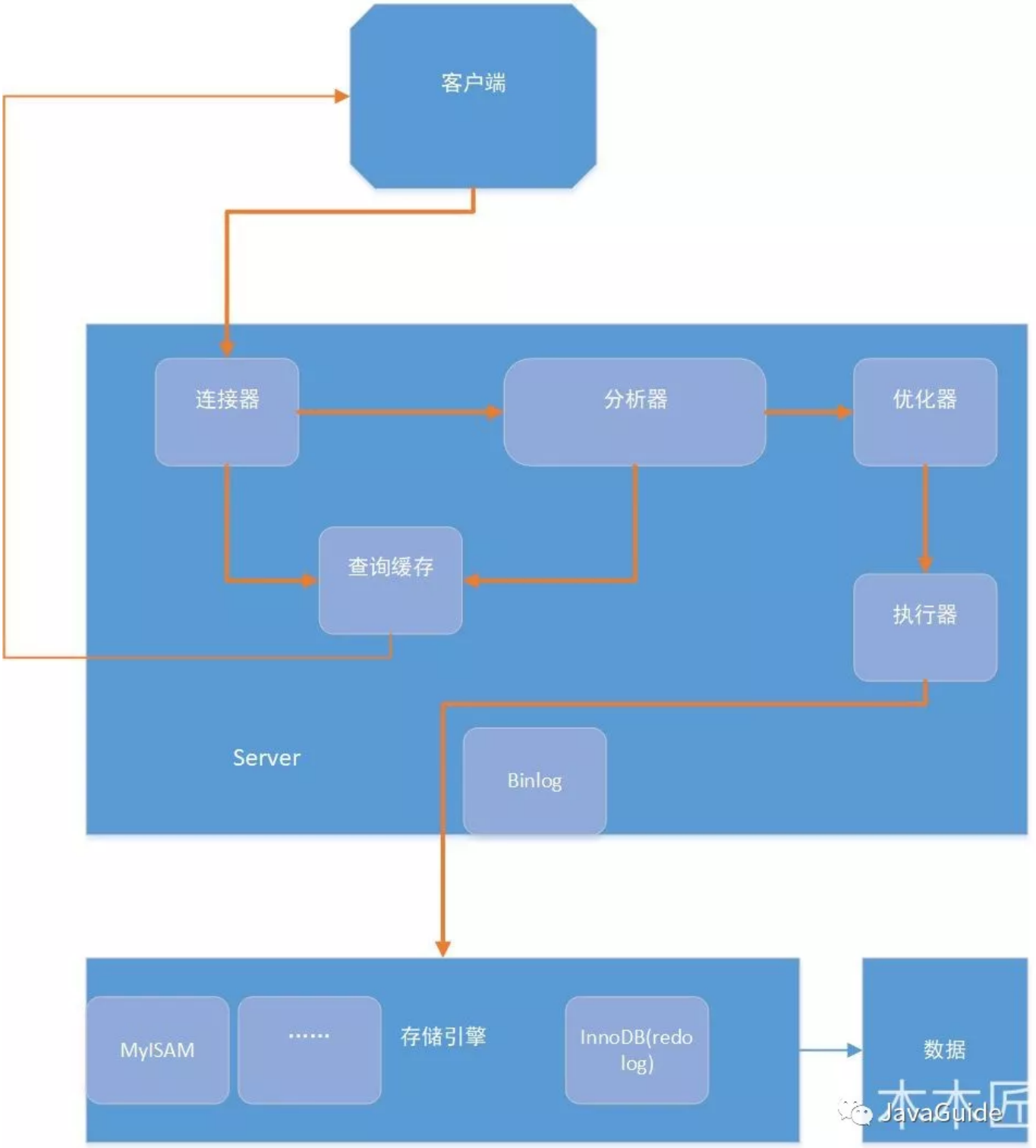
一条SQL语句是如何执行的

MySQL的基础架构



连接器：复杂用户身份认证和权限有关（登录MySQL) **查询缓存**：执行查询语句的时候，会查询缓存（MySQL8.0移除这个功能，这个功能不太实用） **分析器**：没有命中数据的话，SQL语句经过分析器，解析语句目的，检查语句的SQL语法； **优化器**：按照MySQL认为的最优的方法执行； **执行器**：执行语句，然后从存储引擎返回数据；

