FabricSQL：区块链数据的关系查询

余 涛，牛保宁+，樊 星

（太原理工大学 信息与计算机学院，山西 晋中 030600）

摘 要：为解决区块链在查询方面存在的访问方法支持有限、查询功能单一、查询效率低等不足，提出一种区块链数据的关系查询解决方案 FabricSQL，将 Hyperledger Fabric 与 MySQL 结合，把实时存储的区块链有效交易数据同步存储到 SQL 数据库中。为防止数据库中的数据被篡改，提出一种交易数据存储验证机制，在 SQL 数据存储时使用 Salt 加密方法，在数据顺序追加时引入区块链前哈希的思想防止作恶，在用户访问时，进行数据的一致性校验，返回需求数据和校验结果，保证数据的安全性。在设计的实验模型和基于 Fabric 的区块链系统上进行多项查询性能的对比实验，实验结果表明，FabricSQL 同时拥有区块链和关系型数据库的良好特性。

关键词：区块链；关系查询；超级账本；关系型数据库；存储验证机制

中图法分类号：TP31 **文献标识号**：A **文章编号**：1000-7024（2020）10-2988-08

doi：10.16208/j.issn1000-7024.2020.10.047

FabricSQL：Relational query of blockchain data

YU Tao，NIU Bao-ning+，FAN Xing

（College of Information and Computer，Taiyuan University of Technology，Jinzhong 030600，China）

Abstract：To solve the problems of blockchain in terms of query，including limited access method support，single query function and low query efficiency，FabricSQL was proposed as a relational query solution of blockchain data. Hyperledger Fabric was combined with MySQL to conduct real-time storage through blockchain，and the effective transaction data were synchronously stored in SQL database. To prevent the data in the database from being tampered with，a transaction data storage verification mechanism was proposed. When SQL data were stored，the Salt encryption method and prehash were introduced to restrict the evil. When the user queried，the consistency of the data was verified，and the data and the result of verification were returned，to ensure the security of the data. A comparison of multiple query performances on experimental models and fabric-based blockchain systems shows that FabricSQL has good features of both blockchain and relational database.

Key words：blockchain；relational query；Hyperledger Fabric；MySQL；storage verification mechanism

0 引 言

从中本聪描绘的比特币[1]，到 Vitalik Buterin 提出以太坊[2]，再到 Linux 基金会发起的 Hyperledger 开源区块链项目[3]，区块链技术逐步成熟和普及。Hyperledger Fabric 与其它区块链项目的技术类似，是一个使用智能合约、通过所有参与者共同管理交易的账本系统，其主要的特点是私有和许可，因此更适用于企业级区块链项目的开发。基于 Fabric 的区块链系统，能有效解决目前以中心化模式为主的数据管理存在的问题，克服中心化系统的弊端，回避人为作恶的风险，使链上的参与方集体认定、共同维护同一账本[4]。

区块链的设计目的是在不可信、分布式的环境中进行安全、不可变的价值转移。区块链的本质可以理解为去中心化的数据库，但是如果用传统的数据库标准来衡量区块链，不管是在吞吐量、事务延迟、容量还是查询能力上，区块链的效果均显不佳[5]。而区块链的广泛应用必然会要求对区块链数据进行高效查询。目前，区块链查询方面的研究大致有两类：一类是通过区分记录节点能力、优化查询路径解决泛洪循环和响应消息风暴等问题[6,7]，这样做并

收稿日期：2019-07-31；**修订日期**：2019-10-09

基金项目：国家重点研发计划子课题基金项目（2017YFB1401001-01）；国家自然科学基金面上基金项目（61572345）

作者简介：余涛（1993-），女，山西怀仁人，硕士研究生，研究方向为区块链数据管理；+通讯作者：牛保宁（1964-），男，山西太原人，教授，博士生导师，CCF 高级会员，研究方向为大数据管理与分析、数据库系统性能管理、空间数据管理、多媒体数据管理、区块链数据管理等；樊星（1992-），女，山西太原人，博士研究生，CCF 学生会员，研究方向为区块链数据管理。E-mail：350497340@qq.com

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

不能丰富区块链的查询功能；另一类是通过与数据库连接，利用数据库系统的查询能力，丰富查询功能，实现区块链数据的高效查询[5,8-11]，但是，这一途径存在数据库中数据可篡改的问题。针对后一途径存在的问题，本文将区块链与关系型数据库连接后，增加交易验证机制，在保证数据不可篡改特性的前提下，实现区块链丰富的查询功能和高效的查询。

本文的主要贡献如下：

（1）针对区块链查询方面的不足，提出一种区块链数据的关系查询解决方案 FabricSQL，将区块链上的有效交易同步至 SQL 数据库中，丰富区块链系统的查询功能，优化查询方法；

