

18 | 如何自己开发一个大数据SQL引擎？

2018-12-08 李智慧

从0开始学大数据

[进入课程 >](#)



讲述：李智慧

时长 11:20 大小 10.38M



从今天开始我们就进入了专栏的第三个模块，一起来看看大数据开发实践过程中的门道。学习一样技术，如果只是作为学习者，被动接受总是困难的。但如果从开发者的视角看，很多东西就豁然开朗了，明白了原理，有时甚至不需要学习，顺着原理就可以推导出各种实现细节。

各种知识从表象上看，总是杂乱无章的，如果只是学习这些繁杂的知识点，固然自己的知识面是有限的，并且遇到问题的应变能力也很难提高。所以有些高手看起来似乎无所不知，不论谈论起什么技术，都能头头是道，其实并不是他们学习、掌握了所有技术，而是他们是在谈到这个问题的时候，才开始进行推导，并迅速得出结论。

我在 Intel 的时候，面试过一个交大的实习生，她大概只学过一点 MapReduce 的基本知识，我问她如何用 MapReduce 实现数据库的 join 操作，可以明显看出她没学习过这部分

知识。她说：我想一下，然后盯着桌子看了两三秒的时间，就开始回答，基本跟 Hive 的实现机制一样。从她的回答就能看出这个女生就是一个高手，高手不一定要很资深、经验丰富，把握住了技术的核心本质，掌握了快速分析推导的能力，能够迅速将自己的知识技能推进到陌生的领域，就是高手。

这也是我这个专栏的目的，讲述大数据技术的核心原理，分享一些高效的思考和思维方式，帮助你构建起自己的技术知识体系。

在这个模块里，我将以大数据开发者的视角，讲述大数据开发需要关注的各种问题，以及相应的解决方案。希望你可以进入我的角色，跳出纷繁复杂的知识表象，掌握核心原理和思维方式，进而融会贯通各种技术，再通过各种实践训练，最终成为真正的高手。

前面专栏我们提到过，程序员的三大梦想，也就是开发数据库、操作系统和编译器。今天我们通过一个支持标准 SQL 语法的大数据仓库引擎的设计开发案例，看看[如何自己开发一个大数据库 SQL 引擎](#)。

在学习今天的内容前，我们先回顾一下前面讨论过的大数据仓库 Hive。作为一个成功的大数据仓库，它将 SQL 语句转换成 MapReduce 执行过程，并把大数据应用的门槛下降到普通数据分析师和工程师就可以很快上手的地步。这部分内容如果你忘记了，可以返回[专栏第 11 期](#)复习一下。

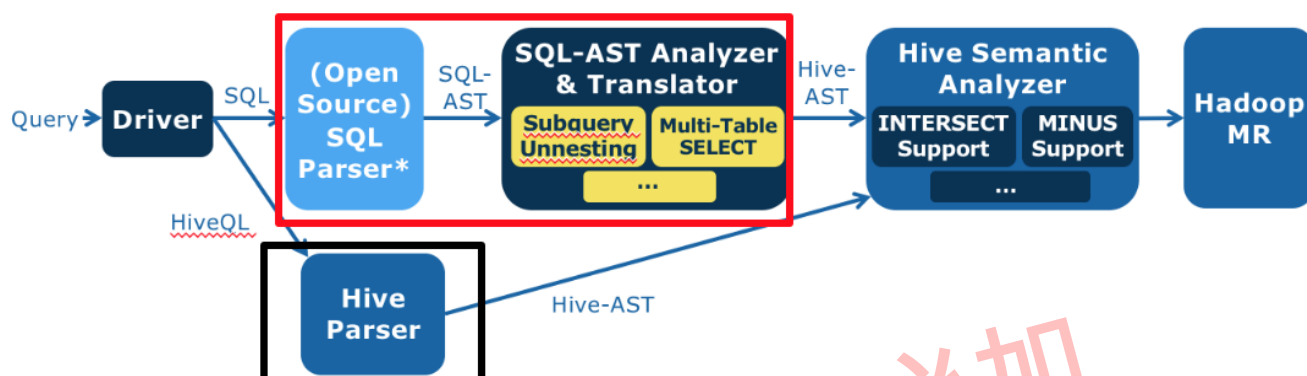
但是 Hive 也有自己的问题，由于它使用自己定义的 Hive QL 语法，这对已经熟悉 Oracle 等传统数据仓库的分析师来说，还是有一定的上手难度。特别是很多企业使用传统数据仓库进行数据分析已经由来已久，沉淀了大量的 SQL 语句，并且这些 SQL 经过多年的修改打磨，非常庞大也非常复杂。我曾经见过某银行的一条统计报表 SQL，打印出来足足两张 A4 纸。这样的 SQL 光是完全理解可能就要花很长时间，再转化成 Hive QL 就更加费力，还不说转化修改过程中可能引入的 bug。

