# 01讲基础架构：一条SQL查询语句是如何执行的

你好，我是林晓斌。

这是专栏的第一篇文章，我想来跟你聊聊MySQL的基础架构。我们经常说，看一个事儿千万不要直接陷入细节里，你应该先鸟瞰其全貌，这样能够帮助你从高维度理解问题。同样，对于MySQL的学习也是这样。平时我们使用数据库，看到的通常都是一个整体。比如，你有个最简单的表，表里只有一个ID字段，在执行下面这个查询语句时：

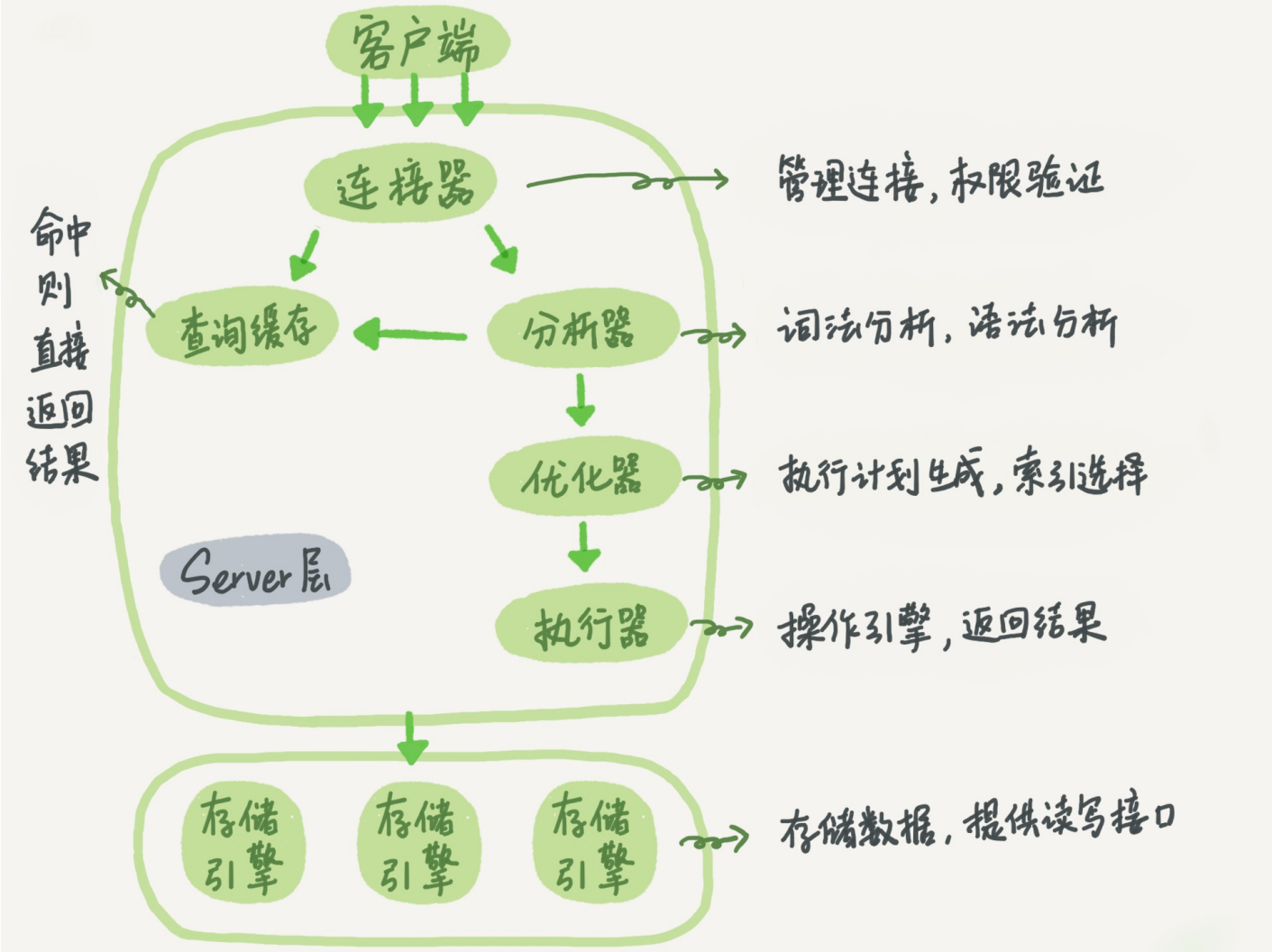
```
mysql> select * from T where ID=10;
```

