tokens from a character stream),
- ✓ Syntactic analysis (making "sentences" or an abstract syntax tree from the tokens),
- ✓ Semantic analysis (making sure these sentences do make sense),
- ✓ Code generation (for compilers) or evaluation (for interpreters).

From the implementation point or view, many different concepts from different layers actually collide in the same code base, so that the actual code organization is as follows:

- ✓ The lexical analysis is implemented in sql/sql_lex.cc, as when parsing regular statements.
- ✓ Syntactic analysis, semantic analysis, and code generation -- all of them -- are done at once, during parsing of the code. From that perspective, the parser behaves as a single pass compiler.

In other words, both the code and the symbol table for the final result are generated on the fly, interleaved with syntactic analysis.

This is both very efficient from a performance point of view, but difficult to understand, from a maintenance point of view.

1.8. Installation

The first decision to make is whether you want to use a production (stable) release or a development release. In the MySQL development process, multiple release series co-exist, each at a different stage of maturity.

The naming scheme in MySQL 5.7 uses release names that consist of three numbers and a suffix; for example, mysql-5.7.1-m1. The numbers within the release name are interpreted as follows:

- ✓ The first number (5) is the major version and describes the file format. All MySQL 5 releases have the same file format.
- ✓ The second number (7) is the release level. Taken together, the major version and release level constitute the release series number.
- ✓ The third number (1) is the version number within the release series. This is incremented for each new release. Usually you want the latest version for the series you have chosen.

For each minor update, the last number in the version string is incremented. When there are major new features or minor incompatibilities with previous versions, the second number in the version string is incremented. When the file format changes, the first number is increased.

Linux/UNIX Installation

The recommended way to install MySQL on a Linux system is via RPM. MySQL AB makes the following RPMs available for download on its web site:

- ✓ MySQL - The MySQL database server, which manages databases and tables, controls user access, and processes SQL queries.
- ✓ MySQL-client - MySQL client programs, which make it possible to connect to and interact with the server.
- ✓ MySQL-devel - Libraries and header files that come in handy when compiling other programs that use MySQL.
- ✓ MySQL-shared - Shared libraries for the MySQL client.
- ✓ MySQL-bench - Benchmark and performance testing tools for the MySQL database server.

The MySQL RPMs listed here are all built on a SuSE Linux system, but they'll usually work on other Linux variants with no difficulty. The steps to proceed for installation are

- ✓ Login to the system using root user.
- ✓ Switch to the directory containing the RPMs:
- ✓ Install the MySQL database server by executing the following command. Remember to replace the filename in italics with the file name of your RPM. [root@host]# rpm -i MySQL-5.0.9-0.i386.rpm Above command takes care of installing MySQL server, creating a user of MySQL,

creating necessary configuration and starting MySQL server automatically. You can find all the MySQL related binaries in /usr/bin and /usr/sbin. All the tables and databases will be created in /var/lib/mysql directory.

- ✓ This is optional but recommended step to install the remaining RPMs in the same manner
  - ✓ [root@host]# rpm -i MySQL-client-5.0.9-0.i386.rpm
  - ✓ [root@host]# rpm -i MySQL-devel-5.0.9-0.i386.rpm
  - ✓ [root@host]# rpm -i MySQL-shared-5.0.9-0.i386.rpm
  - ✓ [root@host]# rpm -i MySQL-bench-5.0.9-0.i386.rpm

Windows Installation

Default installation on any version of Windows is now much easier than it used to be, as MySQL now comes neatly packaged with an installer. Simply download the installer package, unzip it anywhere, and run setup.exe.

Default installer setup.exe will walk you through the trivial process and by default will install everything under C:\mysql.

Test the server by firing it up from the command prompt the first time. Go to the location of the mysqld server which is probably C:\mysql\bin, and type:

mysqld.exe --console

If you are on NT, then you will have to use mysqld-nt.exe instead of mysqld.exe. If all went well, you will see some messages about startup and InnoDB. If not, you may have a permissions issue. Make sure that the directory that holds your data is accessible to whatever user (probably mysql) the database processes run under.

MySQL will not add itself to the start menu, and there is no particularly nice GUI way to stop the server either. Therefore, if you tend to start the server by double clicking the mysqld executable, you should remember to halt the process by hand by using mysqladmin, Task List, Task Manager, or other Windows-specific means.

Verify Installation

You can test whether the MySQL server is working by executing any of the following commands

- ✓ shell> "C:\Program Files\MySQL\MySQL Server 5.0\bin\mysqlshow"
- ✓ shell> "C:\Program Files\MySQL\MySQL Server 5.0\bin\mysqlshow" -u root mysql
- ✓ shell> "C:\Program Files\MySQL\MySQL Server 5.0\bin\mysqladmin" version status proc
- ✓ shell> "C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql" test

If mysqld is slow to respond to TCP/IP connections from client programs, there is probably a problem with your DNS. In this case, start mysqld with the --skip-name-resolve option and use only localhost and IP addresses in the Host column of the MySQL grant tables.