摘要

随着信息技术的发展，网络中存储的数据越来越多，但数据对个人的价值有限，只有通过数据共享才能更好地利用数据的价值。由于隐私数据和单点失效的存在，中心化托管的数据共享技术很难得到数据所有者的信任，因此这就需要一种可靠的分布式基础设施来保证数据的安全共享。区块链技术作为可信的去中心化数据库已经在金融、供应链、医疗等越来越多的领域被广泛使用，大量有价值的数据被存放到区块链上，用户可以通过区块链来安全地共享数据。

在共享数据的过程中，一个常见的需求是对数据的高效检索，但现存的区块链系统并不支持对数据的快速检索。造成这种现象的主要原因是区块链系统在最初设计时没有考虑到丰富的查询需求，区块链底层的键值对数据库只支持快速写入而不支持快速检索，只能追加的数据结构也使得如果想要完全检索必须从尾至头遍历整条区块链。为了解决区块链数据共享系统中检索效率的问题，本文从区块链底层数据结构和存储模式出发，经过了大量的调研工作和基础理论研究后，提出并实现了一种在区块链数据共享系统中进行高效检索的技术方案。本文首先分析了区块链数据共享系统中进行快速检索存在的挑战，包括

针对这些挑战探索相关的传统检索技术并思考其与区块链系统结合的可能性。具体的设计内容上，本方案创新性地将区块链的存储模式由单一的链上存储转变为了链上链下混合存储模式，并设计了一种分布式语义提取算法建立链上数据和链下数据的映射关系。单链检索部分，本方案在链上构建了一个名为默克尔语义搜索树的索引结构，通过结合了merkle树、B+树和帕特丽夏树等结构，提供了多关键字查询、范围查询、模糊查询等功能。在跨链检索部分，本方案首先对不同链上的交易格式进行整合，然后在每条链上抽取跨链委员会组成知识链，委员会中的节点们通过跨链共识对来自不同链上的实体三元组进行处理，构建一个跨链的可视化全局知识图谱，提供跨链检索服务。我们还同时为区块链中的轻节点提供了代理查询和结果验证的功能，轻节点需要连接全节点来进行代理查询，全节点在返回查询结果的同时需要提供对查询结果的验证证明，这样轻节点在自己本地即可对查询结果进行验证。

**关 键 词**：区块链， 数据共享， 分布式检索， 知识图谱

1. 绪论
   1. 研究背景与意义
   2. 国内外研究现状
   3. 本文研究内容及主要贡献
   4. 本文内容组织结构
2. 相关理论与技术

2.1 区块链系统和结构

2.1.1 区块链存储系统

2.1.2 区块链数据结构

2.1.3 区块链查询技术

2.1.4 跨链技术

2.2 信息检索技术

2.2.1 TF-IDF算法

2.2.2 倒排索引

2.2.3 检索模型分类

2.2.4 B+树

2.2.5 pagerank算法

2.2.6 RDF数据模型

2.2.7 知识图谱构建技术

2.3 本章小结

第三章 链上链下混合存储架构中高效搜索技术方案

3.1 背景介绍

3.2 系统设计

3.2.1 整体架构

3.2.2 问题定义

3.2.3 性能指标

3.3 方案描述

3.3.1 系统总体工作流程

3.3.2 链下语义信息抽取模块

3.3.3 链上索引构建模块

3.3.4 单链查询模块

3.3.5 跨链数据整合模块

3.3.6 跨链委员会组建模块

3.3.7 跨链查询模块

3.4 方案分析

3.4.1 检索效率分析

3.4.2 安全性分析

3.5 本章小结

第四章 实验设计与结果分析

4.1 实验环境配置

4.2 实验数据集

4.3 系统性能测试

4.3.1 索引生成开销

4.3.2 系统响应时间

4.3.3 查询准确率和召回率

4.3.4 排序性能测试

4.4 对比实验测试

4.4.1 对比方案介绍

4.4.2 对比实验结果

4.5 本章小结

第五章 总结与展望

**第一章 绪论**

**1.1 研究背景与意义**

随着社会和科技的发展，越来越多的人参与到网络空间活动中，导致了互联网中存储的数据量呈现出了指数级别的增长，而由于数据的所有者不同和存储方式的多样化，形成了大量的信息壁垒和数据孤岛。为了打通不同数据所有者之间的壁垒，学术界和工业界从各方面开展了对于海量数据共享的研究，想要找到一种安全高效的数据共享模式，但仍存在一些问题。

最近几年，随着比特币和以太坊等加密货币系统的成功，区块链技术作为其核心技术受到了越来越广泛的关注。作为一种分布式账本技术，区块链具有去中心化、不可篡改、安全可信、数据公开透明等特点，目前在供应链、农产品溯源、金融基础设置、版权存证等领域已经有了相关应用。

区块链作为一项分布式基础设施，有一个新的应用领域，分布式数据共享[1]。分布式数据共享在之前已经有了相关的尝试，例如Bittorrent协议和ipfs协议[2]，但是，缺乏一种稳定的激励机制，使得这种文件共享系统健康地运转下去，同时，多方写入的场景，也可能导致拜占庭问题的产生。利用区块链的特性，可以有效地解决当前分布式数据共享系统中存在的问题，但区块链本身仍存在数据搜索效率低、不支持复杂搜索等功能。

比特币白皮书中描述的区块链是一种按照时间顺序将区块以链表的形式连接起来的数据结构[3]，通过密码学技术保证了交易数据的不可篡改性，数据需要经过系统中多数节点的共识，才能够以追加的方式添加到区块链的尾部，如果恶意节点想要篡改已经经过共识并存储在区块链内的数据，需要掌握系统中的大部分算力，这在现实中往往是难以实现的，因此通过区块链技术可以解决分布式场景中的信任问题。

在区块链系统中，存储的数据并不是底层的原始数据，底层数据往往是带有一定格式的交易数据，这些数据一般不会以明文的形式存储到区块链上，需要经过散列运算，将得到的哈希值上链，比如比特币采用双sha256散列函数，可以将任意长度的原始交易记录存进区块。在保证安全性的同时，区块链这种只能追加的数据结构也导致了系统性能的下降，大多数区块链系统底层采用的是key-value数据库，通过牺牲读性能来换取写性能的提高[4]。然而，在实际应用中，区块链系统的交易吞吐量往往没有那么大，比特币的交易量只有每秒1笔左右，以太坊的交易写入量为每秒10笔左右[5]，而随着区块链中数据的增加，对链上数据的搜索需求已经逐渐体现了出来，key-value数据库的快速写入优势无法体现，而读取效率低已经成为限制区块链搜索功能发展的瓶颈。

现有的区块链系统在最初设计时，只是针对某种特定事务进行处理，没有考虑丰富查询的需求，数据要么存储在键值对中，要么存储在平面文件中，而这些存储方式都不支持复杂的查询操作。因此，如果实现了区块链上的快速、多关键词的搜索，用户就可以对分布式数据共享系统中的数据进行快速、个性化的搜索，可以更快地找到满足自己需求的数据位置，系统的可用性更强。同时可以利用智能合约对链上资产进行自动分配，系统的安全性更强。

除了区块链系统中存在的检索困难问题之外，另一个需要考虑的问题是不同区块链之间的跨链检索问题。由于不同的区块链系统是异构的，他们底层的数据结构和共识机制都不相同，这就造成了区块链与区块链之间的另一种“数据孤岛”。

跨异构链的信息检索问题可以归为两点：统一数据格式和建立信息关联。从搜索技术的发展史中可以了解到，知识图谱的两大关键技术，RDF（资源描述框架）和语义网（Semantic Web）很好地描述了非结构化数据之间的逻辑关系，RDF通过一种通用的三元组数据格式对非结构化数据进行简洁统一便于移植的形式化描述，语义网则以RDF数据格式为基础构建了一个能够描述知识实体或概念本身以及他们相互之间逻辑关系的智能网络。在这两种技术的基础之上构建的知识图谱，存储的全部都是统一格式的结构化数据，使不同来源的数据之间形成一种结构化状态，用户只需要通过这些关联关系，就可以快速搜索到自己需要的信息。因此，如果可以通过RDF和语义网这两种技术在不同结构的区块链系统中对数据进行统一的格式整合，并在相关数据间建立语义关联关系，就可以构建一个跨异构区块链的“知识图谱”，用户在使用该跨链检索系统进行检索操作时，不需要考虑底层的区块链结构，只需要关注搜索的内容本身，就可以得到自己需要的信息。

**1.2 国内外研究现状**

目前国内外对于区块链系统中的搜索模块的研究还处在起步阶段，主要的处理方法是对区块链存储结构的优化及对区块链索引系统的优化。绝大多数成熟的区块链系统采用的是以LevelDB为代表的key-value分布式数据库。比特币的早期版本采用BerkleyDB[6],之后改为LevelDB。以太坊采用的是LevelDB, Hyperledger Fabric在1.0版本更新后给用户提供了LevelDB和couchDB两种数据库可供用户选择[7]。LevelDB等key-value数据库提供的查询功能较为简单，因此区块链系统的设计者需要在LevelDB数据库的上层封装一个查询层，以提供基本的查询功能。

以太坊在将内存中的数据持久化到磁盘时，将格式化的交易数据通过递归长度前缀编码（RLP编码）转化成字节数组[8]，然后计算转化后字节数组的哈希值，最后将哈希值和编码后的交易数据以<Key,Value>对的形式存到LevelDB数据库中。进行查询操作时，通过解析用户提交的Key值来找到对应的Value字节数组，并将解码后的数据返回给用户。由于以太坊在最初设计之时没有考虑复杂的分析查询，因此没有在系统中加入索引结构，导致系统无法执行关系型查询，同时无法执行Top-K查询，K-NN查询等复杂的查询。

BigchainDB[9]是一种具有区块链特性的分布式数据库结构，底层数据库使用了mongoDB，并且设计了一套丰富的查询语言。但在严格意义上来说，BigchainDB本质上并不是一个传统的区块链系统，同时并不具有区块链的账本全复制特性，只是利用区块链的特性来保护数据的安全性。

Anh Dinh等人提出了一种名为Blockbench[10]的私有区块链评价框架,尝试通过用支持丰富查询功能的存储系统来替换传统区块链的LevelDB数据库。

Li等人提出了EtherQL[11],主要思想是将区块链中的数据拷贝到外部的数据库中作为区块链的搜索层，并在这一查询层上实现了丰富的搜索功能，可以支持Top-K查询、范围查询等复杂的搜索的功能。EtherQL方案不需要对原有区块链系统进行修改，是一种可以兼容多种成熟区块链系统的方案。

BlockchainDB[12]是一种构建在区块链之上的分片数据库系统，主要适用的场景是互不信任的多方实现数据读写的场景，设计了一种将区块链作为数据存储层，并将存储节点进行分片管理的系统。BlockchainDB提供了易用的查询接口和强大的数据一致性保证，提高了区块链存储系统的性能，并降低了使用的复杂度。

外联数据库解决方案的思想是为已有的区块链系统设计外部查询层，另一种对区块链数据进行快速搜索的方法是在区块链系统中内置索引结构。山东大学的李庆忠教授团队提出了一种支持快速搜索的区块链教育证书系统（ECBC）[13]，该方案扩展了以太坊系统中的MPT结构，设计了MPT-Chain索引结构，将学生的学历证书按照相同前缀进行字典存储，利用链上索引结构加快了对学生学历证书的搜索效率。 Ruan等人在2019年提出了LineageChain[14]，通过在区块链上建立了Merkle DAG结构和一个跳表结构，实现了对历史交易数据的快速溯源，并通过智能合约实现了用户存储和搜索的API。

vChain[15]实现了另一种区块链上的搜索方式，思路是将查询工作委托给全节点进行，全节点可通过爬虫或者本地建立数据库副本，从而进行快速查询，同时在区块中加入一种认证数据结构（ADS），全节点需要返回一个查询数据的加密证明，用户通过验证证明可以验证查询结果的完整性。优势是既保证了快速查询，同时又可以保证查询结果的正确性和完整性。但普通用户需要委托全节点进行查询，同时加密证明的验证也需要耗费时间。

华东师范大学的金澈清教授团队提出了“师大链”[16]，将链上数据与链下数据联系起来，为区块链上的交易数据赋予丰富的语义信息，可以对区块数据进行类SQL查询，同时加入了RDBMS对链上和链下的数据进行管理。

经过几年的研究，区块链的跨链技术已经有了比较成熟的解决方案，目前主流的技术有公证人机制，侧链/中继模式和哈希锁定等方式[31]，可以用于实现区块链之间的资产互操作，解决价值孤岛的问题。但是这些跨链方案只关注了跨链的资产交换问题，没有考虑到数据的交互问题，虽然解决的价值孤岛的问题，但数据孤岛和信息壁垒仍然存在，下一步需要考虑的是如何对不同链上的数据进行链接，解决跨链检索的问题。武汉科技大学的任惠[32]等人提出了一种基于知识图谱的区块链语义知识共享机制，设计并实现了一种去中心化的全局知识查询方案，虽然没有提及“跨链”的概念，但为跨异构链的数据检索方案提供了参考。

**1.3** **本文研究内容及主要贡献**

为了解决目前区块链系统中存在的检索效率差，不支持复杂搜索的问题，本毕业设计重点分析现有区块链系统存储结构和索引方式中存在的问题，研究一种在区块链系统中进行快速搜索的方法，在实现对区块链系统中存储的数据进行快速搜索的同时支持较为复杂的分析查询，同时保证查询结果的完整性和可靠性。同时，针对跨区块链信息检索存在的问题，本毕业设计在单链检索的研究基础上研究对多条异构区块链进行信息检索的方法，提出了一种支持多链的信息检索方案，并以知识图谱的形式呈现给用户。

本文的主要内容和贡献如下：

1. 为了建立链下数据和链上数据的协同映射关系,首先对链下信息的语义信息进行提取，提出了一种改进的TF-IDF模型，称为去中心化语义提取算法,通过引入去中心化词频因子、词性因子和位置因子来提升语义特征提取的准确性，解决了传统关键字提取算法中缺乏语义理解的特点,并基于此构建了一个关键字的语义倒排索引,为链上索引结构的构建提供支撑；

2. 在链上构建一个名为默克尔语义搜索树（MST）的索引结构，结合了默克尔树，帕特丽夏树，B+树，哈希指针，AC自动机等技术，利用该索引结构可以对之前提取的链下数据的语义信息和元数据进行多功能的复杂搜索操作，包括多关键字搜索，范围搜索，模糊查询等。由于区块链的追加性质，MST的结构是动态构建的，随着创世区块的生成而生成，并随着新区快的生成而更新，整个操作由所有矿工通过共识机制来完成。针对不同的搜索功能采用了不同的技术手段实现，对于多关键字搜索，通过帕特丽夏树的前缀匹配和压缩特性来实现；B+树用于实现范围搜索功能，针对模糊查询，设计并实现了一个基于AC自动机的模糊态多关键字匹配算法；

3. 为了使得不包含所有账本信息的轻节点（瘦客户端）也可以对系统中的数据进行搜索和验证，设计一种代理查询的功能，即拥有所有账本信息的全节点为轻节点做代理查询和提供验证的操作。首先由轻节点发起查询请求，全节点接收到查询请求后在本地进行查询，然后将得到的查询结果和该查询结果在MST上对应的merkle proof一起返回给轻节点，轻节点收到查询结果和merkle proof后可以通过本地的区块链头部信息对查询结果进行验证。

4．为了提供跨链检索的功能，本文提出了一种基于RDF的去中心化知识抽取方案，可以将异构区块链中不同结构的数据抽取并整合成为实体三元组的形式；在不同的区块链间构建了一个全局的跨链索引结构，并设计了配套的索引生成共识方案；提供了跨链数据查询的通用接口，并在此基础上开发了可视化的知识图谱结构，用户无需考虑底层具体的区块链结构，只需要输入对应的搜索需求，就可以查询到相关的结果。

**1.4 本文****内容组织结构**

本文章节共分五章，组织结构如下：

第一章，介绍研究的背景和意义，对国内外的研究现状进行了整理和分析，针对当前的研究现状分析了在区块链数据共享系统中进行高效检索存在的问题和挑战，最后介绍了本文的主要研究内容及主要贡献。

第二章，介绍了本文研究课题中涉及到的重要理论基础和关键技术，主要包括区块链系统及其结构，例如区块链的存储系统、数据结构、查询技术和跨链技术，以及传统的信息检索技术中使用的相关算法和数据模型。

第三章，针对区块链数据共享系统中的信息检索问题提出了一种链上链下混合架构中的高效搜索方案。首先介绍了方案的系统设计和整体架构，对存在的问题进行了定义并给出了系统需要体现的性能指标。然后对系统的整体工作流程进行了描述，将该方案中的系统整体上分为五个模块，前三个模块结合提供单链查询的功能，后两个模块在单链查询的基础上提供跨链查询和知识图谱可视化的功能，单链检索和跨链检索结合，共同完成了区块链数据共享系统中的高效检索方案。最后，对方案的可行性进行了具体分析，具体体现为检索效率的分析，并对方案的安全性进行了分析。

第四章，介绍了本方案的实验设计与结果分析，实验设计将实验分为系统性能测试和对比试验测试，系统性能测试对方案的性能指标进行测试，通过调节参数对不同状态下系统的检索能力进行测试，对比试验通过将本文系统与国内外其他相关的检索系统进行检索能力对比，以体现本方案的效率优势。根据实验结果，验证了本文提出的链上链下混合架构中的高效检索方案在区块链数据共享系统中的有效性以及相较于其他方案在检索能力和开销成本上的优势。

