BQL新方案

武汉用户画像团队　余启

BQL

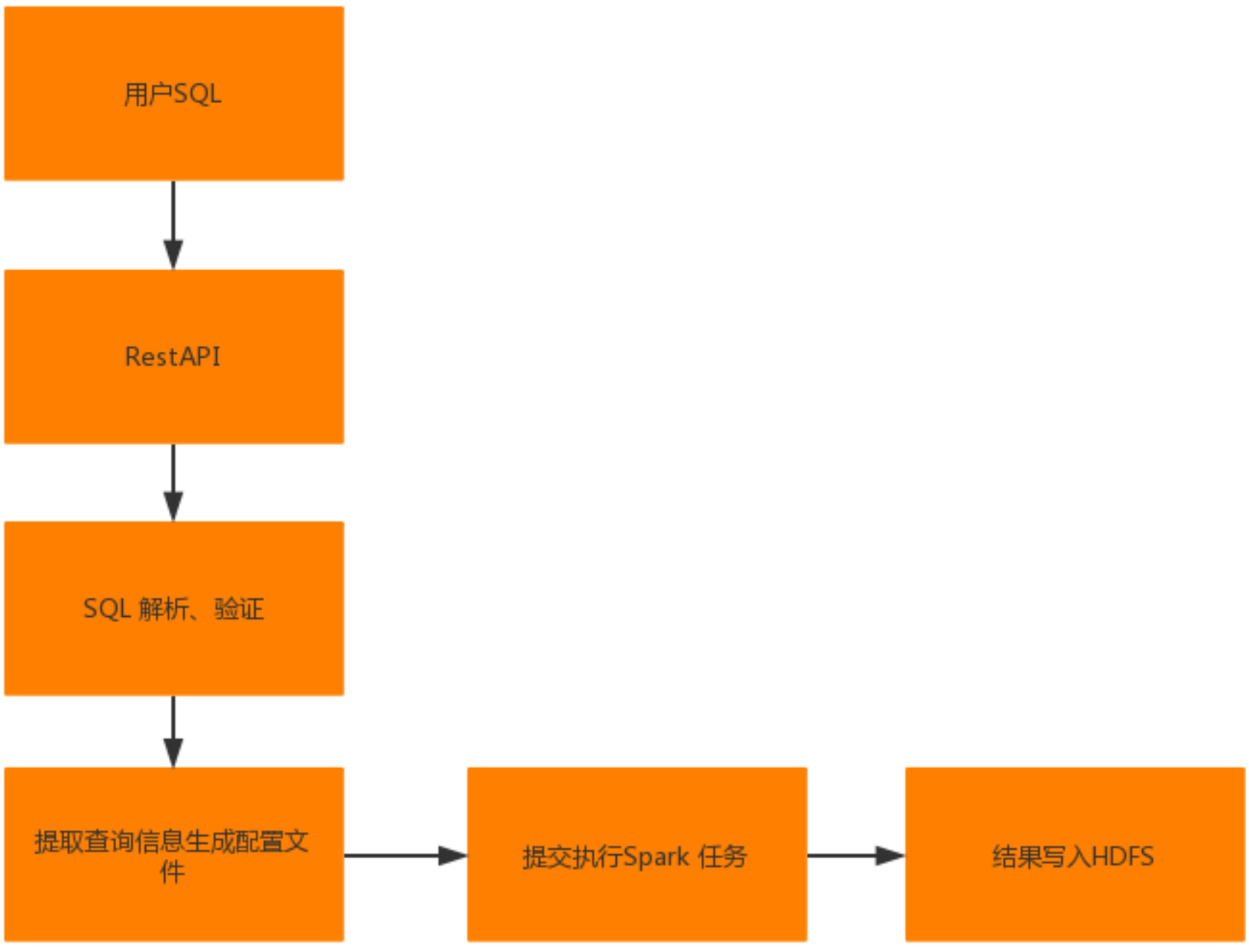
- BQL(Behavior Query Language) 处理用户行为数据，方便客户处理、分析数据