（2）为增强用户查询的准确性，减少不必要的空间占用，通过侦听每笔交易的状态，实时同步区块链上有效交易的信息，并将交易数据转为 SQL 数据库要求的格式存储；

（3）为保证 SQL 数据库的数据安全问题，提出一种交易数据存储验证机制，每笔交易数据对应一个哈希值，在存储时生成，在访问时验证，保证交易数据的安全性。

在设计的实验模型 FabricSQL 和 Hyperledger Fabric 区块链系统上，进行多组查询性能的对比实验，结果表明 FabricSQL 在保证集体验证维护、数据可回溯以及防篡改特性的同时，兼备关系型数据库快速查询、查询功能丰富的特性。

1 相关工作

随着区块链技术不断发展应用，人们对于区块链上数据查找速度和查找功能的要求随之增加。针对区块链查询的研究，目前大致分为两个方向：一是通过区分记录或赋予节点不同的查询能力，优化查询路径，实现区块数据的有效查询；二是将区块链系统与数据库结合，实现查询功能优化。

首先，在第一种研究中，贾大宇等[6,7]在 ElasticChain 模型基础上，提出一种新的容量可扩展区块链系统的高效查询方法 ElasticQM，在用户层、查询层、存储层和数据层实现创新优化，通过在用户层设置数据缓存模块，在查询层增加超级节点、查询叶子节点和查询验证节点 3 种角色，在数据层结合平衡二叉树和梅克尔树的各自特点建立基于 B-M 树的区块链存储结构，提高查询效率。

Tuan 等[12]提出了 Blockbench，可以对集成的私有区块链系统的整体和组件的性能进行评估，并帮助他们识别和改进性能瓶颈。另外，Blockbench 对 3 种主要的区块链系统：Ethereum，Parity 和 Hyperledger 进行全面分析，结果表明，这些系统在传统的数据处理工作负载中还远不能取代现有的数据库系统。

在第二种研究中，McConaghy 等[5]发布的 BigchainDB 是一个区块链数据库，其设计是从分布式数据库开始，使 BigchainDB 继承现代分布式数据库高吞吐量、低延迟、大容量、丰富的查询功能等特征的同时，通过增加区块链去中心化、不可篡改和数字资产创建以及传输的特性，使它具有数据库和区块链的优良属性。2018 年 BigchainDB 2.0[8]发布，网络中的每个节点都有自己本地 MongoDB 数据库，并且引入拜占庭容错机制，允许网络中即使有 1/3 的节点失败，也能够达成一致意见继续进行。Bartoletti 等[9]提出区块链分析的通用框架，支持对比特币和以太坊两种加密货币进行数据分析，框架允许将相关的区块链数据与从外部源检索的数据集成在一起，并将它们组织到一个数据库中，再用数据库管理系统的查询语言进行分析，文献通过一系列用例包括分析区块链元数据、按日期划分的交易数量、交易费和地址标签说明框架支持的不同功能。Li Yang 等[10]提出 EtherQL，是为 Ethereum 开发的一个高效查询层，由同步管理器、链处理、持久化框架和开发者接口 4 个模块组成。EtherQL 通过内置的 Ethereum 客户端实时同步来自 Ethereum 公共网络的区块链数据，并将其存储在 MongoDB 数据库中，提供的接口支持扩展查询、范围查询和 top-k 查询等分析查询。北京众享比特科技有限公司提出 ChainSQL[11]，通过构建 Ripple 与普通数据库结合的区块链数据库，实现先入库再共识的方法，将操作以交易的形式记录在区块链网络中，而真实数据在数据库中查看，如果共识不能通过，则回滚数据库操作，增加数据入库的速度。

本文所提出的 FabricSQL 方法与其它研究相比，是针对联盟链 Fabric 实现的一种区块链数据关系查询解决方案，且设计相应的模块解决其它研究中未能解决的数据库中数据易被篡改的问题。

2 Fabric 系统

Hyperledger Fabric 在一个仅追加复制的账本数据结构中安全地跟踪其执行历史，没有内置的加密货币，且具有高度的模块化，将共识、执行和验证过程完全分离[13]。在 Fabric 网络中，节点是通信的主体，包括客户端节点、CA 节点、peer 节点和排序服务节点，链上的节点共同维护同一账本。

Hyperledger Fabric 交易流程如图 1 所示，交易是由客户端发起的，客户端通过 SDK 调用 CA 服务，交易提案经 SDK 创建后，发送给背书节点，背书节点验证签名并确定提交者是否有权执行操作，同时调用链码并将链码的模拟执行结果等提案响应返回给客户端，客户端判断执行结果是否一致以及背书策略是否满足，再将交易提案、模拟执行结果、背书信息封装为一个交易，签名后一并发给排序服务节点，排序服务节点将交易排序并将生成的区块发送至提交节点，提交节点验证交易的有效性，通过验证的区