MySQL分为两层，一层是Server层，一层是存储引擎； Server层：主要包括连接器，查询缓存，优化器，分析器，执行器等，所有跨存储引擎的功能都在这一层实现，比如存储过程，触发器视图函数的等等 存储引擎；主要负责数据的存储和读取，采用可以更换的插件支持多种存储引擎；

Server层基本介绍

连接器：负责身份认证和权限相关的问题； 主要负责用户登录数据库，进行用户的身份认证，包括校验账户密码，权限等操作，如果用户账户密码都通过，连接器会到权限表中查询改用户的所有权限，之后在这个连接里的权限逻辑判断都是会依赖此时读取的权限数据；

缓存查询：用来缓存所执行的SELECT语句查询的是结果集； 连接建立之后，执行查询语句的时候，会先查询缓存，MySQL会先校验这个sql是否已经执行过，以K-V的形式缓存在内存中，key是查询语句，value是结果集。如果缓存key被命中，就会直接返回给客户端，如果没有命中就会执行后去的操作按成后就会把结果缓存起来，方便下一次调用；真正执行缓存的时候还会校验用户的权限，是否有权限查询这个表； MySQL查询不建议使用缓存，MySQL8.0移除了这个功能，因为查询缓存失效在实际业务场景中非常频繁，表更新的话，表中所有的查询缓存都会被清空；

分析器：MySQL查询缓存没哟命中，那么就会进入到分析器，分析器的功能就是分析sql语句是来干什么的，分析分为两步： 第一步:词法分析，一条SQL语句有多个字符串组层，首先提取其中关键字，比如select，提出查询的表，提出字段名，提出查询条件等等。等做完这些操作之后，就会进入第二步； 第二步：语法分析。主要是判断输入的sql是否正确，是否否和MySQL的语法；

优化器：优化器的作用是让MySQL按照他认为的最优的执行方案执行，比如多个索引的时候如何选择索引，多表查询的时候如何选择关联顺序等等；

执行器：选择了执行方案之后，MySQL开始执行，首先执行前校验该用户有没有权限，如果没有权限，就会返回错误信息，如果有权限就会调用引擎接口，返回接口执行的结果；

语句分析

sql语句分为两种，一种是查询，一种是更新，我们先分析查询语句，语句如下：

查询语句

```
select * from tb_student A where A.age = '18' and A.name = 'Bob';
```

先检查该语句是否有权限，如果没有权限，直接返回错误信息，如果有权限，在MySQL8.0之前会先查询缓存，以这条sql语句为key在内存中查询是否有结果，有直接缓存就返回，没有就执行下一步；

通过分析器进行语法分析，提取sql中语句的关键元素比如提取上面的语句是查询select，以及其他关键词等，然后判断这个sql的语法错误，比如关键词是否证券，检查没有问题就执行下一步；

接下来就是优化器优化确定方案，上面的sql语句：

a.先查询学生表中姓名为“张三”的学生，然后判断是否年龄是 18。 b.先找出学生中年龄 18 岁的学生，然后再查询姓名为“张三”的学生。

那么优化器根据自己的优化算法进行选择执行效率最好的一个方案（优化器认为，有时候不一定最好）。那么确认了执行计划后就准备开始执行了。

进行权限校验，如果没有权限就会返回错误信息，如果有权限就会调用数据库引擎接口，返回引擎的执行结果。

更新语句

```
update tb_student A set A.age='19' where A.name=' 张三 ';
```

其实条语句也基本上会沿着上一个查询的流程走，只不过执行更新的时候肯定要记录日志啦，这就会引入日志模块了，MySQL 自带的日志模块式 binlog（归档日志），所有的存储引擎都可以使用，我们常用的 InnoDB 引擎还自带了一个日志模块 redo log（重做日志）；

先查询到张三这一条数据，如果有缓存就会用到缓存。 然后拿到查询的语句，把age改为19，然后调用接口，写入数据，InnoDB把数据保存在内存中，同时记录redo log进入到prepare中，然后告诉执行器，执行完毕，随时提交。 执行器收到通知之后记录binlog然后调用引擎接口，提交redo log为提交状态；