- 主要特点:
  1. 类SQL
  2. 支持海量数据查询

- 如提取用户，该用户满足使用“百度地图”或者使用”高德地图"

```
select id,text from table A where source=app AND
action=usage AND (text contains
"com.autonavi.minimap" OR text contains
“com.baidu.BaiduMap")
```

BQL执行过程


用户SQL
RestAPI
SQL 解析、验证
提取查询信息生成配置文件
提交执行Spark 任务
结果写入HDFS

主要问题

1.不够通用， 采用硬解

2.逻辑比较复杂、对新员工理解SQL提取过程不友好

3.支持功能有限，缺少常用谓词函数支持

4.不支持常见的函数如时间函数

5.SQL语法很随意 如 where a contains ‘xiaomi.com’这种语法, 用like 或UDF代替

6.中间过程采用 RDD[UserBehavior]和RDD[UserContext]无法自适应列裁剪和增加

目标

- 语法支持SQL标准 如SQL92
- 简化SQL解析优化，不用自已造轮子

语法解析工具调研

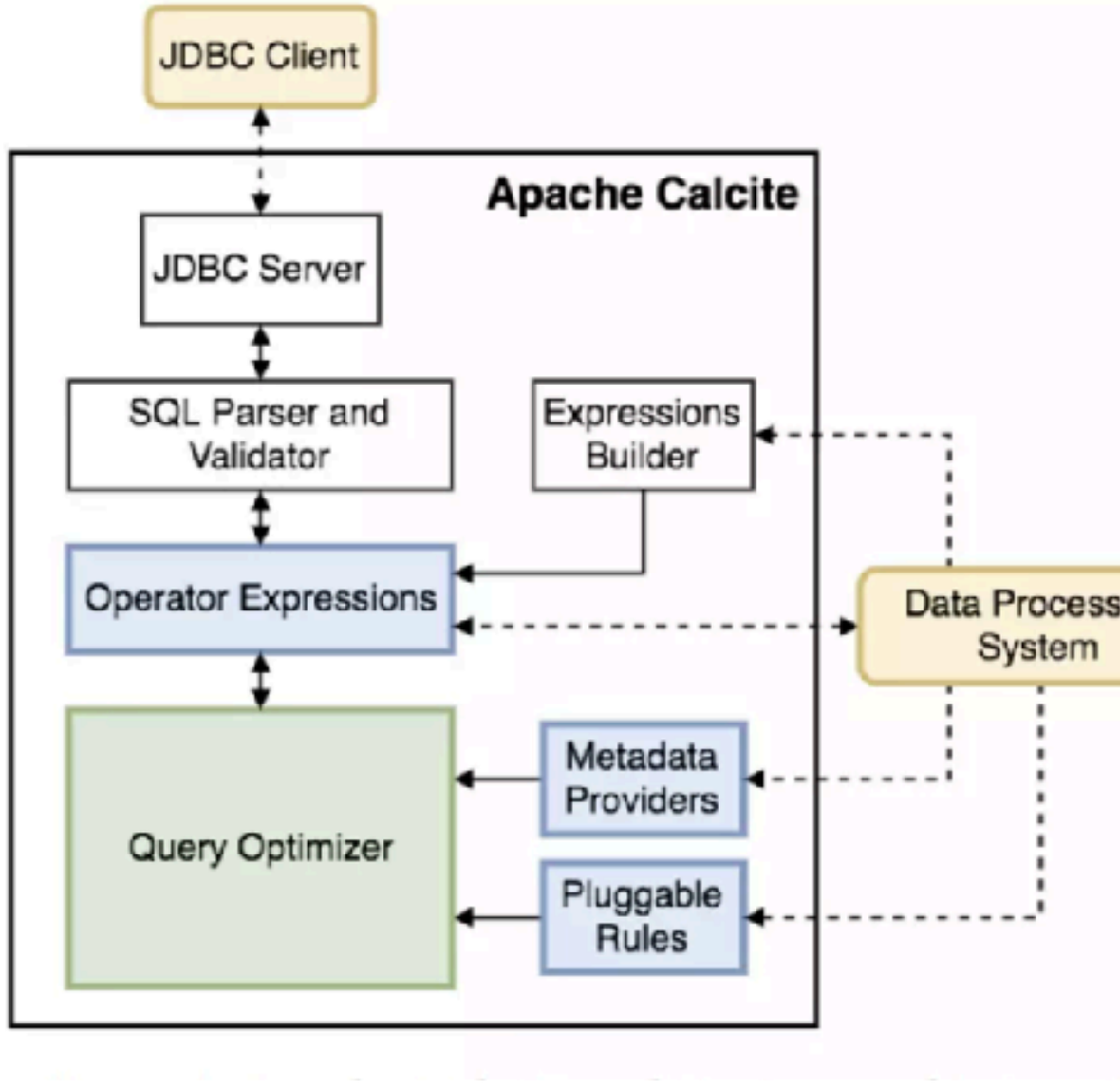
1. Calcite

Calcite是一个独立的集SQL解析、优化、执行的框架, 详细可以参考我的简书: https://www.jianshu.com/p/2dfbd71b7f0f . 其优点:

-Java jar包，Java cc编写，可以很方便以jar包的形式引入
-支持丰富的SQL schema注册、表注册、查询等
-易用的优化器
-易扩展，可以通过修改javacc 文件来扩展对应的语法
-多开源OLAP支持 如Flink、Drill、Hive等

缺点:

公开的资料较少,只有官网有若干文档,
代码对新学者不太友好
https://github.com/yuqi1129/calcite-test

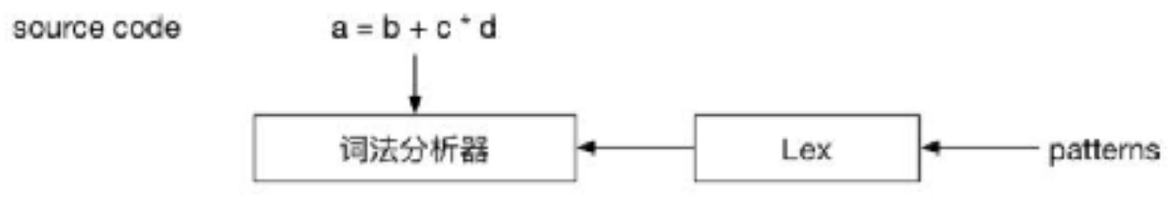


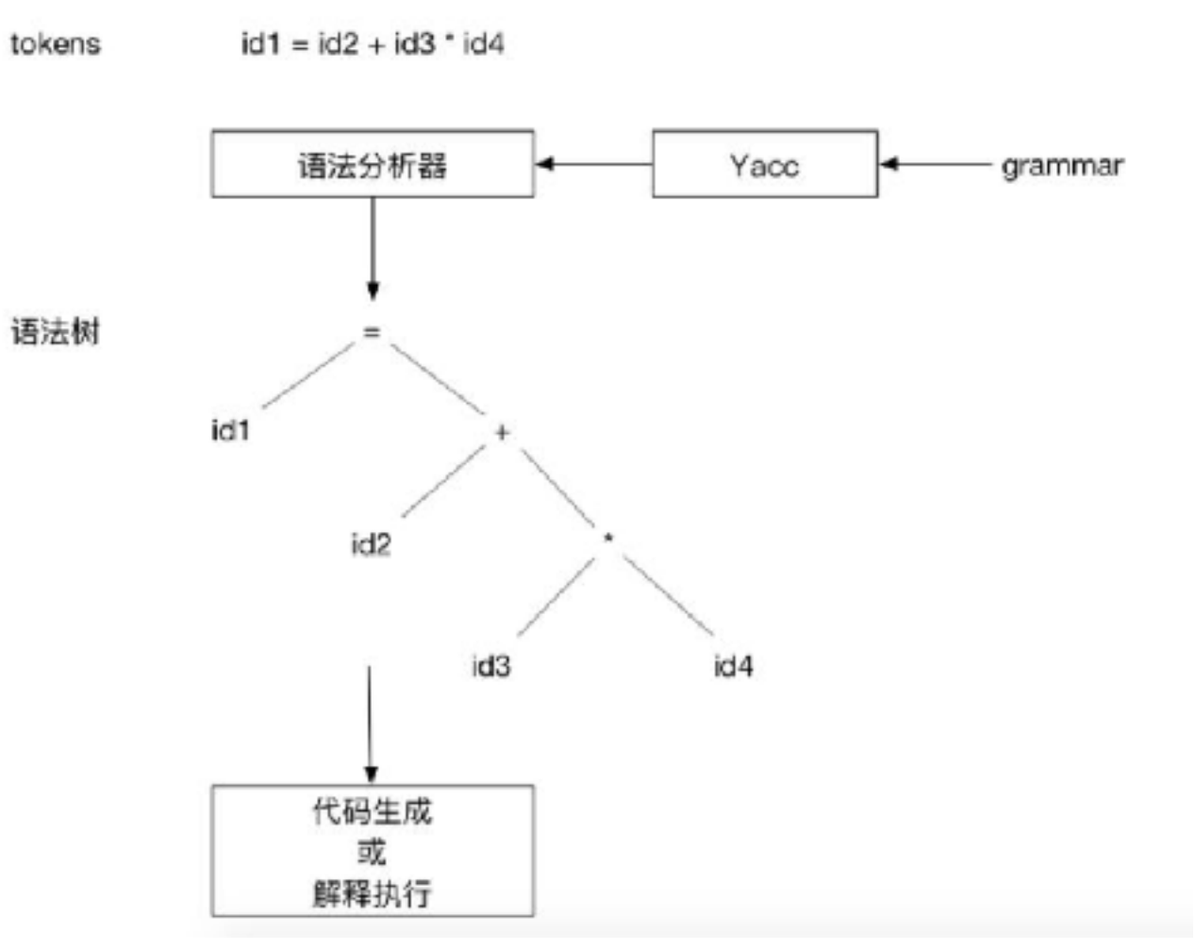
2. Lex + Yacc / Flex + Bison

1.Lex(Lexical Analyzar)， Yacc(Yet another compiler compiler)

- 是一款生成扫描器工具 用以识别文本模式
- C语言 强耦合
- Yacc 是一种工具，将任何一种编程语言的所有语法翻译成针对此种语言的 Yacc 语 法解析器