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

块会被追加到区块链上，与之相关的状态数据库也会进行更新。

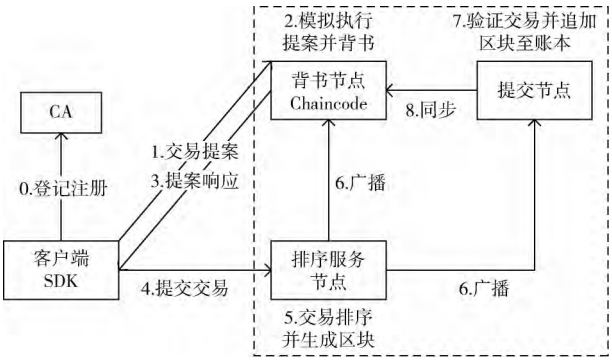

图 1 Hyperledger Fabric 交易流程

Hyperledger Fabric 的存储系统由普通的文件和 key-value 数据库组成。每个区块的区块头和所有交易数据以二进制形式写入 blockfile 文件中，并将区块和交易在 blockfile 中的索引以 key-value 的形式保存。历史数据和区块索引均存储在 LevelDB 数据库中。而状态数据库记录最新的交易执行结果，是所有状态的最新值，支持 LevelDB 和 CouchDB 数据库。在查询账本数据时，从索引数据库中获得文件位置指针，再通过文件位置指针获取所需的数据。

Hyperledger Fabric 中包括账本数据、索引数据、历史数据和状态数据等，各种数据的存储方式不同，即便是基于键值对的存储方式其操作管理的方法也大不相同，在区块提交时，各类存储操作顺序执行，极有可能发生中断或出错等情况。而且由于数据的存储结构受限导致区块链仅支持简单和有限的查询。在用户提出查询请求后，只能通过预先定义的 key 进行查询，难以实现复杂的查询功能，无法满足用户丰富的需求。

从应用开发者角度考虑，区块链没有标准的查询语言，如果开发人员对区块链的内部数据组织方式、存储结构没有较好的理解，将增大数据查询的难度。当用户的需求发生变更时，需要更改链码操作。

3 FabricSQL 系统

本章介绍 FabricSQL 具体细节。3.1 节讨论提出 FabricSQL 的思路，3.2 节给出 FabricSQL 的总体架构，3.3 节、3.4 节分别讨论重要模块的技术细节。

3.1 问题与解决思路

针对区块链查询方面存在的不足，本文将 Fabric 与关系型数据库 MySQL 结合，提出区块链数据的关系查询解决方案 FabricSQL，从而解决 Fabric 中各类数据存储方式复杂、查询功能有限以及需求变更难以实现等问题。但是，将数据从区块链同步存储至数据库的同时，需要解决下面的问题：首先，区块链中存在的无效交易如果直接同步至数据库中，会对查询结果的正确性产生干扰，这里的无效交易是指当区块发送至提交节点后，提交节点会按顺序对块中的交易进行背书策略和读写冲突检查，将不满足背书策略和版本不匹配的交易标记为无效，但是所有有效和无效的交易最终都会追加至区块链上；其次，由于数据库本身安全能力有限，在使用过程中会不可避免暴露数据接口，被外部作恶者通过技术手段篡改数据的案例比比皆是，同时在使用过程中管理员的权限不可控制，存在着不信任因素，因此数据库容易被作恶者篡改的情况，会对数据库中数据的安全性产生影响。

针对上述问题，首先通过增加对交易状态的侦听，将有效交易同步存储至数据库中，从而保证存储数据的正确性。另外提出交易数据的存储验证机制，在 MySQL 数据库中，存储和验证数据时运用加盐思想，即在经过哈希后的每笔交易数据的特定位置增加特定的字符串 Salt，从而改变原始的字符串，实现数据加密，这样即使知道原有的交易信息，在不知道 Salt 的情况下，也会增加一定的破解负担；同时借鉴区块链前哈希的思想，将交易生成的哈希值依序咬合，生成最终的交易哈希值存储在数据库中。当用户发出数据请求时，首先完成对交易数据的一致性校验，然后返回给用户数据和校验结果，从而保证数据的安全性。