第五章，总结与展望，总结了文本研究课题的成果，分析了研究过程与工作中存在的不足和需要改进的地方，并对未来的研究目标进行了展望。

**第二章** **相关理论与技术**

本章重点介绍两个部分，第一部分为区块链系统中与信息检索相关的模块和技术，包括区块链存储系统、数据结构、区块链本身提供的查询技术，以及在不同区块链之间进行互操作的跨链技术。第二部分介绍传统的信息检索技术，包括在搜索引擎、数据库、知识图谱中常用的相关算法和技术。

**2.1** **区块链系统和结构**

区块链是一种分布式账本技术，本质上是一组由互不信任的多个节点维护的只能追加(append-only)的块链式数据结构,由于其去中心化和不可篡改的特性，经常被用做保护数据安全和可信的去中心化数据库。与传统的数据库不同，区块链的存储系统和数据结构都有自己独有的特点，本节将从重点从存储系统和数据结构两方面来介绍区块链系统，并介绍区块链中已有的查询技术和跨链技术。

**2.1.1** **区块链存储系统**

在区块链系统中，存储的数据并不是底层的原始数据，底层数据往往是带有一定格式的交易数据，这些数据一般不会以明文的形式存储到区块链上，需要经过散列(hash)运算，将得到的哈希值上链，比如比特币采用双sha256散列函数[17]，可以将任意长度的原始交易记录存进区块。大多数区块链系统底层采用的是key-value数据库，通过牺牲读性能来换取写性能的提高，像是比特币和以太坊，使用levelDB数据库[23]，hyperledger fabric使用levelDB和couchDB结合的方式[7]。而随着区块链中数据的增加，对链上数据的搜索需求已经逐渐体现了出来，key-value数据库的快速写入优势无法体现，而读取效率低已经成为限制区块链搜索功能发展的瓶颈。

链上和链下混合存储架构架构作为提高区块链系统扩展性的有效方法,已经提出了一些方案，主要的做法是将大部分数据存在链下的分布式对等网络中，例如Bittorrent网络[19]和Ipfs网络[20]，然后将数据的整体散列值或者数据分块后的散列值存放到区块链网络中，比如BTT[21]将Bittorrent网络中存储数据的MD5散列值存放到区块链上，Filecoin[22]将Ipfs网络中文件分块后形成的散列结构merkle-DAG存在区块链上。但他们都是将链下数据的散列值存到链上，保证链上链下数据的一致性，但链上数据不包含链下数据的语义信息等有意义的信息。SEBDB[16]将数据存在链下的数据库中而将数据库的SQL语句存放在链上，使得链上的查询可以对应到链下的数据，但仍没摆脱传统数据库的限制，链下存储的数据必须以格式化的形式存在MySQL等关系型数据库中，对非格式化的链下数据则无法处理。

**2.1.2** **区块链数据结构**

区块链是由一个接一个的区块链接而成的数据结构，每个区块又包含区块头和区块身体两部分，而决定区块的前后关系的核心数据，存在于区块头中，是一种被称作哈希指针的东西，每个区块头部都有一个previousBlockHash，这是对前一个的区块的哈希值的记录，由于链式结构的特殊性，可以认为一个区块头部的previousBlockHash与之前的所有区块都有联系，同时包含时间戳信息，可以认为是这种结构实现了区块链强大的数据安全性保证，然而，这种只能追加的数据结构，意味着如果想要对区块链的所有内容进行检索，你需要从后往前遍历整条链，才能确保你的搜索结果是完整的。由于这种数据结构对只包含区块头部的轻节点并不友好，一些区块链系统对交易等必要数据的验证过程进行了优化，比如，基于UTXO模型，比特币在区块中加入了一个merkle hash tree (MHT)结构[3]，通过层层哈希的方式保证区块内的交易顺序不被篡改，merkle root被存在区块头中，轻节点可以利用merkle root验证区块内交易的真实性。比特币在创始之初就为轻节点提供了便捷的验证功能，这就是简单支付验证（SPV，Simplified Payment Verification）[3]，当想要验证一条交易是否在区块内的某个位置时，只需要让full node提供该交易在区块内部的具体位置和对应的merkle proof即可，事实上，我们只需要在的时间复杂度之内即可对交易在区块内的所在位置进行判断。我们的搜索方案采用了类似SPV的搜索结果验证过程，在第4节的验证部分，我们对我们方案的验证过程进行了介绍。

不同于比特币，以太坊是基于账户的模型，被称为交易驱动的状态机[23]，每个区块的变化都可以看做是一次状态的迁移，可以描述为：

其中，为以太坊的世界状态，为以太坊的状态转换函数，T为一个区块，因为任何计算机系统都可以被抽象成一个状态机，所以这个状态转换函数是区块链范式的基础，也是任何去中心化交易系统的基础。 以太坊将所有账户的状态以RLP编码的方式存储在区块中的一个state trie中，叫做Merkle Patricia Trie (MPT)[23]，MPT的叶子节点是每个以太坊账户的状态，分支信息则存放账户地址的压缩前缀，矿工根据区块内的交易信息来更新MPT中账户的状态，因此最新区块中保存着所有账户的最新状态，MPT的root hash存在区块头中，因此以太坊节点可以通过最新区块中的MPT对账户的当前状态进行搜索，并通过区块头进行验证。

**2.1.3** **区块链查询技术**

区块链底层的数据库大多是以LevelDB为代表的key-value数据库，系统提供的查询功能较为简单，仅能提供针对key的查询、更新和写入等操作。目前的区块链系统大多没有设计针对数据的查询功能，以以太坊为例，接收到用户的查询请求后，首先解析用户查询请求中的Key值，调用LevelDB的Get方法获得Value字符串，然后将Value字符串解码并用Json数据封装后返回给客户端。在以太坊平台上，查询操作包括三种:(1)按照区块高度查找区块头(2)按照区块哈希值查找区块头(3)按照交易哈希值查找交易[4]。由于在设计之初没有考虑到复杂的分析型查询需求，目前的区块链系统大都不支持复杂的语义查询、K-NN查询等方式。

为了增加区块链系统中的搜索功能，最有效的方法是增加查询处理引擎，主流的方案有两个，其中一个是外联数据库方法，是将区块链区块中的数据同步到外部数据库中，通过外部数据库的查询接口提供查询功能。图1是外联数据库方法的结构。

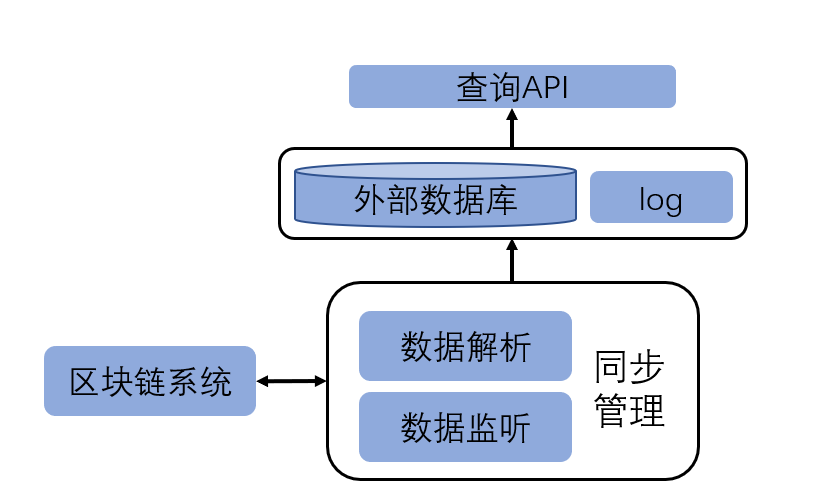


图1 外联数据库方法

内置索引的方式在区块链系统内部建立索引结构，通过矿工共识维护索引结构，构建一个查询层，图2是内置索引方法的结构。

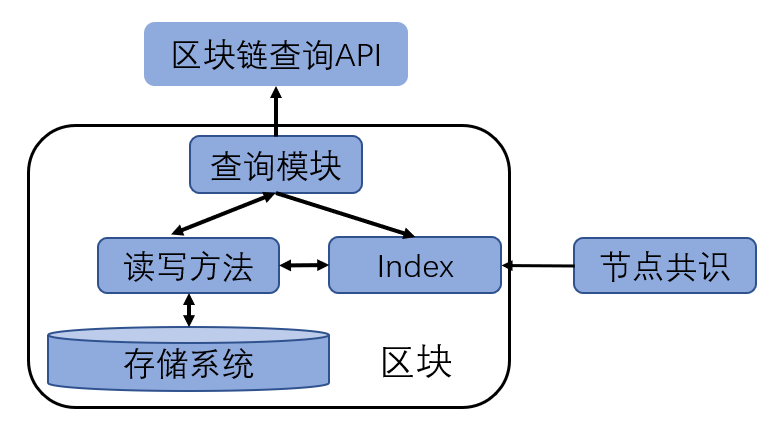


图2 内置索引结构

这两种扩展方式各有利弊，外联数据的方法扩展性强，查询效率高，但需要有一个可信的中心机构提供搜索服务，去中心化程度低，如果想要提供可信的搜索功能，需要同步提供对搜索结果的完整性证明。内置索引的方法由于索引结构经共识得到，因此搜索结果安全可信，但实现的技术难度大。

**2.1.4 跨链技术**

传统的区块链系统大多是独立存在的，链与链之间的共识协议不相同、数据账本不共享，这样会导致价值孤岛和数据孤岛的存在，但随着应用的发展更加广泛和深入，链链互联的需求越来越迫切。跨链技术就是为了解决区块链系统的价值孤岛和数据孤岛而产生的，为了实现区块链间的相互合作，必须通过跨链技术来实现不同链之间的互联互通。目前已经出现的跨链技术有公证人模式、中继链模式、侧链模式等，这些技术可以很好地解决跨链资产交易地原子性和安全性等问题。

**2.2 信息检索技术**

信息检索是用户进行信息查询的主要方法和手段，有广义和狭义之分，广义的信息检索包括信息的存储和检索，狭义的信息检索为信息的“查找”或“搜索”，目的都是针对用户的信息需求，通过信息检索的技术和方法返回满足用户需求的相关信息。在数据库出现之后，为了对数据库中存储的数据进行查询，人们在数据库管理系统（DBMS）中实现了数据库查询功能，特别是针对关系型数据库，利用结构化查询语言（SQL）可以快速查找到想要查询的数据。而随着互联网上海量非结构化数据的出现，结构化查询语言已经无法满足用户的搜索需求，因此全文索引技术[24]出现，全文索引技术是对大量非结构化数据进行搜索的关键技术，也是google，bing等搜索引擎的核心技术[25]。由于区块链系统中存放的数据是半结构化或非结构化的，所以本节主要介绍全文索引技术中使用的相关算法和技术，同时针对于跨链检索，本节还将介绍语义搜索和知识图谱中存在的相关技术。

**2.2.1 TF-IDF算法**

TF-IDF算法是一种在信息检索和数据挖掘领域中常见的加权技术，是传统的统计算法，可以用于评估一个词在一个文档集或语料库中对于某一个文档的重要性[26]。TF-IDF表示一个词的重要程度随着它在文件中出现的次数增大而增大，同时随着在整个文档集中出现的频率增大而减小。TF-IDF的主要思想是如果一个词在某一篇文章中频繁出现而在其他文章中较少出现，则认为该词对于该文章在所有文章中进行分类很重要。

TF-IDF实际上是TF \* IDF，其中，TF(Term Frequency)指词条在数据中出现的频率，计算公式如下：

其中，表示关键字i在数据j中出现的次数，分母则表示数据j中的总词数，IDF(inverse document frequency)则用于表示词条的重要程度，计算公式为：

其中N表示数据总数，是包含关键字i的数据在N中出现的次数，为了避免分母为0，经常使用拉普拉斯平滑对进行处理。

TF-IDF算法具有简单易统计且结果具有代表性的优势，但只通过词频来衡量一个关键词的重要程度还不够全面，并且无法体现词的位置信息和词性等其他重要语义信息。

**2.2.2 倒排索引**

在无结构大数据的全文索引技术中，倒排索引是最为核心的一项技术[27]，与倒排索引相对的是正向索引（forward index），正向索引是指在搜索引擎中每个文件都对应一个文件的ID，而文件的内容则被表示为一系列关键字的集合，通过文件的ID可以搜索到文件对应的关键字。当用户对语料库中的文档进行搜索时，正向索引需要遍历索引库中的所有文档，找到包含用户输入关键词的文档，再根据文档中该关键词的重要性排名后将结果反馈给用户，这样的搜索模式是低效且成本巨大的，在海量的互联网数据中使用正向索引进行搜索并不可行。因此，将正向索引进行转置后得到的倒排索引，把文件ID到关键词组合的映射转变为了关键词到文件ID的映射，每个关键词都对应包含这个关键词的一系列文件，这样，通过搜索用户需求的关键词，就可以找到于其匹配的文档。

倒排索引主要由两部分组成，第一部分是单词词典，单词词典记录了文档集合中出现过的所有单词，并对每个单词记录了一个指向“倒排列表”的指针；第二部分是倒排列表，记录了包含单词词典中某个单词的所有文档和该词在文档中的位置信息，倒排列表顺序地存放在磁盘的一些文件中，这些文件被称为“倒排文件”。倒排索引的构建方法有转置法和归并法两种，转置法主要是通过将正向索引进行拆分后转换为倒排索引的方法，该方法是顺序执行的，因此不易用于并行处理操作；归并法则是每次将内存中的数据写入磁盘时，将包括词典在内的其他所有中间结果一并写入磁盘，生成临时倒排文件，然后对生成的多个临时倒排文件进行多路归并，输出得到最终的倒排文件。

**2.2.3 检索模型分类**

信息检索模型，主要是用于计算用户搜索需求和搜索结果内容相关度的理论基础和核心组件，经典的信息检索模型有布尔模型、向量模型和概率模型[28]。

布尔模型是基于集合论和布尔代数的检索模型，查询请求基本上是由and,not和or构成的布尔表达式，可以用于多关键字检索的场景中。通常，一种多关键字搜索中关键字组合的形式为<key1,key2,key3>,用and ,or等连接词进行连接，有时也可以称之为布尔搜索，用户可能会想搜索包含某几个确定关键字或者是包含几个关键字的其中一个或几个关键字的数据。比如“Blockchain”∧(“Retrieval”∨“Search”)，表示用户想要得到包含Blockchain并且包含 Retrieval与Search两者其中至少一个的搜索结果。

向量模型也叫做向量空间模型（Vector Space Model），它把每个文档表示成一个t维的向量，这个t维的向量可以是单词、词组、N-gram片段等，其中最常用的是单词。向量空间模型的定义如下:

**给定一个文档 ，D符合以下两条约定：**

**（1）各个特征项)互异（既没有重复）；**

**（2）各个特征项无先后顺序关系（即不考虑文档的内部结构）**

**在以上两个约定下，可以把特征项看成一个n维坐标系，而权重为相应的坐标值，因此，一 个 本就表示为维空间中的一个向量，我们称 为文本D的向量或向量空间模型，如图3所示：**

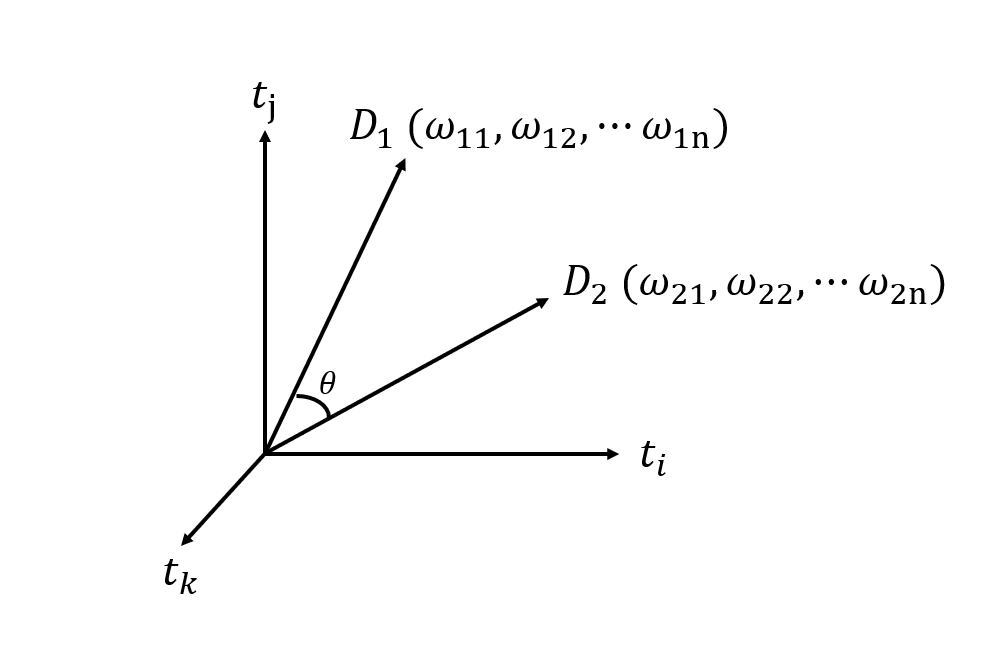


图3 向量空间模型

TF-IDF算法是向量空间模型中最常用的方法。

概率模型是基于概率假设的检索模型，它的基本原理为：给定一个用户的查询请求和语料库中的一篇文档,概率模型会尽可能的给出用户可以搜索到相关文档的概率。概率检索模型与贝叶斯分类的思想非常接近，但是存在着本质区别，概率检索模型的检索目标不是分类，而是根据查询请求和文档相关度的概率大小为文档进行排序。