缺点: 太过复杂、语言不相通; 代码开发比较入远; 需要重复造轮子





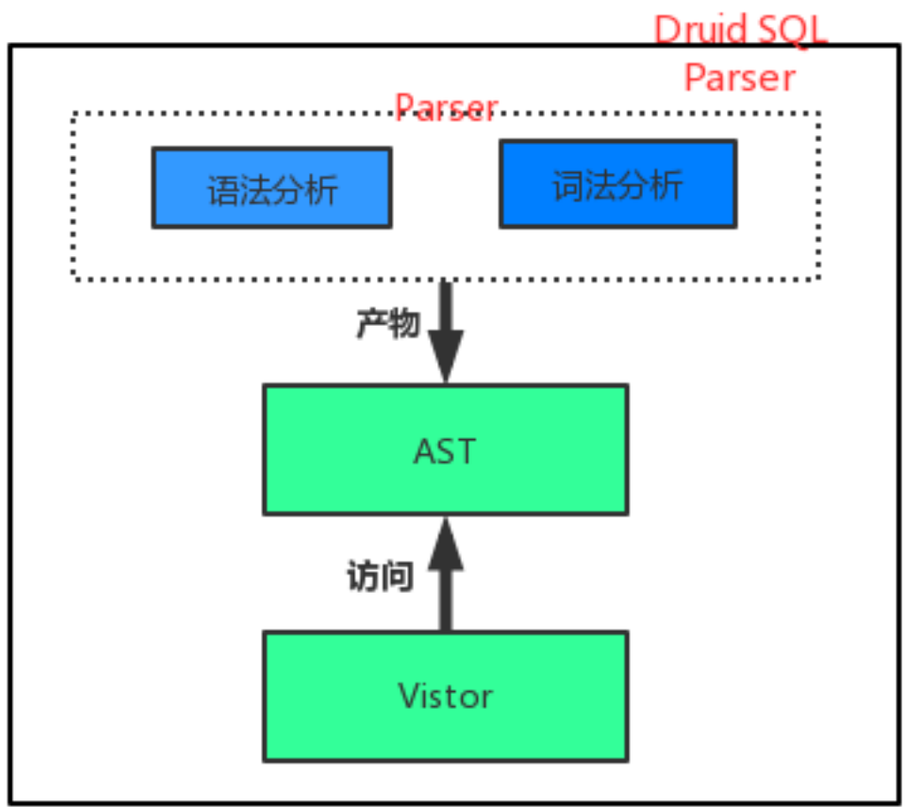
3. ANTLR

一款词法器（Lexer）、语法分析器(parser)和树分析器(Tree Parser)
- Java 编写 LL top-down 算法
- 在开源的Presto, Spark都采用这个方案， 用户只需要自定义语法文件，然后ANTLR根据该文件生成特定的语法分析代码(SqlBaseLexer和SqlBaseParser)。
- SqlBaseLexer 解析语法 SqlBaseLexer 进行表验证、类型验证等操作。
- 相对于calcite 来说Antlr缺少必要optimizer以及相应的优化框架，全部需要手动自已实现，Spark Catalyst。

优点是ANTLR使用场景较广，语法不仅仅限于SQL,如计算器，可定制化强。
Example: https://stackoverflow.com/questions/1931307/antlr-is-there-a-simple-example

4. Druid SQL Parser

是Druid 重要组成之一，目前能支持较为完备的SQL语法， 其特点是
- parser性能非常好， 支持自定义visitor AST, 同时支持常见的方言，没有相应的优化器，AST后需要自行解析优化。
- 采用类似于ANTLR 语法树结构，自行读取SQL、生成Statement 再生成AST（FAST SQL)
-
具体可以参考以下
https://github.com/alibaba/druid/wiki/SQL-Parser



5. TiDB parser

使用go yacc 根据预定义的SQL语法规则文件parser.y生成 SQL 语法解析器, 采用go语言编写， 详细可以看看5.2中有关介绍

6. 其它可以参考

https://en.wikipedia.org/wiki/Comparison_of_parser_generators

总结

综合以上信息，打算选用Calcite作为相应的SQL解析与优化框架

- 有内置的优化器, 可插拔
  RBO/CBO

- SQL 各种标准语法支持
  oracle/MySQL

- 语法易扩展
  JavaCC

- 支持各种Adapter
  JDBC

Calcite实现步骤

- SQL

```
select cast(text as int) + 1, count(1) from b where text <> '3' group by
