(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 111488611 A
(43)申请公布日 2020.08.04

(21)申请号 202010271043.9

(22)申请日 2020.04.08

(71)申请人 北京瑞策科技有限公司
地址 100085 北京市海淀区上地九街9号一层108

(72)发明人 吉建勋 杨慧

(51)Int.Cl.
G06F 21/62(2013.01)
G06F 21/64(2013.01)
G06Q 40/04(2012.01)
H04L 29/08(2006.01)
G06F 16/2455(2019.01)
G06F 16/22(2019.01)

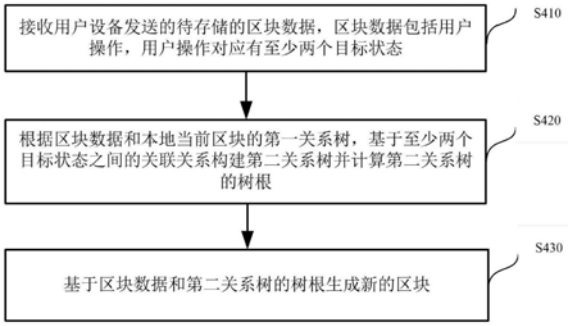
权利要求书2页 说明书13页 附图5页

(54)发明名称

业务数据区块链的关系数据存储方法及装置

(57)摘要

本发明提供了一种业务数据区块链的关系数据存储方法及装置,涉及区块链技术领域,包括区块链节点接收用户设备发送的待存储的区块数据,所述区块数据包括用户操作,所述用户操作对应有至少两个目标状态;根据所述区块数据和本地当前区块的第一关系树,基于所述至少两个目标状态之间的关联关系构建第二关系树并计算所述第二关系树的树根;基于所述区块数据和所述第二关系树的树根生成新的区块。通过区块存储用户操作数据,关系树存储用户操作后的状态,基于待存储的用户操作,可以更新的关系树,以便客户端直接从更新的关系树获取相应的数据,可以提升数据存储的效率。并且保证了数据的真实性以及有效性,提升了数据的安全性。



CN 111488611 A

1.一种业务数据区块链的关系数据存储方法,应用于业务数据区块链系统,所述系统包括多个区块链节点,其特征在于,所述方法包括:

区块链节点接收用户设备发送的待存储的区块数据,所述区块数据包括用户操作,所述用户操作对应有至少两个目标状态;

根据所述区块数据和本地当前区块的第一关系树,基于所述至少两个目标状态之间的关联关系构建第二关系树并计算所述第二关系树的树根,其中,所述第二关系树包括两个目标状态之间的关联关系;

基于所述区块数据和所述第二关系树的树根生成新的区块,其中,所述新的区块包括区块头和区块体,所述区块头包括第二关系树的树根,所述区块体包括区块数据。

2.根据权利要求1所述的关系数据存储方法,其特征在于,所述目标状态为用户操作后全局状态,所述关系树指示的关联关系包括用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息中的一种或多种;其中,所述用户操作为用户对业务数据的操作。

3.根据权利要求2所述的关系数据存储方法,其特征在于,关系树中叶子节点的value以哈希表或数组的方式存储用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息;或,

关系树的叶子节点的value用于存储关系子树的树根,所述关系子树存储用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息,所述关系子树为默克尔树或MPT树。

4.根据权利要求1所述的关系数据存储方法,其特征在于,所述用户操作包括时间戳、操作用户地址、被操作地址、操作类型、操作的值、积分地址、用户对用户操作数据的签名以及用户操作数据的哈希值中的一种或多种;其中,所述操作类型包括用户对实体的操作和对积分的操作,所述被操作地址包括对实体的操作地址和其他操作用户的地址。

5.根据权利要求1所述的关系数据存储方法,其特征在于,所述用户设备为客户端或服务器。

6.一种业务数据区块链的关系数据存储装置,应用于业务数据区块链系统,所述系统包括多个区块链节点,其特征在于,所述装置包括:

接收模块,用于接收用户设备发送的待存储的区块数据,所述区块数据包括用户操作,所述用户操作对应有至少两个目标状态;

构建模块,用于根据所述区块数据和本地当前区块的第一关系树,基于所述至少两个目标状态之间的关联关系构建第二关系树并计算所述第二关系树的树根,其中,所述第二关系树包括两个目标状态之间的关联关系;

生成模块,用于基于所述区块数据和所述第二关系树的树根生成新的区块,其中,所述新的区块包括区块头和区块体,所述区块头包括第二关系树的树根,所述区块体包括区块数据。

7.根据权利要求6所述的关系数据存储装置,其特征在于,所述目标状态为用户操作后全局状态,所述关系树指示的关联关系包括用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息中的一种或多种;其中,所述用户操作为用户对业务数据的操作。

8.一种区块链节点,其特征在于,包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序,所述处理器执行所述程序时实现权利要求1-5任意一项所述的方法。

9.一种区块链系统,其特征在于,包括多个区块链节点以及多个用户设备,所述区块链

节点用于实现权利要求1-5任意一项所述的方法。

10.一种计算机可读存储介质,所述计算机可读存储介质上存储有计算机程序,所述计算机程序被处理器执行时实现权利要求1-5任意一项所述的方法。

业务数据区块链的关系数据存储方法及装置

技术领域

[0001] 本发明涉及区块链技术领域，尤其是涉及一种业务数据区块链的关系数据存储方法及装置。

背景技术

[0002] 目前，区块链技术是运用加密算法、共识机制等技术的分布式存储账本。随着区块链技术的运用，越来越多的互联网数据会存储在区块链上。

[0003] 现有的区块链中，只能存储交易数据，交易数据包括转账方地址、接收方地址以及转账金额；针对各种业务数据(例如：存证数据、溯源数据、金融数据、旅游数据、搜索数据、自媒体数据、调研数据、广告数据、电商数据、社区数据、知识问答数据、知识付费数据、共享单车数据、招聘数据、生活服务数据、租房数据、投票数据、OTO数据(也称为线上到线下数据)、社交数据、点赞数据、评价数据、网约车数据等互联网相关数据)而言，不仅需要在区块链上表达出数据本身，还需要在区块链上表达出数据之间的关联关系。

[0004] 因此，如何实现在区块链上存储业务数据，换句话说，如何实现业务数据区块链，成为亟待解决的问题。

发明内容

[0005] 本发明的目的在于提供一种业务数据区块链的关系数据存储方法及装置，以缓解了区块链中数据存储的效率低的技术问题。

[0006] 第一方面，实施例提供一种业务数据区块链的关系数据存储方法，应用于业务数据区块链系统，所述系统包括多个区块链节点，所述方法包括：

[0007] 区块链节点接收用户设备发送的待存储的区块数据，所述区块数据包括用户操作，所述用户操作对应有至少两个目标状态；

[0008] 根据所述区块数据和本地当前区块的第一关系树，基于所述至少两个目标状态之间的关联关系构建第二关系树并计算所述第二关系树的树根，其中，所述第二关系树包括两个目标状态之间的关联关系；

[0009] 基于所述区块数据和所述第二关系树的树根生成新的区块，其中，所述新的区块包括区块头和区块体，所述区块头包括第二关系树的树根，所述区块体包括区块数据。

[0010] 在可选的实施方式中，所述目标状态为用户操作后全局状态，所述关系树指示的关联关系包括用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息中的一种或多种；其中，所述用户操作为用户对业务数据的操作。