**2.2.4 B+树**

B+树和二叉树、平衡二叉树一样，是一个经典的数据结构。B+树由B树和索引顺序访问方法演化而来，常用于在MySQL等关系型数据库中构建索引。B+树的定义是为磁盘或者其他存储设备设计的一种平衡查找树，是B树的扩展，可以进行有效的插入、删除和搜索操作[29]。B+树通常用于数据库和操作系统的文件系统中，其特点是能够保证其中数据是稳定有序的，B+树的插入和修改操作可以保持在比较稳定的O(log n)时间复杂度。B+树通常具有几个特性:数据只存储在叶子节点上，且所有的叶子节点位于同一层，叶子节点以单链表的形式连接从而提升查询的效率；内部节点只存储键值，且由于主存储器的大小有限，主存储器中只存储内部节点，叶子节点被存储在辅助存储器中。图4为B+树的结构，其中最下面一层为叶子节点。

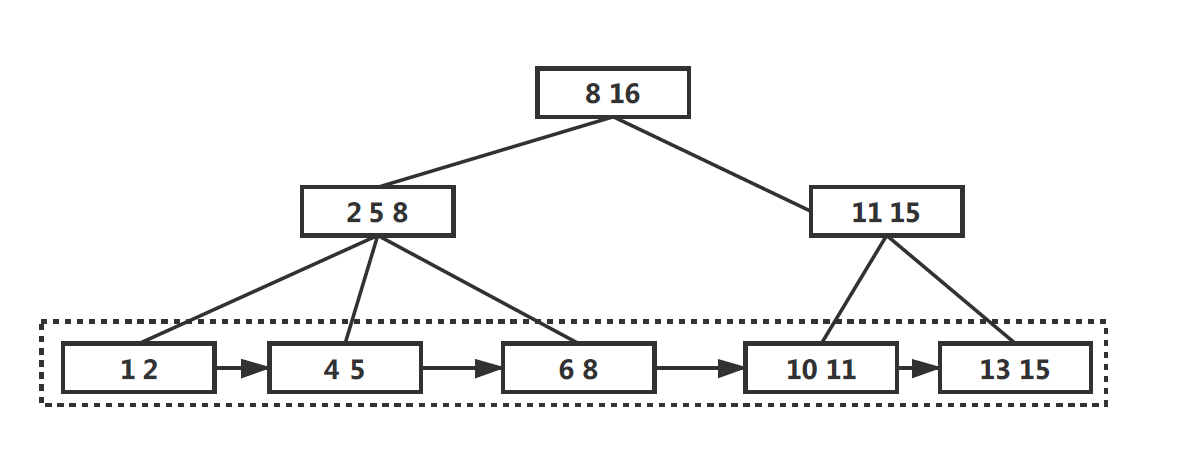


图4 B+树的结构

**2.2.5 pagerank算法**

Pagerank是一种根据网页之间的超链接来计算网页排名的技术，是Google公司用于体现网页之间相关性和重要性的关键算法，也经常被用作评价搜索引擎优化程度的指标。Pagerank算法由谷歌创始人Lawrence Edward Page和Sergey Brin在斯坦福大学共同发明[30]。Pagerank的核心思想是，如果一个网页被很多其他网页链接，则该网页的重要性比较大，pagerank值较高，同时，如果一个pagerank值很高的网页链接到另一个网页，则该网页的pagerank值也会提高。

Pagerank算法主要分为两步：

1.给每个网页设置一个初始的PR值，代表网页的被访问概率，假设网页的初始PR值为，其中N为网页的总数量，在一般的情况下，所有网页的PR值总和为1。

2.通过投票算法不断迭代，更新每个网页的PR值，举例来说，A、B、C三个网页都有进入D网页的链接，则D网页的PR值为A、B、C三个网页PR值的总和，计算公式如下：

继续进行计算，A网页除了链接到D网页之外，还同时链接到了B网页和C网页，因此用户通过A网页访问B、C、D三个网页的概率均为1/3，同理，如果B网页同时包含对A、D两个网页的链接，则进入这两个网页的概率为1/2，C如果只能链接到D网页，则C所有的跳转网页都进入了D网页，这样A对D网页的投票量为1/3，B为1/2，C为1，因此D网页的PR值计算公式变为:

PR(D)=PR(A)/3+PR(B)/2+PR(C)

经过总结，我们可以得出一个网页的PR值计算公式应该为：

PR(w) =

其中，是所有链接到网页w的网页集合，网页n是属于集合的一个网页，L(n)是网页n的出度（对外链接的数量）。

**2.2.6 RDF数据模型**

RDF（Resource Description Framework）是一种资源描述框架[33]，其中资源可以是任何文本内容，网络上的文档等，RDF是Web中进行数据交换的标准模型，可以在底层数据格式不同的情况下进行数据合并，有利于在网络中形成人机可读，并且可以由机器自动识别的文档。RDF扩展了Web的链接结构，使用URI(统一资源标识符)来定义和命名事物之间的关系，并通过一种简单的二元关系模型来表示事物之间的语义关系，即RDF三元组。RDF三元组常用来表示实体之间的关系或者实体某个属性的值，结构为或。RDF三元组互相连接可以构成RDF图，一个RDF图的实例如图5所示:

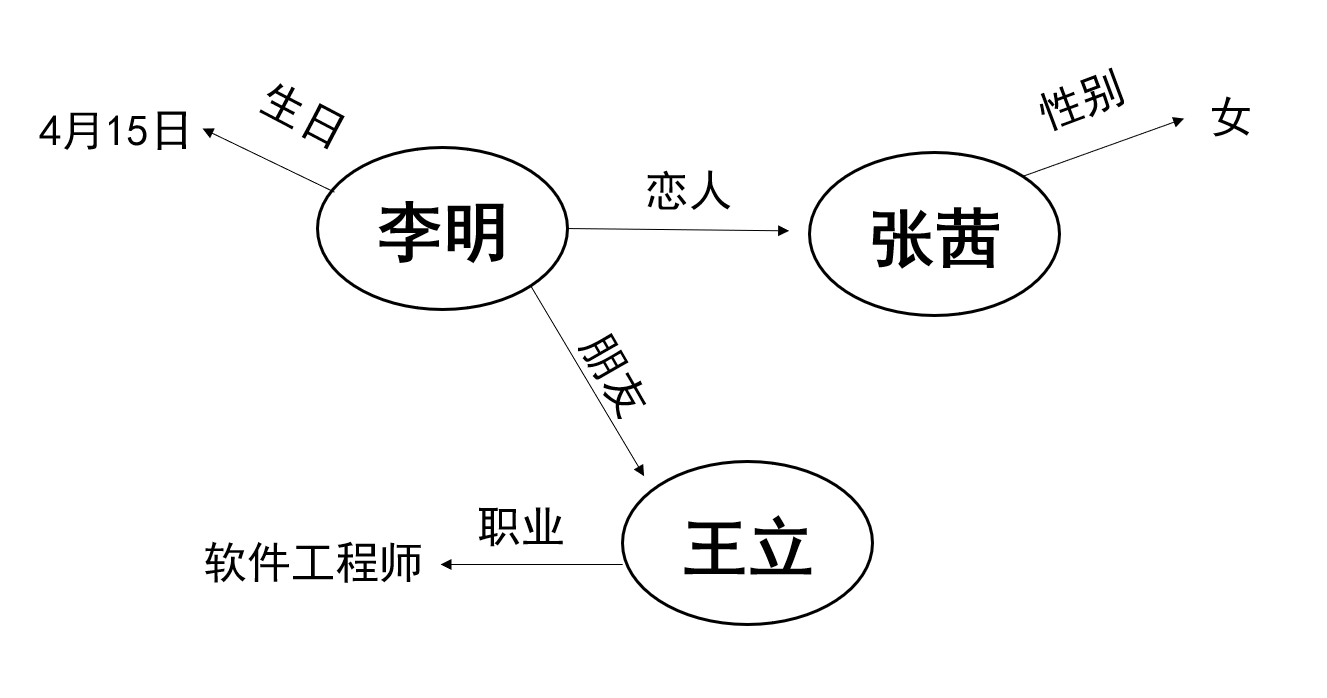


图5 RDF图实例

**2.2.7 知识图谱构建技术**

知识图谱实质上是通过关系（边）将语义实体链接起来的语义知识网，以图的形式呈现，并可以对存储的知识进行推理和进一步分析。知识图谱通过将不同类型的数据整合为人们易于使用的结构化数据，目前已经被广泛应用在语义搜索、智能问答、情报分析、金融风险控制等领域。知识图谱构建技术主要分为知识抽取、知识表示、知识融合和知识推理四大核心步骤[34]。

知识抽取主要的研究内容是如何自动或半自动地从不同异构的数据中抽取实体、属性、关系等知识要素。知识表示通过互相关联的三元组等形式对抽取出来的知识进行较为合理的表示。知识融合主要是将多个不同知识库中的知识进行整合，形成一个统一的知识库，主要解决的是实体对齐的问题，将不同知识库对实体的描述进行统一化的整合，获得对一个实体较为完整全面的描述。知识推理则是根据已有的数据模型和数据，挖掘出隐含的知识或识别出错误知识，对现有的知识图谱进行扩充和纠正。除了这四个核心步骤之外，还有知识存储、质量评估等步骤，知识图谱的构建是一个系统化的工程，涉及到各种信息检索或自然语言处理的知识，目前的技术仍不算成熟，还有很多挑战需要研究人员去解决。

**2.3 本章小结**

本章根据第一章介绍的区块链信息检索的研究背景和基本现状，从区块链系统和传统的检索技术两方面介绍了本文相关的理论知识。首先介绍了区块链系统中与查询相关的结构和技术，体现了区块链查询模式的特殊性，随后分类介绍了传统信息检索中使用的不同技术，并分析了它们在实现查询功能中的优劣势，以探索其与区块链结合的可能性。这些相关知识为本文后续方案的设计提供了坚实的理论和技术基础。

**第三章 链上链下混合存储架构中高效搜索技术方案**

**3.1 背景介绍**

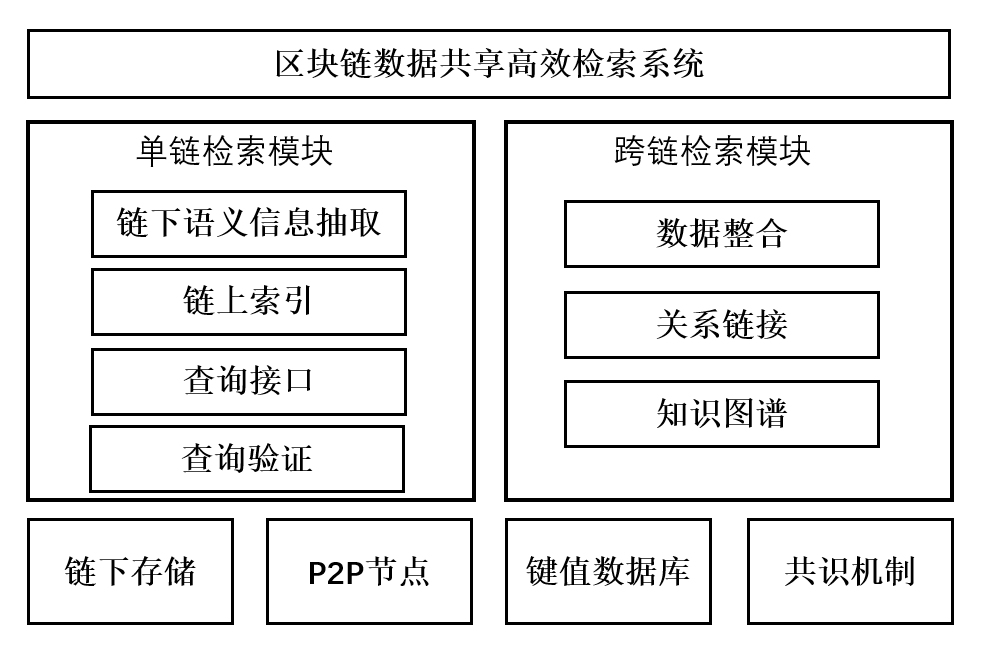
最近几年，区块链作为去中心化的可信数据库在多个领域受到了广泛关注，由于其多方参与、信息公开透明、交易可追溯等特点，往往被用于存放大量有价值的数据，区块链的节点之间可以对这些数据进行安全的共享。目前主流的区块链数据存储模式已经有单一的链上存储转变为了链上存储和链下存储结合的模式[35]，针对链上链下混合存储的数据提供实时的搜索功能是当前需求的重点。同时，为了防止数据孤岛的产生，在不同的区块链系统中进行跨区块链的搜索也是研究的难点之一。

本篇论文提出了一种通过建立索引结构在链上链下混合存储架构中进行分布式搜索的方案，并通过知识图谱的构建来实现跨链搜索的功能。本章首先通过结合区块链分布式数据库和传统索引技术、知识图谱技术以及语义知识共享背景，提出了一种新颖的针对区块链数据共享系统的高效检索方案，随后详细介绍了系统的架构设计，总体工作流程，具体模块设计，并对方案的检索效率和安全性进行了定性分析，以体现本文方案的优越性。

**3.2 系统设计**

**3.2.1 整体架构**

本文提出的在区块链数据共享系统中进行高效检索的方案是、基于多节点和多链的去中心化网络结构，方案的整体架构如图4所示，其中底层的P2P节点、键值数据库、共识机制等是区块链原有的技术，为检索模块的构建提供环境。链下存储模块是用户独立的存储空间，为检索模块提供内容支持。整体系统主要分为单链检索和跨链检索两个部分，两部分都遵循去中心化的特性，单链检索又分为链下语义信息抽取模块、链上索引构建模块和单链查询模块，跨链搜索分为数据整合和查询两个模块。在两个模块之上，本方案向需要检索服务的用户提供了单链检索和跨链检索的接口，单链检索通过rpc接口调用的方式，跨链检索则提供了可视化的知识图谱。

图4 整体架构图

**3.2.2 问题定义**

在本文提出的系统中存在单链搜索和跨链搜索两部分，在这两部分中用户的搜索需求也不完全相同，本节分别针对这两部分的搜索需求来对本方案中的搜索模型进行分析和形式化的定义。

第一部分是单链搜索部分，在单链搜索中用户最常用的功能就是关键字搜索功能，包括单关键字搜索、多关键字搜索和模糊关键字搜索等，这也是搜索引擎的常用功能之一，但现有的区块链系统并没有实现这种关键字搜索的功能。通过我们的链上索引结构，这种关键字搜索成为了可能，通常，一种多关键字搜索中关键字组合的形式为<key1,key2,key3>,用AND OR等连接词进行连接，有时也可以称之为布尔搜索，用户可能会想搜索包含某几个确定关键字或者是包含几个关键字的其中一个或几个关键字的数据。比如“Blockchain”∧(“Retrieval”∨“Search”)，表示用户想要得到包含Blockchain并且包含 Retrieval与Search两者其中至少一个的搜索结果。

第二部分是跨链搜索部分，与单链搜索不同，跨链搜索中用户更关注信息之间的“关系”，搜索的对象往往是以实体的方式存在，而不同的实体之间存在着关联，例如，用户可能想要搜索“比特币的创始人是谁”，其中“比特币”是一个实体，“创始人”则是实体之间的关系。另一方面，同一个用户可能在多条链上都有与其相关联的数据，由于每个链的应用领域可能不同，使得这些数据之间的关联度不大，但在有些应用场景比如对一个人进行信用评估时，可能需要对他在各个领域的行为都进行收集，这也就需要跨链的搜索服务。

**3.2.3 性能指标**

在本文的方案中，最主要的功能是对信息的高效检索，不同搜索功能的响应时间是系统的核心性能指标，同时，考虑到区块链系统的去中心化特性，没有一个可信的节点对整个系统进行统一管理，因此搜索结果的完整性和正确性是我们方案需要考虑的主要问题。

* 响应时间：针对不同的搜索需求，系统的时间消耗是否是可以接受的。
* 正确性：搜索结果是否返回期望的搜索结果。
* 完整性：搜索结果是否返回全部的搜索结果。

上述三个指标是单链搜索和跨链搜索的通用性能指标，需要注意的是，在跨链检索模块中，知识图谱的构建速率和搜索结果的排序能力也是非常重要的指标，因此在跨链检索模块中需要对这两个指标进行重点测试。

在安全性方面，由于拜占庭问题的存在，系统中可能存在恶意的矿工节点，恶意的矿工节点可能会对链上的索引结构进行恶意的修改，进行搜索的全节点也可能将不完整或者具有误导性的搜索结果返回给轻节点。因此，为了保证搜索结果的安全可靠，本文的方案需要设计一种针对轻节点的查询结果验证方法，以保证轻节点接收到的查询结果是正确可靠的，轻节点只需要在本地即可对全节点提供的查询结果进行验证而无需进行其他的操作。