2012 年那会，我还在 Intel 亚太研发中心大数据团队，当时团队决定开发一款能够支持标准数据库 SQL 的大数据仓库引擎，希望让那些在 Oracle 上运行良好的 SQL 可以直接运行在 Hadoop 上，而不需要重写成 Hive QL。这就是后来的[Panthera](#)项目。

在开发 Panthera 前，我们分析一下 Hive 的主要处理过程，大体上分成三步：

1. 将输入的 Hive QL 经过语法解析器转换成 Hive 抽象语法树 (Hive AST) 。
2. 将 Hive AST 经过语义分析器转换成 MapReduce 执行计划。
3. 将生成的 MapReduce 执行计划和 Hive 执行函数代码提交到 Hadoop 上执行。

Panthera 的设计思路是保留 Hive 语义分析器不动，替换 Hive 语法解析器，使其将标准 SQL 语句转换成 Hive 语义分析器能够处理的 Hive 抽象语法树。用图形来表示的话，是用红框内的部分代替黑框内原来 Hive 的部分。



红框内的组件我们重新开发过，浅蓝色的是我们使用的一个开源的 SQL 语法解析器，将标准 SQL 解析成标准 SQL 抽象语法树 (SQL AST)，后面深蓝色的就是团队自己开发的 SQL 抽象语法树分析与转换器，将 SQL AST 转换成 Hive AST。

那么标准 SQL 和 Hive QL 的差别在哪里呢？

标准 SQL 和 Hive QL 的差别主要有两个方面，一个是语法表达方式，Hive QL 语法和标准 SQL 语法略有不同；另一个是 Hive QL 支持的语法元素比标准 SQL 要少很多，比如，数据仓库领域主要的测试集 TPC-H 所有的 SQL 语句 Hive 都不支持。尤其是 Hive 不支持复杂的嵌套子查询，而对于数据仓库分析而言，嵌套子查询几乎是无处不在的。比如下面这样的 SQL，在 where 查询条件 exists 里面包含了另一条 SQL 语句。


复制代码

```
1 select o_orderpriority, count(*) as order_count
2 from orders
3 where o_orderdate >= date '[DATE]'
4 and o_orderdate < date '[DATE]' + interval '3' month
5 and exists
6 ( select * from lineitem
7 where l_orderkey = o_orderkey and l_commitdate < l_receiptdate )
8 group by o_orderpriority order by o_orderpriority;
```

所以开发支持标准 SQL 语法的 SQL 引擎的难点，就变成**如何将复杂的嵌套子查询消除掉**，也就是 where 条件里不包含 select。


SQL 的理论基础是关系代数，而关系代数的主要操作只有 5 种，分别是并、差、积、选择、投影。所有的 SQL 语句最后都能用这 5 种操作组合完成。而一个嵌套子查询可以等价转换成一个连接 (join) 操作。