3.2 总体架构

FabricSQL 由应用层、网络层、数据处理层和存储层 4 个层次构成。系统架构图如图 2 所示。

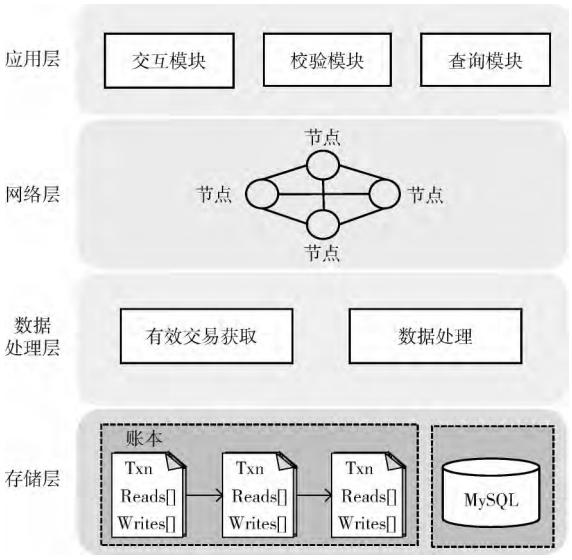

图 2 FabricSQL 系统架构

应用层负责与区块链网络交互和查询逻辑的实现，包括交互模块、查询模块和校验模块。交互模块包含与区块链网络交互的 SDK 程序在内的应用程序。查询模块在收到用户查询申请后，与存储层交互。校验模块对查询模块的返回值进行一致性校验，通过验证的结果返回给交互模块。

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

网络层由排序服务节点、peer 节点、CA 节点构成。由于区块链数据的关系查询模型 FabricSQL 是在 Hyperledger Fabric 系统的基础上实现的，所以，网络层的结构保持不变。

数据处理层包括有效交易获取模块和数据处理模块，负责组织区块链上的有效交易数据，并转换为可存储在 SQL 数据库中的关系型数据。

存储层由两部分组成，分别是存储区块链信息的普通文件和存储由数据处理层输出的交易数据的 MySQL 数据库，它们存储着验证通过的每笔交易信息，不同点是，SQL 数据库只存储有效交易。

3.3　交易数据同步处理

FabricSQL 的交易数据同步处理模块在对区块链有效交易信息判断、同步和提取后，处理成 SQL 数据库要求的格式存储。FabricSQL 系统模型如图 3 所示，一条设备资产的交易信息在区块链网络中经过背书、打包、成块和验证后写入区块链中，其数据源是用户提交的原始数据，经过 SDK 程序，实现对 Fabric-CA 节点和 Fabric 网络的访问。在完成区块上链后，Fabric 会发出特定的 event 来协助客户端程序，客户端通过 SDK 捕获 event 事件，获取交易信息的侦听状态，并将其提供给数据处理模块处理其中的有效交易，最终将有效交易的数据存储在 MySQL 数据库中。

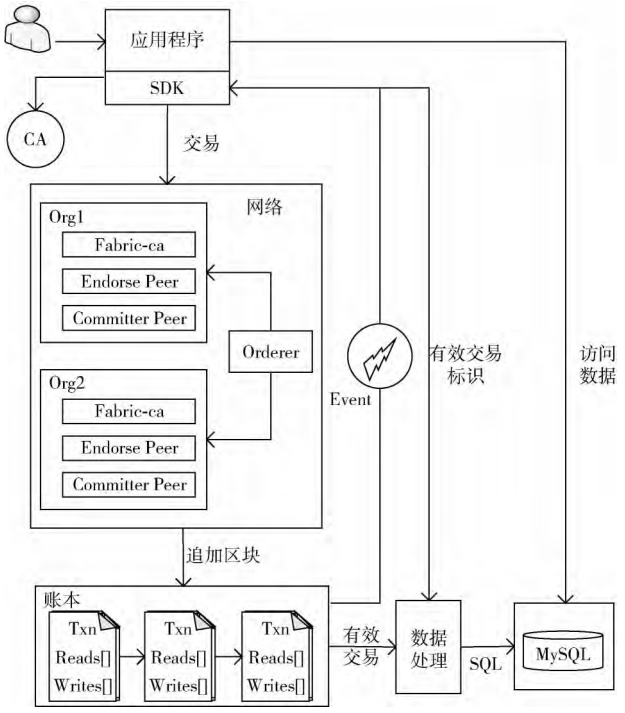

图 3　FabricSQL 系统模型

FabricSQL 在区块链网络中的交易过程如算法 1 所示，首先判断用户的身份证书，确定用户注册信息，然后连接区块链网络，当交易被认可，成功发送给排序服务节点前，需要注册侦听器。侦听器存在的目的是检测交易状态，因为区块链上的交易数据包括有效交易和无效交易，如果直接将区块链上的所有交易数据直接同步至 SQL 数据库中，不仅会增加存储负担，还会对查询结果的正确性产生直接影响。最后，当包含交易的块被提交至区块链时，已注册的侦听器会被触发。