**3.3 方案描述**

**3.3.1 系统总体工作流程**

本节将在3.2.1节给出的系统整体架构的基础上，介绍本方案设计系统的总体工作流程。图5是我们方案的总体工作流程，

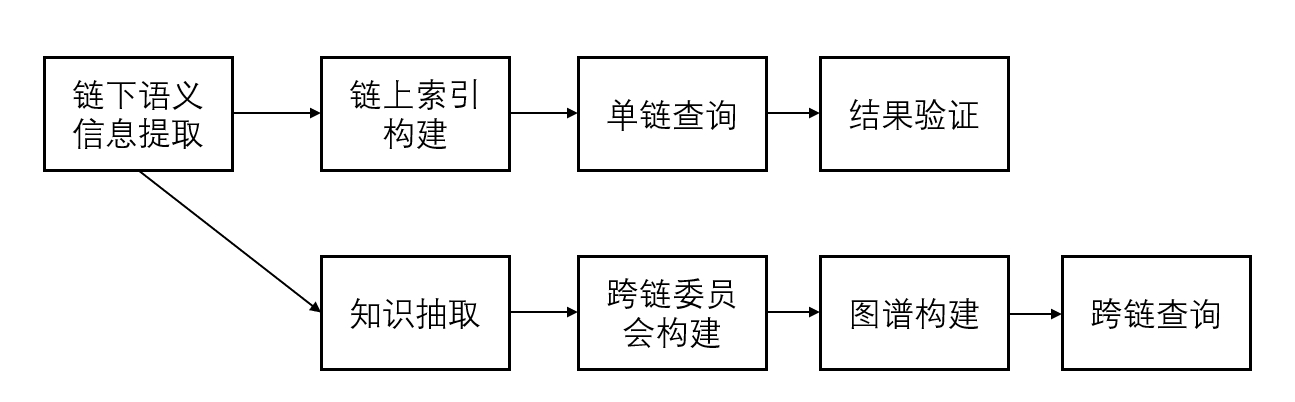


图5 总体工作流程

由于单链检索和跨链检索的步骤不同，本节将分别对单链检索和跨链检索的工作流程进行描述，由于数据存储在链下，因此单链搜索和跨链搜索第一步均需要对链下的语义信息进行提取，由于单链检索和跨链检索的工作流程并不相同，在这里通过两个架构流程图对两种检索方式分别进行介绍，图6显示了单链查询的工作流程，用户首先要从链下存储的数据中抽取出语义信息，以交易的方式提交到区块链上，收到交易后全节点们通过共识机制构建并维护一个全局的索引结构，通过索引结构提供对链上链下混合存储架构中数据的检索功能，所有的全节点均可在自己本地通过链上的索引结构进行检索，而只包含区块头部的轻节点需要连接全节点来进行代理检索，全节点收到代理检索请求后，将检索结果和验证对象返回给轻节点，轻节点获取结果后在本地即可对检索结果的正确性进行验证。

图7是跨链检索的工作流程，首先通过与单链检索相同的方式对链下数据的语义信息进行抽取，在每条单独的链上构建索引，然后通过知识抽取的方式对不同格式的数据进行整合，整合为统一的格式之后在每条链上抽取一部分节点作为跨链委员会，不同链上的委员会之间组成一条知识链，通过委员会间的跨链共识构建跨链知识图谱，最终通过知识图谱提供跨链检索的功能。

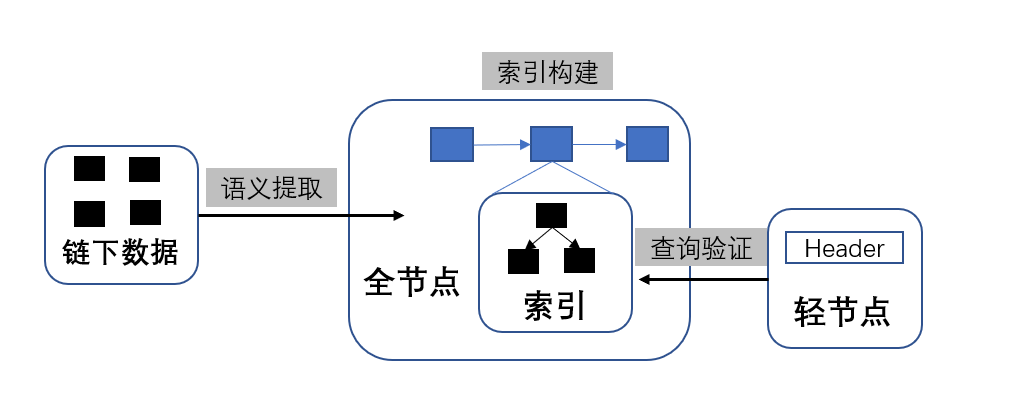


图6 单链检索工作流程

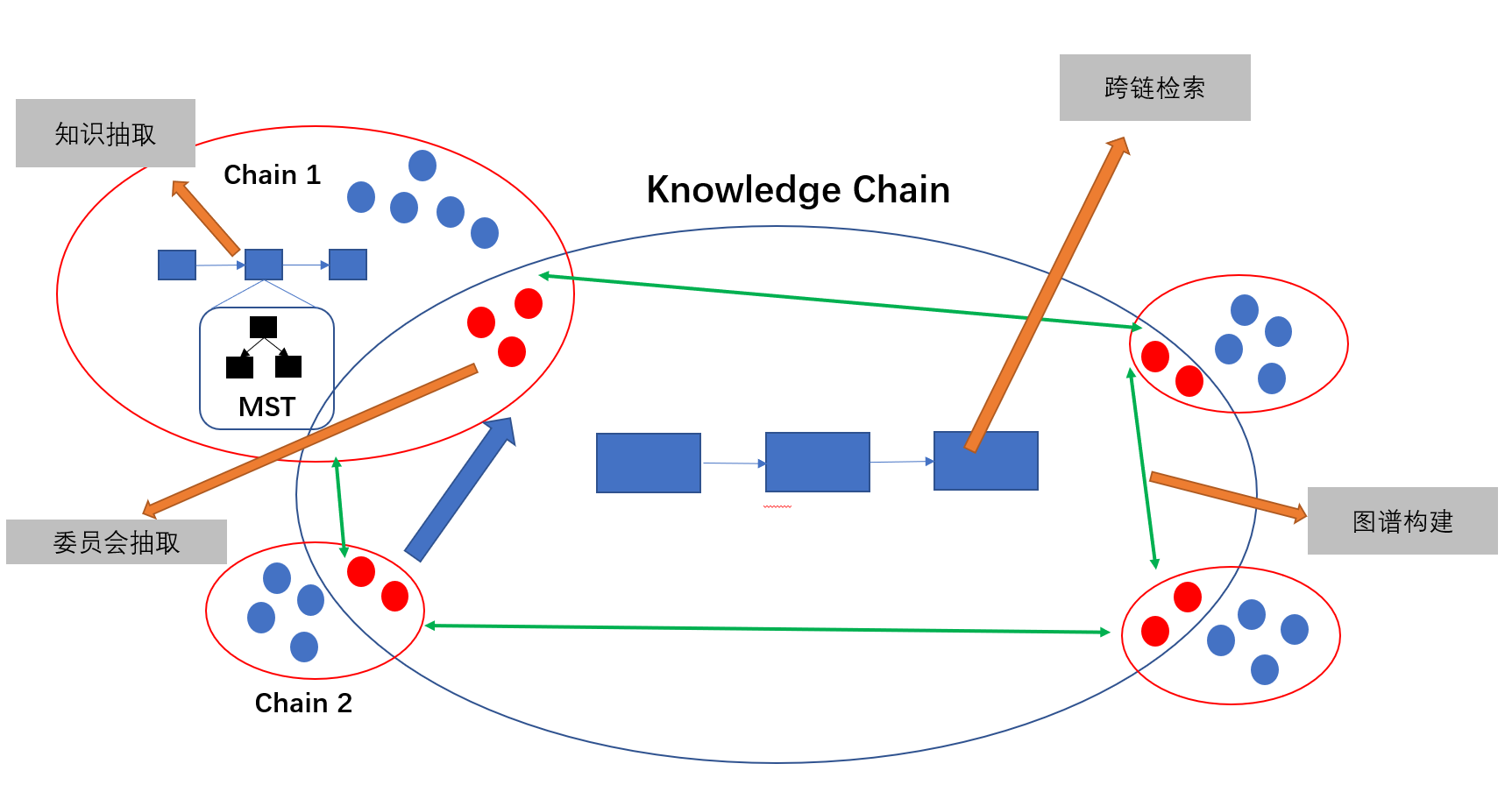


图7 跨链检索工作流程

**3.3.2 链下语义信息抽取模块**

为了建立链下数据和链上数据的协同映射关系，首先，本方案中的数据共享节点对链下信息的语义信息进行提取，我们提出了一种改进的TF-IDF模型，我们称为去中心化语义提取（Decentralization Semantic Extraction, DSE）算法，通过引入去中心化词频因子、词性因子和位置因子来提升语义特征提取的准确性，解决了传统关键字提取算法中缺乏语义理解的特点，并基于此构建了一个关键字的语义倒排索引。

TF-IDF在2.2.1节中已经介绍过，在这里我们用代表TF，代表IDF，传统的TF-IDF的计算结果为和的乘积，但TF-IDF仅能通过频次来对关键字进行权重计算，在链下数据中，由于关键字对数据分类的贡献大小不一，我们采用去中心化词频因子(DecentralizationTerm Frequency Factor,D-TFF)对对关键字出现的频率进行去中心化处理，得到的公式如下：

其中为关键字i在数据j中的出现次数，为关键字i在所有数据中出现的平均次数。通过D-TFF的去中心化处理，可以提高专属词的权重大小。

再者，为了对词性的权重进行区分，我们将词性的权值进行定义，不同词性的关键字权值定义如下：

另一方面，大多数数据在开始和结束时包含的内容往往更为重要，应该给与更高的权重，因此我们将数据中关键词第一次出现的位置排成一个序列，以数据中总字数为总长度，以1为单元刻度，将序列最中间的位置定为初始坐标，计算其他词距离初始坐标的距离，距离越远，权重越大，定义位置影响因子（Position factor）公式如下:

其中，为增加的权重倍数，d为词i距离初始坐标的距离，∈[0,L/2),L为序列的长度。

因此，我们的特征词权重的计算公式为：

随后，矿工维护一个全局的倒排列表，包含三部分：key word，blockNumber，W，值得一提的是，在这里我们没有用数据marker而是用了BlockNumber，这是因为我们的索引结构最终会有一个指向数据所在区块的哈希指针，BlockNumber则是我们的主要搜索内容。

**3.3.3 链上索引构建模块**

在对链下的数据进行了抽取之后，我们以抽取的链下信息为基础，在链上的区块身体中建立一个索引结构，这个索引结构又矿工们共同维护，随着新区块的生成而更新，并始终在最新的区块中保持最新状态。

在我们的搜索框架中，链上索引结构跟随创世区块生成而随之生成，并随着新区块的生成而随之改变。3.3.2节中，我们提取出了链下数据的语义信息并构建了一个关键字的倒排列表，利用倒排列表，我们可以对单个关键词进行快速实时查询，而为了将搜索功能扩展到多关键字，我们构建了一个新型的名为默克尔语义搜索树(MST)的索引结构，通过结合了倒排列表、merkle tree、哈希指针等技术，并利用了前缀压缩等特性，可以更高效地对链上数据进行多关键词搜索。由于区块链的追加性质，MST结构是动态构建的，每生成一个新的区块就会对MST的结构进行更新。

在MST中存在三种不同的节点：分支节点、扩展节点、叶子节点，每个节点只能成为其中一类节点，节点详细结构在图8中展示。分支节点只存放索引相关信息，并提供指向下一层节点的指针；叶子节点位于MST的最底层，代表了一条搜索路径的结束，搜索结果存放在叶子节点中，结果为一个指向数据对应交易所在区块的哈希指针；如果一个节点是分支节点，则除了包含指向下一层的指针之外，节点本身也包含对应的搜索结果,不同节点的类型图如下，rootNode是一个比较特殊的节点，如果MST中只有rootNode一个节点，则rootNode为LeafNode，如果有了孩子节点，则为branchNode。

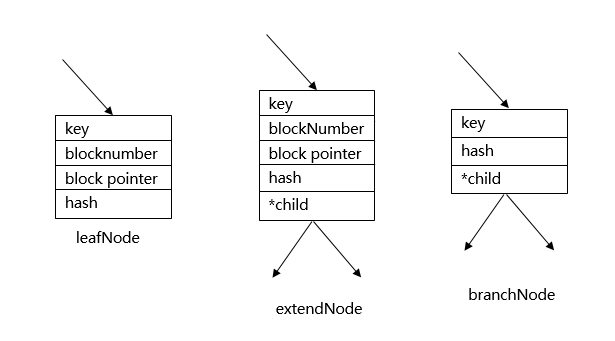


图 8 节点类型图

对于倒排列表中有序的关键词集合，矿工们会依照词权重W的高低依次将<blockNumber,key word{key1,key2…}>对插入MST中，每次插入时都需要从root节点开始插入，root节点的插入算法如下：

Algorithm 1: MST root-node insert

keyword must be inserted by ­

Input: as indexInfo(blockNumber and a list of key word) and in as inverted index

1. .sort by in
2. for all in list do
3. true <- flag;
4. for all ∈ root.child do
5. If == root..
6. false <- flag;
7. root..insert ();
8. end for
9. if flag == true do
10. root.child.append(new node(info));
11. if len(root.child) == 1
12. true <- root.isExtend;
13. end for

为了解释上述算法中的排列，我们用一个数据集的例子来模拟一个插入和查询过程，如果你想要插入的式子为：{“blockchain” ∧(”query”∨”search”)},可能的查询为{“blockchain” ∧”query”}或者{“search” ∧“blockchain”}，通过我们的DSE模型，我们消除了这种歧义，我们根据词权重的值对搜索语句中输入的关键字组合进行排序，将权值较大的关键字排在前面，防止了对关键词组合进行递归插入操作。

MST对关键字的存储利用了字典树(trie)中的共同前缀思想，在对关键字列表进行插入时，将具有相同前缀关键字组合的数据保存到同一路径下，并结合Patricia Trie[23]的思想，将只有一个子节点的路径进行压缩，充分释放了存储空间，MST的具体结构如图9所示。



图9 MST结构

类似于merkle tree，MST的每个节点都有一个hash value，对于叶子节点来说，这个哈希值是对叶子节点中存储的关键字以及区块指针进行的一次散列运算，运算的公式为：

对于分支节点来说，这个哈希值是对关键字及其孩子节点的哈希值进行的一次散列运算，运算的公式为：

对于扩展节点来说，这个哈希值是对关键字、区块指针以及孩子节点的哈希值进行的一次散列运算，运算的公式为：

对于节点之间的关系，父节点通过子节点的哈希值找到对应的子节点，root节点保存的根哈希值作为MSTrootHash字段存储在区块的头部。

为了节省存储空间，我们不在区块中存储与倒排索引相关的信息，因为MST中包含所有的关键词相关信息，如果作为一个新加入的节点，你想要恢复倒排列表，可以在本地利用MST进行恢复，或者连接另一个全节点同步他的倒排列表，然后用自己的MST进行验证。

在算法1中，我们对根节点的子节点进行了插入操作，针对节点的插入操作，我们用下面的算法进行描述：

Algorithm 2: MST node insert

Input:info as indexInfo

Output:node

1. ln <- len(node.key)
2. li <- len(info)
3. while ∃i: i≤ln and i≤li
4. for each ∈info.key ,each ∈ node.key do
5. if ≠
6. go to split;
7. end for
8. end while
9. if li < ln
10. extendnode <- node
11. go to split;
12. if li = ln
13. switch: node type
14. case: LeafNode and extendNode
15. node.value.append(blockNumber)
16. break;
17. case: branchNode
18. node.value.append(blockNumber);
19. extendNode <- node;
20. break;
21. if li >ln
22. if node.type == leaf
23. extendNode <- node;
24. end if
25. node.child.append(info[i:li]);
26. split: node.child.append(info[0:i]) and node.child.append(info[i:li]);
27. end

加入了MST索引结构之后，我们的系统已经可以较好地支持多关键词搜索和模糊搜索，由于底层采用了levelDB数据库，levelDB数据库底层采用了LSM（Log Structured-Merge Tree），并没有天然支持范围查询的属性，但由于MST的索引结构需要从根节点搜索到叶子节点，所以我们在没有修改底层存储模式的情况下加入了B+树特性，将MST的节点顺序进行再次排序，使得每个节点在对应一个关键字的Key值的同时，还具有B+树的范围key值，加入了B+树特性的MST结构如图10所示：

