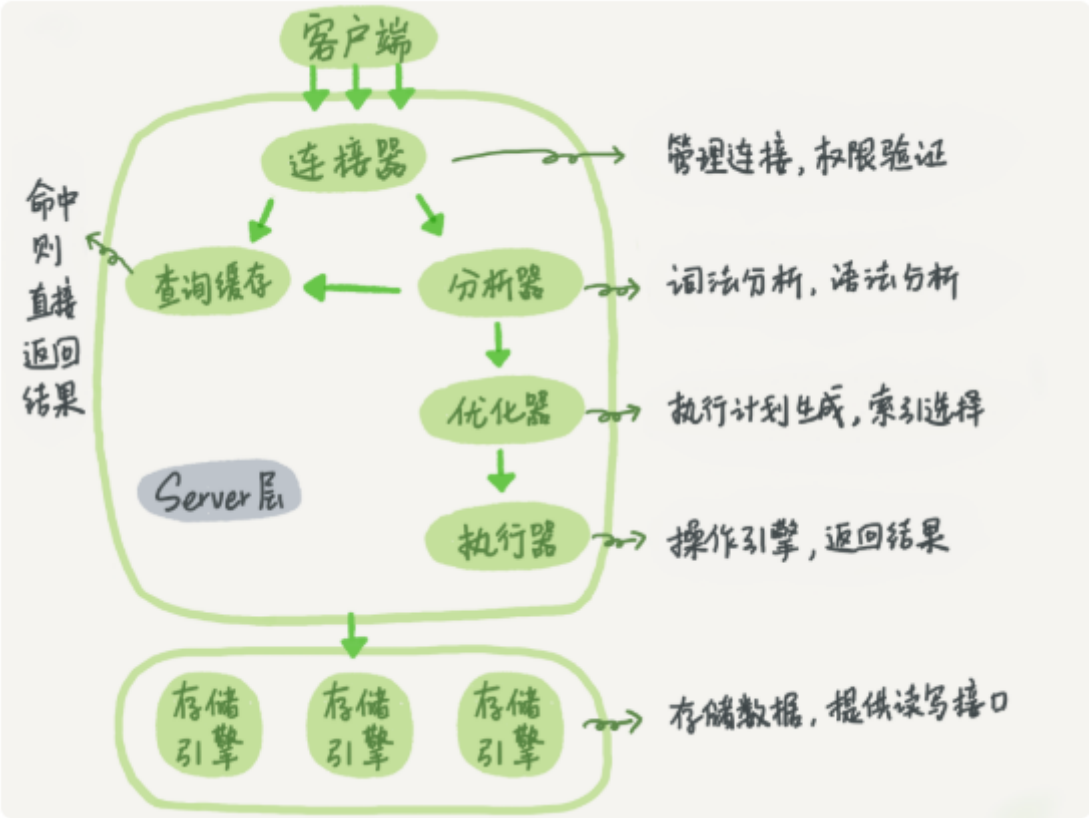
1.mysql基本架构



从图中可以看出，mysql的架构可以分为server层和存储引擎层。存储引擎层是采用插件式的设计，主要提供了数据的读写操作，从mysql5.5.5版本开始，默认的存储引擎是InnoDb。 Server层包括了连接器、查询缓存、分析器、优化器、执行器等，涵盖了mysql的大多数核心服务，所有的内置函数，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。

2.连接器

连接器负责跟客户端建立连接、获取权限、维持、管理连接。一条连接命令如下:

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

连接器首先会判断账号密码是否正确，校验功通过之后会去查询这个账号的权限。之后所有的权限验证都依赖此时查询到的权限，这意味着一个用户成功建立连接后，再也无法修改这个用户的权限，除非断开连接之后重连。 连接完成后，如果没有后续操作，这个连接就会处于空闲状态，mysql默认空闲8小时会自动断开连接。 mysql连接分为长连接和短连接，由于建立连接消耗很大，所以尽可能少的使用短连接。 但是全部使用长连接之后会出现一个问题，有些时候mysql的内存增长的特别快，这是因为mysql执行过程中临时使用的内存是管理在连接对象里的。这些资源在断开连接的时候才会释放，就会导致OOM，从现象上看就是mysql异常重启。 有两个解决方案:

- 定时断开长连接。使用一段时间后或者程序判断执行过一个占用内存大的查询后断开连接，之后重连。
- 如果使用mysql5.7以上版本，可以在每次执行一个占用内存大的查询后，通过执行**mysql_reset_connection**来重新初始化连接资源，这个操作并不会重连和重做权限验证。

3.查询缓存

当成功建立连接之后，执行就来到了查询缓存。mysql拿到一个查询请求会先去查询缓存是否之前执行过这条语句，如果执行过会以key-value的形式存储在缓存中。如果命中缓存，直接返回结果，效率非常高。 但大多数情况不建议使用查询缓存，查询缓存失效的频率会非常高。对更新压力大的数据库来说，查询缓存还没使用就被一个更新操作给清空了。除非是静态业务表，很长时间不会更新，这种时候才是查询缓存的适用场景。

4.分析器

分析器首先会做词法分析，输入一条sql语句，分析器会先识别出里面的字符串分别是什么，代表什么。做完词法分析之后，分析器就开始做语法分析，根据词法分析的结果，语法分析器会判断这条sql是否满足mysql的语法，如果语法错误就会返回第一次出现错误的位置。这个阶段也会判断表是否存在，列是否存在等。

5.优化器

经过了分析器之后，mysql就知道你要做什么了，但具体执行前，还需要经过优化器的处理。 优化器是在表有多个索引的时候，决定使用哪个索引；或一个语句多表关联(join)时，决定各个表的连接顺序。例如下面这个语句：

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

- 可以先从t1表中查询出c=10的记录的ID值，再根据ID值关联到t2表，在判断t2表里d的值是否等于20.
- 也可以先从t2表查询出d=20的记录的ID值，再根据ID关联t1表，在判断t1里c的值是否等于10.

这两种执行方法的逻辑结果都是一样的，但是不同的执行方法的效率是不同的，优化器就会选择执行效果最优的方法(但也有可能并不是最优)

6.执行器

优化器选择了最优执行方案之后就会交给执行器去执行，但是在执行前，会做权限判断，判断这个用户是否有查询权限，如果没有权限就会返回错误。（如果是命中查询缓存，则会在结果返回的时候做权限验证。） 如果有权限，就会打开表，打开表查询的时候会根据表的引擎定义去使用这个引擎提供的接口。 执行器将所有满足条件的结果返回，至此，这个语句就执行完成了。