算法 1：FabricSQL 在区块链网络中的交易过程

输入：用户提交的原始交易数据

输出：交易上链，触发侦听器

```
(1) try {
(2)     wallet = new FileSystemWallet (walletPath)
(3)     if (! wallet.exists (user)) {
(4)         return }
(5)     连接区块链网络，加入通道，调用链码
(6)     transaction=
        contract.createTransaction ('transferDevice')
(7)     //完成侦听器的注册功能
(8)     transaction.addCommitListener ( {callback
        function: Data2SQL})
(9)     transaction.submit (transaction args)
(10)    断开区块链网络连接}
(11) catch {异常处理}
```

数据处理层，在侦听器触发后，执行回调函数 Data2SQL，通过对交易状态分析，返回所需的目标交易。所以，本模块以区块中的每笔交易作为一个处理单元，将有效交易的信息进行提取、分解，处理成不同的数据格式传递至后续的存储层模块。FabricSQL 的数据处理过程如算法 2 所示。

算法 2：FabricSQL 数据处理过程

输入：链上的每笔交易数据

输出：每笔有效交易数据

```
(1) function Data2SQL (error, transId, status, block-
Number) {
(2)     if status ! = VALID:
(3)         return
(4)     result = 由 transId 定位的交易信息
(5)     obj = JSON.parse ( result.transactionEnve-
        lope.payload.data)
(6)     trans _ time = result.transactionEnvelope.pay-
        load.Header.channel _ header.timestamp
(7)     data=obj ['senderID','receiverID','trans _
        time','deviceID','transId', 'blockNumber']
(8)     insertData (data)
(9)}
```

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

3.4 交易数据存储验证机制

FabricSQL 的交易数据存储验证机制是在 MySQL 数据库中利用 Salthash 值，实现对交易数据的存储和验证。Salthash 是指对每笔交易数据进行散列后，在数据的任意固定位置插入特定的字符串 Salt，再进行散列操作，这样可以极大地降低数据被篡改的风险。

在 FabricSQL 存储层中的区块链文件系统和关系型数据库 MySQL 都存储着验证通过的交易数据，不同点是，SQL 数据库中只存储有效交易的信息。当交易数据传至 SQL 数据库时，每条交易数据顺序追加，更新交易表和哈希表。在 SQL 数据库中，交易表各字段的值记录着每笔交易的重要信息，哈希表存储着每笔交易的 Salthash 值，如式（1）所示，它是将 Salt 值、上一笔交易的 Salthash 值和此笔交易的哈希值相组合，并使用 SHA256 加密算法对其散列，得到的散列值即为此笔交易的最终哈希值。如果为第一笔交易，Salthash 值是 Salt 值和此笔交易信息的哈希值组合哈希后的结果。FabricSQL 在 MySQL 数据库存储数据的过程如算法 3 所示，首先与数据库建立连接，然后将数据分别插入到对应表中

$$cur\_Salthash = sha256(Salt + pre\_Salthash + sha256(transdata)) \tag{1}$$

算法 3：FabricSQL 在 MySQL 数据库中存储数据的过程

输入：每笔有效交易数据

输出：数据存储在 MySQL 数据库中

```
(1) function insertData (data) {
(2)     连接 MySQL 数据库
(3)     connection.execute (
        'insert into t_transactions values (?)', data)
(4)     //pre_Salthash 指上一笔交易的 Salthash 值
(5)     pre_Salthash = connection.execute (
        'select Salthash from t_hash order by tid
        desc limit 1')
(6)     //cur_Salthash 指当前交易的 Salthash 值
(7)     cur_Salthash=sha256 (pre_Salthash+Salt
        +sha256 (transdata))
(8)     connection.execute ('insert into t_hash
        values (?)', (transId, cur_Salthash))
(9)     断开连接
(10)}
```

为防止数据发生更改的情况，数据库中不会存储 Salt 值。Salthash 值之间的连接方式也使得有一笔交易发生篡改时，作恶者需要更改这条交易之后的所有交易的 Salthash 值，这必然会提高作恶的成本。所以，Salthash 值的存储和连接方式，实现交易数据不可篡改的特性。SQL 数据库中数据存储的流程如图 4 所示。

