第一步：应用程序把查询SQL语句发给服务器端执行

我们在数据层执行SQL语句时，应用程序会连接到相应的数据库服务器，把SQL语句发送给服务器处理。

第二步：服务器解析请求的SQL语句

SQL计划缓存，经常用查询分析器的朋友大概都知道这样一个事实，往往一个查询语句在第一次运行的时候需要执行特别长的时间，但是如果你马上或者在一定时间内运行同样的语句，会在很短的时间内返回查询结果。原因是：

1. 服务器在接收到查询请求后，并不会马上去数据库查询，而是在数据库中的计划缓存中找是否有相对应的执行计划。如果存在，就直接调用已经编译好的执行计划，节省了执行计划的编译时间。

2. 如果所查询的行已经存在于数据缓冲存储区中，就不用查询物理文件了，而是从缓存中取数据，这样从内存中取数据就会比从硬盘上读取数据快很多，提高了查询效率。数据缓冲存储区会在后面提到。

如果在SQL计划缓存中没有对应的执行计划，服务器首先会对用户请求的SQL语句进行语法效验，如果有语法错误，服务器会结束查询操作，并用返回相应的错误信息给调用它的应用程序。

注意：此时返回的错误信息中，只会包含基本的语法错误信息，例如select 写成selec等，错误信息中如果包含一列表中本没有的列，此时服务器是不会检查出来的，因为只是语法验证，语义是否正确放在下一步进行。

语法符合后，就开始验证它的语义是否正确。例如，表名、列名、存储过程等等数据库对象是否真正存在，如果发现有不存在的，就会报错给应用程序，同时结束查询。

接下来就是获得对象的解析锁，我们在查询一个表时，首先服务器会对这个对象加锁，这是为了保证数据的统一性，如果不加锁，此时有数据插入，但因为没有加锁的原因，查询已经将这条记录读入，而有的插入会因为事务的失败会回滚，就会形成脏读的现象。

接下来就是对数据库用户权限的验证。SQL语句语法，语义都正确，此时并不一定能够得到查询结果，如果数据库用户没有相应的访问权限，服务器会报出权限不足的错误给应用程序，在稍大的项目中，往往一个项目里面会包含好几个数据库连接串，这些数据库用户具有不同的权限，有的是只读权限，有的是只写权限，有的是可读可写，根据不同的操作选取不同的用户来执行。稍微不注意，无论你的SQL语句写的多么完善，完美无缺都没用。

解析的最后一步，就是确定最终的执行计划。当语法、语义、权限都验证后，服务器并不会马上给你返回结果，而是会针对你的SQL进行优化，选择不同的查询算法以最高效的形式返回给应用程序。例如在做表联合查询时，服务器会根据开销成本来最终决定采用hash join,merge join ，还是loop join，采用哪一个索引会更高效等等。不过它的自动化优化是有限的，要想写出高效的查询SQL还是要优化自己的SQL查询语句。