我们看到的只是输入一条语句，返回一个结果，却不知道这条语句在MySQL内部的执行过程。

所以今天我想和你一起把MySQL拆解一下，看看里面都有哪些“零件”，希望借由这个拆解过程，让你对MySQL有更深入的理解。这样当我们碰到MySQL的一些异常或者问题时，就能够直戳本质，更为快速地定位并解决问题。

下面我给出的是MySQL的基本架构示意图，从中你可以清楚地看到SQL语句在MySQL的各个功能模块中的执行过程。

更多课程请加QQ1042887154，微信lovelu_110获取




MySQL的逻辑架构图

大体来说，MySQL可以分为Server层和存储引擎层两部分。

Server层包括连接器、查询缓存、分析器、优化器、执行器等，涵盖MySQL的大多数核心服务功能，以及所有的内置函数（如日期、时间、数学和加密函数等），所有跨存储引擎的功能都在这一层实现，比如存储过程、触发器、视图等。

而存储引擎层负责数据的存储和提取。其架构模式是插件式的，支持InnoDB、MyISAM、Memory等多个存储引擎。现在最常用的存储引擎是InnoDB，它从MySQL 5.5.5版本开始成为了默认存储引擎。

也就是说，你执行create table建表的时候，如果不指定引擎类型，默认使用的就是InnoDB。不过，你也可以通过指定存储引擎的类型来选择别的引擎，比如在create table语句中使用engine=memory, 来指定使用内存引擎创建表。不同存储引擎的表数据存取方式不同，支持的功能也不同，在后面的文章中，我们会讨论到引擎的选择。

从图中不难看出，不同的存储引擎共用一个**Server层**，也就是从连接器到执行器的部分。你可以先对每个组件的名字有个印象，接下来我会结合开头提到的那条SQL语句，带你走一遍整个执行流程，依次看下每个组件的作用。

### 连接器

第一步，你会先连接到这个数据库上，这时候接待你的就是连接器。连接器负责跟客户端建立连接、获取权限、维持和管理连接。连接命令一般是这么写的：

```
mysql -h$ip -P$port -u$user -p
```

输完命令之后，你就需要在交互对话里面输入密码。虽然密码也可以直接跟在-p后面写在命令行中，但这样可能会导致你的密码泄露。如果你连的是生产服务器，强烈建议你不要这么做。

连接命令中的mysql是客户端工具，用来跟服务端建立连接。在完成经典的TCP握手后，连接器就要开始认证你的身份，这个时候用的就是你输入的用户名和密码。

- 如果用户名或密码不对，你就会收到一个"Access denied for user"的错误，然后客户端程序结束执行。
- 如果用户名密码认证通过，连接器会到权限表里面查出你拥有的权限。之后，这个连接里面的权限判断逻辑，都将依赖于此时读到的权限。

这就意味着，一个用户成功建立连接后，即使你用管理员账号对这个用户的权限做了修改，也不会影响已经存在连接的权限。修改完成后，只有再新建的连接才会使用新的权限设置。

连接完成后，如果你没有后续的动作，这个连接就处于空闲状态，你可以在show processlist命令中看到它。文本中这个图是show processlist的结果，其中的Command列显示为“Sleep”的这一行，就表示现在系统里面有一个空闲连接。

```
mysql> show processlist;
+----+------+-----------------+------+---------+------+----------+------------------+
| Id | User | Host            | db   | Command | Time | State    | Info             |
+----+------+-----------------+------+---------+------+----------+------------------+
|  5 | root | localhost:27710 | test | Sleep   |   16 |          | NULL             |
|  6 | root | localhost:27712 | test | Query   |    0 | starting | show processlist |
+----+------+-----------------+------+---------+------+----------+------------------+
2 rows in set (0.00 sec)
```