比如这条 SQL

 复制代码

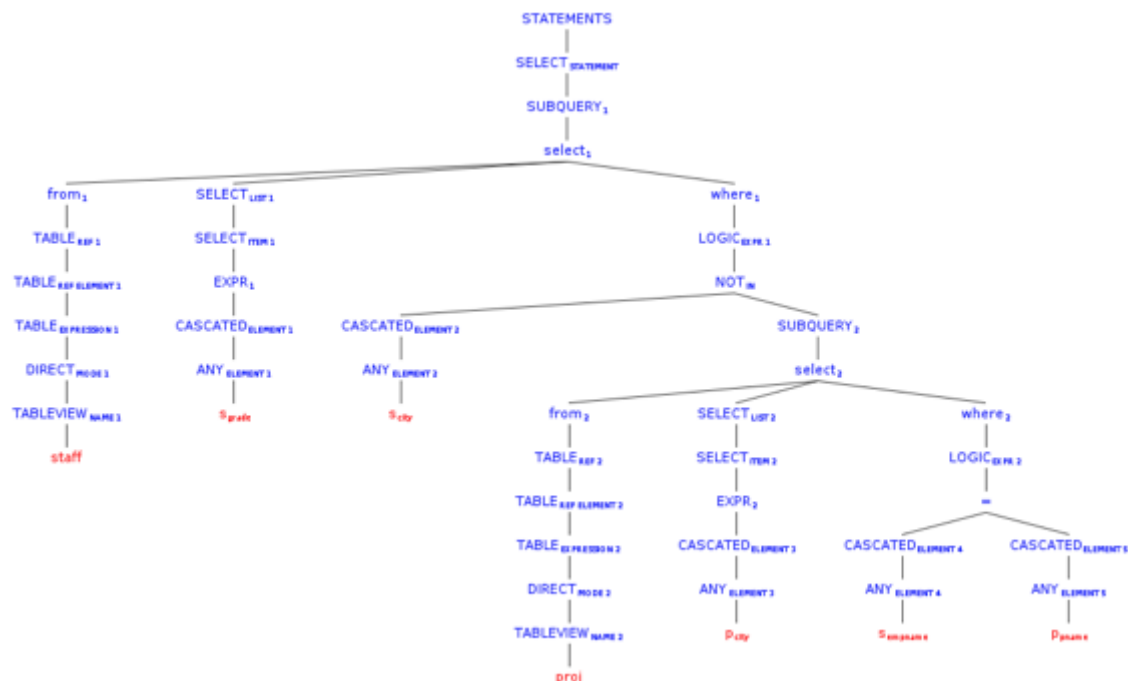
```
1 select s_grade from staff where s_city not in (select p_city from proj where s_empname=
```

这是一个在 where 条件里嵌套了 not in 子查询的 SQL 语句，它可以用 left outer join 和 left semi join 进行等价转换，示例如下，这是 Panthera 自动转换完成得到的等价 SQL。这条 SQL 语句不再包含嵌套子查询，

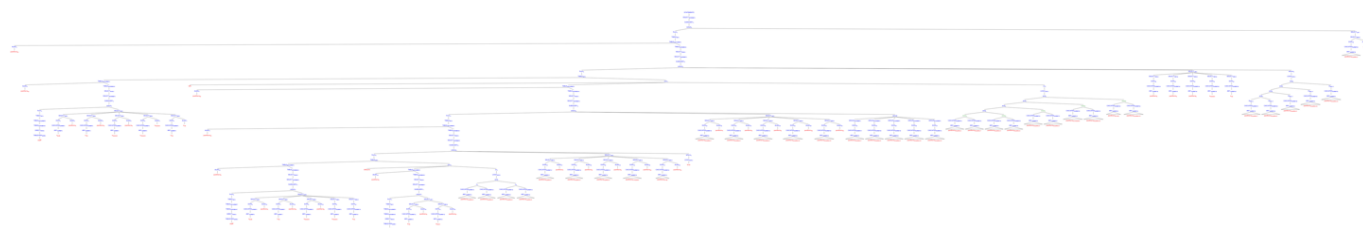
 复制代码

```
1 select panthera_10.panthera_1 as s_grade from (select panthera_1, panthera_4, panthera_6
```

通过可视化工具将上面两条 SQL 的语法树展示出来，是这样的。



这是原始的 SQL 抽象语法树。



这是等价转换后的抽象语法树，内容太多被压缩的无法看清，不过你可以感受一下（笑）。

那么，在程序设计上如何实现这样复杂的语法转换呢？当时 Panthera 项目组合使用了几种经典的设计模式，每个语法点被封装到一个类里去处理，每个类通常不过几十行代码，这样整个程序非常简单、清爽。如果在测试过程中遇到不支持的语法点，只需为这个语法点新增加一个类即可，团队协作与代码维护非常容易。