[0011] 在可选的实施方式中，关系树中叶子节点的value以哈希表或数组的方式存储用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息；或，关系树的叶子节点的value用于存储关系子树的树根，所述关系子树存储用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息，所述关系子树为默克尔树或MPT树。

[0012] 在可选的实施方式中，所述用户操作包括时间戳、操作用户地址、被操作地址、操

作类型、操作的值、积分地址、用户对用户操作数据的签名以及用户操作数据的哈希值中的一种或多种;其中,所述操作类型包括用户对实体的操作和对积分的操作,所述被操作地址包括对实体的操作地址和其他操作用户的地址。

[0013] 在可选的实施方式中,所述用户设备为客户端或服务器。

[0014] 第二方面,实施例提供一种业务数据区块链的关系数据存储装置,应用于业务数据区块链系统,所述系统包括多个区块链节点,所述装置包括:

[0015] 接收模块,用于接收用户设备发送的待存储的区块数据,所述区块数据包括用户操作,所述用户操作对应有至少两个目标状态;

[0016] 构建模块,用于根据所述区块数据和本地当前区块的第一关系树,基于所述至少两个目标状态之间的关联关系构建第二关系树并计算所述第二关系树的树根,其中,所述第二关系树包括两个目标状态之间的关联关系;

[0017] 生成模块,用于基于所述区块数据和所述第二关系树的树根生成新的区块,其中,所述新的区块包括区块头和区块体,所述区块头包括第二关系树的树根,所述区块体包括区块数据。

[0018] 在可选的实施方式中,所述目标状态为用户操作后全局状态,所述关系树指示的关联关系包括用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息中的一种或多种;其中,所述用户操作为用户对业务数据的操作。

[0019] 在可选的实施方式中,关系树中叶子节点的value以哈希表或数组的方式存储用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息;或,关系树的叶子节点的value用于存储关系子树的树根,所述关系子树存储用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息,所述关系子树为默克尔树或MPT树。

[0020] 在可选的实施方式中,所述用户操作包括时间戳、操作用户地址、被操作地址、操作类型、操作的值、积分地址、用户对用户操作数据的签名以及用户操作数据的哈希值中的一种或多种;其中,所述操作类型包括用户对实体的操作和对积分的操作,所述被操作地址包括对实体的操作地址和其他操作用户的地址。

[0021] 在可选的实施方式中,所述用户设备为客户端或服务器。

[0022] 第三方面,实施例提供一种区块链节点,包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序,所述处理器执行所述程序时实现前述实施方式任意一项所述的方法。

[0023] 第四方面,实施例提供一种区块链系统,包括多个区块链节点以及多个用户设备,所述区块链节点用于实现前述实施方式任意一项所述的方法。

[0024] 第五方面,实施例提供一种计算机可读存储介质,所述计算机可读存储介质上存储有计算机程序,所述计算机程序被处理器执行时实现前述实施方式任意一项所述的方法。

[0025] 本发明提供的一种业务数据区块链的关系数据存储方法及装置,应用于业务数据区块链系统,所述系统包括多个区块链节点。区块链节点接收用户设备发送的待存储的区块数据,所述区块数据包括用户操作,所述用户操作对应有至少两个目标状态;根据所述区块数据和本地当前区块的第一关系树,基于所述至少两个目标状态之间的关联关系构建第二关系树并计算所述第二关系树的树根,其中,所述第二关系树包括两个目标状态之间的

关联关系;基于所述区块数据和所述第二关系树的树根生成新的区块,其中,所述新的区块包括区块头和区块体,所述区块头包括第二关系树的树根,所述区块体包括区块数据。通过区块存储用户操作数据,关系树存储用户操作后的状态,基于待存储的用户操作,可以更新的关系树,以便客户端直接从更新的关系树获取相应的数据,可以提升数据存储的效率。并且保证了数据的真实性以及有效性,提升了数据的安全性。

附图说明

[0026] 为了更清楚地说明本发明具体实施方式或现有技术中的技术方案,下面将对具体实施方式或现有技术描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图是本发明的一些实施方式,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其他的附图。

[0027] 图1为本申请公开了一种应用场景示意图;

[0028] 图2为现有的一种将区块链的账户状态数据组织成MPT状态树的示意图;

[0029] 图3为现有的一种MPT状态树上的节点node复用的示意图;

[0030] 图4为本申请公开了一种业务数据区块链的关系数据存储方法流程示意图;

[0031] 图5为本申请公开了一种业务数据区块链的区块头结构示意图;

[0032] 图6为本申请公开了一种业务数据区块链的关系数据存储装置结构示意图;

[0033] 图7为本申请公开了一种区块链节点结构示意图。

具体实施方式

[0034] 为使本发明实施例的目的、技术方案和优点更加清楚,下面将结合本发明实施例中的附图,对本发明实施例中的技术方案进行清楚、完整地描述,显然,所描述的实施例是本发明一部分实施例,而不是全部的实施例。通常在此处附图中描述和示出的本发明实施例的组件可以以各种不同的配置来布置和设计。

[0035] 因此,以下对在附图中提供的本发明的实施例的详细描述并非旨在限制要求保护的本发明的范围,而是仅仅表示本发明的选定实施例。基于本发明中的实施例,本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其他实施例,都属于本发明保护的范围。

[0036] 应注意到:相似的标号和字母在下面的附图中表示类似项,因此,一旦某一项在一个附图中被定义,则在随后的附图中不需要对其进行进一步定义和解释。

[0037] 本申请实施例提供了一种业务数据区块链的关系数据存储方法及装置。图1示出了为本申请提供的区块链网络的示例性架构。

[0038] 如图1所示,该示例性架构可以包括一个或多个用户设备101和一个或多个区块链节点102,当区块链节点102和用户设备101为多个时,形成区块链网络,其中,该区块链网络中的区块链节点可以包括存储节点和出块节点。用户设备101可以用于与区块链节点102进行交互,例如,用户设备101发送用户操作给区块链节点;区块链节点将该用户操作存储在区块链中;用户设备101还可以向区块链节点发送查询请求,该查询请求用于查询该区块链中的数据。

[0039] 值得说明的是,本实施例架构并不限定其实现其他功能,例如用户设备101也可以

作为区块链节点等。

[0040] 为便于对本申请实施例的理解，下面将结合附图以具体实施例作进一步的解释说明，实施例并不构成对本申请实施例的限定。

[0041] 区块链一般被划分为三种类型：公有链(Public Blockchain)，私有链(PrivateBlockchain)和联盟链(Consortium Blockchain)。此外，还有多种类型的结合，比如私有链+联盟链、联盟链+公有链等不同组合形式。其中去中心化程度最高的是公有链。公有链以比特币、以太坊为代表，加入公有链的参与者可以读取链上的数据记录、参与交易以及竞争新区块的记账权等。

[0042] 而且，各参与者(即节点)可自由加入以及退出网络，并进行相关操作。私有链则相反，该网络的写入权限由某个组织或者机构控制，数据读取权限受组织规定。简单来说，私有链可以为一个弱中心化系统，参与节点具有严格限制且少。这种类型的区块链更适合于特定机构内部使用。

[0043] 基于区块链的基本特性，区块链通常是由若干个区块构成。在这些区块中分别记录有与该区块的创建时刻对应的时间戳，所有的区块严格按照区块中记录的时间戳，构成一条在时间上有序的数据链条。