    text having count(1) > 2 order by cast(text as int) + 1 limit 3
```

- RelNode

```
LogicalSort(sort0=[$0], dir0=[ASC], fetch=[3])
    LogicalProject(EXPR$0=[+(CAST($0):INTEGER, 1)], EXPR$1=[$1])
        LogicalFilter(condition=[>($1, 2)])
                LogicalAggregate(group=[{0}], EXPR$1=[COUNT()])
                    LogicalProject(TEXT=[$3], $f1=[1])
                            LogicalFilter(condition=[<>($3, '3')])
                                EnumerableTableScan(table=[[B]])
```

- CodeGen

```
Translate(RelNode  relnode) {
   If (relnode has input) {
      Input = translate(relnode.input)
      …
   } else {
      //table scan
      createRDD from HDFS
   }

   Case project
       input.project  -> Dynamic make class and func
   case  filter
       input.filter
   case sort
       input.sort
   case join
   ….
}
```

存在的问题

- 关键字支持，如Sample
  修改calcite parser.jj 文件
  可以参考:
  https://github.com/yuqi1129/mysql/tree/master/mysql-common/src/main/codegen/templates

- 复杂类型支持，如Map、List
  提供了Map和Array类型
  select map, list from t
  select map[‘id’], list(1) from t
  实现对应的方法体即可

后期计划

- 完善各种Select、Filter等列计算CodeGen
- 接入Join操作
- 丰富函数UDF