客户端如果太长时间没动静，连接器就会自动将它断开。这个时间是由参数wait_timeout控制的，默认值是8小时。

如果在连接被断开之后，客户端再次发送请求的话，就会收到一个错误提醒： Lost connection to MySQL server during query。这时候如果你要继续，就需要重连，然后再执行请求了。

数据库里面，长连接是指连接成功后，如果客户端持续有请求，则一直使用同一个连接。短连接则是指每次执行完很少的几次查询就断开连接，下次查询再重新建立一个。

建立连接的过程通常是比较复杂的，所以我建议你在使用中要尽量减少建立连接的动作，也就是尽量使用长连接。

但是全部使用长连接后，你可能会发现，有些时候MySQL占用内存涨得特别快，这是因为MySQL在执行过程中临时使用的内存是管理在连接对象里面的。这些资源会在连接断开的时候才释放。所以如果长连接累积下来，可能导致内存占用太大，被系统强行杀掉（OOM），从现象看就是MySQL异常重启了。

怎么解决这个问题呢？你可以考虑以下两种方案。

1. 定期断开长连接。使用一段时间，或者程序里面判断执行过一个占用内存的大查询后，断开连接，之后要查询再重连。
2. 如果你用的是MySQL 5.7或更新版本，可以在每次执行一个比较大的操作后，通过执行 mysql_reset_connection来重新初始化连接资源。这个过程不需要重连和重新做权限验证，但是会将连接恢复到刚刚创建完时的状态。

### 查询缓存

连接建立完成后，你就可以执行select语句了。执行逻辑就会来到第二步：查询缓存。

MySQL拿到一个查询请求后，会先到查询缓存看看，之前是不是执行过这条语句。之前执行过的语句及其结果可能会以key-value对的形式，被直接缓存在内存中。key是查询的语句，value是查询的结果。如果你的查询能够直接在这个缓存中找到key，那么这个value就会被直接返回给客户端。

如果语句不在查询缓存中，就会继续后面的执行阶段。执行完成后，执行结果会被存入查询缓存中。你可以看到，如果查询命中缓存，MySQL不需要执行后面的复杂操作，就可以直接返回结果，这个效率会很高。

**但是大多数情况下我会建议你不要使用查询缓存，为什么呢？因为查询缓存往往弊大于利。**

查询缓存的失效非常频繁，只要有对一个表的更新，这个表上所有的查询缓存都会被清空。因此很可能你费劲地把结果存起来，还没使用呢，就被一个更新全清空了。对于更新压力大的数据库来说，查询缓存的命中率会非常低。除非你的业务就是有一张静态表，很长时间才会更新一次。比如，一个系统配置表，那这张表上的查询才适合使用查询缓存。

好在MySQL也提供了这种“按需使用”的方式。你可以将参数query_cache_type设置成DEMAND，这样对于默认的SQL语句都不使用查询缓存。而对于你确定要使用查询缓存的语句，可以用SQL_CACHE显式指定，像下面这个语句一样：

```
mysql> select SQL_CACHE * from T where ID=10;
```

需要注意的是，MySQL 8.0版本直接将查询缓存的整块功能删掉了，也就是说8.0开始彻底没有这个功能了。

### 分析器

如果没有命中查询缓存，就要开始真正执行语句了。首先，MySQL需要知道你要做什么，因此需要对SQL语句做解析。

分析器先会做“词法分析”。你输入的是由多个字符串和空格组成的一条SQL语句，MySQL需要识别出里面的字符串分别是什么，代表什么。

MySQL从你输入的"select"这个关键字识别出来，这是一个查询语句。它也要把字符串“T”识别成“表名T”，把字符串“ID”识别成“列ID”。

做完了这些识别以后，就要做“语法分析”。根据词法分析的结果，语法分析器会根据语法规则，判断你输入的这个SQL语句是否满足MySQL语法。

如果你的语句不对，就会收到“You have an error in your SQL syntax”的错误提醒，比如下面这个语句select少打了开头的字母“s”。

```
mysql> elect * from t where ID=1;

ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL serve
```