[0044] 对于物理世界产生的真实数据，可以将其构建成区块链所支持的标准的交易(transaction)格式，然后发布至区块链，由区块链中的节点设备进行共识，并在达成共识后，由区块链中作为记账节点的节点设备，将这笔交易打包进区块，在区块链中进行持久化存证。

[0045] 在区块链领域，有一个重要的概念就是账户(Account)；以以太坊为例，以太坊通常将账户划分为外部账户和合约账户两类；外部账户就是由用户直接控制的账户；而合约账户则是由用户通过外部账户创建的，包含合约代码的账户(即智能合约)。

[0046] 当然，对于一些基于以太坊的架构而衍生出的区块链项目，还可以对区块链支持的账户类型，进行进一步的扩展，在本说明书中不进行特别限定。

[0047] 对于区块链中的账户而言，通常会通过一个结构体，来维护账户的账户状态。当区块中的交易被执行后，区块链中与该交易相关的账户的状态通常也会发生变化。

[0048] 以以太坊为例，账户的结构体通常包括Balance，Nonce，Code和storage等字段。其中：

[0049] Balance字段，用于维护账户目前的账户余额；

[0050] Nonce字段，用于该账户的交易次数；它是用于保障每笔交易能且只能被处理一次的计数器，有效避免重放攻击。

[0051] code字段，用于维护该账户的合约代码；在实际应用中，code字段中通常仅维护合约代码的hash值；因而，code字段通常也称之为codehash字段。对于外部账户而言，该字段为空值。

[0052] storage字段，用于维护该账户的存储(默认为空)。在实际应用中，storage字段仅维护基于账户的存储内容构建的MPT(Merkle Patricia Trie)树的根节点；因此，storage字段通常也称之为storageRoot字段。

[0053] 其中，对于外部账户而言，以上示出的code字段和storage字段为空值。

[0054] 而大多数区块链项目，通常都会使用Merkle树；或者，基于Merkle树的数据结构，

来存储和维护数据。以以太坊为例，以太坊使用了MPT树(一种Merkle树变种)，作为数据组织形式，用来组织和管理账户状态、交易信息等重要数据。

[0055] 以太坊针对区块链中需要存储和维护的数据，设计了三颗MPT树，分别是MPT状态树、MPT交易树和MPT收据树。

[0056] MPT状态树，是区块链中所有账户的账户状态数据(state)，组织成的MPT树；MPT交易树是区块中的交易数据(transaction)，组织成的MPT树；MPT收据树，是区块中的交易执行完毕后生成的与每笔交易对应的交易收据(receipt)，组织成的MPT树。以上示出的MPT状态树、MPT交易树和MPT收据树的根节点的hash值，都会被添加至区块头中。

[0057] 其中，MPT交易树和MPT收据树，与区块相对应，每一个区块都有自己的MPT交易树和MPT收据树。而MPT状态树是一个全局的MPT树，并不与某一个特定的区块相对应，而是涵盖了区块链中所有账户的账户状态数据。

[0058] 对于组织成的MPT交易树、MPT收据树和MPT状态树，最终都会在采用多级数据存储结构的Key-Value型数据库(比如，LevelDB)中进行存储。

[0059] 而采用多级存储结构的上述数据库，通常可以被划分为n级数据存储；例如，各级数据存储可以依次设为L0，L1，L2，L3....L(n-1)；对于上述数据库中的各级数据存储而言，等级编号越小通常级别越高；例如，L0存储的是最新的若干区块的数据，L1存储的是次新的若干区块数据，依次类推。

[0060] 其中，各级数据存储对应的存储介质的读写性能，通常也可以存在性能差异；级别高(即等级编号较小的)的数据存储对应的存储介质的读写性能，可以高于级别低的数据存储对应的存储介质的读写性能。

[0061] 例如，在实际应用中，级别高的数据存储，可以使用读写性能较高的存储介质；而级别低的数据存储，可以使用单位成本低，且容量较大的存储介质。

[0062] 在实际应用中，随着区块高度的增长，在数据库中存储的数据，会包含很多历史数据；而且，区块号越小的区块中的数据越久远，越不重要。因此，为了降低整体的存储成本，通常需要对不同区块高度的数据进行“区别对待”；

[0063] 例如，可以将区块号较小的区块中的数据，存储至成本较低的存储介质上；而将区块号较大的区块中的数据，存储在成本较高的存储介质上。

[0064] 在针对数据库中存储的MPT交易树、MPT收据树和MPT状态树等数据进行分级存储时，由于MPT交易树和MPT收据树，与各个区块相对应，实际上是“区块间无关”的数据；因此，对于MPT交易树和MPT收据树，很容易进行分级存储；例如，直接按照MPT交易树和MPT收据树上的node所属的区块号进行数据迁移即可完成分级存储。

[0065] 基于此，本说明书将不再具体阐述MPT交易树和MPT收据树的分级存储，而重点阐述MPT状态树的分级存储。

[0066] 请参见图2，图2为本说明书示出的一种将区块链的账户状态数据组织成MPT状态树的示意图。

[0067] MPT树，是一种经过改良的，融合了Merkle树和Trie字典树(也称之为前缀树)两种树形结构的优点的Merkle树变种。

[0068] 在MPT树中通常包括三种数据节点，分别为叶子节点(leaf node)，扩展节点(extension node)和分支节点(branch node)。

[0069] 叶子节点,表示为[key,value]的一个键值对,其中key是种特殊十六进制编码。

[0070] 扩展节点,也是[key,value]的一个键值对,但是这里的value是其他节点的hash值(hash指针)。也就是说通过hash指针链接到其他节点。

[0071] 分支节点,因为MPT树中的key被编码成一种特殊的16进制的表示,再加上最后的value,所以分支节点是一个长度为17的list,前16个元素对应着key中的16个可能的十六进制字符(一个字符对应一个半字节nibble)。如果有一个[key,value]对在这个分支节点终止,最后一个元素代表一个value值,即分支节点既可以是搜索路径的终止也可以是路径的中间节点。

[0072] 假设需要组织成MPT状态树的账户状态数据如下表1所示:

[0073]

账户地址(Key)							账户状态(Value)
a	7	1	1	3	5	5	state1
a	7	7	d	3	3	7	state2
a	7	f	9	3	6	5	state3
a	7	7	d	3	9	7	state4

[0074] 表1

[0075] 在表1中,账户地址是由若干16进制的字符构成的字符串。账户状态state,是由上述Balance,Nonce,Code和storage等字段构成的结构体。

[0076] 最终按照表1中的账户状态数据组织成的MPT状态树,参见图2所示;如图2所示,按照表1中的账户状态数据组织成的MPT状态树,是由4个叶子节点,2个分支节点,和2个扩展节点构成。

[0077] 在图2中,prefix字段为扩展节点和叶子节点共同具有的前缀字段。该prefix字段的取值,在实际应用中可以用于表示节点类型。

[0078] prefix字段的取值为0,表示包含偶数个nibbles的扩展节点;如前所述,nibble表示半字节,由4位二进制组成,一个nibble可以对应一个组成账户地址的字符。

[0079] prefix字段的取值为1,表示包含奇数个nibble(s)的扩展节点;

[0080] prefix字段的取值为2,表示包含偶数个nibbles的叶子节点;

[0081] prefix字段的取值为3,表示包含奇数个nibble(s)的叶子节点。

[0082] 而分支节点,由于其是并列单nibble的前缀节点,因此分支节点不具有上述prefix字段。

[0083] 扩展节点中的Shared nibble字段,对应该扩展节点所包含的键值对的key值,表示账户地址之间的共同字符前缀;比如,上表中的所有账户地址均具有共同的字符前缀a7。Next Node字段中填充下一个节点的hash值(hash指针)。

