28 带你认识MySQL的系统架构

更新时间：2020-05-13 09:44:47

> 什么是路？就是从没路的地方践踏出来的，从只有荆棘的地方开辟出来的。 —— 鲁迅

我们几乎每天都会接触 MySQL，大概也能说出几个构成 MySQL 的组件，但是，你知道各个组件之间是如何协同工作的吗（这也称之为 MySQL 的逻辑架构）？数据库就是用于实现对数据的读写，那么，你知道 MySQL 是怎么组织存储文件的吗（这也称之为 MySQL 的物理架构）？最后，从一条 SQL 语句的输入、执行到返回结果，你知道这里面都经历了些什么吗？想要搞清楚这些问题，就来看一看这一节的内容吧。

1. MySQL 的物理架构

数据库的核心目的即实现对数据的读写，而数据最终一定会以文件的形式保存在磁盘上，所以，MySQL 的物理架构指的就是文件的存储结构。另外，MySQL 又会以不同的文件存储不同类型的数据，而这些不同类型的数据又会以不同的方式存储在磁盘上。当然，你不需要记住所有的文件及类型，重要的是理解它的设计思想。

1.1 MySQL Base 目录

MySQL 中的文件主要存储于两个目录中：Base 目录和数据目录。Base 目录主要包含库文件和可执行文件，即系统数据文件。关于这个目录，我们可以看看官网对它的描述：

> Path to installation directory. All paths are usually resolved relative to this.

它的意思是：Base 目录指定了 MySQL 的安装路径（也就是 MySQL 的安装目录），而这个全路径可以解决相对路径所造成的问题。MySQL 使用 basedir 参数来记录 Base 目录，我们可以查看这个变量，从而确定目录的绝对路径。如下所示：

```
mysql> SHOW VARIABLES LIKE 'basedir';
+---------------+-----------------------------------------+
| Variable_name | Value                                   |
+---------------+-----------------------------------------+
| basedir       | /usr/local/mysql-5.7.28-macos10.14-x86_64/ |
+---------------+-----------------------------------------+
1 row in set (0.00 sec)
```

Base 目录中最核心的就是 MySQL 提供的可执行文件（位于 bin 目录下面），例如：mysql 可以用于登录服务器、mysqladmin 可以用来检查服务器的配置和当前状态、mysqldump 可以导出数据表文件等等。

1.2 MySQL 数据目录

与 Base 目录类似，MySQL 也使用一个参数来标记数据目录，我们也可以查询它。如下所示：

```
mysql> SHOW VARIABLES LIKE 'datadir';
+---------------+-----------------------+
| Variable_name | Value                 |
+---------------+-----------------------+
| datadir       | /usr/local/mysql/data/ |
+---------------+-----------------------+
1 row in set (0.00 sec)
```

在搞清楚数据目录存储了哪些文件之前，我们先去看一看 `/usr/local/` 目录（Base 目录和数据目录的上一级目录），以此把 Base 目录和数据目录的关系搞清楚。如下所示：

```
~ ll /usr/local
total 0
lrwxr-xr-x  1 root   30 10 24 11:02 mysql -> mysql-5.7.28-macos10.14-x86_64
drwxr-xr-x 13 root  416 10 24 11:02 mysql-5.7.28-macos10.14-x86_64
```

所以，`/usr/local/mysql/` 只是一个“软链接”，它最终指向了 MySQL 的 Base 目录。而数据目录则位于 Base 目录中的 `/data/` 目录中，这也是 MySQL 的默认存储结构。

数据目录主要存储两类文件：日志文件和数据文件。同时，这两类文件又可以细分为很多种类。对于日志文件，主要有：

- 错误日志文件：以 err 结尾，记录 MySQL 服务器启动、关闭和运行过程中产生的重大错误信息
- 二进制日志文件：默认以 mysql-bin 开头，数字结尾。记录所有数据库、表的更新事件
- 慢查询日志文件：以 slow.log 结尾，记录执行时间超过 long_query_time 变量指定时间的 SQL 语句

我们在 MySQL 中创建的每一个数据库都会在数据目录下面创建同名的目录，这里面存储了数据表对应的数据（文件）和索引（文件）。例如：我当前的数据库中存在 imooc_mysql 和 sbtest 两个库，相对应的，它们就会有同名目录存在。

```
~ ls -l /usr/local/mysql/data/
drwxr-x---  39 _mysql _mysql    1248 12 27 11:11 imooc_mysql
drwxr-x---   3 _mysql _mysql      96 12  4 13:44 sbtest
```

