

1、当我们执行一条sql查询，coordinator接收到这条sql语句以后，它会有一个sql的语法解析器去把sql语法解析变成一个抽象的语法树AST，这抽象的语法书它里面只是进行一些语法解析，如果你的sql语句里面，比如说关键字你用的是int而不是Integer，就会在语法解析这里给暴露出来

2、如果语法是符合sql语法规范，之后会经过一个逻辑查询计划器的组件，他的主要作用是，比如说你sql里面出现的表，他会通过connector的方式去meta里面把表的schema，列名，列的类型等，全部给找出来，将这些信息，跟语法树给对应起来，之后会生成一个物理的语法树节点，这个语法树节点里面，不仅拥有了它的查询关系，还拥有类型的关系，如果在这一步，数据库表里某一列的类型，跟你sql的类型不一致，就会在这里报错

3、如果通过，就会得到一个逻辑的查询计划，然后这个逻辑查询计划，会被送到一个分布式的逻辑查询计划器里面，进行一个分布式的解析，分布式解析里面，他就会去把对应的每一个查询计划转化为task

4、在每一个task里面，他会把对应的位置信息全部给提取出来，交给执行的plan，由plan把对应的task发给对应的worker去执行，这就是整个的一个过程

这是一个通用的sql解析流程，像hive也是遵循类似这样的流程，不一样的地方是distribution planner和executor pan，这里是各个引擎不一样的地方，前面基本上都一致的

**最后，一些零散的知识点**

presto适合pb级的海量数据查询分析，不是说把pb的数据放进内存，比如一张pb表，查询count，vag这种有个特点，虽然数据很多，但是最终的查询结果很小，这种就不会把数据都放到内存里面，只是在运算的过程中，拿出一些数据放内存，然后计算，在抛出，在拿，这种的内存占用量是很小的，但是join这种，在运算的中间过程会产生大量的数据，或者说那种查询的数据不大，但是生成的数据量很大，这种也是不合适用presto的，但不是说不能做，只是会占用大量内存，消耗很长的时间，这种hive合适点

presto算是hive的一个补充，需要尽快得出结果的用presto，否则用hive

work是部署的时候就事先部署好的，work启动100个，使用的work不一定100个，而是根据coordinator来决定拆分成多少个task，然后分发到多少个work去

一个coordinator可能同时又多个用户在请求query，然后共享work的去执行，这是一个共享的集群

coordinator和discovery server可以启动在一个节点一个进程，也可以放在不同的node上，但是现在公司大部分都是放在一个节点上，一个launcher start会同时把上述两个启动起来

对于presto的容错，如果某个worker挂掉了，discovery server会发现并通知coordinator

但是对于一个query，是没有容错的，一旦一个work挂了，那么整个qurey就是败了

因为对于presto，他的查询时间是很短的，与其查询这里做容错能力，不如重新执行来的快来的简单

对于coordinator和discovery server节点的单点故障，presto还没有开始处理这个问题貌似

http://blog.csdn.net/jiangshouzhuang/article/details/52461274