[0084] 分支节点中的16进制字符0～f字段,对应该分支节点所包含的键值对的key值;如果该分支节点为账户地址在MPT树上的搜索路径上的中间节点,则该分支节点的Value字段可以为空值。0～f字段中用于填充下一个节点的hash值。

[0085] 叶子节点中的Key-end,对应该叶子节点所包含的键值对的key值,表示账户地址的最后几个字符。从根节点搜索到叶子节点的搜索路径上的各个节点的key值,构成了一个完整的账户地址。该叶子节点的Value字段填充账户地址对应的账户状态数据;例如,可以

对上述Balance,Nonce,Code和storage等字段构成的结构体进行编号后,填充至叶子节点的Value字段。

[0086] 进一步的,如图2所示的MPT状态树上的node,最终也是以Key-Value键值对的形式存储在数据库中;

[0087] 其中,当MPT状态树上的node在数据库中进行存储时,MPT状态树上的node的键值对中的key,为node所包含的数据内容的hash值;MPT状态树上的node的键值对中的Value,为node所包含的数据内容。

[0088] 也即,在将MPT状态树上的node存储至数据库时,可以计算该node所包含的数据内容的hash值(即对node整体进行hash计算),并将计算出的hash值作为key,将该node所包含的数据内容作为value,生成Key-Value键值对;然后,将生成的Key-Value键值对存储至数据库中。

[0089] 由于MPT状态树上的node,是以node所包含的数据内容的hash值为Key,node所包含的数据内容为value进行存储;因此,在需要查询MPT状态树上的node时,通常可以基于node所包含的数据内容的hash值作为key来进行内容寻址。而采用“内容寻址”,对于一些“内容重复”的node,则通常可以进行“复用”,以节约数据存储的存储空间。

[0090] 如图3所示,图3为本说明书示出的一种MPT状态树上的node复用的示意图。

[0091] 在实际应用中,区块链每产生一个最新区块,则在该最新区块中的交易被执行之后,区块链中与这些被执行的交易相关账户的账户状态,通常也会随之发生变化;

[0092] 例如,当区块中的一笔“转账交易”执行完毕后,与该“转账交易”相关的转出方账户和转入方账户的余额(即这些账户的Balance字段的取值),通常也会随之发生变化。

[0093] 而节点设备在区块链产生的最新区块中的交易执行完毕后,由于当前区块链中的账户状态发生了变化,因此节点设备需要根据区块链中所有账户当前的账户状态数据,来构建MPT树,用于维护区块链中所有账户的最新状态。

[0094] 也即,每当区块链中产生一个最新区块,并且该最新区块中的交易执行完毕后,导致区块链中的账户状态发生变化,节点设备都需要基于区块链中所有账户最新的账户状态数据,重新构建一颗MPT树。

[0095] 换句话说,区块链中每一个区块,都有一个与之对应的MPT状态树;该MPT状态树,维护了在该区块中的交易在执行完毕后,区块链中所有账户最新的账户状态。

[0096] 而需要说明的是,一个最新区块中的交易执行完毕后,可能仅仅会导致部分账户的账户状态发生变化;因此,在更新MPT状态树时,并不需要基于区块链中所有的账户当前的状态数据,重新构建一颗完整的MPT状态树,而只需要在该最新区块之前的区块对应的MPT状态树的基础上,对部分账户状态发生变化的账户对应的node进行更新即可。而对于MPT状态树上与账户状态未发生变化的账户对应的node而言,由于这些node为发生数据更新,可以直接复用该最新区块之前的区块对应的MPT状态树上相应的node即可。

[0097] 如图3所示,假设表1中的账户状态数据,为Block N中的交易执行完毕后,区块链上所有账户的最新账户状态;基于表1中的账户状态数据组织成的MPT状态树,仍如图2所示。

[0098] 假设当Block N+1中的交易执行完毕后,导致上述表1中的账户地址为“a7f9365”的账户状态,由“state3”更新为“state5”;此时,在Block N+1更新MPT状态树时,并不需要

基于Block N+1中的交易执行完毕后，区块链中所有的账户当前的状态数据，重新构建一颗MPT状态树。

[0099] 请参见图3，在这种情况下，可以仅将Block N对应的MPT树上(即图2示出的MPT状态树)，“key-end”为“9365”的叶子节点中的Value，由“state3”更新为“state5”，并继续更新从root节点到该叶子节点的路径上的所有节点的hash指针；也即，当MPT状态树上的叶子节点发生更新，由于该叶子节点整体的hash值发生更新，那么从根节点到该叶子节点的路径上的所有的节点的hash指针也会随之发生更新。例如，请继续参见图3，除了需要更新“key-end”为“9365”的叶子节点中的Value值以外，还需要更新该叶子节点的上一个分支节点(Branch Node)的f字段中填充的，指向该叶子节点的哈希指针；进一步的，还可以继续向根节点追溯，继续更新该分支节点的上一个根节点(Root Extension Node)的“Next Node”字段中填充的，指向该分支节点的hash指针。

[0100] 而除了以上发生更新的节点以外，其它未发生更新的节点，都可以直接复用BlockN的MPT状态树上对应的节点即可；

[0101] 其中，由于Block N对应的MPT树，最终需要作为历史数据进行保留；因此，在BlockN+1更新MPT状态树时，对于这些发生更新的node，并不是对Block N对应的MPT状态树上原来的node的基础上，直接进行修改更新，而是在Block N+1对应的MPT树上重新创建这些发生更新的node。

[0102] 也即，对于与Block N+1对应的MPT状态树上，实际上只需要重新创建少量发生更新的node，对于其它未发生更新的node，可以通过直接复用Block N对应的MPT状态树上对应的节点。

[0103] 例如，如图3所示，对于Block N+1对应的MPT状态树上，实际上只需要重新创建少量发生更新的node；比如，图3中仅需要重新创建一个作为根节点的扩展节点、一个分支节点和一个叶子节点；对于未发生更新的node，可以通过在该MPT状态树上这些重新创建的node中，添加指向Block N对应的MPT状态树上的相应node的hash指针来完成node的复用。而Block N对应的MPT状态树上那些更新前的node，将作为历史账户状态数据进行保存；比如，图3示出的“key-end”为“9365”，且Value为“state3”的叶子节点，将作为历史数据进行保留。在以上例子中，以Block N+1的MPT状态树上的少量node发生内容更新，可以“复用”上一个区块Block N的大多数node为例进行了说明。而在实际应用中，Block N+1的MPT状态树上也可能会较上一个区块Block N新增node。

[0104] 在这种情况下，该新增的node虽然无法直接从上一个区块Block N的MPT树中进行复用，但有可能从更早之前的区块的MPT状态树上进行“复用”；

[0105] 例如，Block N+1的MPT状态树上新增的node，虽然在Block N的MPT状态树上出现过，但出现在更早的Block的MPT状态树上；比如，出现在Block N-1的MPT状态树上；因此，Block N+1的MPT状态树上新增的node，可以直接复用Block N-1的MPT状态树上对应的node即可。

[0106] 以上是现有的区块链中的存储结构说明，下面对本申请中的业务数据区块链的存储结构进行说明。

[0107] 图4为本申请提供的一种业务数据区块链的关系数据存储方法流程示意图。如图4所示，应用于业务数据区块链系统，区块链系统包括多个区块链节点，例如图1中的区块链