当确定好执行计划后，就会把这个执行计划保存到SQL计划缓存中，下次在有相同的执行请求时，就直接从计划缓存中取，避免重新编译执行计划。

第三步：语句执行

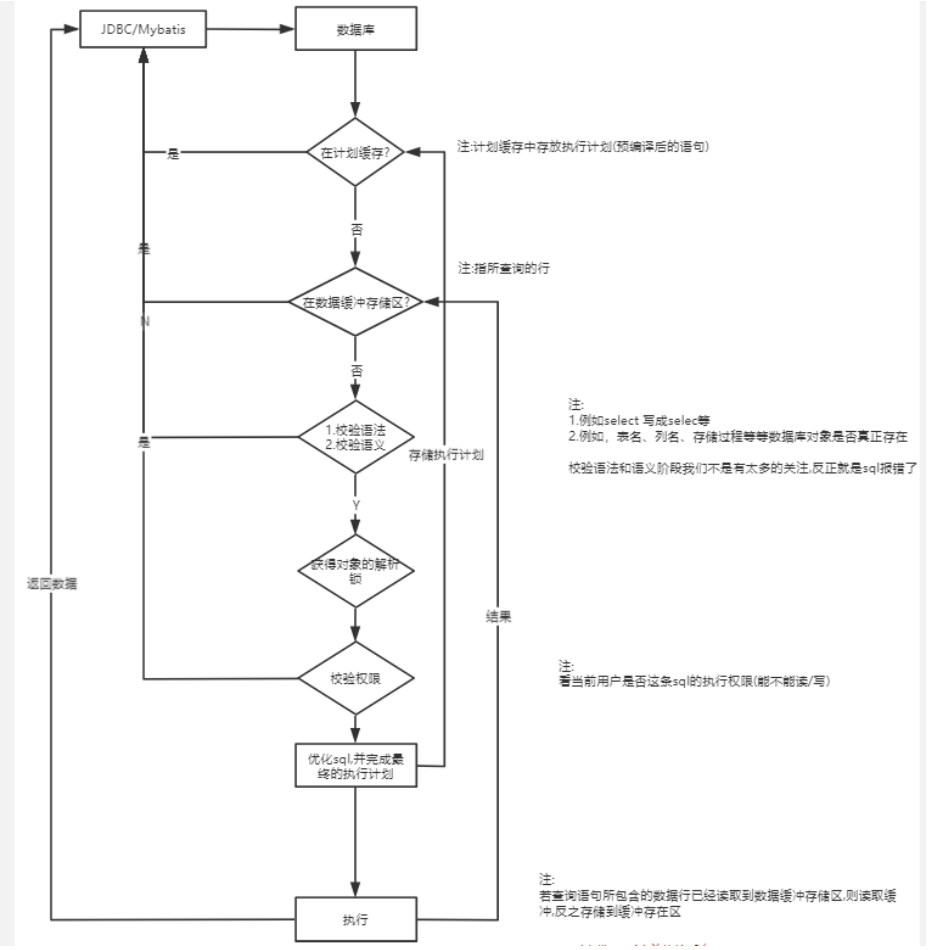
服务器对SQL语句解析完成后，服务器才会知道这条语句到底表态了什么意思，接下来才会真正的执行SQL语句。

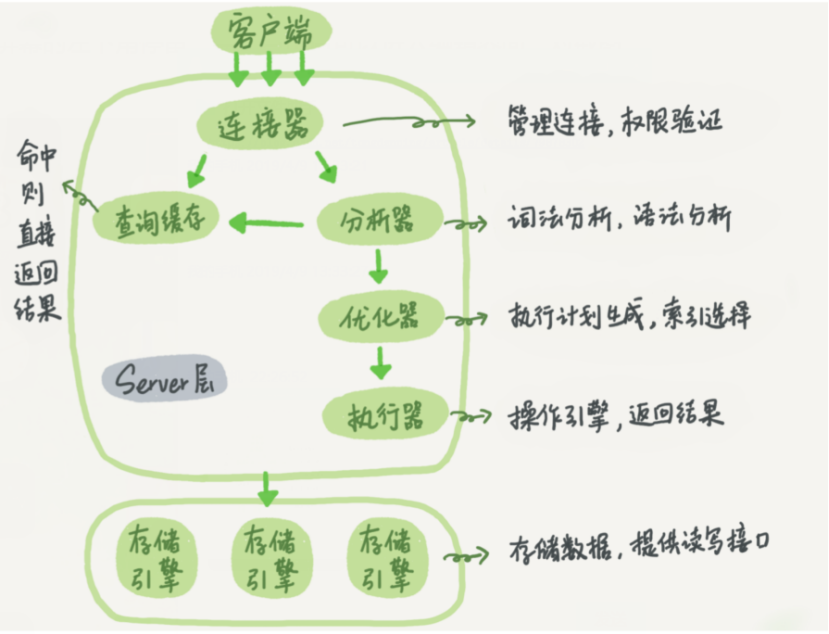
此时分两种情况：

1. 如果查询语句所包含的数据行已经读取到数据缓冲存储区的话，服务器会直接从数据缓冲存储区中读取数据返回给应用程序，避免了从物理文件中读取，提高查询速度。

2. 如果数据行没有在数据缓冲存储区中，则会从物理文件中读取记录返回给应用程序，同时把数据行写入数据缓冲存储区中，供下次使用。

来源：https://www.cnblogs.com/wuyun-blog/p/4697144.html





> https://www.cnblogs.com/ChangAn223/p/10686639.html

sql的解析顺序:


```
FROM <left_table>
ON <join_condition>
<join_type> JOIN <right_table>
WHERE <where_condition>
GROUP BY <group_by_list>
HAVING <having_condition>
SELECT
DISTINCT <select_list>
ORDER BY <order_by_condition>
LIMIT <limit_number>
```

https://www.bilibili.com/video/BV12b411K7Zu?p=189