**http://blog.csdn.net/hguisu/article/details/7106342**

**1.MySQL整体逻辑架构**

我们先下图看看MySQL整体逻辑架构(MySQL’s Logical Architecture)

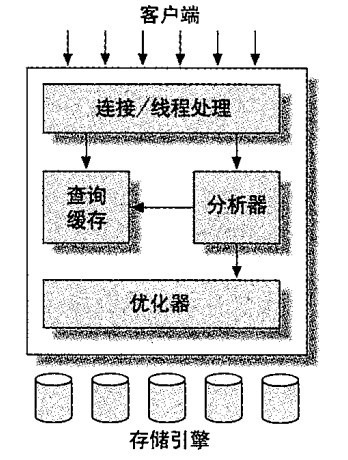


                      图1

**第一层，即最上一层**，所包含的服务并不是MySQL所独有的技术。它们都是服务于C/S程序或者是这些程序所需要的 ：连接处理，身份验证，安全性等等。

**第二层值得关注**。这是MySQL的核心部分。通常叫做 SQL Layer。在 MySQL据库系统处理底层数据之前的所有工作都是在这一层完成的，包括权限判断， sql解析，行计划优化， query cache 的处理以及所有内置的函数(如日期,时间,数学运算,加密)等等。各个存储引擎提供的功能都集中在这一层，如存储过程，触发器，视 图等。

**第三层包括了存储引擎**。通常叫做StorEngine Layer ，也就是底层数据存取操作实现部分，由多种存储引擎共同组成。它们负责存储和获取所有存储在MySQL中的数据。就像Linux众多的文件系统 一样。每个存储引擎都有自己的优点和缺陷。服务器是通过存储引擎API来与它们交互的。这个接口隐藏 了各个存储引擎不同的地方。对于查询层尽可能的透明。这个API包含了很多底层的操作。如开始一个事 物，或者取出有特定主键的行。存储引擎不能解析SQL，互相之间也不能通信。仅仅是简单的响应服务器 的请求。

**连接管理和安全**

在服务器内部，每个client连接都有自己的线程。这个连接的查询都在一个单独的线程中执行。这些线程轮流运行在某一个CPU内核(多核CPU)或者CPU中。服务器缓存了线程，因此不需要为每个client连接单独创建和销毁线程 。

当clients（也就是应用程序）连接到了MySQL服务器。服务器需要对它进行认证(Authenticate)。认证是基于用户名，主机，以及密码。对于使用了SSL（安全套接字层）的连接，还使用了X.509证书。clients一连接上，服务器就验证它的权限 （如是否允许客户端可以查询world数据库下的Country表的数据）。

**优化和执行**

MySQL会解析查询，并创建了一个内部数据结构（解析树）。然后对其进行各种优化。这些优化包括了，查询语句的重写，读表的顺序，索引的选择等等。用户 可以通过查询语句的关键词传递给优化器以便提示使用哪种优化方式，这样即影响了优化器的优化方式。另外，用户也可以请求服务器给出优化过程的各种说明，以获知服务器的优化策略，为用户提供了参数基准，以便用户可以重写查询，架构和修改相关服务器配置，便于mysql更高效的运行。

优化器并是不关心表使用了哪种存储引擎，但是存储引擎对服务器优化查询的方式是有影响的。优化器需要知道存储引擎的一些特性：具体操作的性能和开销方面的信息，以及表内数据的统计信息。例如，存储引擎支持哪些索引类型，这对于查询是非常有用的。

在解析查询之前，要查询缓存，这个缓存只能保存查询信息以及结果数据。如果请求一个查询在缓存中存在，就不需要解析，优化和执行查询了。直接返回缓存中所存放的这个查询的结果。

**2.MySQL逻辑模块组成**

虽然从上图1看起来 MySQL 架构非常的简单，就是简单的两部分而已，但实际上每一层 中都含有各自的很多小模块，尤其是第二层 SQL Layer ，结构相当复杂的。下面我们就分别 针对 SQL Layer 和 Storage Engine Layer 做一个简单的分析。我们看下图体系结构：