一般语法错误会提示第一个出现错误的位置，所以你要关注的是紧接“use near”的内容。

### 优化器

经过了分析器，MySQL就知道你要做什么了。在开始执行之前，还要先经过优化器的处理。

优化器是在表里面有多个索引的时候，决定使用哪个索引；或者在一个语句有多表关联（join）的时候，决定各个表的连接顺序。比如你执行下面这样的语句，这个语句是执行两个表的join：

```
mysql> select * from t1 join t2 using(ID)  where t1.c=10 and t2.d=20;
```

- 既可以先从表t1里面取出c=10的记录的ID值，再根据ID值关联到表t2，再判断t2里面d的值是否等于20。
- 也可以先从表t2里面取出d=20的记录的ID值，再根据ID值关联到t1，再判断t1里面c的值是否等于10。

这两种执行方法的逻辑结果是一样的，但是执行的效率会有不同，而优化器的作用就是决定选择使用哪一个方案。

优化器阶段完成后，这个语句的执行方案就确定下来了，然后进入执行器阶段。如果你还有一些疑问，比如优化器是怎么选择索引的，有没有可能选择错等等，没关系，我会在后面的文章中单独展开说明优化器的内容。

### 执行器

MySQL通过分析器知道了你要做什么，通过优化器知道了该怎么做，于是就进入了执行器阶段，开始执行语句。

开始执行的时候，要先判断一下你对这个表T有没有执行查询的权限，如果没有，就会返回没有权限的错误，如下所示(在工程实现上，如果命中查询缓存，会在查询缓存放回结果的时候，做权限验证。查询也会在优化器之前调用precheck验证权限)。

```
mysql> select * from T where ID=10;


ERROR 1142 (42000): SELECT command denied to user 'b'@'localhost' for table 'T'
```

如果有权限，就打开表继续执行。打开表的时候，执行器就会根据表的引擎定义，去使用这个引擎提供的接口。

比如我们这个例子中的表T中，ID字段没有索引，那么执行器的执行流程是这样的：

1. 调用InnoDB引擎接口取这个表的第一行，判断ID值是不是10，如果不是则跳过，如果是则将这行存在结果集中；
2. 调用引擎接口取“下一行”，重复相同的判断逻辑，直到取到这个表的最后一行。
3. 执行器将上述遍历过程中所有满足条件的行组成的记录集作为结果集返回给客户端。

至此，这个语句就执行完成了。

对于有索引的表，执行的逻辑也差不多。第一次调用的是“取满足条件的第一行”这个接口，之后循环取“满足条件的下一行”这个接口，这些接口都是引擎中已经定义好的。

你会在数据库的慢查询日志中看到一个rows_examined的字段，表示这个语句执行过程中扫描了多少行。这个值就是在执行器每次调用引擎获取数据行的时候累加的。

在有些场景下，执行器调用一次，在引擎内部则扫描了多行，因此**引擎扫描行数跟rows_examined并不是完全相同的。**我们后面会专门有一篇文章来讲存储引擎的内部机制，里面会有详细的说明。

### 小结

今天我给你介绍了MySQL的逻辑架构，希望你对一个SQL语句完整执行流程的各个阶段有了一个初步的印象。由于篇幅的限制，我只是用一个查询的例子将各个环节过了一遍。如果你还对每个环节的展开细节存有疑问，也不用担心，后续在实战章节中我还会再提到它们。

我给你留一个问题吧，如果表T中没有字段k，而你执行了这个语句 select * from T where k=1, 那肯定是会报“不存在这个列”的错误： “Unknown column ‘k’ in ‘where clause’”。你觉得这个错误是在我们上面提到的哪个阶段报出来的呢?

