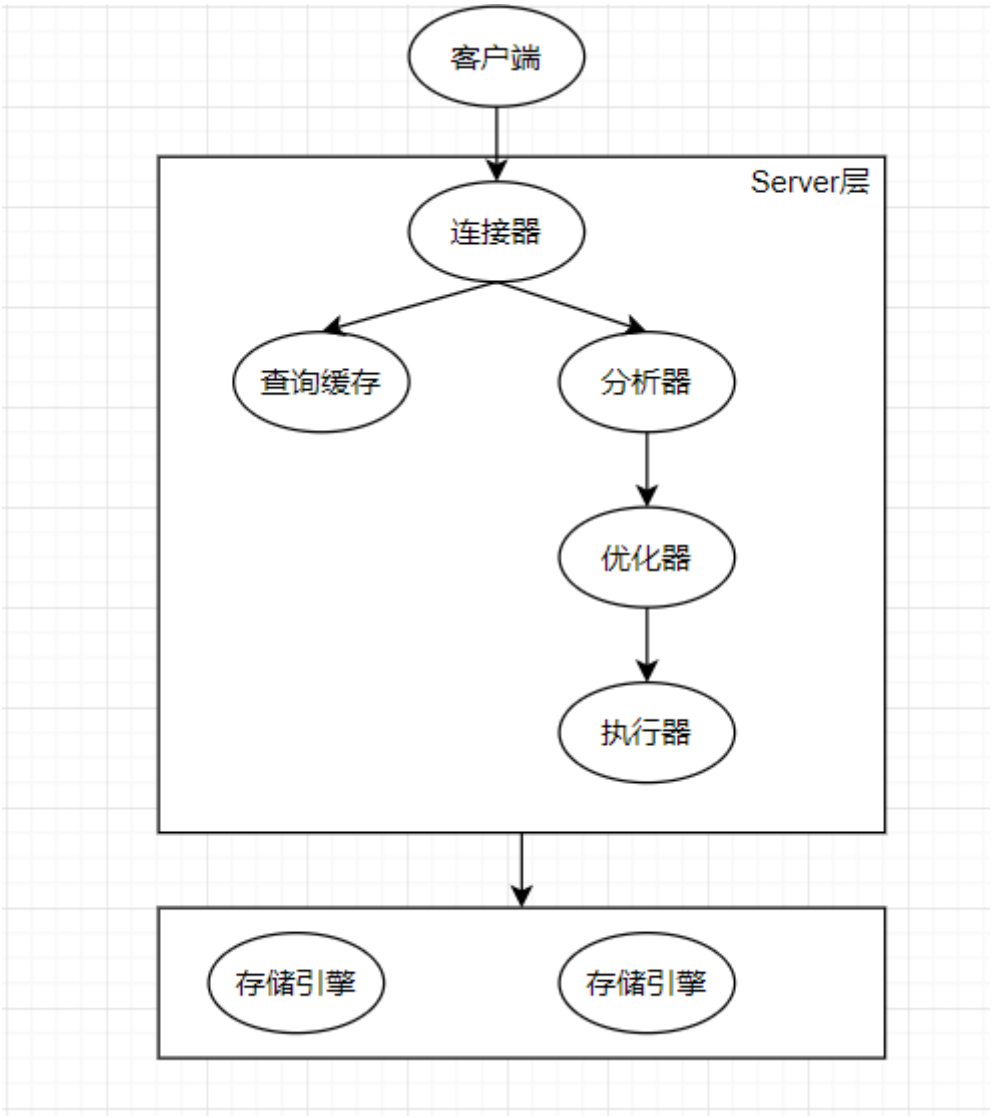
一、基础架构：一条SQL查询语句是如何执行的?

一、逻辑架构图



Server层涵盖mysql大多数核心功能，以及所有内置函数、跨存储引擎的功能（存储过程、触发器、视图）

- 连接器：管理连接，权限验证
- 查询缓存：命中缓存直接返回结果
- 分析器：词法分析、语法分析
- 优化器：生成执行计划，选择索引

- 执行器：调用引擎的接口，返回结果
- 存储引擎：存储数据，提供读写接口

二、连接器

1、连接器的职责

- 负责与客户端建立连接
- 权限验证
- 维持和管理连接

2、连接器的特性

- 为避免建立连接的开销，要尽量使用长连接
- 使用长连接，mysql内存占用增长，解决方案
  - 定期断开长连接
  - mysql5.7或以后版本，可在执行一次比较大的操作后，通过执行mysql_reset_connection来重新初始化连接资源。

3、客户端与连接器连接流程

- 客户端连接命令：mysql -h$ip -P$port -u$user -p
- 客户端与连接器通过tcp协议通信
- 连接建立后，开始验证用户名和密码
  - 密码验证通过后，连接器去权限表读取用户权限，该权限信息在连接的生命周期内有效
- 连接建立后无后续动作，该连接将处于空闲状态。
  - show processlist命令查看连接

- wait_timeout参数，连接断开超时间隔，默认8小时

三、查询缓存

- mysql收到一个查询请求后会先去查询缓存判断之前是否执行过这条语句。执行过的语句会以key-value对的形式，被缓存在内存中。key为查询语句，value为查询结果。
- 若缓存中找不到，继续执行后续阶段。执行完成后，执行结果会被存入查询缓存中。
- 大多情况下不用查询缓存的原因
  - 查询缓存的失效很频繁，只要对一个表的更新，这个表上的所有查询缓存都会被清空。
  - 查询缓存比较适用于不经常变的表，如系统配置表
- 通过参数query_cache_type设为demand，默认不会使用查询缓存了，通过select SQL_CACHE * from T 里显式使用
- mysql8.0版本删除了查询缓存模块

四、分析器

- mysql需要知道你要做什么，因此需要对sql语句做解析
- 词法分析：识别语句中字符串分别是什么，代表着什么
- 语法分析：根据语法规则，判断sql语句是否符合mysql语法

五、优化器

- 优化器是在表中有多个索引的时候，决定使用哪个索引
- 在一个语句有多表关联（join）的时候，决定各个表的连接顺序。

例子：select * from t1 join t2 using(ID) wherr t1.c=10 and t2.d=20

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

两种执行方法的结果是一样的，但执行效率不同，而优化器的作用就是决定使用哪个方案

六、执行器

- 执行之前，权限验证
- 验证通过后，执行器根据表的引擎定义，去使用引擎提供的接口

示例：select * from t where id=10;如果id字段没有索引，执行器的执行流程：

- 调用InnoDB引擎接口取表的第一行，判断id值是否为10，若不是则跳过，若是则将这行存储在结果集中。
- 调用引擎接口取下一行，重复相同的判断逻辑，直到取到表的最后一行
- 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
- 对于有索引的表，执行逻辑差不多，第一次调用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，这些接口都是引擎定义好的。

可以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。

七、问题及思考

按理说应该是先到分析器再到查询缓存，为什么连接器到查询缓存直接有一道线?

- 查询缓存存储的是key-value，key为查询语句，不需要进行分析就能判断有无