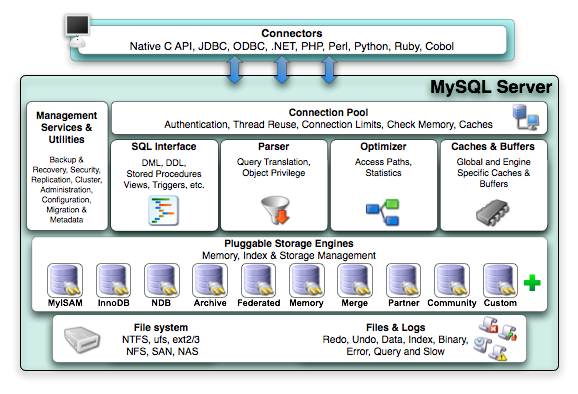


                                                            图2

**1.Connectors**

指的是不同语言中与SQL的交互

**2 Management Serveices & Utilities：**

系统管理和控制工具

**3 Connection Pool: 连接池**

管理缓冲用户连接，线程处理等需要缓存的需求。

负责监听对 MySQL Server 的各种请求，接收连接请求，转发所有连接请求到线程管理模块。每一个连接上 MySQL Server 的客户端请求都会被分配（或创建）一个连接线程为其单独服务。而连接线程的主要工作就是负责 MySQL Server 与客户端的通信，  
接受客户端的命令请求，传递 Server 端的结果信息等。线程管理模块则负责管理维护这些连接线程。包括线程的创建，线程的 cache 等。

**4 SQL Interface: SQL接口。**

接受用户的SQL命令，并且返回用户需要查询的结果。比如select from就是调用SQL Interface

**5 Parser: 解析器。**

SQL命令传递到解析器的时候会被解析器验证和解析。解析器是由Lex和YACC实现的，是一个很长的脚本。

在 MySQL中我们习惯将所有 Client 端发送给 Server 端的命令都称为 query ，在 MySQL Server 里面，连接线程接收到客户端的一个 Query 后，会直接将该 query 传递给专门负责将各种 Query 进行分类的解析器，然后转发给各个对应的处理模块。  
主要功能：  
a . 将SQL语句进行语义和语法的分析，分解成数据结构，然后按照不同的操作类型进行分类，然后做出针对性的转发到后续步骤，以后SQL语句的传递和处理就是基于这个结构的。  
b.  如果在分解构成中遇到错误，那么就说明这个sql语句是不合理的

**6 Optimizer: 查询优化器。**

SQL语句在查询之前会使用查询优化器对查询进行优化。就是优化客户端请求的 query（sql语句） ，根据客户端请求的 query 语句，和数据库中的一些统计信息，在一系列算法的基础上进行分析，得出一个最优的策略，告诉后面的程序如何取得这个 query 语句的结果

他使用的是“选取-投影-联接”策略进行查询。  
       用一个例子就可以理解： select uid,name from user where gender = 1;  
       这个select 查询先根据where 语句进行选取，而不是先将表全部查询出来以后再进行gender过滤  
       这个select查询先根据uid和name进行属性投影，而不是将属性全部取出以后再进行过滤  
       将这两个查询条件联接起来生成最终查询结果

**7 Cache和Buffer： 查询缓存。**

他的主要功能是将客户端提交 给MySQL 的 Select 类 query 请求的返回结果集 cache 到内存中，与该 query 的一个 hash 值 做  
一个对应。该 Query 所取数据的基表发生任何数据的变化之后， MySQL 会自动使该 query 的Cache 失效。在读写比例非常高的应用系统中， Query Cache 对性能的提高是非常显著的。当然它对内存的消耗也是非常大的。

如果查询缓存有命中的查询结果，查询语句就可以直接去查询缓存中取数据。这个缓存机制是由一系列小缓存组成的。比如表缓存，记录缓存，key缓存，权限缓存等

**8 、存储引擎接口**

存储引擎接口模块可以说是 MySQL 数据库中最有特色的一点了。目前各种数据库产品中，基本上只有 MySQL 可以实现其底层数据存储引擎的插件式管理。这个模块实际上只是 一个抽象类，但正是因为它成功地将各种数据处理高度抽象化，才成就了今天 MySQL 可插拔存储引擎的特色。  
     从图2还可以看出，MySQL区别于其他数据库的最重要的特点就是其插件式的表存储引擎。MySQL插件式的存储引擎架构提供了一系列标准的管理和服务 支持，这些标准与存储引擎本身无关，可能是每个数据库系统本身都必需的，如SQL分析器和优化器等，而存储引擎是底层物理结构的实现，每个存储引擎开发者 都可以按照自己的意愿来进行开发。  
    注意：存储引擎是基于表的，而不是数据库。