感谢你的收听，欢迎你给我留言，也欢迎分享给更多的朋友一起阅读。


精选留言

林晓斌

感谢大家的积极评论。

答案是分析器。

@threefat 提到Oracle，MySQL确实在设计上受Oracle影响颇深。

@圈圈圆圆 高性能MySQL里面概念学得好。

@Yezhiwei 等几位，说正文里面已经暗示答案的同学，你们这么机智

后面每期的问题，我们会在下一期的结尾的给出解释，不过只是“参考”哦，以评论区的质量，我估计到时候只需要引用优秀评论就好了 ^_^

2018-11-16 16:41

一步

我认为是优化器的，优化器会进行优化分析，比如用先执行哪个条件，使用哪个索引。如果没有对应的字段就会报错的，我看其他评论说是执行器，原因是这个时候才打开表获取数据，但是表的字段不是数据啊，是事先定义好的，所以可以直接读取的，不需要打开表。这是我的看法，希望老师点评一下是否正确

2018-11-14 08:13

作者回复

“不是执行器”这一点，你分析得很好

2018-11-14 09:55

threefat

课后答案:分析器。Oracle会在分析阶段判断语句是否正确，表是否存在，列是否存在等。猜测MySQL也这样。

2018-11-14 22:08

作者回复

MySQL确实在设计上受Oracle影响颇深。就是这样哈，分析器做了这个工作。另外有评论里面举例高性能MySQL 里面概念的同学、说正文里面已经暗示答案的同学，远程点赞了哈　　就喜欢你们这么机智的回答

2018-11-14 23:08

李志博

应该是分析器吧，老师我还有个问题，如果有memory 引擎，还有redis 存在的必要吗，应该是有的吧，两者的场景?

2018-11-14 09:41

作者回复

这个后面专门有一篇《要不要用memory引擎》会说明，先说结论，redis很有必要

2018-11-14 10:45

圈圈圆圆

《高性能mysql》里提到解析器和预处理器。

解析器处理语法和解析查询, 生成一课对应的解析树。

预处理器进一步检查解析树的合法。比如: 数据表和数据列是否存在, 别名是否有歧义等。如果通过则生成新的解析树，再提交给优化器。

所以我觉得课后习题的错误应该发生在在分析器处理阶段>_<

2018-11-14 01:28

郭

有个问题不太明白，为什么对权限的检查不在优化器之前做?

2018-11-14 01:42

作者回复

有些时候，SQL语句要操作的表不只是SQL字面上那些。比如如果有个触发器，得在执行器阶段（过程中）才能确定。优化器阶段前是无能为力的

2018-11-14 09:53

yezhizi

源码安装完MySQL之后,使用Debug模式启动

mysqld --debug --console &后，

mysql> create database wxb;

Query OK, 1 row affected (0.01 sec)

mysql> use wxb;

Database changed

mysql> create table t(a int);

Query OK, 0 rows affected (0.01 sec)

mysql> select * from t where k=1;

ERROR 1054 (42S22): Unknown column 'k' in 'where clause'

T@4: | | | | | | | | | | error: error: 1054 message: 'Unknown column 'k' in 'where clause''

Complete optimizer trace:

答案就很清楚了

2018-11-14 11:06

作者回复

嗯嗯，Trace 日志是个好东西

2018-11-14 11:38

Geek_75aada

执行器 这个阶段才会打开表 才会知道表有哪些列

2018-11-14 00:35

ditiki

请问能否在专业名词旁边加上它们的英文名称，比如分析器，优化器等等。谢谢!