我们进入到任何一个存在表的数据库目录中，可以发现，每一张表都对应着两个文件（也是两种类型），即 frm 文件和 ibd 文件（以后缀结尾）。这其中：

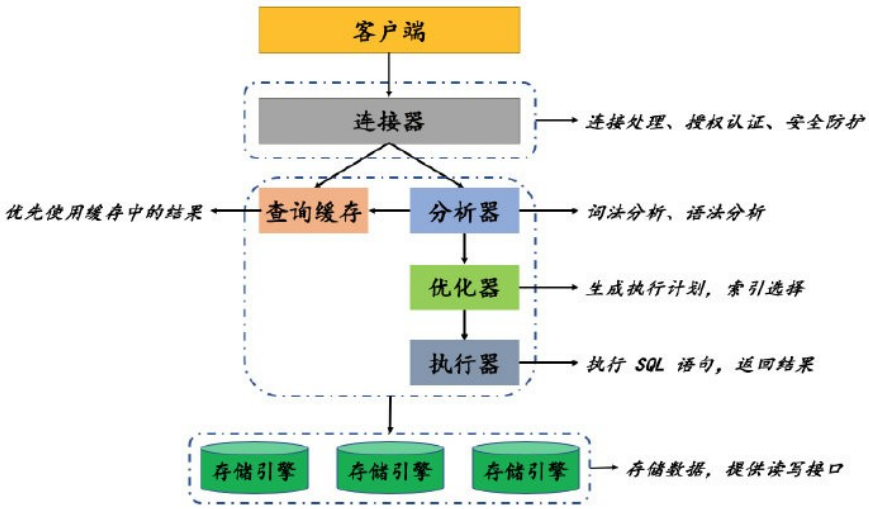
- frm 文件：存储数据表的结构信息，它与操作系统和存储引擎无关，即对于任何环境下安装的 MySQL，都会存在这个文件
- ibd 文件：存储数据表中的数据和索引信息，它是与存储引擎相关的

2. MySQL 逻辑架构

MySQL 的物理架构主要是“各种”文件的存储结构，而逻辑架构则是 MySQL 中的各个组件，以及组件之间的协作关系。理解物理架构能够搞清楚 MySQL 怎样组织系统所需的文件，而理解逻辑架构则是为了搞清楚 MySQL 的工作方式（过程）。

2.1 逻辑架构分层

MySQL 是典型的 C/S 系统，它的逻辑架构中定义了很多组件，每个组件负责数据库复杂功能的一部分。通常，我们会把这个整体的逻辑架构分为三层：服务层、核心层和存储层。分层（自顶向下每个虚线框都是一层）的逻辑架构如下图所示：



每一层实现的功能总结如下：

- **服务层**：所谓服务，即服务于客户端，主要是接受客户端的连接请求，并对请求进行授权认证
- **核心层**：这一层提供了 MySQL 的核心功能服务，包括查询解析、分析、优化、缓存。另外，对于跨存储引擎的功能也在这一层实现，例如：存储过程、视图等等
- **存储层**：存储引擎，负责数据存取，且对于 InnoDB 来说还需要做事务处理

2.2 服务层

服务层是 MySQL 逻辑架构的最上层（即连接器），它直接“暴露”给客户端，主要实现了三点功能：

- 客户端连接到来时，连接器会从线程池中分配一个线程（如果没有达到最大连接数限制）进行连接，此后，客户端的所有操作都会在这个线程上进行
- 连接器会对发起连接请求的用户进行鉴权处理，包括：用户名、密码、主机或 ip 地址等等
- 连接器会根据用户的权限来判断用户是否有权限执行某项操作

2.3 核心层

核心层提供了 MySQL 服务器的所有逻辑功能，当然包含的组件也较多。除了图中所绘的之外，这一层还承担了 MySQL 系统管理的角色（同样也是由组件提供）。系统管理组件实现的功能主要包括以下四点：

- 数据库的备份与恢复
- 数据库集群管理
- 数据库分区、分库、分表的管理
- 数据库元数据的管理

接下来，按照各个组件的工作顺序，我们依次来看一看核心层中的各个组件都实现了怎样的功能：

- 分析器
    - 首先它会对查询语句进行语法分析，如果存在错误，立即返回错误信息
    - 语法检查通过后，分析器会询问 “查询缓存”，如果命中缓存，直接返回结果
    - 否则，生成语法树，并为每个查询语句生成 SQL_ID
- 查询缓存
    - 它在所有的客户端会话之间共享，基于 SQL_ID 执行查询，且只有完全相同的 SQL 语句才可能会命中（缓存的换入换出导致历史缓存失效）
    - 如果命中了缓存，则查询语句直接从缓存中取数据，不再 “经历” 下面的组件
- 优化器
    - 主要作用是对查询语句进行优化，包括调整字段顺序、决定表的读取顺序以及挑选合适的索引等等
    - 虽然优化器是与存储引擎分离的，但是不同的存储引擎会影响优化效果
- 执行器
    - 执行查询语句（这里面会调用存储引擎的接口），返回结果给客户端

