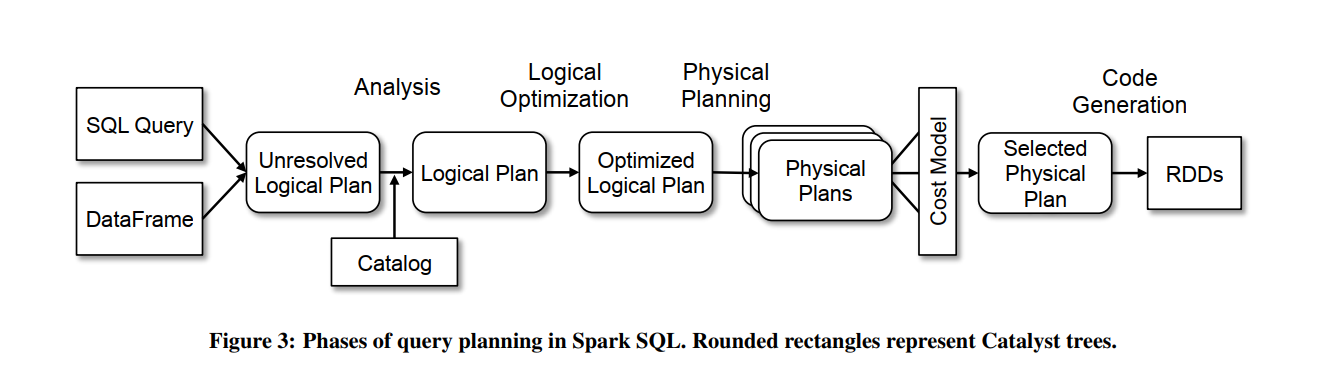
# Using Catalyst in Spark SQL

|  |
| --- |
| 在SparkSQL中, 把一条SQL语句转化成对应的RDD任务, 完成这一过程(后续用"解析SQL到RDD"过程代替这一过程)全部在Catalyst框架实现.  "解析SQL到RDD"这一过程, 主要包含了4个组成部分, 如下图所示 |



## analyzing a logical plan to resolve references(分析逻辑计划用于解析引用)

|  |
| --- |
| SparkSQL开始是一个待计算的关系, 要么是被SQL parser返回的abstract syntax tree(AST), 要么是使用API构造的DataFrame对象.  这两种情况下,待计算的关系都包含unresolved 属性引用或者关系. (简单说,就是缺失schema)  SparkSQL用Catalyst Rules和一个Catalog对象从元数据找到schema, 构建Logical Plan. |

## logical plan optimization(逻辑计划优化)

|  |
| --- |
| Logical Optimiztion阶段应用标准的RBO(rule-based optimization)去优化Logical Plan.  RBO包括constant folding、predicate pushdown(谓词下推)、projection pruning、null propagation、Boolean expression simplification等等规则. |

## physical planning(物理计划)

|  |
| --- |
| The Physical Planning阶段, 根据Optimized Logical Plan生成一个或多个物理计划. 用cost model选择一个计划. 这个阶段, CBO(cost-based optimization)仅仅用于选择join算法. 针对数据较小的表, Spark SQL使用broadcast join. |

## code generation(代码生成)

|  |
| --- |
| 在每台有数据的机器上生成Java二进制代码(移动数据不如移动代码). |