使用装饰模式的语法等价转换类的构造，Panthera 每增加一种新的语法转换能力，只需要开发一个新的 Transformer 类，然后添加到下面的构造函数代码里即可。

[复制代码](#)

```
1 private static SqlASTTransformer tf =
2     new RedundantSelectGroupItemTransformer(
3     new DistinctTransformer(
4     new GroupElementNormalizeTransformer(
5     new PrepareQueryInfoTransformer(
6     new OrderByTransformer(
7     new OrderByFunctionTransformer(
```




```

8      new MinusIntersectTransformer(
9      new PrepareQueryInfoTransformer(
10     new UnionTransformer(
11     new Leftsemi2LeftJoinTransformer(
12     new CountAsteriskPositionTransformer(
13     new FilterInwardTransformer(
14     //use leftJoin method to handle not exists for correlated
15     new CrossJoinTransformer(
16     new PrepareQueryInfoTransformer(
17     new SubQUnnestTransformer(
18     new PrepareFilterBlockTransformer(
19     new PrepareQueryInfoTransformer(
20     new TopLevelUnionTransformer(
21     new FilterBlockAdjustTransformer(
22     new PrepareFilterBlockTransformer(
23     new ExpandAsteriskTransformer(
24     new PrepareQueryInfoTransformer(
25     new CrossJoinTransformer(
26     new PrepareQueryInfoTransformer(
27     new ConditionStructTransformer(
28     new MultipleTableSelectTransformer(
29     new WhereConditionOptimizationTransformer(
30     new PrepareQueryInfoTransformer(
31     new InTransformer(
32     new TopLevelUnionTransformer(
33     new MinusIntersectTransformer(
34     new NaturalJoinTransformer(
35     new OrderByNotInSelectListTransformer(
36     new RowNumTransformer(
37     new BetweenTransformer(
38     new UsingTransformer(
39     new SchemaDotTableTransformer(
40     new NothingTransformer())))))))))))))))))))))))))))))))))));

```

而在具体的 Transformer 类中，则使用组合模式对抽象语法树 AST 进行遍历，以下为 Between 语法节点的遍历。我们看到使用组合模式进行树的遍历不需要用递归算法，因为递归的特性已经隐藏在树的结构里面了。

 复制代码

```

1  @Override
2  protected void transform(CommonTree tree, TranslateContext context) throws SqlXlateEx
3      tf.transformAST(tree, context);
4      trans(tree, context);
5  }
6
7  void trans(CommonTree tree, TranslateContext context) {