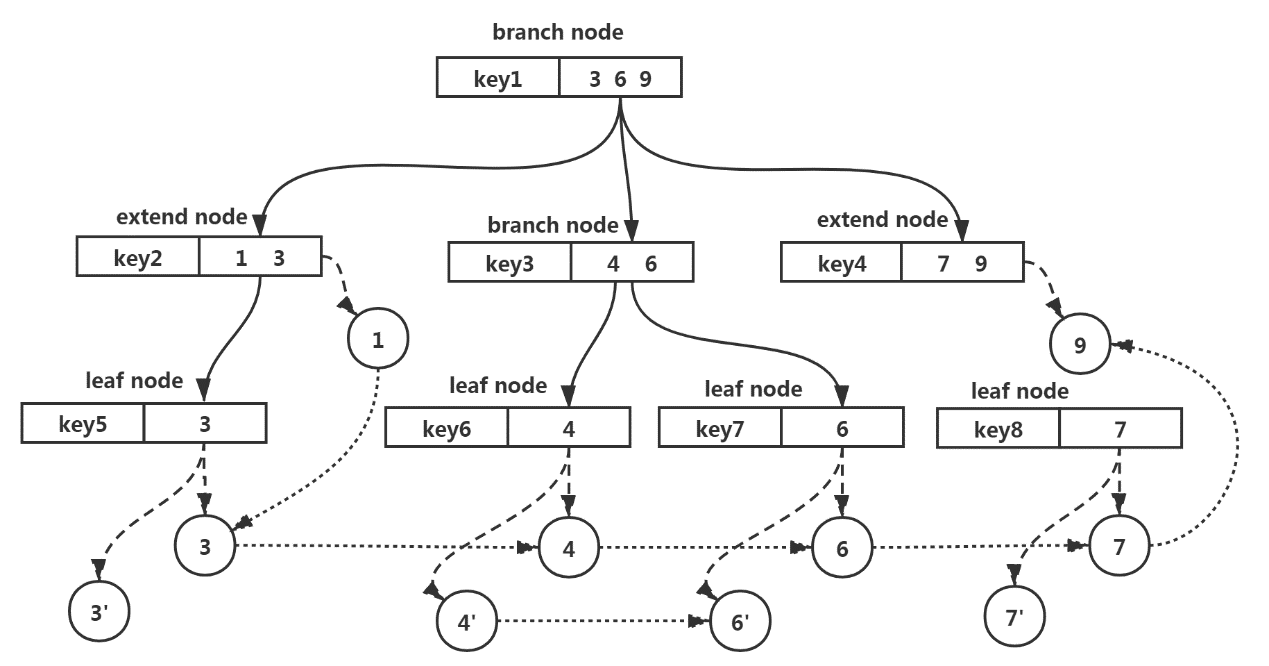


图 10 MST-B+ tree结构,由1到9的链为关键字key1的B+树有序节点链，4’和6’的节点链是关于关键字key3的B+树有序节点链

与传统的B+树相比，我们的MST-B+ tree结构是关键字-数据量强耦合的，因此不需要对磁盘进行频繁的IO操作，只需要矿工针对每条路径上的相同关键字对应的文件范围进行一次排序即可。

**3.3.4 单链查询模块**

在获取到用户的搜索输入之后，我们需要将用户输入的关键字组合与MST中的索引路径进行匹配，并将匹配结果返回给用户。在匹配的过程中我们用到了Multiple keywords Aho-Corasick automaton匹配算法，与KMP[36]类似，但我们的系统中的公共前缀并不是一个相同的字符串，而是一组相同的关键字集合，首先，我们将MST映射为图5所示的AC自动机，图中用双圆圈表示的状态是接收状态，对应的是MST中的扩展节点和叶子节点。在这里我们改进的一点是，如果你想要进行模糊查询，并且在没有到达终态前所有关键字就已经匹配完毕，则到达的该状态变为模糊接收状态，此时自动机会返回该状态之后的所有结果。因为输入关键词顺序已经经过倒排列表进行过排序，因此匹配过程中不会出现递归匹配的现象。在提出多关键字匹配算法之前，我们先提出以下相关概念：

定义1 自动机MACM是一个五元组M=(Q, Σ, δ, , F)，Q={}是有穷状态集，对应MST上的所有节点，Σ是单词表，即系统中的倒排列表，δ是转移函数，输入一个新的关键词通过转移函数可以转到下一个状态，是MACM的初始状态，对应MST上的root节点，F是MACM的中止状态，对应MST的叶子节点和扩展节点。自动机的结构如图11所示。

定义2 对于状态∈Q,pref()表示的所有前驱状态，即pref()={} ,suff()表示的后继状态，即suff()={}



图11 AC自动机的结构

多关键字匹配算法：

Algorithm 3: MACM match

Input:MACM m, MST mst，query list key=<K>,query type: certain or fuzzy

Output:result blockNumber=<R>,VO

Let state is the initial state of m

1. state <-
2. for each in K do
3. if δ(state,) ∈Q then
4. state <-δ(state,)
5. if state ∈F
6. goto Result
7. if K.next.isempty and type == fuzzy
8. goto fuzzyResult
9. else
10. return NULL
11. Result: node <- MACMToMSTInSamePos(state)
12. Add all node.value to R
13. create ProofList
14. pathNode <-node
15. node <-node.parent
16. while node ≠ rootNode
17. for each node.child except pathNode do
18. Add node. to ProofList
19. pathNode <- node
20. node <- node.parent
21. VO <- ProofList,mst.rootHash
22. return <R,VO>
23. fuzzyResult: create a <R,VO>list RVO
24. for all suff(state) do
25. Add result(suff(state)) to RVO
26. return RVO

利用MACM match算法，我们可以得到关键字搜索结果R和一个搜索结果的验证对象(verification object)VO，得到了查询结果R和VO之后，用户需要对查询的结果进行验证，首先需要对查询结果进行SRV(简单检索验证)验证，利用查询结果和VO计算出查询结果对应的MSTrootHash,并与本地blockheader中的MSTrootHash进行对比，如果匹配，则证明全节点返回的是合法的查询结果。因为查询的结果是区块指针而不是具体的交易信息，用户需要继续向全节点请求查询结果对应的相关区块，为了快速定位到交易在区块中的位置，采用了布隆过滤器的快速匹配方法，利用布隆过滤器来定位交易的具体位置，布隆过滤器的用法在本节下文中进行介绍。

在MST建立成功之后，我们已经可以通过RPC调用的方式实现数据的查询操作，像是Json-rpc和grpc等，但为了给轻节点以更好的查询体验，我们设计了一套智能合约的具有RESTful风格的查询接口，智能合约接口如下所示:

**searchByKeyword()**

**searchByRange()**

**searchByFuzzyInfo()**

**verifySearchResult()**

在区块链中加入了一个索引结构，会不可避免的增加区块链存储的成本，为了避免索引结构不断膨胀导致区块结构爆炸，我们采用了对结构的优化，有效地减少了索引结构的存储成本。优化结构包括节点指针和布隆过滤器两部分。

第一个优化结构为节点指针，由于MST包含区块链上所有的索引信息，随着索引信息的不断添加，索引结构的规模会不断增大，但每个区块内矿工们可能只对MST的部分结构进行更新，因此我们设计了一个节点指针的结构，如果在一个区块内，MST的部分结构没有进行更新，则不需要在此区块中保存该部分结构，只需要令该部分子树的父亲节点添加一个节点指针，指向的位置为该部分子树最后一次发生变化的位置，这种方式有效地减少了块内的存储成本，加入了节点指针的区块链描述如图12所示:

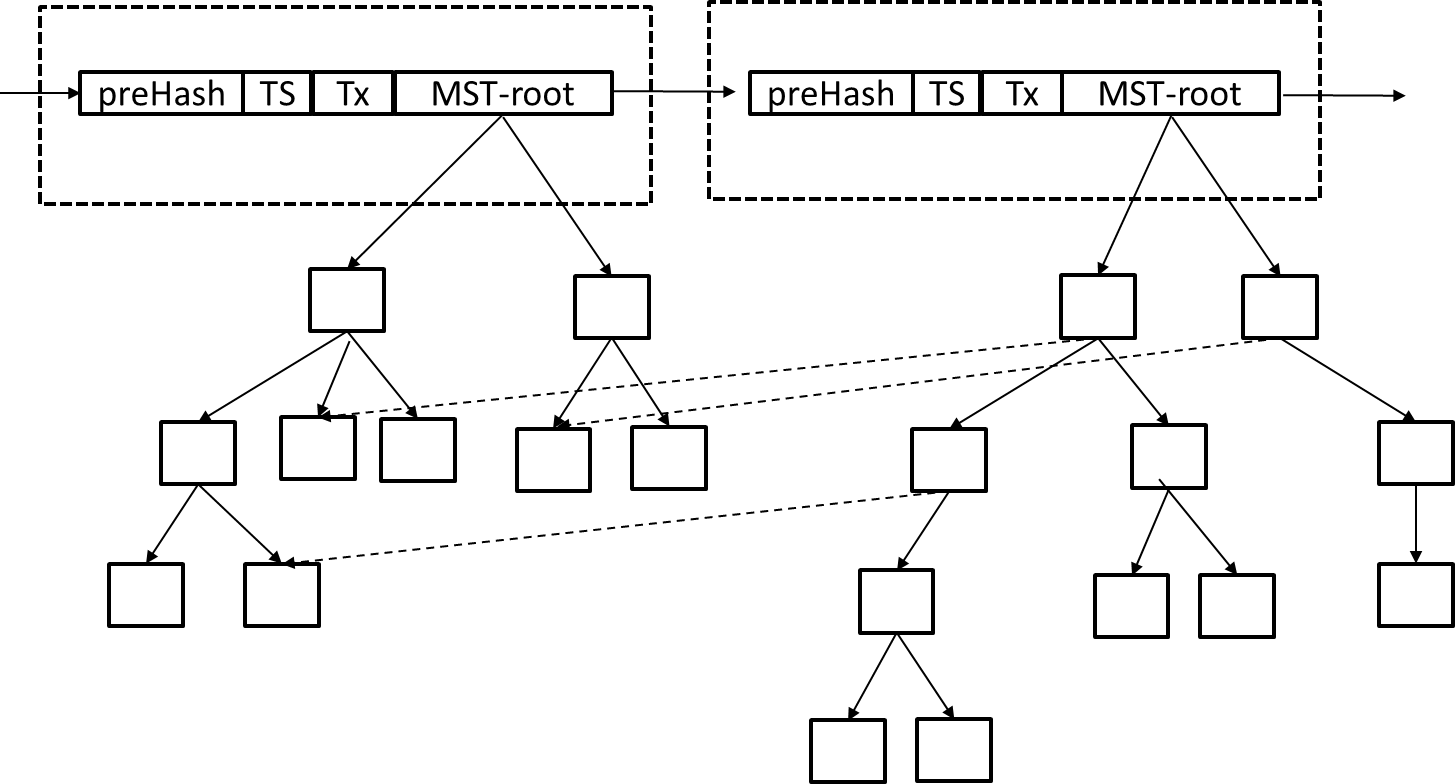


图12 加入了节点指针后的结构，图中的虚线为父节点的节点指针，指向的是在MST状态变化后没有发生改变的子树部分

第二个优化结构是布隆过滤器，MST的结构导致了当有之前没有出现过的新关键字插入时，MST的结构可能会发生较大改变，为了快速判断一个待插入的关键字是否是新关键字，我们采用了一个布隆过滤器的结构，我们为倒排索引中的关键字列表映射了一个bit vector，叫做bitmap，对于每个关键字我们都进行k次不同的哈希操作，然后将哈希值模128后得到的数据位在bitmap中置1，当一个的新的关键字来的时候，我们对进行多重哈希操作：

如果结果为1，则该关键字很可能为一个已经存在的关键字，布隆过滤器的结构如图13所示：

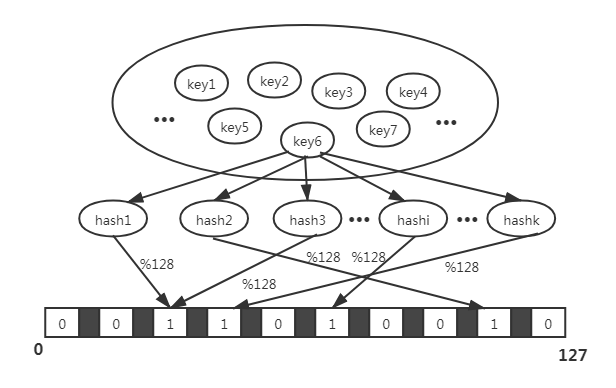


图13 关键字布隆过滤器

**3.3.5 跨链数据整合模块**

我们的方案支持在异构链系统中进行跨链的信息检索，在跨链检索模块中，由于异构区块链中的数据结构如交易结构和区块结构等都不相同，因此首先需要对不同链上不同格式的数据进行整合，整合为支持构建索引的统一格式，并以此为基础在多链间构建索引。

在2.2.6节的介绍中可知，RDF是一种对非结构化数据和半结构化数据进行格式统一的有效手段，因此在我们的方案中选择通过RDF构建实体三元组来进行异构链上的数据整合，步骤如下：

1.我们在每条独立的区块链上建立MST索引结构，使其中的数据以关键字组的形式存在；

2.定义实体三元组抽取规则，包括两部分，实体抽取和关系抽取。其中实体抽取为定义实体集合{ ，，…}，结合单链检索模块中的分布式语义提取算法，根据词性和相关度进行分类，将以名词词性为主的关键词集合按照文档相关性从高到低进行排序，作为可能的实体集合；关系抽取与实体抽取步骤类似，通过分布式语义提取算法和相关度排序得到可能的关系集合{，，…}…，然后遍历整个关键词组，找到某个关系的所在位置，从该关系所在的位置向左右两边寻找距离最近的实体词，得到实体三元组<>。

3.关系抽取还有一个特殊情况存在，就是属性抽取，属性抽取是一种特殊的关系抽取，因为属性抽取是对实体的某个属性的值进行抽取，三元组中的第三部分属性值是一种特殊的实体，是不能与其他实体产生关系的实体，因此需要在实体集中手工判断并筛选出一些可能为属性词的实体集合{，，…}，找到属性值之后，向前搜寻与该属性值最近的实体值，并以实体值和属性值之间的关系作为属性关系，构建属性三元组<>。

经过我们的数据整合之后，支持跨链检索的区块链上的交易格式变成了图14所示的模式，交易1为提交实体属性三元组的交易，交易2为提交实体关系三元组的交易。

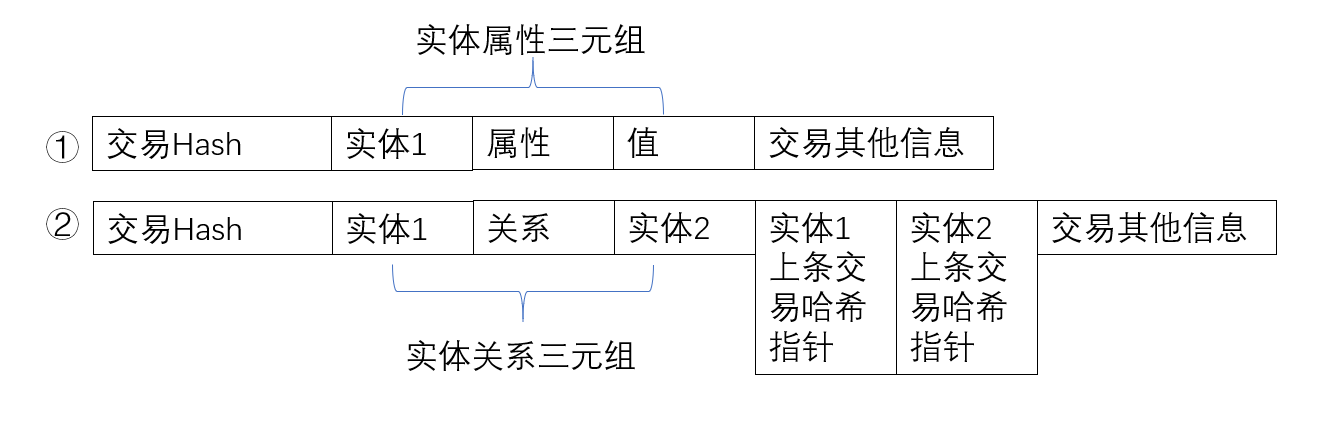


图14 跨链数据整合后的交易格式

**3.3.6 跨链委员会组建模块**

在对每条链上进行RDF三元组抽取之后，结构不统一的问题已经解决，需要解决进一步的共识不统一的问题，要进行跨链检索，就必须在多条链间构建一个可被所有链上的节点共识的统一索引结构。为解决共识不统一的问题，我们在多条链间构建了一条知识链，知识链由每个独立链上抽取的部分节点组成，通过知识证明共识机制来构建一个基于RDF三元组的全局索引结构。我们使用委员会抽取的方式在每条链上按照一定比例抽取部分节点参与知识链的共识，并隔一段时间进行委员会成员的替换,跨链共识委员会的具体组建方法如下:

由于信息收集侧链以域（domain）为单位进行划分，所以我们的中继链节点组成，也是以域（domain）为划分。即，来自同一个区块链的节点，属于同一个域，共享同一个域编号（源链编号ID）。

参考欧盟委员会形成机制，我们对中继链组成过程做如下约束：

数学定义：令，其中，令。表示第i个域。

1.委员会总人数为c；

2.各个区块链系统的节点数分配为：

3.各区块链系统，按照节点配额，选举节点作为委员会节点。

说明：为了保证每个区块链系统至少有一个节点加入委员会中，我们选择向上取整。

每个区块链自己选举出节点，我们推荐类似于董事会的PoS机制来选举出各自的节点。也就是比重越大越有机会成为中继链中的节点。

具体实现路线是：实现一个周期选举机制。当中继链中的周期运行到快要结束时，由当前出块者节点（领导者）触发重新选举机制。然后每条区块链中的节点将收到重选信号进行下一轮的竞争。

当跨链委员会构建完成之后，委员会中的节点开始对知识链中的交易进行处理，首先进行的是知识关联，委员会节点们首先将交易中提交的关系进行知识关联，将三元组组合映射为图结构，映射的过程如图15所示：