节点102,该方法具体可以包括如下步骤:

[0108] S410,接收用户设备发送的待存储的区块数据,区块数据包括用户操作,用户操作对应有至少两个目标状态;

[0109] 该业务数据区块链可以用于存储用户操作数据,该用户操作数据可以为用户在互联网上的操作数据,用户操作数据包括存证数据、溯源数据、金融数据、旅游数据、搜索数据、自媒体数据、调研数据、广告数据、电商数据、社区数据、知识问答数据、知识付费数据、共享单车数据、招聘数据、生活服务数据、租房数据、投票数据、线上到线下数据、社交数据、点赞数据、评价数据以及网约车数据中的一种或多种。

[0110] 在一个示例中,用户操作数据包括时间戳、操作用户地址、被操作地址、操作类型、转账的值、积分地址、用户对用户操作数据的签名以及用户操作数据的哈希值中的一种或多种;其中,操作类型包括用户对实体的操作和转账操作,被操作地址包括对实体的操作地址和其他用户的地址。

[0111] 该目标状态可以为用户操作后的全局状态。其中,用户操作后的全局状态包括积分信息、用户信息以及实体信息中的一种或多种;其中,用户操作为用户对业务数据的操作。

[0112] 举例来说,实体可以是互联网电子商务平台的商品,用户可以发起购买该商品的操作,就是针对该商品实体的转账操作;用户也可以发起针对该商品的评论或点赞操作,就是针对该商品实体的用户操作。

[0113] 积分可以是用户发行的积分,例如:用户可以通过公证后,将房子证券化后,在区块链上发行一个积分,这个积分在区块链上唯一表示该房产;也可以是,将房子证券化,发行预定数量的积分,拥有该积分即拥有该房子的所有权,享有该房子租赁或者变卖的对等所有权。除此之外,区块链上拥有原生积分,该原生积分可以用于约束用户在区块链上的操作,避免用户无限制的使用区块链系统的资源;例如:区块链上的用户注册后,在区块链上进行操作,需要抵押原生积分,才能将用户操作数据上链。

[0114] 作为一个示例,用户信息(如:Account字段)、积分信息(如:Asset字段)、实体信息(如:Object字段)中任意一种状态信息都可以采用MPT状态树来实现,各种MPT状态树的树根存储在区块头中。如图5所示,区块存储的用户操作(如:Action字段),和链上数据库存储的收据信息(如:Receipt字段)的MPT状态树的树根也存储在区块头中。

[0115] 该用户设备为客户端或服务器。

[0116] S420,根据区块数据和本地当前区块的第一关系树,基于至少两个目标状态之间的关联关系构建第二关系树并计算第二关系树的树根。其中,第二关系树包括两个目标状态之间的关联关系。

[0117] 该关系树可以为用户操作后全局状态之间的关系。关系树指示的关联关系包括用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息中的一种或多种;其中,用户操作为用户对业务数据的操作。

[0118] 作为一个示例,实体信息-用户信息(如:Object-Account字段)、用户信息-积分信息(如:Account-Asset字段)、实体信息-积分信息(如:Object-Asset字段)中任意一种关联关系都可以采用MPT关系树来实现,各种MPT关系树的树根存储在区块头中。如图5所示,区块存储的用户操作(如:Action字段),和链上数据库存储的收据信息(如:Receipt字段)的

MPT关系树的树根也存储在区块头中。

[0119] 在一个示例中，关系树存储的关联关系包括实体信息和用户信息的关联关系；其中，一个实体信息对应一个或多个用户信息。

[0120] 例如：在互联网的电子商务平台(如：淘宝、天猫、京东商城)上，一件商品，有多位不同用户购买；在互联网上的自媒体平台(如：微信、微博)上，一条自媒体数据，多位用户进行了点赞或评论；这类实体与用户之间的一对多关系，可以采用上述实体信息-用户信息的关联关系表达出来。不仅方便了业务数据上链，也方便了用户对业务数据进行查询。

[0121] 在另一个示例中，关系树的关联关系包括用户信息和积分信息的关联关系；其中，一个用户信息对应一个或多个积分信息。

[0122] 此时，一位用户有可能拥有多种积分；例如：一位用户既拥有区块链上的原生积分，还拥有其他类型的积分。上述关联关系能够描述该用户与积分之间的一对多关系，同时也方便该用户查询。需要说明的是，在区块链系统中的用户，至少会拥有该区块链系统的原生积分；有可能拥有其他用户发行或者自己发行的积分。

[0123] 在另一个示例中，关系树存储的关联关系包括实体信息和积分信息的关联关系；其中，一个实体信息对应一个或多个积分信息。

[0124] 此时，相当于在区块链上，实现了实体的证券化；例如：一件商品的所有权，对应一个积分，积分的转账对应该商品所有权的转移；一套房子的所有权，对应预定数量的积分，拥有积分的数量对应该房子的所有权比例，即拥有该房子出售或者租赁同比例的价款。

[0125] 所述积分数据的属性类型包括关联属性和非关联属性；所述用户信息和/或所述实体信息的属性类型包括关联属性和非关联属性。其中，所述关联属性的属性值根据所述用户操作数据自动增减。

[0126] 需要说明的是，区块链系统中的区块包括区块头和区块体，区块头存储的是摘要信息，区块体存储的是转账信息(也称交易信息)。本说明书中用户操作是存储在区块的区块体中，用户操作的摘要值(也称哈希值)存储在区块头中；本说明书中的链上状态数据库也是区块链系统存储的一部分，每个区块链节点都有链上状态数据库和区块，来存储本说明书中的业务数据。

[0127] 第一关系树为本地存储前的业务数据区块链的关系树。

[0128] 基于第一关系树和待存储的区块数据生成的第二关系树为本地存储后的关系树。

[0129] 其中，对于关系树的构架方式可以包括多种，具体可以参见前述图2-图3中的描述。

[0130] 例如，关系树中叶子节点的value以哈希表或数组的方式存储用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息；或，关系树的叶子节点的value用于存储关系子树的树根，关系子树存储用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息，关系子树为默克尔树或MPT树。

[0131] S430，基于区块数据和第二关系树的树根生成新的区块，其中，新的区块包括区块头和区块体，区块头包括第二关系树的树根，区块体包括区块数据。

[0132] 通过区块存储用户操作数据，关系树存储用户操作后的状态，基于待存储的用户操作，可以更新的关系树，以便客户端直接从更新的关系树获取相应的数据，可以提升数据存储的效率。并且保证了数据的真实性以及有效性，提升了数据的安全性。

[0133] 上述中，采用状态树或关系树的方式存储这些信息，用户可以通过地址信息进行查询；针对用户信息、积分信息和实体信息，直接采用用户地址、积分地址以及实体地址信进行信息查询；针对实体信息-用户信息、用户信息-积分信息以及实体信息-积分信息，分别采用实体地址、用户地址以及实体地址进行信息查询。

[0134] 本说明书中，实体信息-用户信息、用户信息-积分信息以及实体信息-积分信息中任意一种信息被组织成MPT状态树的数据结构，也就是数据逻辑结构；采用的存储结构可以是KV数据库，例如：LevalDB数据库，也可以采用支持属性查询的MySql和MongoDB数据库，这些数据库是物理的存储结构。