```

```

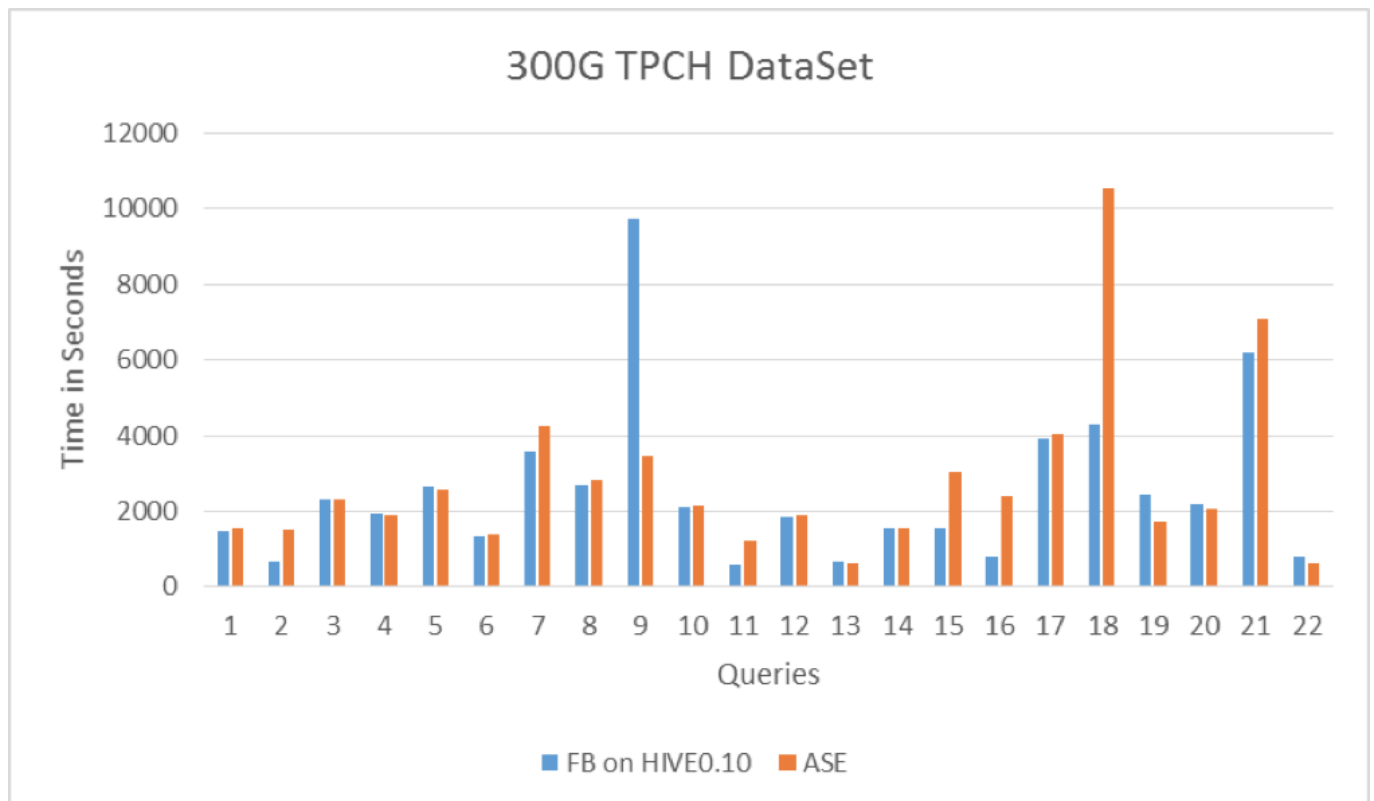
8    // deep firstly
9    for (int i = 0; i < tree.getChildCount(); i++) {
10       trans((CommonTree) (tree.getChild(i)), context);
11    }
12    if (tree.getType() == PantheraExpParser.SQL92_RESERVED_BETWEEN) {
13       transBetween(false, tree, context);
14    }
15    if (tree.getType() == PantheraExpParser.NOT_BETWEEN) {
16       transBetween(true, tree, context);
17    }
18 }

```

将等价转换后的抽象语法树 AST 再进一步转换成 Hive 格式的抽象语法树，就可以交给 Hive 的语义分析器去处理了，从而也就实现了对标准 SQL 的支持。

当时 Facebook 为了证明 Hive 对数据仓库的支持，Facebook 的工程师手工将 TPC-H 的测试 SQL 转换成 Hive QL，我们将这些手工 Hive QL 和 Panthera 进行对比测试，两者性能各有所长，总体上不相上下，这说明 Panthera 自动进行语法分析和转换的效率还是不错的。

Panthera (ASE) 和 Facebook 手工 Hive QL 对比测试结果如下。



事实上，标准 SQL 语法集的语法点非常多，我们经过近两年的努力，绞尽脑汁进行各种关系代数等价变形，依然没有全部适配所有的标准 SQL 语法。

我在开发 Panthera 的时候，查阅了很多关于 SQL 和数据库的网站和文档，发现在我们耳熟能详的那些主流数据库之外还有很多名不见经传的数据库，此外，还有大量的关于 SQL 的语法解析器、测试数据和脚本集、各种周边工具。我们经常看到的 MySQL、Oracle 这些产品仅仅是整个数据库生态体系的冰山一角。还有很多优秀的数据库在竞争中落了下风、默默无闻，而更多的支撑起这些优秀数据库的论文、工具，非业内人士几乎闻所未闻。