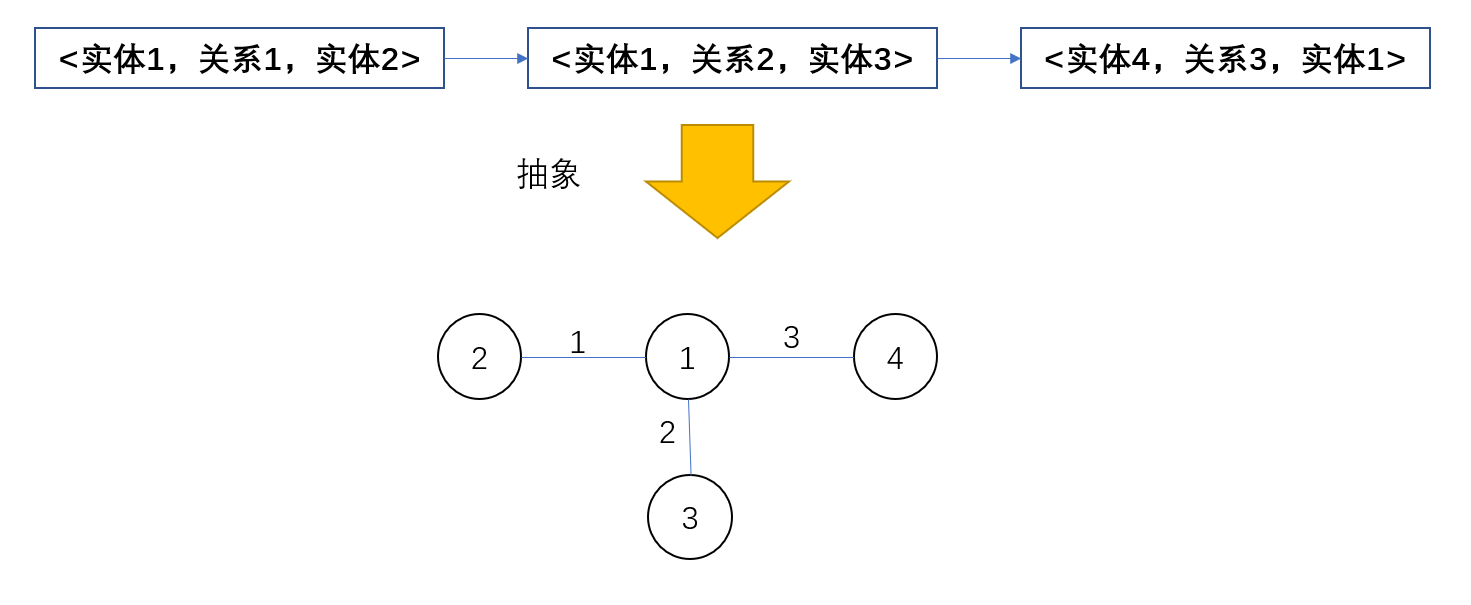


图15 实体三元组映射为图结构

然后我们采用一种改进的pagerank算法对交易构成的知识图进行处理，将交易中的实体映射为一个有向网络图，实体作为图的节点，关系作为图的边。具体过程如下：

1.为每个节点分配初值，值为节点实体在源链上的TF-IDF相关性的值；

2.计算节点的rank值，由节点的入度和出度及边上关系在源链上的TF-IDF相关性的值决定；

3.重复迭代。

**3.3.7 跨链查询模块**

主要的内容是对知识链中节点生成的知识图进行存储，并实现用户查询的接口。由于区块链底层的数据库大多为K-V键值型数据库，不能很方便地存储知识图数据，因此，我们在每个知识链的节点上安装了Neo4j图数据库，并在K-V数据库和图数据库之间建立映射关系，存储模式如图16所示：

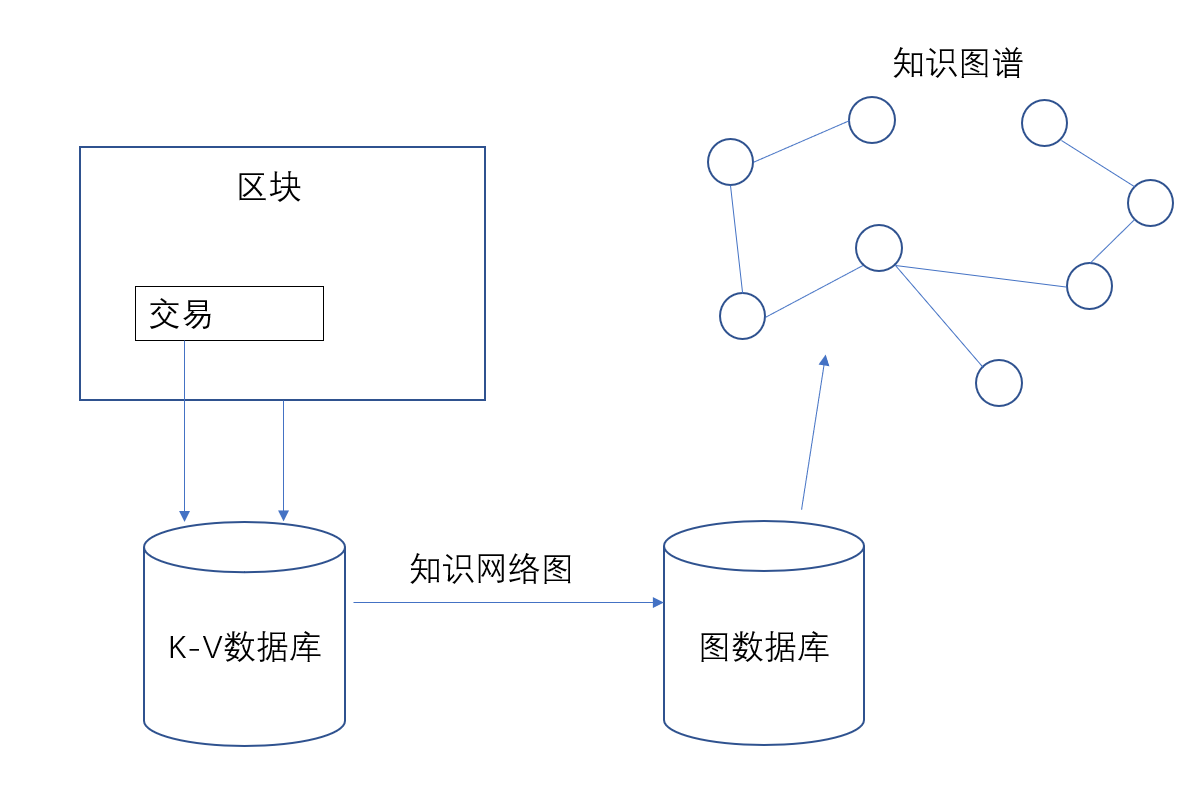


图16 知识图存储模式

在添加图数据库的基础上对跨链查询的接口进行封装，使对外的查询接口以知识图谱的形式体现，用户无需考虑底层具体的区块链结构，只需要输入感兴趣的信息就可以得到相关的数据，总体上来说，本方案跨链检索部分可以分为5层，具体分层如图17所示：



图17 跨链检索五层结构

**3.4 方案分析**

**3.4.1 检索效率分析**

系统的检索效率主要分为三个部分，我们的方案使用索引结构来提升检索效率，区块链中原始的搜索方法需要从尾至头遍历整条区块链上的数据，才能找到想要搜索的结果，搜索的时间复杂度为O(mn)，其中m为区块数量，n为每个区块的交易数，随着区块数量的不断增长，搜索的时间将变得难以接受。与原始区块链的搜索方法不同，我们的方案通过使用索引结构来对链上的数据进行搜索，搜索的时间复杂度仅为O(logN),其中N为索引结构包含的节点数，随着区块数量的不断增长，我们方案的效率优势越来越明显。在跨链检索部分，由于本方案经过知识链共识和知识图谱的脸上构建过程，已经形成了标准化的知识图谱，因此跨链查询的检索效率与普通的知识图谱大致相当。

**3.4.2 安全性分析**

对于我们系统的安全性分析，主要针对的是数据安全和区块链系统安全两个方面，对于数据安全方面，用户正常的访问数据过程是通过链上的索引结构找到感兴趣的数据，然后通过对应交易中元数据的引导找到数据在链下的具体存储位置。一个恶意的攻击者可能会通过搜索一条数据，得到数据的链下位置之后，通过寻找数据存储点是否存在漏洞，从而进行“拖库攻击”，以把用户数据库中的所有数据导出，针对这种攻击方式，本方案建议用户将自己的链下数据存放入分布式的存储系统中，包括IPFS系统、分布式数据库等，或者用户通过对链下数据进行加密，增加权限控制等限制等方式，由数据所有者根据自己数据的重要程度自行进行安全防护。

另一方面，由于区块链是访问和检索数据的核心模块，区块链保证了数据的不可篡改和公开可验证，显然试图对链上数据进行修改是非常困难的，但由于开放且去中心化的交易模式，任何节点都可以合法地通过上传任何数据来对链上的索引结构进行修改，对手可能试图用一种特殊的拒绝服务攻击方式，通过不停上传大量无效数据来干扰索引结构的生成，由于无法手动判断恶意节点上传数据的价值，所以无效数据也会被存入到我们的索引结构当中，但我们的系统针对无效数据对索引结构的干扰采取了限制措施，我们在系统中采用了搜索频率平衡算法，对于长时间没有被搜索的数据，设置一个更新时间周期，每周期定时对MST的结构进行更新，将搜索频率低的数据所在的子树部分向右移动，而用户通过MST进行搜索是由左至右进行搜索的，因此敌手的攻击不会对搜索效率产生较大的影响。

**3.5 本章小结**

本章首先对我们方案的整体架构进行了介绍，确定了主要的检索模型和性能指标，随后将整个区块链数据共享系统中的检索方案分为单链检索和跨链检索两个主要模块，并把这两个模块共分为六个子模块分别进行描述。最后从检索效率和安全性两方面对我们的方案进行了较为全面的分析。

**第四章 实验设计与结果分析**

本章将对本本提出的高效检索方案进行仿真实现，并分别对单链检索和跨链检索两个模块从检索的响应时间、准确性、完整性和安全性等方面进行全面评估，通过设计多组对照试验来验证本方案的可行性，并通过与其他类似方案的对比来体现本方案的优越性。

**4.1 实验环境配置**

实验平台和测试软硬件环境：我们的实验运行在64位的Linux 服务器中(Ubuntu 16.04)，Intel core i7-7700CPU和16GB内存，我们选取的平台是以太坊的golang客户端geth1.8。

**4.2 实验数据集**

为了全面且准确地评估本文所提方案的检索效率、检索能力和安全性，我们的实验采用了多个不同领域的数据集对我们实现的系统进行了多组不同参数的实验测试，其中针对单链检索和跨链检索两个模块我们选用的不同的数据集进行测试，本节将介绍我们实验中使用的数据集。

**单链部分**

1.亚马逊音乐标签数据集(AM)

亚马逊音乐标签数据集是一个亚马逊电商推进系统提供的一个关于blues音乐 meta信息的公开数据集，包含大约305KB的meta数据和对应的35MB的音乐信息数据，meta数据采用json格式描述。

2.steam bundle 数据集(SB)

steam bundle 数据集是steam游戏平台提供的一个用户行数据集，包括user-id,game-title,bundle-name和value，我们使用该数据集进行范围查询的测试。

3.交通情况数据集(taxi)

交通情况数据集是香港大学提供的一个公开的数据集，包括四个城市罗马、上海、博洛尼亚和科隆的4000辆出租车24小时的行驶数据，车辆的行驶数据格式为<carID,time, longitude&latitude,speed,whether carry passenger>，因为我们的系统共享的数据不仅是区块链相关的交易数据，由于智慧城市的发展，未来区块链上存储的数据可能来自于边缘物联网设备提供的相关数据，因此我们在测试性能时考虑到了未来的数据共享需求。

**跨链部分**

针对跨链检索的实验数据集，由于跨链检索往往设计多条不同领域的区块链之间的联合搜索，因此只选择单一领域的数据集无法体现跨链检索的实际需求，因此我们选择了维基百中文维基百科的语料库，大小为1.98GB。

**4.3 系统性能测试**

**4.3.1 索引生成开销**

由于我们的系统在区块链中添加了索引结构，且索引结构需要通过矿工们的共识过程来建立和更新，这就不可避免地延长了共识的时间并且增加了矿工们的存储开销。因此我们需要首先对索引结构的生成开销进行测试，以验证系统是否可用。

我们首先测试了索引结构生成的相关开销，包括时间开销和空间开销，并针对四种搜索功能进行了响应时间的测试，在三个数据集中分别测试了全节点和清节点的搜索响应时间。之后我们进行了特性对比，我们控制倒排列表和MST为变量，对结构添加前后系统的搜索效率和检索能力进行对比。参数对比方面，我们测试了我们系统对不同关键词数的检索能力，同时比较了DSE算法中参数的改变对搜索相关性的影响。最后我们与两个区块链委托查询的方案进行了整体性能对比。

