|  |  |
| --- | --- |
| 交底书名称 | 一种求解spark sql语句的覆盖集的方法 |
| 技术联系人姓名 | 马殿军 |
| 技术联系人电话 | 13269325833 |
| 技术联系人Email | madianjun@jd.com |

（技术联系人信息用于与外部代理沟通，发明人信息在ERP专利申请系统中填写）

注意事项：

1、代理人并不是技术专家，交底书要使代理人能看懂，尤其是完整技术方案，一定要写得全面、清楚。

2、在后续与专利代理人进行沟通时，对于代理人的疑问应认真讲解，要求补充的材料应及时补充（禁止通过私人邮箱与代理人沟通）。

3、常用检索网站：www.soopat.com（SOOPAT），http://so.baiten.cn/（佰腾），patents.google.com（谷歌专利）。

# 1. 现有技术

/\* 应记载某个应用场景或者解决某个技术问题当前所采用的技术，可以概述该技术，也可以仅给出参考文献的链接或相关专利号。

在某些对sql语句作分析的场景下，需要求出能代表大量sql语句的一个较小的sql语句集合，该集合中语句的特征覆盖了所有语句的特征，称该集合为“sql语句覆盖集”（简称覆盖集）。举一个示例场景：当需要把hive sql语句迁移到spark sql平台时，迁移之前要对语句的执行结果作校验；如果语句数量巨大，则校验工作量也巨大，所以可以求一个覆盖集，只校验该覆盖集中的语句，如果覆盖集中的语句校验都成功，则表示原始语句的校验也是成功的。

目前已有的求解覆盖集的通用算法可参考：<https://www.geeksforgeeks.org/set-cover-problem-set-1-greedy-approximate-algorithm/>，本专利是针对spark计算引擎的sql语句分析场景，将算法作适当修改，以求解出sql语句的覆盖集。