这个认识给了我很大触动，我一直期待我们中国人能开发出自己的操作系统、数据库、编程语言，也看到有很多人前仆后继投入其中。但是这么多年过去了，大部分努力惨淡收场，小部分结果沦落为笑柄，成为人们饭后的谈资。我曾经思考过，为什么会这样？

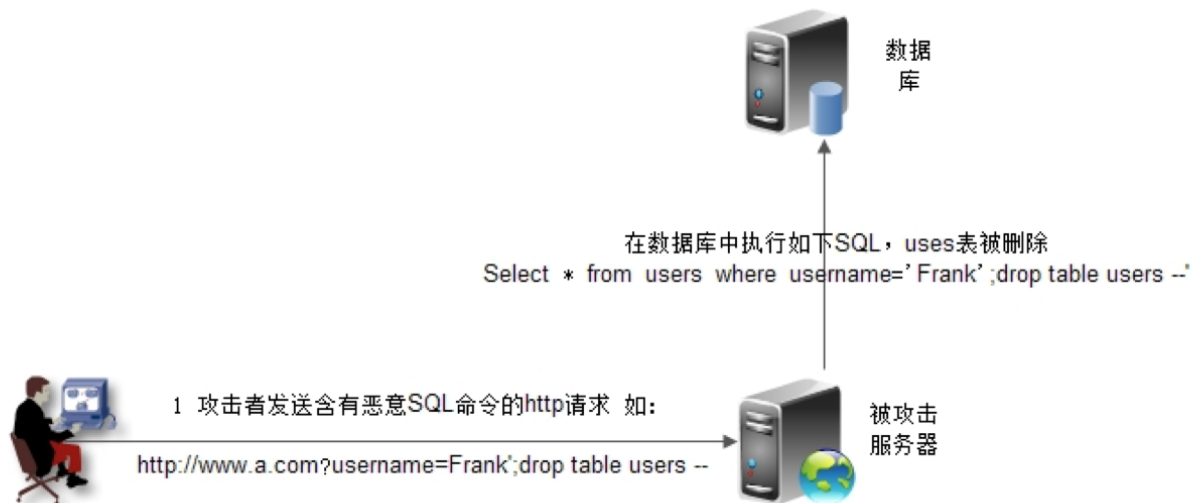
开发 Panthera 之后，我想，我们国家虽然从事软件开发的人数很多，但是绝大多数都在做最顶层的应用开发，底层技术开发和研究的人太少，做出来的成果也太少。我们在一个缺乏周边生态体系、没有足够的竞争产品的情况下，就想直接开发出自己影响力的操作系统、数据库、编程语言，无异于想在沙漠里种出几棵参天大树。

不过，庆幸的是，我也看到越来越多有全球影响力的底层技术产品中出现中国人的身影，小草已经在默默生长，假以时日，料想必有大树出现。

今天我讲的是一个 SQL 引擎是如何设计出来的，也许在你的工作几乎不可能去开发 SQL 引擎，但是了解这些基础的知识，了解一些设计的技巧，对你用好数据库，开发更加灵活、有弹性的系统也会很有帮助。

思考题

SQL 注入是一种常见的 Web 攻击手段，如下图所示，攻击者在 HTTP 请求中注入恶意 SQL 命令（`drop table users;`），服务器用请求参数构造数据库 SQL 命令时，恶意 SQL 被一起构造，并在数据库中执行。



但是 JDBC 的 PreparedStatement 可以阻止 SQL 注入攻击，MyBatis 之类的 ORM 框架也可以阻止 SQL 注入，请从数据库引擎的工作机制解释 PreparedStatement 和 MyBatis 的防注入攻击的原理。

欢迎你点击“请朋友读”，把今天的文章分享给好友。也欢迎你写下自己的思考或疑问，与我和其他同学一起讨论。



从 0 开始学大数据

智能时代你的大数据第一课

李智慧

同程艺龙交通首席架构师
前 Intel 大数据架构师



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言 (15)

写留言



三木子

2018-12-09

12

Oracle会有走向衰落吗？

展开

作者回复：会，正在走向



欧嘉权Feli...