时间开销：与普通的区块链相比，我们系统主要的耗时在于索引结构MST的生成，索引生成的时间开销，可以用以下公式表示：

表示时间间隔内，索引生成的耗时由索引插入操作的耗时和索引更新的耗时组成，索引更新的耗时包括MST更新的耗时以及倒排列表更新的耗时。

空间开销：区块占用的存储空间，因为区块中包含交易的数量可能不相同，所以我们需要分别测试不同交易数据规模下区块占用存储空间的大小。

为了测试加入索引结构对区块生成的影响，我们分别测试了不加索引结构与添加了索引结构后的时间开销和空间开销，结果如图18和图19所示：

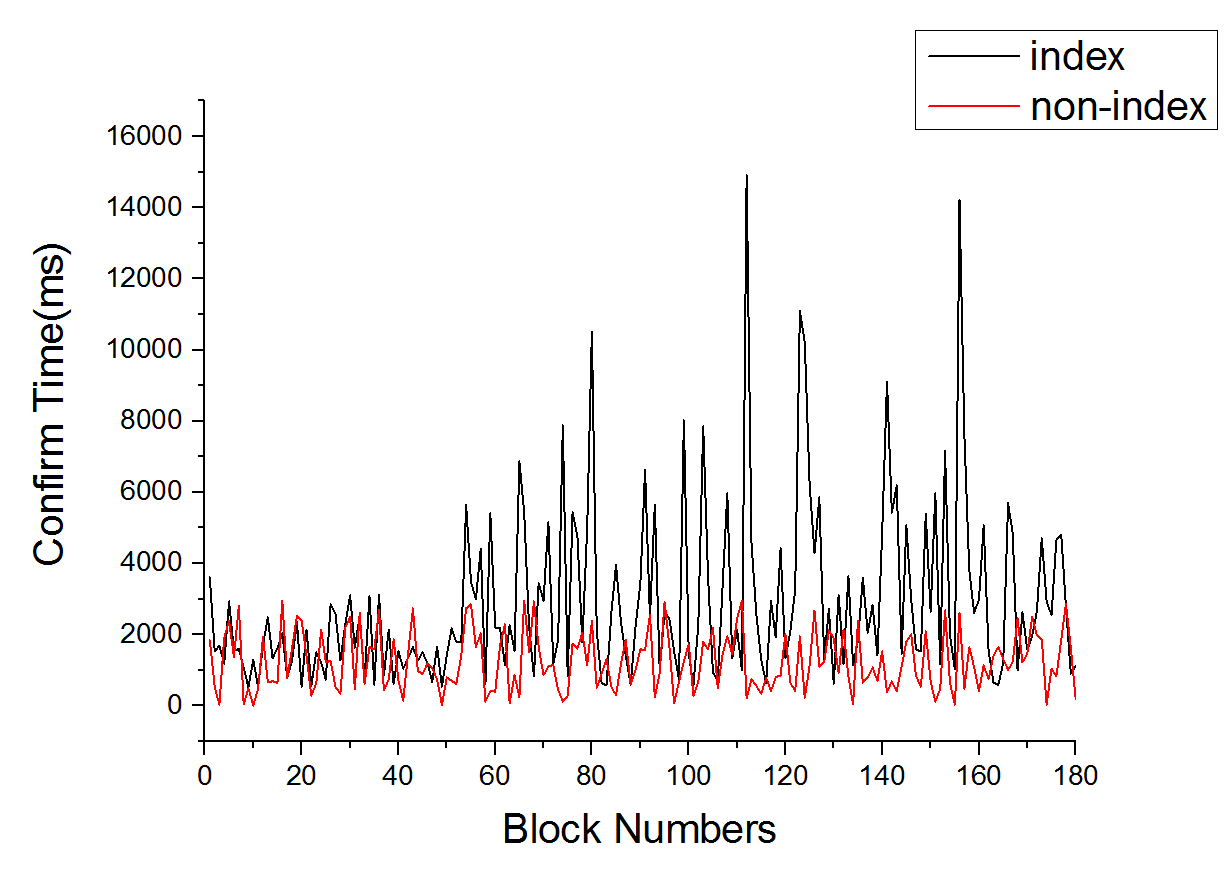


图18 时间开销

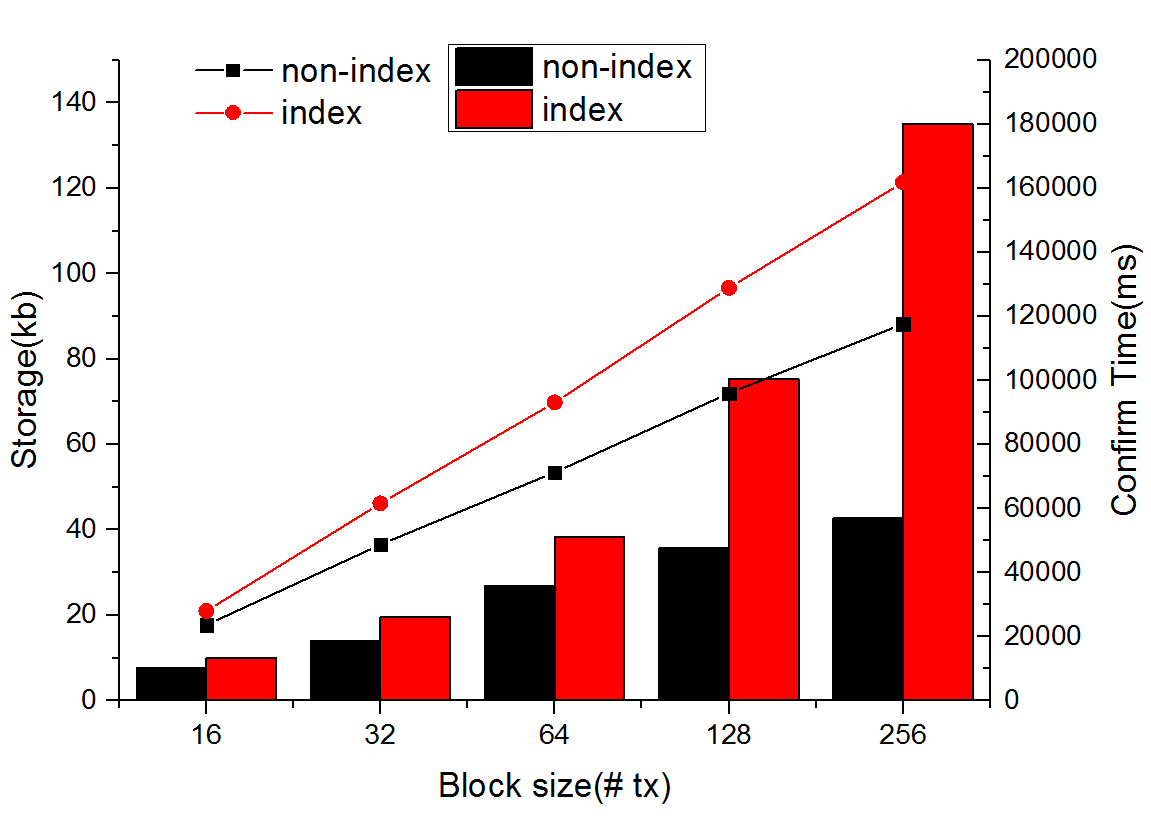


图19 空间开销

图18显示了在不同区块数量下，每个区块的生成时间，根据图18的结果显示，加入MST后的系统出块平均耗时比不加索引结构的出块耗时多了0.5秒左右，这是可以接受的。

图19显示了包含不同交易数量的区块的平均大小，测试结果显示，在非交易密集的区块内加入MST的区块只有较小的存储空间增长，根据目前以太坊系统的吞吐量，加入MST之后区块的平均大小不会出现较大的增长。

**4.3.2 系统响应时间**

对于系统在单链检索模块的响应时间，我们测试了3个不同数据集(AM,SB,taxi)下的四种搜索方式（基本搜索，语义关键字搜索，模糊搜索，范围搜索）的平均查询时间和最差情况下的查询时间，其中基本查询包括以太坊的getBalanceByAddress,getBlockByHash,和getTransactionByHash。由于轻节点采用的是委托检索的方式，需要与全节点进行网络通信并对搜索结果进行验证，所以在测试响应时间时对轻节点和全节点分别进行测试。

系统的响应时间如表1所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Query approach | Basic query | | | | | | Semantic (keyword) query | | | | | |
| dataset | AM | | SB | | taxi | | AM | | SB | | taxi | |
| node type | light node | full node | light node | full node | light node | full node | light node | full node | light node | full node | light node | full node |
| Mean(ms) | 2 | 2 | 2 | 2 | 2 | 2 | 5 | 4 | 6 | 5 | - | - |
| Worse case(ms) | 4 | 3 | 4 | 3 | 4 | 3 | 106 | 33 | 60 | 23 | - | - |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Range query | | | | | | Fuzzy query | | | | | |
| AM | | SB | | taxi | | AM | | SB | | taxi | |
| light node | full node | light node | full node | light node | full node | light node | light node | full node | light node | light node | full node |
| - | - | 19 | 17 | 21 | 20 | 11 | 10 | 13 | 11 | 13 | 12 |
| - | - | 156 | 145 | 273 | 251 | 29 | 26 | 33 | 29 | 42 | 38 |

表1 四种查询的响应时间

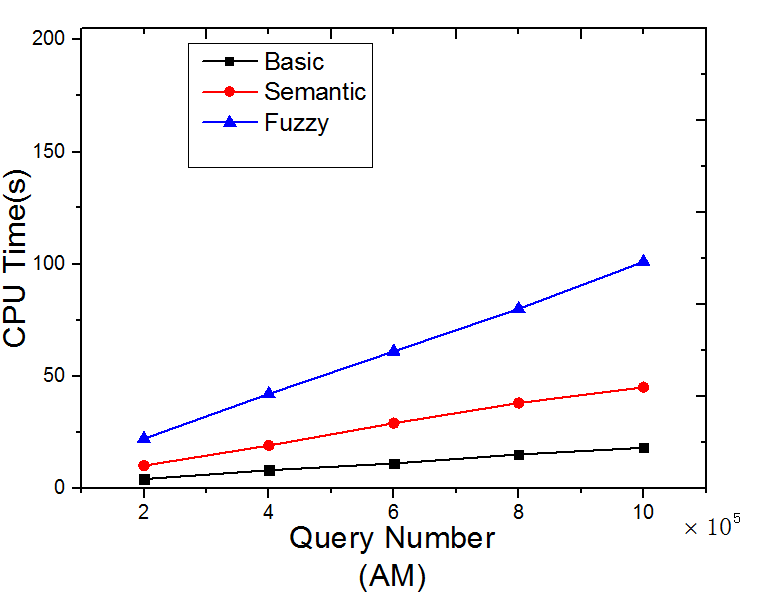
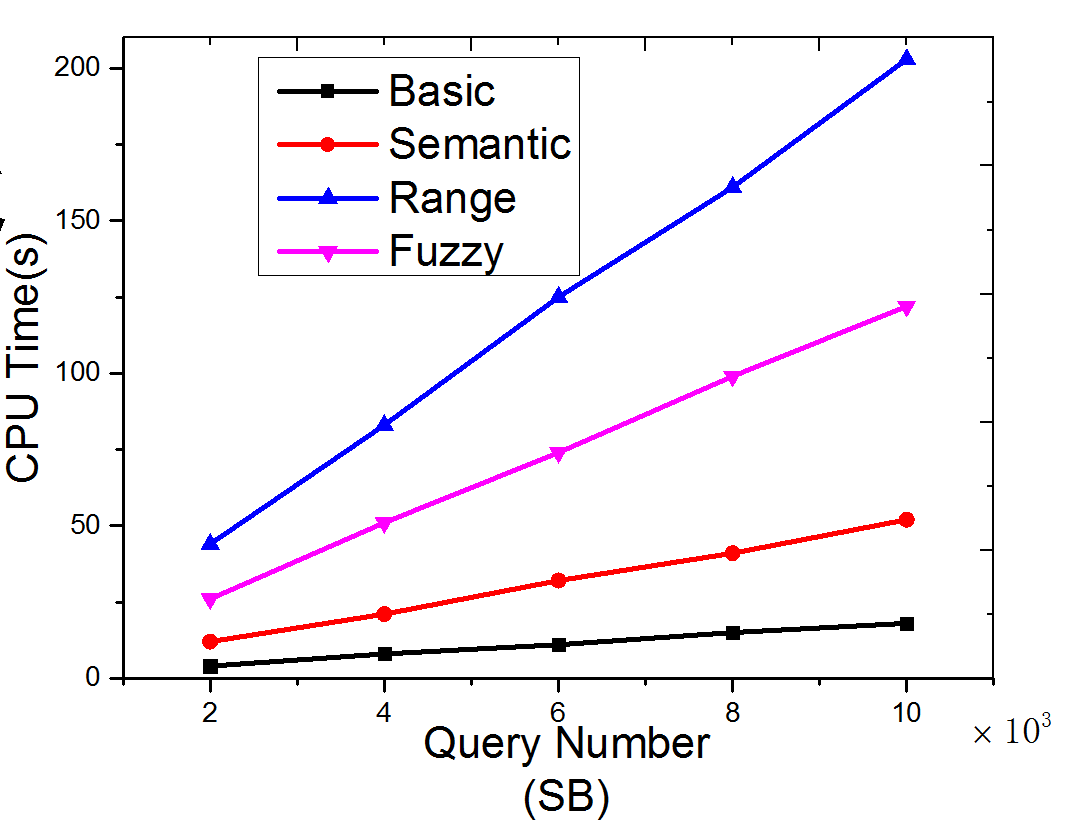
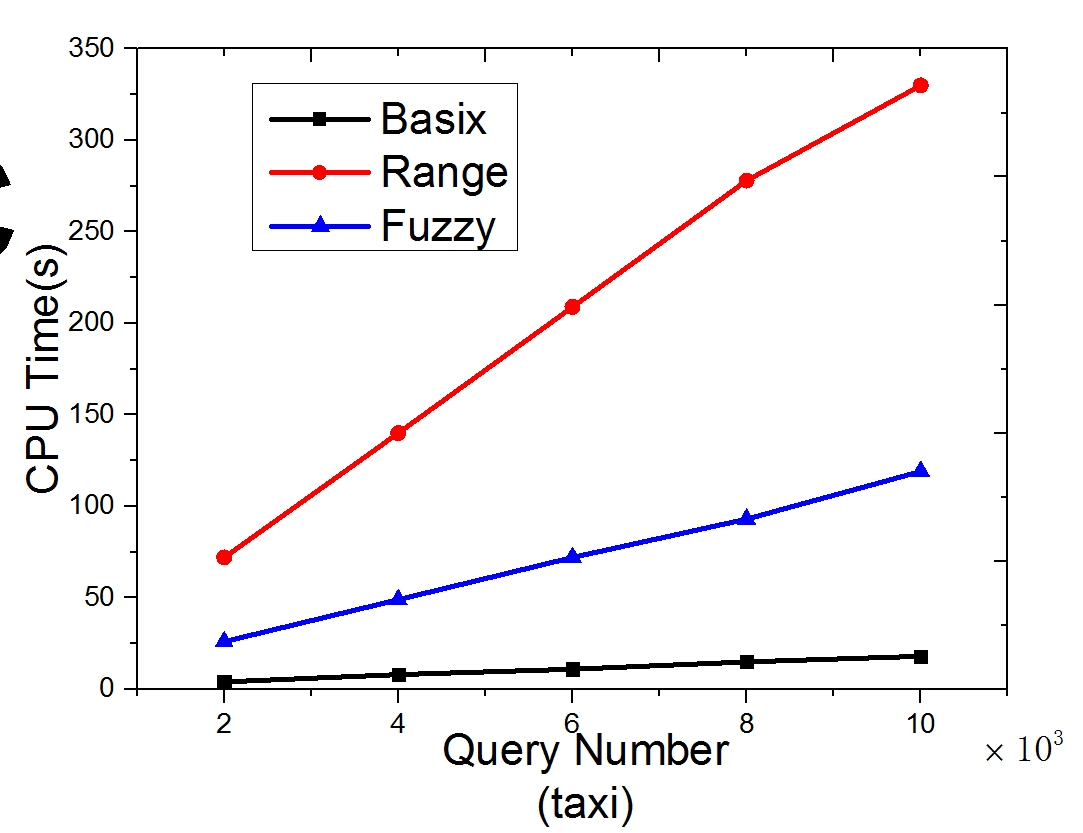
  

图20 三个不同数据集查询的CPU时间

通过表1可以得到，我们的系统针对四种不同的搜索方式，平均响应时间都在20ms以内，而通过对比轻节点和全节点的响应时间，我们可以发现，轻节点的搜索相应时间比全节点略长，证明我们的代理搜索验证效率较高，轻节点对搜索的验证时间较短。我们统计了不同查询规模下3个数据集对于不同查询的累积响应时间，统计的结果如图20所示。

为了验证我们系统对关键字搜索的性能，我们采用控制变量的方法对系统的关键字搜索性能进行测试，测试的数据集是AM，我们采取了两个对比方案，第一个是有MST和无MST，由于以太坊没有原生的关键字搜索功能，因此我们写了一个智能合约，通过采取循环扫描的方式来测试无MST结构的情况下以太坊对关键字搜索的性能。另一个对比方案是取消了倒排列表，如果没有了倒排列表，我们就必须将DSE算法对链下数据提取出的关键字进行随机排序，导致插入MST的关键字组合也必须进行全排列操作，我们比较了无到排列表和有倒排列表的情况下系统分别对于不同查询条数的累计响应时间。

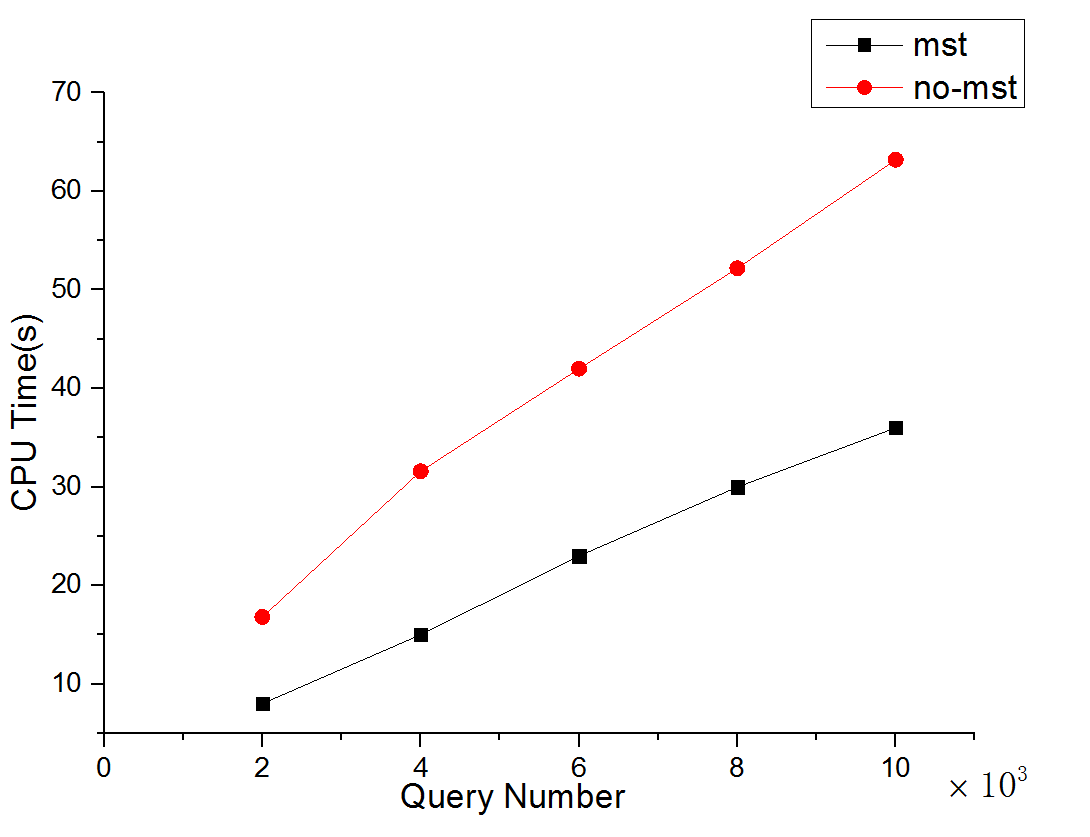
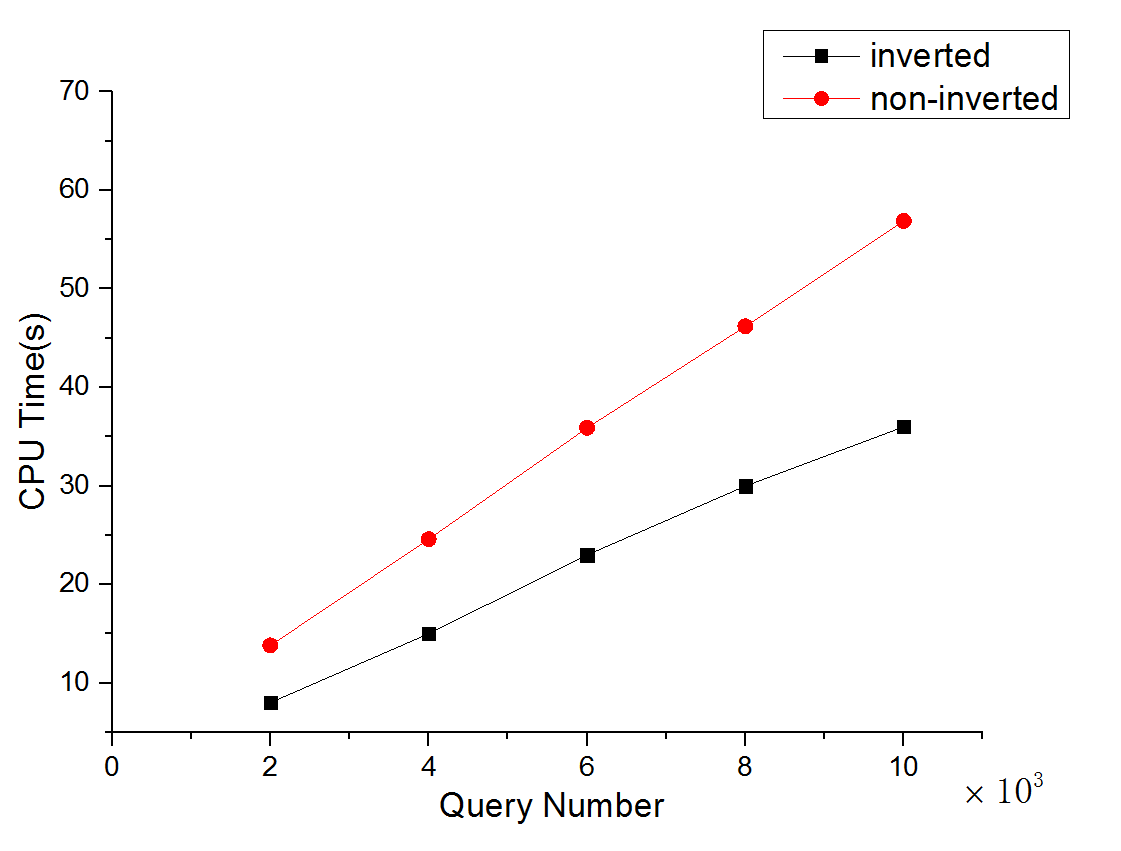
 