[0135] 本说明书能够让业务数据上链，让业务数据的各种关联关系也能够上链；业务数据上链的好处包括但不限于真实可信、可溯源、数据权属归属于个人等。

[0136] 本申请通过区块存储用户操作数据，链上状态树和关系树，能够将业务数据的各种属性存储在区块链上，从而解决业务数据上链的问题；此外，链上存储的状态树和关系树，也方便用户进行访问。

[0137] 此外，需要说明的是，本说明书公开了业务数据上链，采用区块存储的用户操作数据，链上存储的用户信息和实体信息；用户操作数据、用户信息以及实体信息也可以被称为三维数据模型，区别于现有技术中的用户地址和转账余额的二维数据。上述的用户信息、实体信息、积分信息以及它们之间的关联信息，也成为多状态信息；多状态信息的存储上链，也就意味着业务数据的上链。

[0138] 图6为本申请实施例提供的一种业务数据区块链的关系数据存储装置结构示意图。如图6所示，应用于业务数据区块链系统，系统包括多个区块链节点，装置包括：

[0139] 接收模块601，用于接收用户设备发送的待存储的区块数据，区块数据包括用户操作，用户操作对应有至少两个目标状态；

[0140] 构建模块602，用于根据区块数据和本地当前区块的第一关系树，基于至少两个目标状态之间的关联关系构建第二关系树并计算第二关系树的树根，其中，第二关系树包括两个目标状态之间的关联关系；

[0141] 生成模块603，用于基于区块数据和第二关系树的树根生成新的区块，其中，新的区块包括区块头和区块体，区块头包括第二关系树的树根，区块体包括区块数据。

[0142] 在一些实施例中，目标状态为用户操作后全局状态，关系树指示的关联关系包括用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息中的一种或多种；其中，用户操作为用户对业务数据的操作。

[0143] 在一些实施例中，关系树中叶子节点的value以哈希表或数组的方式存储用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息；或，关系树的叶子节点的value用于存储关系子树的树根，关系子树存储用户信息-积分信息、实体信息-积分信息以及实体信息-用户信息，关系子树为默克尔树或MPT树。

[0144] 在一些实施例中，用户操作包括时间戳、操作用户地址、被操作地址、操作类型、操作的值、积分地址、用户对用户操作数据的签名以及用户操作数据的哈希值中的一种或多种；其中，操作类型包括用户对实体的操作和对积分的操作，被操作地址包括对实体的操作地址和其他操作用户的地址。

[0145] 在一些实施例中，用户设备为客户端或服务器。

[0146] 可以理解，本实施例的业务数据区块链的关系数据存储装置与图4所示的方法实施例相对应，因此，以上关于图4所示的方法实施例的描述同样适用于本实施例的装置，在此不再赘述。

[0147] 图7示出了本申请实施例所提供的一种区块链节点结构示意图，该计算机设备可以包括：处理器710、存储器720、输入/输出接口730、通信接口740和总线750。其中处理器710、存储器720、输入/输出接口730和通信接口740通过总线750实现彼此之间在设备内部的通信连接。处理器710用于执行存储器720中存储的可执行模块，例如图4所示的方法实施例对应的计算机程序。

[0148] 对于上述图7中，处理器可以采用通用的CPU(Central Processing Unit，中央处理器)、微处理器、应用专用集成电路(Application Specific Integrated Circuit，ASIC)、或者一个或多个集成电路等方式实现，用于执行相关程序，以实现本申请实施例所提供的技术方案。

[0149] 存储器可以采用ROM(Read Only Memory，只读存储器)、RAM(Random Access Memory，随机存取存储器)、静态存储设备，动态存储设备等形式实现。存储器可以存储操作系统和其他应用程序，在通过软件或者固件来实现本申请实施例所提供的技术方案时，相关的程序代码保存在存储器中，并由处理器来调用执行。

[0150] 输入/输出接口用于连接输入/输出模块，以实现信息输入及输出。输入输出/模块可以作为组件配置在设备中(图中未示出)，也可以外接于设备以提供相应功能。其中输入设备可以包括键盘、鼠标、触摸屏、麦克风、各类传感器等，输出设备可以包括显示器、扬声器、振动器、指示灯等。

[0151] 通信接口用于连接通信模块(图中未示出)，以实现本设备与其他设备的通信交互。其中通信模块可以通过有线方式(例如USB、网线等)实现通信，也可以通过无线方式(例如移动网络、WIFI、蓝牙等)实现通信。

[0152] 总线包括一通路，在设备的各个组件(例如处理器、存储器、输入/输出接口和通信接口)之间传输信息。

[0153] 需要说明的是，尽管上述设备仅示出了处理器、存储器、输入/输出接口、通信接口以及总线，但是在具体实施过程中，该设备还可以包括实现正常运行所必需的其他组件。此外，本领域的技术人员可以理解的是，上述设备中也可以仅包含实现本申请实施例方案所必需的组件，而不必包含图中所示的全部组件。

[0154] 专业人员应该还可以进一步意识到，结合本文中所公开的实施例描述的各示例的单元及算法步骤，能够以电子硬件、计算机软件或者二者的结合来实现，为了清楚地说明硬件和软件的可互换性，在上述说明中已经按照功能一般性地描述了各示例的组成及步骤。这些功能究竟以硬件还是软件方式来执行，取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能，但是这种实现不应认为超出本申请的范围。

[0155] 结合本文中所公开的实施例描述的方法或算法的步骤可以用硬件、处理器执行的软件模块，或者二者的结合来实施。软件模块可以置于随机存储器(RAM)、内存、只读存储器(ROM)、电可编程ROM、电可擦除可编程ROM、寄存器、硬盘、可移动磁盘、CD-ROM、或技术领域内所公知的任意其它形式的存储介质中。

[0156] 以上所述的具体实施方式，对本申请的目的、技术方案和有益效果进行了进一步详细说明，所应理解的是，以上所述仅为本申请的具体实施方式而已，并不用于限定本申请的保护范围，凡在本申请的范围之内，所做的任何修改、等同替换、改进等，均应包含在本申请的保护范围之内。