# 2. 现有技术的缺点

/\* 需要指出现有技术存在的缺点，本发明也不能克服的缺点无需提供。

/\* 应根据现有技术的实现过程，有针对性地说明缺点产生的原因。

现有的算法没有针对spark sql语句的处理方法，本专利的解决对象只是spark sql。

# 3. 本发明技术方案

3.1 本发明所要解决的技术问题（即发明目的）

/\* 描述本发明所要解决的技术问题，与“2.现有技术的缺点”部分指出的缺点相对应。

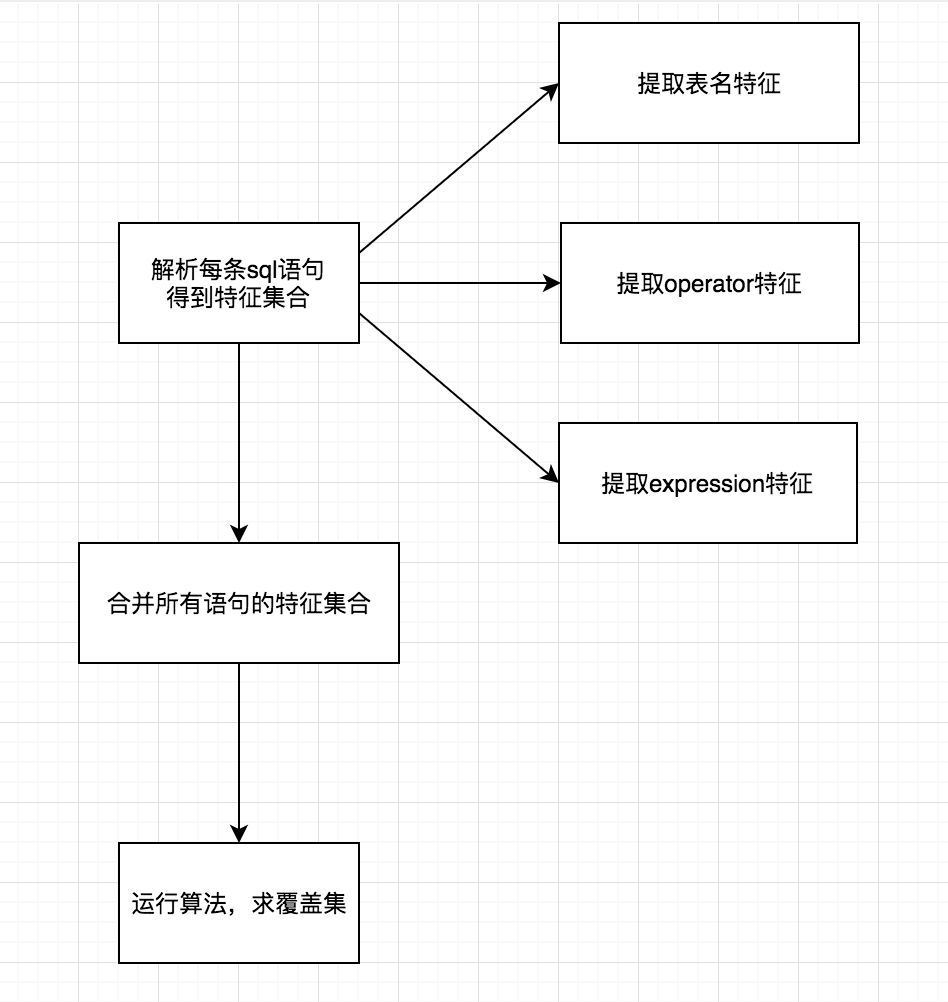
本发明主要解决的是对一个较大的spark sql语句集合作特征提取，运行一个算法求出能覆盖该集合的一个较小的集合，这个小集合可以用在后续的各种sql语句分析工作中，从而大大减少sql语句分析的工作量。

3.2 本发明的完整技术方案的详细阐述

/\* 这是本文档最重要的部分，需要详细完整的阐述，不能光有原理，也不能仅有功能性介绍或操作说明。

/\* 在描述具体的技术方案时，必须结合附图（方法型专利按照数据流向或实现步骤抽象框图，装置型专利按照组成部件抽象框图）进行说明，每个附图都应当有对应的文字描述。如果本发明方案包含多个主题，方法与装置等，则需要分别进行描述。

1. 流程



1. 解析sql语句：调用spark sql的api接口，得到逻辑查询计划，它表示的是一个树型结构，其中包含了三部分内容：表名、operator(算子)、expression(表达式)，其中operator节点和expression节点都具有各自的父子关系，把这种父子关系作为特征。这样，每条sql语句都得到一个自己的特征集合。
2. 合并所有sql语句的特征集合，得到一个特征全集。
3. 在特征全集上运行求覆盖集的算法，得到sql语句的覆盖集。

以下是一个sql语句的解析示例。

语句为：

INSERT OVERWRITE TABLE table1 PARTITION (p, q)

SELECT a, COUNT(DISTINCT(b))

FROM table2

WHERE c = 'final\_base\_details'

and d >= 10

and d <= 50

and e not in ('X', 'Y')

GROUP BY a

表名为：table2

Operator节点树为：

-- InsertIntoHiveTable

---- Aggregate

------ Project

-------- Filter

---------- HiveTableRelation

Expression节点树为（一般每个operator都有一棵expression树）：

InsertIntoHiveTable：

( InsertIntoHiveTable 0 )

Aggregate：

( Aggregate 2 )

-- ( aggregateExpressions 2 )

---- ( AttributeReference 0 )

---- ( Alias 1 )

------ ( Cast 1 )

-------- ( AggregateExpression 1 )

---------- ( Count 1 )

------------ ( AttributeReference 0 )

-- ( groupingExpressions 1 )

---- ( AttributeReference 0 )

Project：

( Project 0 )

Filter：

( Filter 1 )

-- ( And 2 )