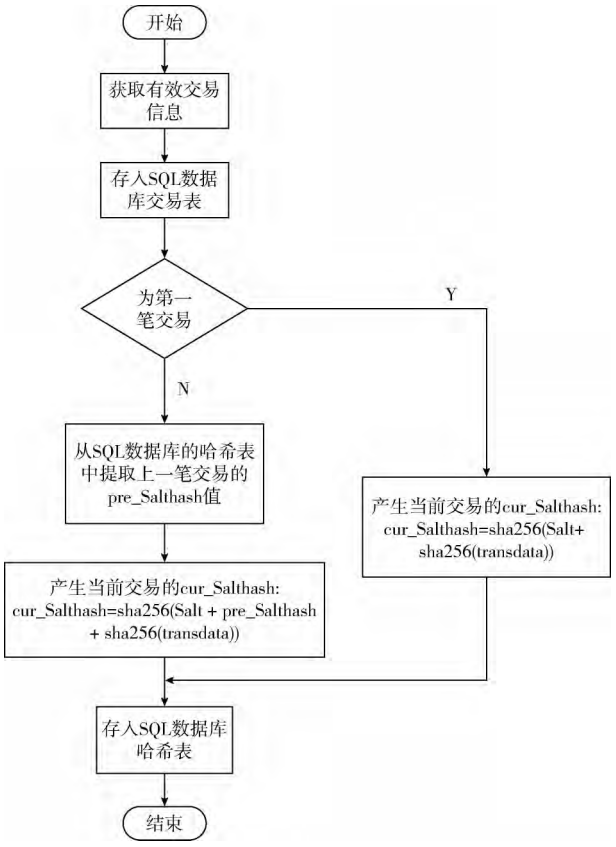

图 4 MySQL 数据库中数据存储的流程

应用层除了完成与区块链网络的交互，也负责查询逻辑的实现。数据查询校验的流程如图 5 所示。当收到用户的查询申请后，查询模块向存储层提交查询请求，存储层将交易表各字段的值和哈希表中上一笔交易的 pre_Salthash 值，返回给校验模块，校验模块将返回值与 Salt 值相组合，使用 SHA256 加密算法对其散列，再将得到的散列值与哈希表返回的此笔交易的 cur_Salthash 值进行比较，如果校验结果一致，将查询结果返回给交互模块，如果校验结果不一致，说明 SQL 数据库中的数据有被作恶者篡改的可能，返回的数据结果不可信。如果是数据库中的第一笔交易，验证原理一致，只是不需要加入 pre_Salthash 值进行验证。当要查看用户或资产的详细信息时，交易表会关联用户数据表和资产数据表，返回用户所需的全部信息。

4 实验与分析

4.1 理论分析

为了更好地进行对比实验，分别搭建了 Hyperledger Fabric 区块链系统以及本文所设计的 FabricSQL 系统。在实验中，Hyperledger Fabric 区块链系统选用相比 LevelDB 而言，查询功能更加丰富的 CouchDB 数据库作为原区块链的状态数据库，FabricSQL 选择 MySQL 作为查询数据库，虽然 CouchDB 支持原生的 json 和字节数组的操作，但是相

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

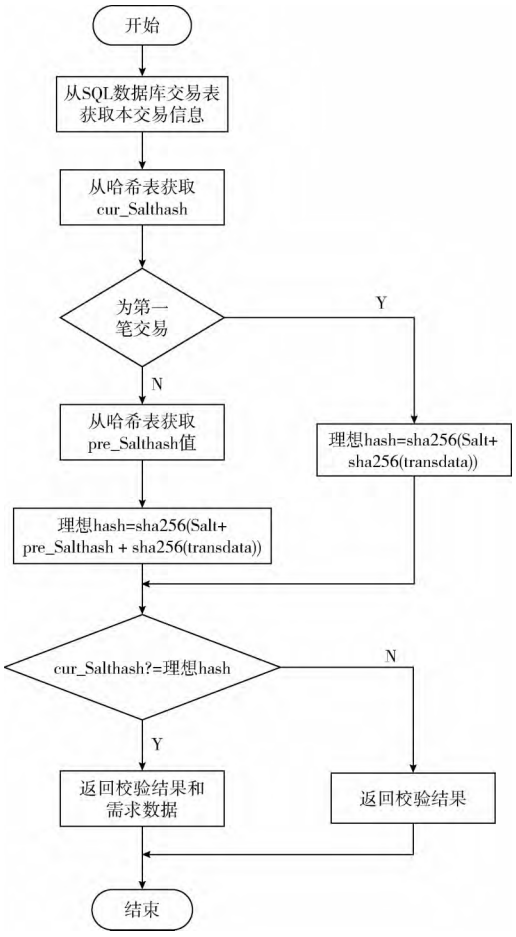

图 5　数据查询校验流程

较而言，关系型数据库只需要一条简单的语句就可以实现复杂和丰富的查询，查询功能的可扩展性强，可以更好的对数据进行分析，满足用户的需求。

4.2　实 验

4.2.1　实验环境和实验数据

实验的开发环境为 Intel Core i5-3210M 2.50 GHz CPU 和 12 GB 内存的 PC 机。利用 VMware Workstation14.1.2 建立虚拟机，以模拟真实节点，每个节点为内存 1 GB 硬盘大小为 20 GB 的 ubuntu16.04 系统。其中，Fabric 系统，利用 IBM 的 Hyperledger Fabric v1.4.0（https：//github.com/hyperledger/fabric.git）开源项目搭建环境，使用 Go 语言编写 Chaincode，使用 Node.js 语言编写用于交互操作的 SDK 程序，包括用户注册、交易创建和交易查询等功能；FabricSQL 系统在硬件方面与 Fabric 系统相同，仍在同一计算机上使用相同的 VMware 软件模拟设备结点，每个结点所拥有的计算资源和存储资源均与 Fabric 系统的资源相同，保证两组实验的硬件资源并无差异。特殊的是，FabricSQL 添加了独立的数据库结点并在 SDK 程序中，增加对交易状态判断、有效交易提取和验证等功能，还针对 MySQL 数据库开发相应的工具程序用于实现数据库相关的增删查改等一系列基本操作。