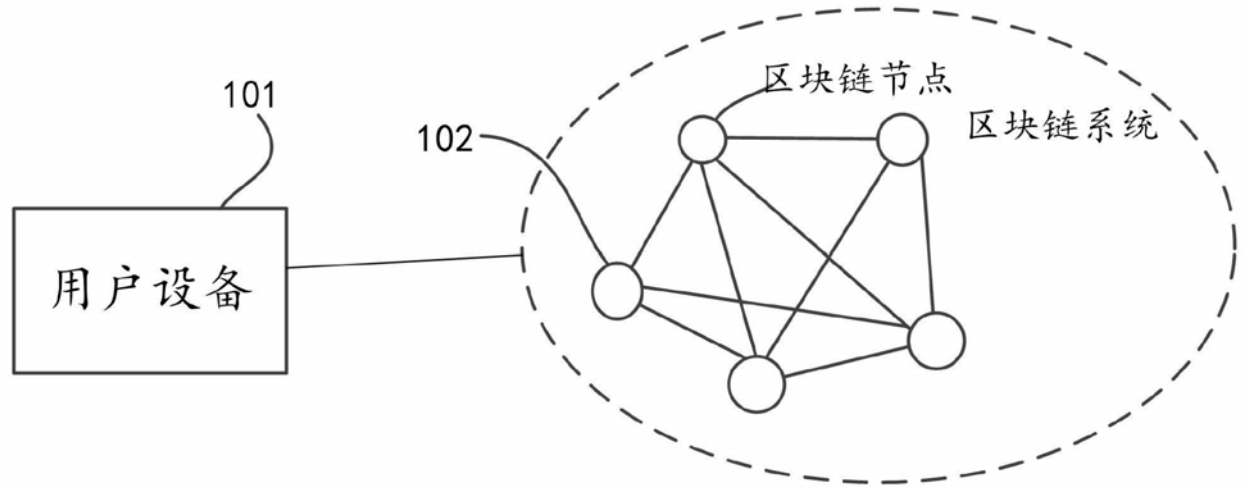


图1

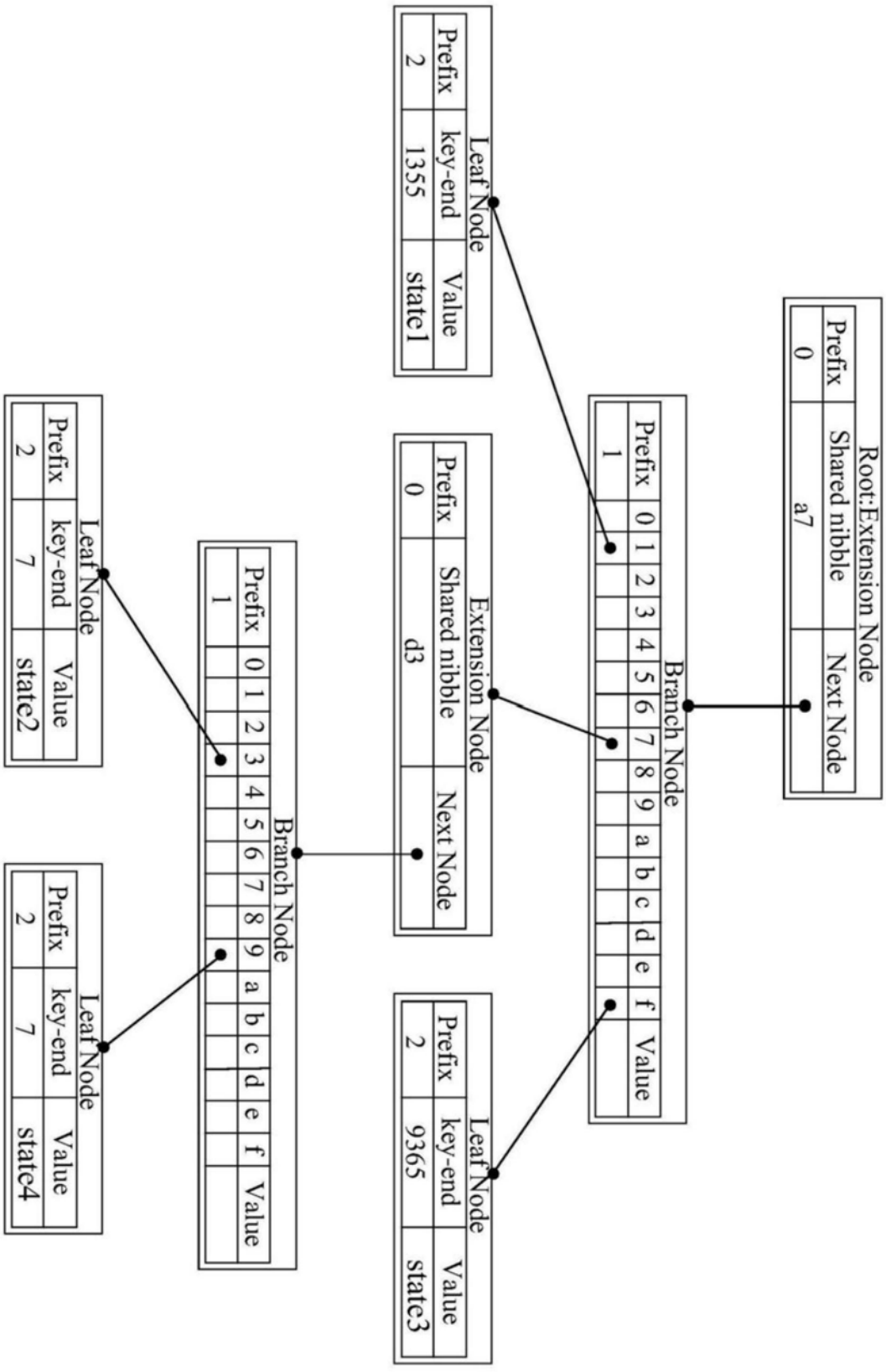

Root:Extension Node
Prefix
Shared nibble
Next Node
0
a7
Branch Node
Prefix
0 1 2 3 4 5 6 7 8 9 a b c d e f Value
1
Leaf Node
Prefix
key-end
Value
2
1355
state1
Extension Node
Prefix
Shared nibble
Next Node
0
d3
Leaf Node
Prefix
key-end
Value
2
9365
state3
Branch Node
Prefix
0 1 2 3 4 5 6 7 8 9 a b c d e f Value
1
Leaf Node
Prefix
key-end
Value
2
7
state2
Leaf Node
Prefix
key-end
Value
2
7
state4