---- ( And 2 )

------ ( And 2 )

-------- ( And 2 )

---------- ( And 2 )

------------ ( IsNotNull 1 )

-------------- ( AttributeReference 0 )

------------ ( IsNotNull 1 )

-------------- ( AttributeReference 0 )

---------- ( EqualTo 2 )

------------ ( AttributeReference 0 )

------------ ( Literal 0 )

-------- ( GreaterThanOrEqual 2 )

---------- ( AttributeReference 0 )

---------- ( Literal 0 )

------ ( LessThanOrEqual 2 )

-------- ( AttributeReference 0 )

-------- ( Literal 0 )

---- ( Not 1 )

------ ( In 2 )

-------- ( AttributeReference 0 )

-------- ( Literal 0 )

-------- ( Literal 0 )

HiveTableRelation：没有expression树。

该语句的特征集为：

{

// 表名特征：

('table2', 'TABLE'),

// operator特征：

('Aggregate','Project'), ('Project','Filter'), ('InsertIntoHiveTable','Aggregate'), ('Filter','HiveTableRelation'),

// expression特征：

('And', 'And'), ('groupingExpressions', 'AttributeReference'), ('EqualTo', 'Literal'),

('EqualTo', 'AttributeReference'), ('Aggregate', 'groupingExpressions'), ('GreaterThanOrEqual', 'Literal'),

('Alias', 'Literal'), ('And', 'Not'), ('And', 'LessThanOrEqual'), ('Filter', 'And'), ('Not', 'In'),

('And', 'GreaterThanOrEqual'), ('Alias', 'AttributeReference'), ('GreaterThanOrEqual', 'AttributeReference'),

('LessThanOrEqual', 'Literal'), ('Aggregate', 'aggregateExpressions'), ('LessThanOrEqual', 'AttributeReference'),

('In', 'Literal'), ('aggregateExpressions', 'AttributeReference'), ('IsNotNull', 'AttributeReference'),

('And', 'EqualTo'), ('And', 'IsNotNull'), ('aggregateExpressions', 'Alias'), ('In', 'AttributeReference'),

('Count', 'AttributeReference'), ('Alias', 'Cast'), ('Cast', 'AggregateExpression'),

('AggregateExpression', 'Count')