2018-12-08

8

如果用statement jdbc会简单拼接字符串然后作为sql执行
preparedstatement就会进行预编译 对其中的换行符等字符做转义 对注入的sql会起到混淆的作用

mybatis这些orm框架也是基于preparedstatement mybatis尽量使用#占位符

展开



纯洁的憎恶

2018-12-17

7

基于SQL的大数据仓库引擎panthera的核心任务是把SQL语义与Hive AST对应起来。难点是SQL的语义远比Hive AST丰富，而幸运的事SQL丰富的表意逻辑主要源于它的嵌套子语句，这在Hive AST中是不存在的。但是SQL的嵌套子语句可以等价于若干join操作。

为了在工程上降低实现难度，特意为每个语法点设计一个对象（类），这就将复杂问题...

展开

作者回复：清晰



杰之7

3



2018-12-09

从这篇文章开始，大数据技术的体系架构知识就告一段落，但并不意味着已经完全掌握，我需要不断的回顾体系架构原理，这样才会有可能沿着这个思路有自己的一点拓展。回到本篇内容，如何自己开发SQL引擎，由于Hive语法元素少于标准SQL,且不支持复杂的嵌套子查询，所以开发的难点就是如何将复杂嵌套消除转化成标准SQL。在老师讲述的过程中，主要通过装饰模式等价转化类的构造，对于每一种新的语法通过开发新的...

展开 ▾



galen

2018-12-08

👍 3

因为SQL语句在程序运行前已经进行了预编译，在程序运行时第一次操作数据库之前，SQL语句已经被数据库分析，编译和优化，对应的执行计划也会缓存下来并允许数据库已参数化的形式进行查询，当运行时动态地把参数传给PreparedStatement时，即使参数里有敏感字符如 or '1=1'也数据库会作为一个参数一个字段的属性值来处理而不会作为一个SQL指令，如此，就起到了SQL注入的作用了

展开 ▾



明亮

2018-12-20

👍 2

可以推荐一些SQL解析的来源产品和资料吗

展开 ▾



Well_Ksun

2018-12-17

👍 2

老师能简单聊聊presto吗？也是facebook开源出来的。据说AWS 的 Athena 就是基于Presto 的产品。



周翔

2018-12-09

👍 2

预编译就是sql命令的模板归模板，参数归参数，参数就不会混合到原sql，改变原有的sql语句和执行逻辑。除了数据访问框架本身prepare statement功能，还可以从服务层接收参数就可以屏蔽。

展开 ▾



~



Oliver-08



保证用户输入的内容仅仅且只能作为查询条件，涉及一些字符转义操作

展开 ∨



cwx0220

2019-03-04



正常情况下，当http请求参数带有sql语句的条件时，在 SQL 注入中，参数值作为 SQL 指令的一部分，会被数据库进行编译/解释执行。当使用了 PreparedStatement，带占位符 (?) 的 sql 语句只会被编译一次，之后执行只是将占位符替换为参数值，并不会再次编译/解释，因此从根本上防止了 SQL 注入问题。

展开 ∨



程序员小灰

2018-12-08



PreparedStatement会先初始化SQL，语句中只有部分变量可以被 “?” 替代，所以sql注入的攻击方式无效。



北京知府

2019-05-12



```
private static SqlASTTransformer tf =  
    new RedundantSelectGroupItemTransformer(  
        new DistinctTransformer(  
            new GroupElementNormalizeTransformer(  
                new PrepareQueryInfoTransformer(...
```

展开 ∨

作者回复: 事实上是非常简单~~ 将非常复杂的SQL语法转换解耦合，有兴趣建议看看更详细的代码



小老鼠

2019-01-17



Hive QL也支持预编译吗?

展开 ∨



桃园悠然在

2018-12-29



请问老师，如果目前手头使用的hive可以支持嵌套子查询，是否说明已经定制优化过了？



反应慢

2018-12-14



之前有一段时间在学习函数式编程，和sql一样属于命令式编程，所以感觉以函数去解析sql是水到渠成的事情。不过至今没有动手去实现一下，比较遗憾

作者回复: 赞，spark也往这个方向走。