本实验的数据集来自设备资产管理数据（https：//github.com/Y-tao/FabricSQL），共 6 万用户，约 25 万条交易数据，区块链上已完成的交易数据量大小为 1 GB。

4.2.2　实验分析

在 FabricSQL 和 Fabric 系统上分别进行多组对照实验，通过查询数据，对比分析 FabricSQL 模型的防篡改能力、查询效率等多项性能。

实验 1：在 FabricSQL 上，不经过更改数据的正确流程，手动恶意篡改数据库内的多项原始实验数据，然后进行正常的查询操作，测试 FabricSQL 模型的防篡改能力是否有效。

防篡改测试方案和结果如图 6 所示，当更改一笔设备的拥有者后，执行查询操作，因为一致性校验无法通过，发现篡改，所以会有错误提示。而且，在 SQL 数据库的哈希表中，Salthash 值首尾相连，如果中间有改动，必然会对后面的数据产生影响。通过实验结果可以发现，FabricSQL 具有良好的防篡改能力。

1.在MySQL数据库上进行篡改操作：

mysql> select * from t_transactions where tid='129308';

tid	deviceID	senderID	receiverID	createtime	transID	blockNum
129308	DEV999	USER53479	**USER32155**	2019-5-23 1:44:35	d2fe54f36728b18fd1fe6735ba6f1dd2757cf74d0cf97b7dda5f99f94c4ea4a5	135290

mysql> update t_transactions set receiverID='USER0' where tid=129308;

mysql> select * from t_transactions where tid='129308';

tid	deviceID	senderID	receiverID	createtime	transID	blockNum
129308	DEV999	USER53479	**USER0**	2019-5-23 1:44:35	d2fe54f36728b18fd1fe6735ba6f1dd2757cf74d0cf97b7dda5f99f94c4ea4a5	135290

2.在FabricSQL上进行查询操作：

root@fabricsql:~/fabric # python verify.py 999

error!!!

done

图 6　FabricSQL 防篡改测试

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

实验 2：在 FabricSQL 和基于 Fabric 的区块链系统上分别录入相同的原始数据，然后分别查询一笔交易的响应时间，进行对照实验。再通过更改交易量的规模，观察分析 FabricSQL 模型在不同数据基数上的查询效率及是否比区块链系统的原始查询操作有所优化。

状态数据查询响应时间如图 7 所示，由于单个查询以毫秒为单位完成，为了最小化随机错误的影响，实验中查询时间是 1000 个独立查询的平均时间。图中横坐标为已上链的交易数据量大小，分别为 500 M 和 1000 M，纵坐标是以 10 为基数的对数刻度，反映 Fabric 和 FabricSQL 进行查询操作的响应时间，FabricSQL 的查询时间中已包含对每笔交易的安全性进行验证的时间代价。通过对实验结果分析，可以看出 FabricSQL 的查询效率比区块链系统的原始查询效率有显著提升。

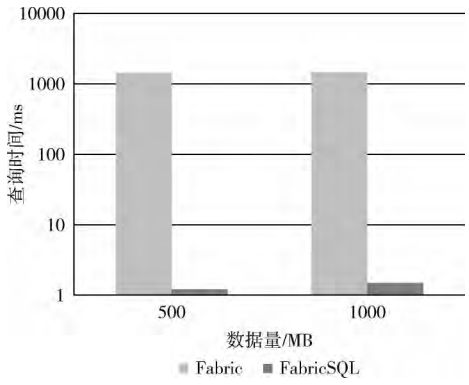

图 7 状态数据的查询时间

实验 3：在设计的实验模型和基于 Fabric 的区块链系统上，基于相同版本号的多条历史交易数据，分别进行溯源查询操作。再通过更改交易的版本号，观察分析 FabricSQL 模型在不同版本号下的查询效率及是否比区块链系统的溯源查询有所优化。

历史数据查询时间如图 8 所示，在数据库中按照不同的版本号筛选出候选数据，在候选数据中随机查询 5 个设备，计算平均查询时间。实验中，分别在 FabricSQL 和 Fabric 上对版本号为 10，20，30，40，50 的交易数据进行溯源查询，通过对实验结果分析，可以看出 FabricSQL 比区块链系统的溯源查询效率有显著提升，且查询代价主要集中于最新版本的查询时间。

