**Hive源码分析(三)-执行流程分析**

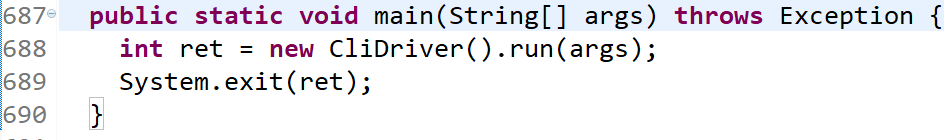
2016年12月24日 16:22:08

# 1.启动脚本:

$HIVE\_HOME/bin/hive --> $HIVE\_HOME/bin/ ext/cli.sh  可以看到入口类org.apache.hadoop.hive.cli.CliDriver

# 2.入口类:

apache-hive-3.1.0-src/cli/src/java/org/apache/hadoop/hive/cli/CliDriver.java



(1) 参数解析OptionsProcessor( -f -hiveconf -d -i 等等参数解析)

process\_stage1  如果是hiveconf的参数就System.setProperty , 并把-d 和 -hivevar的加入到hiveVariables中

(2) log4j 日志加载 initHiveLog4j( 此时会调用HiveConf 并将里面的一些静态变量初始化了，获取日志的一些配置)

(3) 创建CliSessionState

创建HiveConf，设置hiveJar = hive-exec-1..1.0.jar

(4) 参数解析 process\_stage2 并将参数设置到CliSessionState中(-S -database -e -f -v -i 等)

(5) SessionState.start (CliSessionState设置到SessionState中)

通过Hive(apache-hive-3.1.0-src/ql/src/java/org/apache/hadoop/hive/ql/metadata/Hive.java) 创建HiveDb

通过Hive getMSC 获取MetaStoreClient

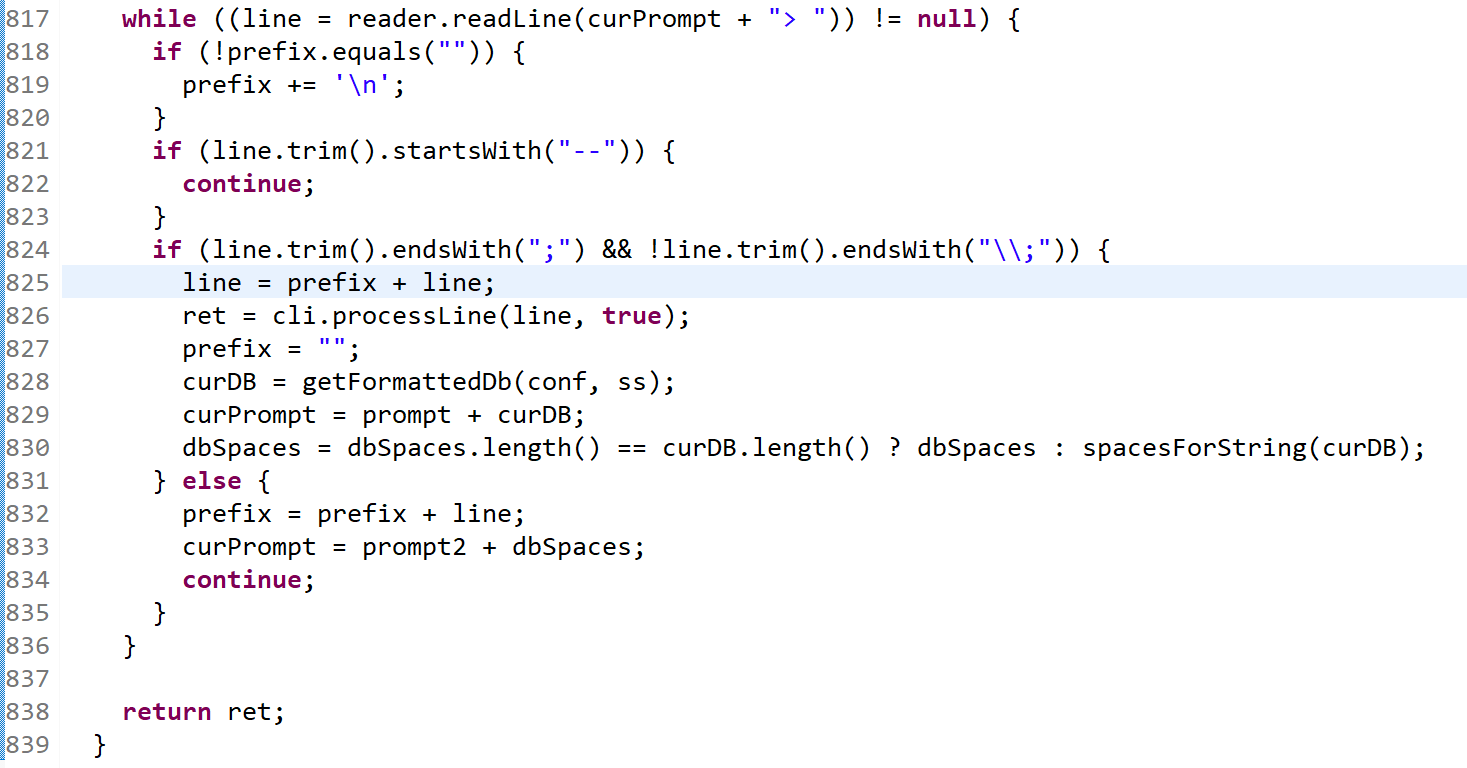
(6) executeDriver  执行

创建 cliDriver

如果-e 执行processLine(ss.execString)