图2

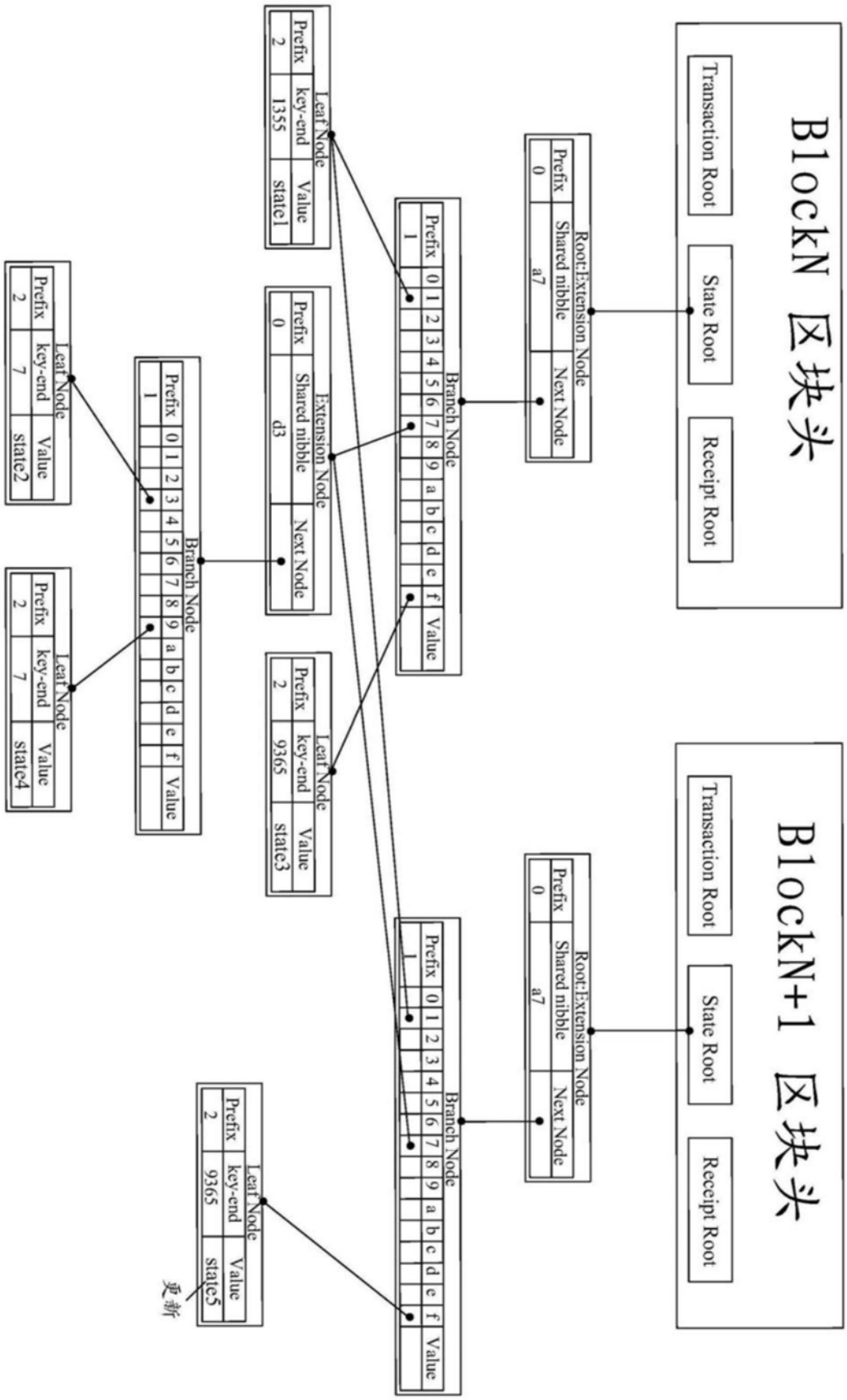

BlockN 区块头
Transaction Root
State Root
Receipt Root
BlockN+1 区块头
Transaction Root
State Root
Receipt Root
Root:Extension Node
Prefix
Shared nibble
Next Node
0
a7
Branch Node
Prefix
0 1 2 3 4 5 6 7 8 9 a b c d e f Value
1
Extension Node
Prefix
Shared nibble
Next Node
0
d3
Leaf Node
Prefix
key-end
Value
2
1355
state1
Leaf Node
Prefix
key-end
Value
2
9365
state3
Leaf Node
Prefix
key-end
Value
2
7
state2
Leaf Node
Prefix
key-end
Value
2
7
state4
Leaf Node
Prefix
key-end
Value
2
9365
state5
更新

图3

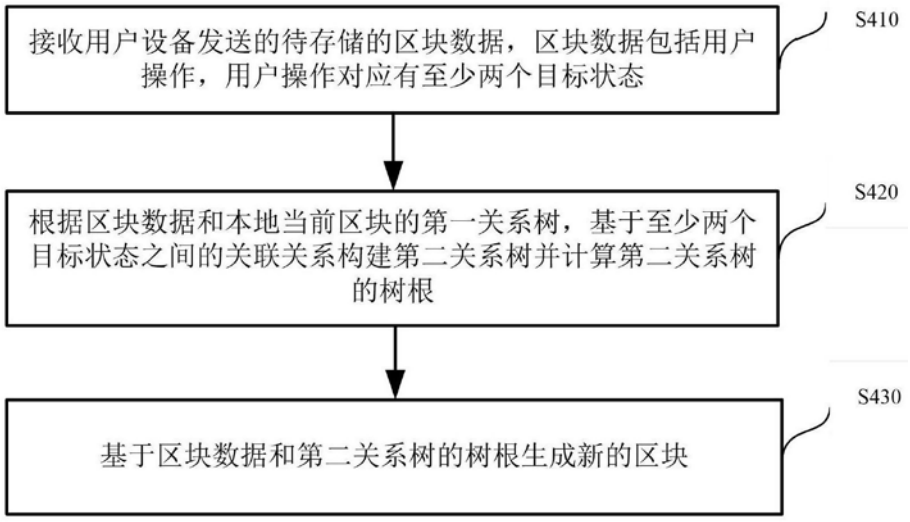

接收用户设备发送的待存储的区块数据，区块数据包括用户操作，用户操作对应有至少两个目标状态
S410
根据区块数据和本地当前区块的第一关系树，基于至少两个目标状态之间的关联关系构建第二关系树并计算第二关系树的树根
S420
基于区块数据和第二关系树的树根生成新的区块
S430

图4

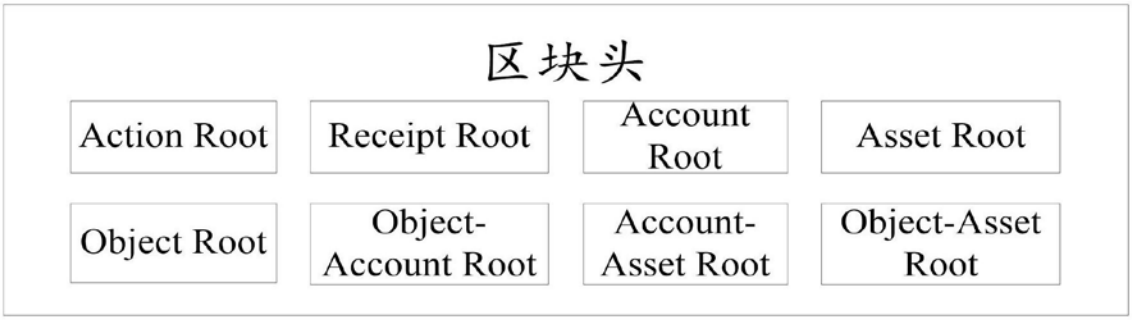

区块头
Action Root
Receipt Root
Account Root
Asset Root
Object Root
Object-Account Root
Account-Asset Root
Object-Asset Root

图5

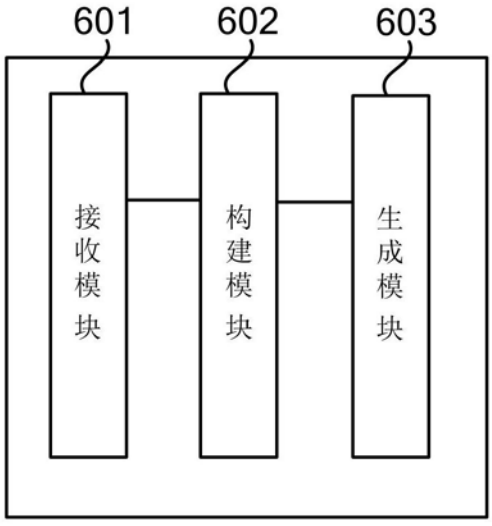

601
602
603
接收模块
构建模块
生成模块

图6

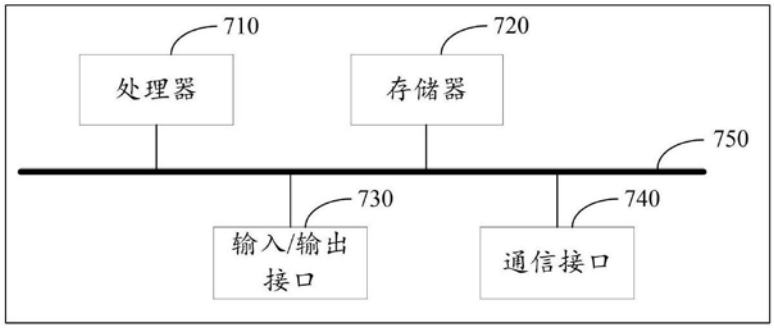


图7