}

1. 覆盖集算法

定义以下变量：

：语句的索引，值为从1到n。

：第i条语句的特征集。

：的列表，是按大小的降序排列。

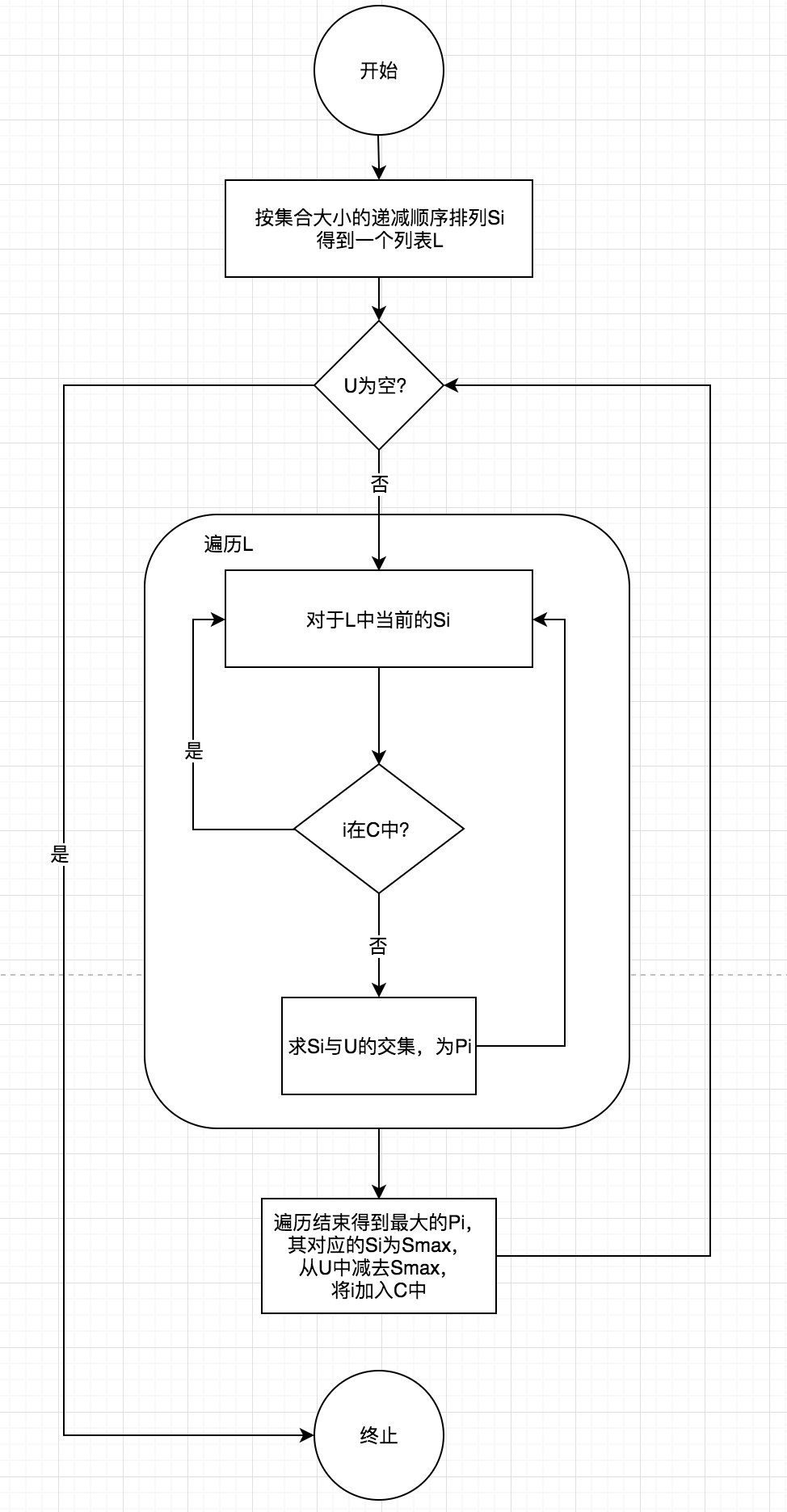
：所有语句的特征集的并集，即：。

：每一次迭代中与的交集。

：使得最大的，即：)。

：所求的覆盖集，是i的集合。

算法流程：



1. 将按集合大小降序排列，得到一个列表。
2. 如果不为空，则遍历中每个；如果为空，则算法终止。
3. 对于中当前的，如果i已经在中，则遍历下一个；否则，求与的交集为。
4. 按第(3)步完成对的遍历，求出最大的，并记录它对应的记为。
5. 从中减去，将的语句索引加入中；回到第(2)步。

每一次迭代，都会得到一个,它的语句就属于最终的覆盖集；同时也会将减小，直到变为空集，完成对的覆盖。

3.3 本发明希望保护的技术创新点

/\* 指出技术方案中希望保护的技术关键点，并概括说明该关键点的技术原理。

本发明的技术创新点是根据现有的求覆盖集的通用算法进行改进，使其能够应用在对spark sql的分析场景中，该算法求出的覆盖集将是原始集合的一个很小的子集，大大减少了后期对sql语句的分析和测试工作。

3.4 针对3.3中的技术方案，是否还有别的替代方案同样能完成发明目的？

/\* 替代方案可以是完整技术方案的替代，也可以是部分结构或者步骤的替代。

3.5交底书中技术术语的名词解释

/\* 记载交底书中出现的专业技术术语、缩写、外文的解释。

spark：大规模数据处理计算引擎。