如果-f  执行processFile(ss.fileName)

创建ConsoleReader  while 循环读取用户输入，遇到“；”为一个完整的命令，执行该命令(cli.ProcessLine 处理)



# 3. ProcessLine 每条语句处理流程

  (1) 设置中断信号处理 --> processCmd

如果是quit, exit 退出

如果是source 加载文件 processFile 处理

如果是!处理shell

processLocalCmd 处理正常语句(通过不同的CommandProcessor 处理不同的语句)

通过hiveCommand 执行不同的Processor

set           SetProcessor，设置修改参数,设置到SessionState的HiveConf里。   
dfs           DfsProcessor，使用hadoop的 FsShell运行hadoop的命令。   
add         AddResourceProcessor  添加到SessionState的resource\_map里，运行提交job的时候会写入 Hadoop的Distributed Cache。   
delete    DeleteResourceProcessor从SessionState的resource\_map里删除。

list         ListResourceProcessor  列出资源

如果为hiveCommand 为null 执行Driver中的run 函数

(2)  Driver run 执行 runInternal

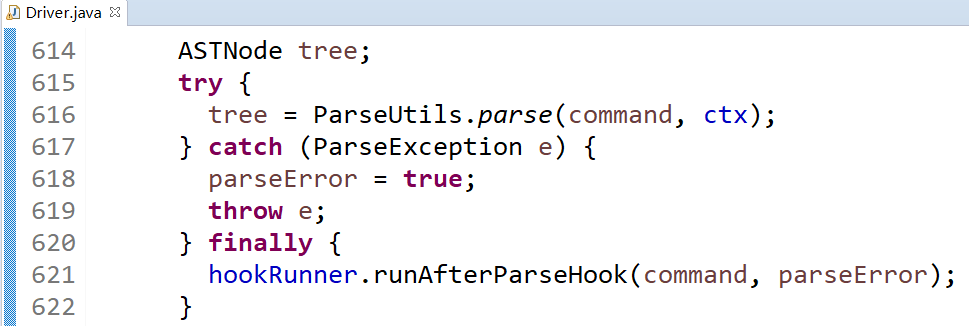
获取hive.exec.driver.run.hooks 执行自己定义的一些hook函数，hive 默认为空

compileInternal 进行编译

execute 执行

(3) compileInternal 编译

词法分析(通过HiveParse。 进行解析)  此时会获取语句的TOK 类型(例如TOK\_SHOWDATABASES, TOK\_SHOWTABLES, TOK\_QUERY)



      语义分析BaseSemanticAnalyzer sem = SemanticAnalyzerFactory.get(conf, tree);

            通过不同的tree.getType 来创建不同的语义分析

                TOK\_EXPLAIN --->  ExplainSemanticAnalyzer

                TOK\_EXPLAIN\_SQ\_REWRITE --> ExplainSQRewriteSemanticAnalyzer

                TOK\_LOAD     ---> LoadSemanticAnalyzer

                TOK\_EXPORT -->ExportSemanticAnalyzer

                TOK\_IMPORT -->ImportSemanticAnalyzer

                DDLSemanticAnalyzer 操作(ALTER, SHOW GRANT 等等操作)

                TOK\_QUERY 用的SeManticAnalyzer(select , insert)

     hive.semantic.analyzer.hook ，在执行sem.analyze 前后执行preAnalyze 和 postAnalyze

     plan = new QueryPlan()

     plan.getFetchTask().initialize(conf,plan,null)  初始化FetchTask

     schema = getSchema()  获取返回的schema

     sem.skipAuthorization() 授权操作 doAuthorization

# 4. DDLSemanticAnlyzer和 SeManticAnalyzer分析

(1) DDLSemanticAnlyzer -->analyzeInternal -->不同的TOK 不同处理–>例如TOK\_SHOWDATABASES -->analyzeShowDatabases --> rootTasks 添加DDLWork任务–> setFetchTask(获取返回的schema)

(2) SemanticAnlyzer–>analyzeInternnal --> doPhase1(分析AST树，AST拆分成查询子块) -->getMetaData(获取元数据信息) -->genOpTree(生成逻辑执行计划) -->Optimizer.optimize（优化逻辑执行计划)-->TaskCompile.compile（执行物理执行计划，此处可以选择MR，TEZ，Spark，通过TaskCompileFactory具体的去获取）

compile的过程中会把任务放到rootTasks里。

# 5. doAuthorization

doAuthorization --> isAuthorizationModeV2 -->getAuthorizationMode–>setupAuth–>通过hive.security.authenticator.manager 获取authenticator-->通过hive.security.authorization.manager 获取authorizer

getAuthorizeProviderManager 来获取通过返回是否为null 来判断是auth V1 还是authV2 ， 如果class 是继承HiveAuthorizationProvider那返回对应的class

如果不是返回null。

如果为null 通过HiveUtils.getAuthorizerFactory 获取授权Factory. 此时返回的是auth V2 版本。

# 6.编译后的execute执行

(1) 通过hive.exec.pre.hooks  Get all the pre execution hooks and execute them

(2) getRootTasks 获取 所有的task

(3) 将task 放入driverCxt 中（driverCxt.addToRunnable)

(4) 通过TaskRuner 启动任务。

TaskRunner.runSequential --> tsk.executeTask() -->execute(不同的task执行不同execute)

任务类型:       可以查砍apache-hive-3.1.0-src\ql\src\java\org\apache\hadoop\hive\ql\exec\TaskFactory.java 中查看具体有哪些任务类型。

主要DDLWork， FunctionWork， MapredWordk， TezWork(如果物理执行计划是通过tez)