总的来说，MySQL 通过分析器知道了查询语句想要做什么、通过优化器知道了应该怎样做、最后通过执行器决定执行，返回结果。其中，如果查询缓存 “已经有了结果”，则不再需要优化器和执行器 “那样做”。

2.4 存储层

虽然表面上说存储引擎是 MySQL 逻辑架构中的一层，但其实它才是真正与文件打交道的子系统。那么，关于存储层，我们这里简单的做三点说明：

- MySQL 区别于其他数据库最大的地方就是插件式的表存储引擎，存储引擎供应商根据 MySQL 提供的文件访问抽象接口来定制文件访问的规则
- MySQL 服务器则通过其定义的 API 与存储引擎进行通信，且这种通信过程是 “透明的”
- 存储引擎的实现过程也就是适配 MySQL 读写接口的过程

3. SQL 语句在 MySQL 中的执行过程

到目前为止，我们已经理清了 MySQL 的物理和逻辑架构，但是，以上的内容也确实太偏重于理论化。所以，为了更好的理解 MySQL 的 “内涵”，接下来，我们去看一看一条 SQL 语句在 MySQL 中是怎样被执行的。

3.1 查询语句的执行过程

我们日常编写的 SQL 语句可以分为两类：查询（SELECT）语句和更新（INSERT、UPDATE、DELETE）语句。我们先来看一看相对比较简单的查询语句，举个例子，执行：

```
SELECT * FROM worker WHERE name = 'pony';
```

MySQL 执行这条查询语句的过程如下：

- 校验当前的会话是否有执行权限，如果没有，直接返回错误信息
- 权限条件满足，询问“查询缓存”（以当前的 SQL 语句为 KEY 去 Map 中查询），如果命中，返回结果
- 没有命中缓存，则查询语句交给分析器，提取 SQL 语句中的关键元素，构造语法树（包含对语法错误的校验）。其中，关键元素包含
    - SELECT 关键字，确定是查询请求
    - worker 可以确定查询的数据表以及数据列（还会对列进行校验）
    - `name = 'pony'` 设定为查询条件
- 优化器确定执行方案，对于当前的查询语句主要是选择合适的索引
- 执行器调用存储引擎接口，执行查询，并返回引擎的执行结果

可见，执行过程中发挥最大作用、也是最复杂的组件是分析器和优化器。特别地，优化器会制定出很多执行方案，并根据自己的优化算法选择执行效率最高的一个方案（需要特别注意，这是优化器自身认定的最优）。

3.2 更新语句的执行过程

更新语句的执行与查询语句是非常类似的，假设我们执行如下的插入语句：

```
INSERT INTO worker(`type`, `name`, `salary`, `province`, `city`) VALUES('A', 'qinyi', 1000, '安徽省', '宿州市');
```

首先，执行这条语句时，同样会查询缓存，这是为了省去语法和词法分析的过程。如果没有命中缓存，那就继续走分析器和优化器的“原有”流程。最后，到达执行器：

- 调用存储引擎的 API 接口，写入数据
- InnoDB 会把数据保存在内存中，并记录 redo log，此时，redo log 进入 prepare 状态
- 执行器记录 Binlog，随后通知存储引擎，redo log 进入 commit 状态

可见，更新过程与查询过程在执行时最大的不同点是日志（虽然查询时也会有查询日志和慢查询日志的产生）和事务。其中，InnoDB 使用 redo log 两阶段提交的方式解决了数据一致性的问题。

4. 总结

想要完全搞清楚 MySQL 的系统架构，这无疑是一件非常难的事，如无必要，我也并不建议你去这样做。但是，你至少应该知道 MySQL 是如何组织它的内部文件，以及各个组件的基本功能。当然，最好在别人问你 SQL 语句是如何执行的，你能解释得出来。

5. 问题

浏览 MySQL 的安装（Base）目录，搞清楚它的物理架构？

自己画一画 MySQL 的逻辑分层，并在这个过程中思考各个组件所提供的功能？

以你在工作中使用到的 SQL 语句为例，解释说明它在 MySQL 中的执行过程？

6. 参考资料

《高性能 MySQL（第三版）》

MySQL 官方文档：Installing MySQL on macOS

MySQL 官方文档：General Information

MySQL 官方文档：Security

MySQL 官方文档：MySQL Server Administration

MySQL 官方文档：InnoDB Architecture

}

27 服务器性能调优，你知道怎么做吗？

29 你知道 SQL 分析器的实现原理是什么吗？