5 结束语

Hyperledger Fabric 通过文件系统和 key-value 数据库实现对区块链数据的查询，但是存在着对其存储数据的访问方法支持有限、查询功能单一且查询功能的可扩展性不强、查询效率较低等问题。本文针对区块链在查询方面的不足，提出一种区块链数据的关系查询解决方案 FabricSQL，该方案在区块链网络中完成对交易数据验证和更新后，根据返回的事件状态提取其中有效交易的数据，并将其按照防篡改方案存储至 SQL 数据库中，依据存储验证机制实现对区块链数据的关系查询。在实验模型和 Hyperledger Fabric 上进行多组查询性能的对比实验结果表明，本文提出的区块链数据的关系查询解决方案 FabricSQL 具有丰富的查询功能、良好的防篡改性能以及高效的查询效率。

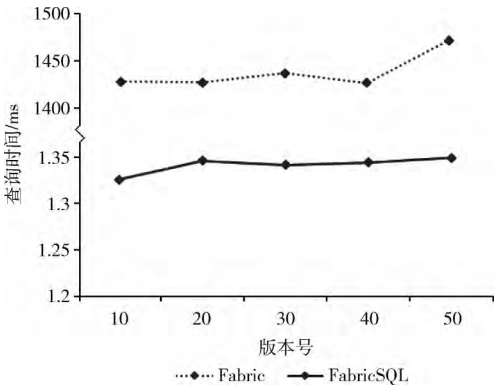

图 8 历史数据的查询时间

区块链的广泛应用需要良好的查询性能支持，后续还需要与其它区块链数据库方案在防篡改能力、空间复杂度和时间复杂度等方面进行比较，从而改进实验方法，优化 FabricSQL 方案。

参考文献：

[1] Nakamoto S. Bitcoin：A peer-to-peer electronic cash system [EB/OL]. [2008-10-31]. https://bitcoin.org/bitcoin.pdf.

[2] Buterin V. Ethereum：A next-generation smart contract and decentralized application platform [EB/OL]. [2015-01-05]. https://github.com/ethereum/wiki/wiki/White-Paper.

[3] Cachin C. Architecture of the hyperledger blockchain fabric [EB/OL]. [2016-07-25]. https://www.zurich.ibm.com/dccl/papers/cachin_dccl.pdf.

[4] FENG Xiang，LIU Tao，WU Shouhe，et al. Blockchain development combat：Key technologies and case studies of Hyperledger Fabric [M]. Beijing：Mechanical Industry Press，2018：245-248 (in Chinese). [冯翔，刘涛，吴寿鹤，等. 区块链开发实战：Hyperledger Fabric 关键技术与案例分析[M]. 北京：机械工业出版社，2018：245-248.]

[5] McConaghy T，Marques R，Müller A，et al. BigchainDB：A scalable blockchain database [EB/OL]. [2016-06-08]. http://blockchain.jetzt/wp-content/uploads/2016/02/bigchaindb.

[6] Jia Dayu，Xin Junchang，Wang Zhiqiong，et al. ElasticChain：Support very large blockchain by reducing data redundancy [C] //Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

and Big Data. Springer，2018：440-454.

[7] JIA Dayu，XIN Junchang，WANG Zhiqiong，et al. Efficient query method for capacity scalable blockchain system [J]. Journal of Software，2019，30 (9)：2655-2670 (in Chinese). [贾大宇，信俊昌，王之琼，等. 容量可扩展区块链系统的高效查询方法 [J]. 软件学报，2019，30 (9)：2655-2670.]

[8] McConaghy T，Marques R，Müller A，et al. BigchainDB 2.0 the blockchain database [EB/OL]. [2018-05-14]. https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf.

[9] Bartoletti M，Lande S，Pompianu L，et al. A general framework for blockchain analytics [C] //Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers. ACM，2017：7-23.

[10] Li Yang，Zheng Kai，Yan Ying，et al. EtherQL：A query layer for blockchain system [C] //International Conference on Database Systems for Advanced Applications. Springer，2017：556-567.

[11] Beijing PeerSafe Technology Limited Company. White paper for blockchain database application platform [EB/OL]. [2017-01-22]. http://chainsql.net/PDF/ChainSQL-whitepaper.pdf.

[12] Dinh T T A，Wang J，Chen G，et al. Blockbench：A framework for analyzing private blockchains [C] //Proceedings of the ACM International Conference on Management of Data. ACM，2017：1085-1100.

[13] Androulaki E，Barger A，Bortnikov V，et al. Hyperledger fabric：A distributed operating system for permissioned blockchains [C] //Proceedings of the Thirteenth EuroSys Conference. ACM，2018：30-44.

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net