2018-11-14 04:57

作者回复

是好建议，下一篇开始这么执行哈

2018-11-14 10:14

zhxilin°C+

优化器阶段。个人理解：

1.编译原理告诉我们做一个Parser，核心有三步：词法分析、语法分析和语义分析；

2.分析器负责词法分析和语法分析，即将查询分解成一个个Identifier（词法分析），构造一棵解析树，整棵树只确保没有语法错误（语法分析），比如检查标识符是否有效、语句是否闭合等；

3.优化器紧接着分析上一步的解析树，这时才解决分析器阶段无法做的语义分析，也就是会检查表名、列表等是否正确、是否存在。

所以此题我认为是优化器阶段。

2018-11-14 09:51

Jason

1，连接

连接管理模块，接收请求；连接进程和用户模块，通过，连接线程和客户端对接

2，查询

查询缓存 Query Cache

分析器，内建解析树，对其语法检查，先from，再on，再join，再where.....；检查权限，生成新的解析树，语义检查（没有字段k在这里）等

优化器，将前面解析树转换成执行计划，并进行评估最优

执行器，获取锁，打开表，通过meta数据，获取数据

3，返回结果

返回给连接进程和用户模块，然后清理，重新等待新的连接

请大神指正

2018-11-14 16:43

作者回复

最后是“等待新的请求”哈，其他部分很不错呢

2018-11-14 18:48

尼古拉斯·赵四

我创建了一个没有select权限的用户，执行select * from T where k=1，报错“select command denied”，并没有报错“unknown column”，是不是可以说明是在打开表之后才判断读取的列不存在?

2018-11-14 13:56

作者回复

这个是一个安全方面的考虑。你想想一个用户如果没有查看这个表的权限，你是会告诉他字段不对还是没权限？如果告诉他字段不对，其实给的信息太多了，因为没权限的意思还包含了：没权限知道字段是否存在

2018-11-14 16:18

xuan

答案是分析器，理由如下：

根据文章分析器主要做两个事情，先做词法分析后做语法分析,词法分析主要做的是根据mysql的关键字进行验证和解析，而语法分析会在词法解析的基础上进一步做表名和字段名称的验证和解析；

看完整篇文章有个疑问望解答，在执行阶段为什么需判断对表是否有执行查询的权限，而不是在分析阶段去做；根据mysql中的information_schema库的存储信息理论上可以在分析器阶段判断是否有权限。

2018-11-14 12:59

志平

请问您的逻辑架构图是用什么工具画的？谢谢

2018-11-15 21:14

峰

分析器，感觉老师故意只讲了词法语法解析，却没有讲语义解析。很坏⊙▿⊙！

2018-11-15 11:22

作者回复

被…被发现了

2018-11-15 12:33

luffy

应该是分析器 1.从功能模块的职责来考虑 2.从性能来考虑，错误越晚发现成本越高。

2018-11-14 09:08

啊油酱

迫不及待想看第一篇文章。 加班回来收拾下刚好到这个点。 希望自己加油。

2018-11-14 00:14

作者回复

好辛苦，希望语音版会让你学起来轻松些

2018-11-14 00:28

Kids Return

建议github上面建立一个仓库 方便大家一起在上面完善笔记脑图之类的 这样我就不用记笔记了

2018-11-14 13:56

WL

为该讲总结了几个问题, 大家复习的时候可以先尝试回答这些问题检查自己的掌握程度:

1.
MySQL的框架有几个组件, 各是什么作用?
2.
Server层和存储引擎层各是什么作用?
3.
you have an error in your SQL syntax 这个保存是在词法分析里还是在语法分析里报错?
4.
对于表的操作权限验证在哪里进行?
5.
执行器的执行查询语句的流程是什么样的?

2018-11-24 11:24

作者回复

非常好

2018-11-24 14:37

lionetes

wait_timeout 是客户端 非交互式的连接时间，如果程序连接mysql SERVER，是交互连接,关联的时间参数为 interactive_timeout, 这两个时间参数需要尽量一致吗,一般设置多少合适?
query_cache_size 参数虽然不用了,我想确认下,关闭情况是 query_cache_size=0 要匹配参数query_cache_type=off吗？ 还是直接query_cache_size=0 即可?

2018-11-14 15:09

作者回复

第一个问题：是的，这两个尽量设置成相同。值的话取决于业务。如果你面对的是成熟的开发（比如公司内部团队），可以设置小些，分钟级别就行。

第二个问题：这两个都可以，不过用query_cache_type会好些（代码判断路径更短）

2018-11-14 22:57