图21 图22

结果如图21和图22所示，图22的结果表明，加入了MST结构之后，相较于单纯的以太坊，平均搜索时间降到原先的1/2以下。图22则表明，在没有倒排列表的情况下，单纯的MST对系统搜索性能的提升较为有限，因此如果想要达到最佳的搜索效果，需要保证倒排列表和MST都正常工作。

**4.3.3 查询准确率和召回率**

我们首先通过测试了不同关键字规模下，对系统搜索查准率和召回率的影响，我们分别测试了在关键字的数量为1-5时系统检索的查准率和召回率，同时，为了兼顾查准率和查全率，我们采用二者的调和平均值，F值，F值的计算公式如下：

其中，P表示查准率，R表示召回率，表示P和R所占的比重大小，在这里，我们将定为0.5，因为我们认为查准率和召回率同样重要，测试的结果如表2所示：

|  |  |  |  |
| --- | --- | --- | --- |
| Key Number | P | R | F |
| @1 | .733 | .549 | .627 |
| @2 | .809 | .588 | .681 |
| @3 | .863 | .665 | .751 |
| @4 | .889 | .773 | .826 |
| @5 | **.909\*** | **.802\*** | **.852\*** |
| non-inverted list | .615 | .261 | .366 |

表2 不同关键字数量下的查准率、召回率以及F值指标

通过表2的结果，我们可以发现，随着关键字数量的增加，系统的查准率在不断增加，增加的幅度逐渐变小，召回率也在不断增加，增加的幅度先增大后减小，总体上来说，随着关键字数量的增多，系统的查准率和召回率在提升。值得一提的是，我们测试了在没有倒排列表情况下查准率和召回率的大小，结果发现，系统的查准率和召回率都发生了下降，特别是在召回率方面。

随后，我们用相关性指标MRR，测试了我们的DSE算法对关键字排名的性能，我们选取了的两种情况，对=L/(n=1,2,3,4)的四种不同取值进行MRR统计，统计的结果如表3：

|  |  |  |
| --- | --- | --- |
| n(=L/) | MRR | MRR |
| 1 | 0.502 | 0.502 |
| 2 | 0.611 | 0.636 |
| 3 | **0.677** | **0.692** |
| 4 | 0.652 | 0.661 |

表3 取1和2的情况下，值的变化对DSE算法关键字排序性能的影响

通过表3可以得到，随着距离初始坐标越来越远，DSE算法的关键字排序性能先升后降，当=L/8时，DSE的关键字排序性能达到最佳。

**4.3.4 跨链图谱测试**

跨链检索部分，我们对知识图谱的关系检索能力和可视化功能进行了测试，测试过程如下：

1．测试关系检索能力

我们在知识链中以WEB服务的方式提供跨链查询服务，用户可以通过HTTP的GET方法来查询自己想要的结果，请求URL格式为：节点IP.节点端口号.链ID.节点ID.实体内容，返回结果格式为：查询应答码.查询结果。一条查询的实例如下：

请求实例：http://192.168.80.156:7545/ chainID=1&nodeID=1&entity=比特币

返回实例：

{

"message": "success",

"data": {

"entity": "比特币",

"domain": "百科",

"Rs list": [

["中文名","比特币"],

["外文名","Bitcoin"],

["种类","加密数字货币"],

[流通平台","网络"],

["创始人", "中本聪" ],

["缩写","BTC" ],

["诞生时间","2009年1月3日"],

["总量","2100万个"],

["最小单位","“聪”（satoshi），1聪=0.00000001BTC"],

["共识机制""POW工作量证明"]

],

}

}

然后我们在web页面端对知识图谱的效果进行展示，我们的检索页面如图23所示：



图23 跨链检索页面

输入关键字“比特币”后，我们得到的可视化图谱如图24所示：

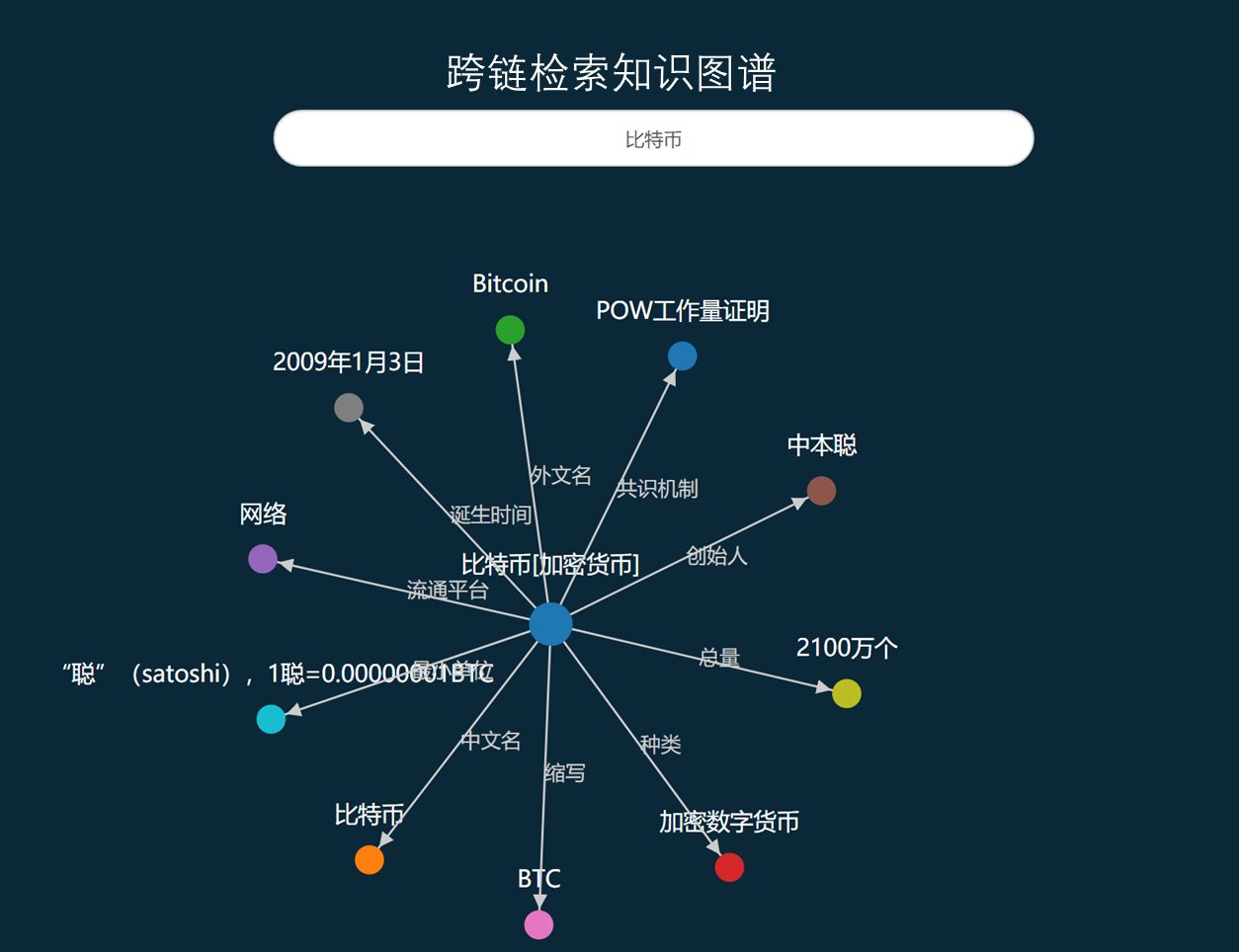


图24可视化图谱效果

点击其中一个属性，可以看到所在的链代码和区块号，如图25所示：

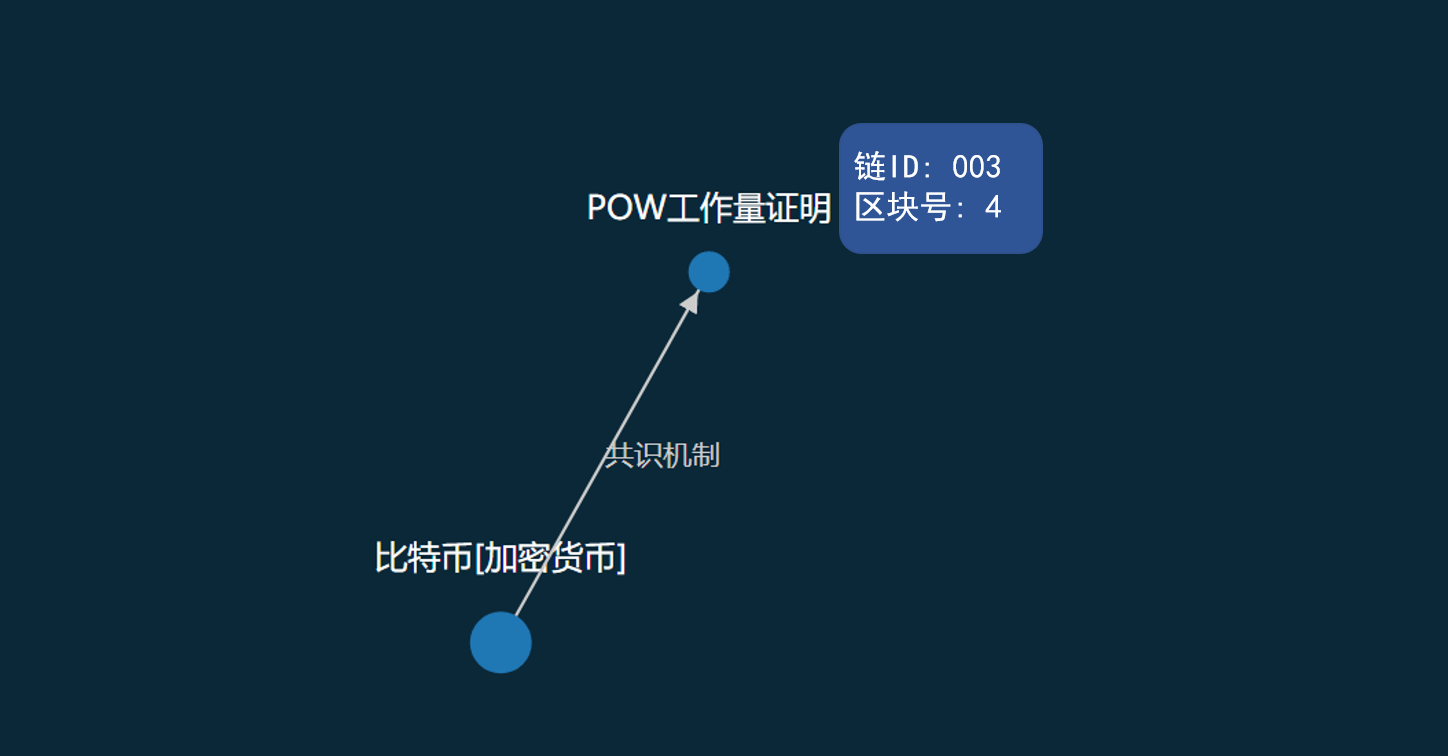


图25 属性所在位置显示

**4.4 对比实验测试**

**4.4.1 对比方案介绍**

目前针对区块链系统的高效检索方案比较有代表性的工作是vChain[15]和SEBDB[16]，这两个项目都针对区块链上的高效语义检索进行了研究，vChain主要针对代理查询的可验证性进行了方案优化，SEBDB则主要设计了一种可用性和扩展性更强的区块链数据库管理系统，我们主要在检索效率方面与这两个系统进行比较。

**4.4.2 对比实验结果**

为了测试我们项目的检索性能以及验证我们项目的优越性，我们将我们的方案与vChain和SEBDB两个方案进行了整体性能比较，三个项目都支持代理查询和非代理查询，同时对于查询结果提供一个对结果的验证对象（VO），我们主要对代理查询的功能进行测试，我们分别比较了三个项目的VO大小和VO的验证时间，我们通过数据集SB进行测试，进行了查询18000条关键字查询和范围查询的实验，测试了不同区块数量下的VO所占的存储空间，以及对于不同的搜索结果条数，对VO的累计验证时间，测试结果如图26所示：

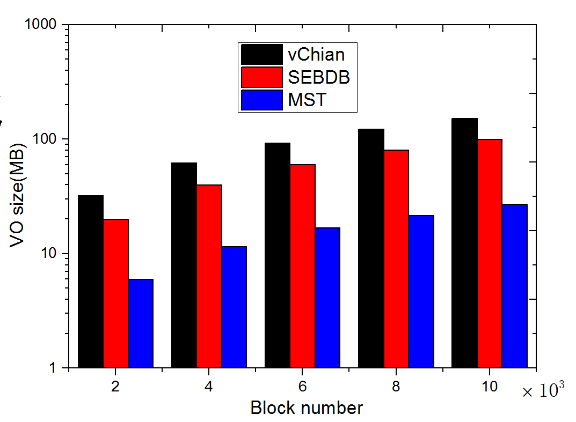
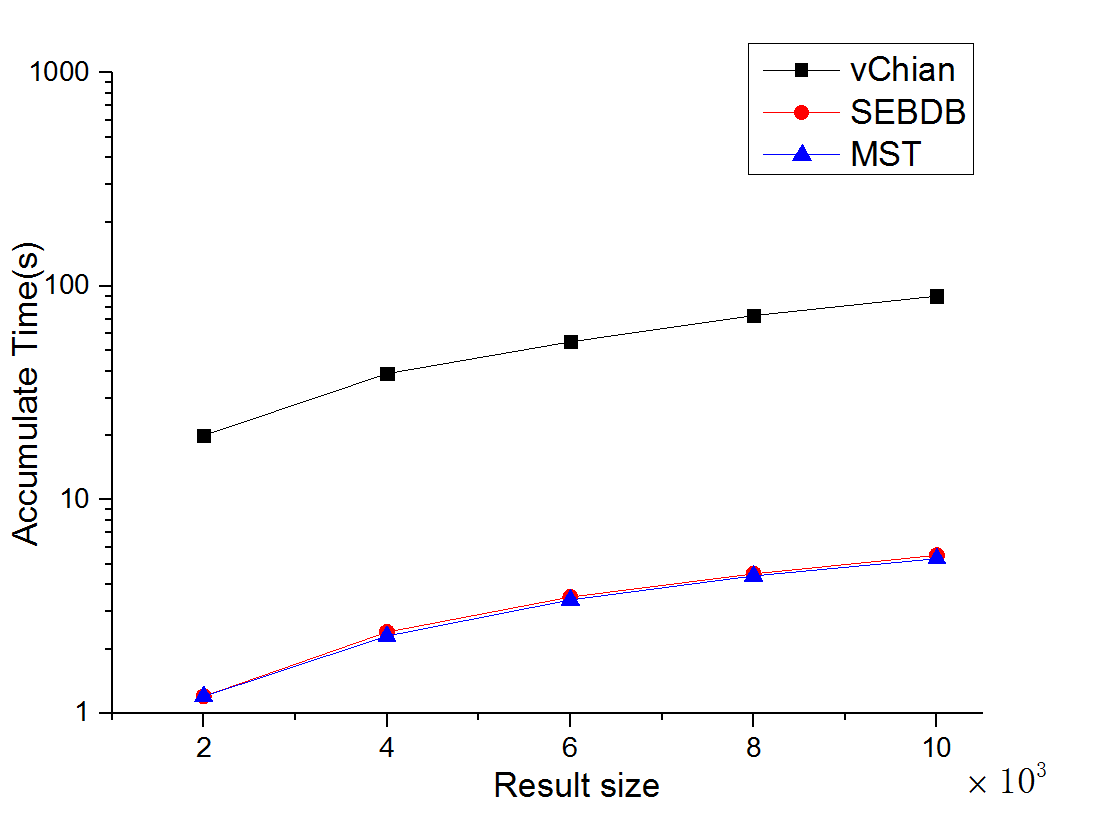
 

图26 整体对比

图26显示出了我们系统与另外两个区块链搜索系统的性能对比结果，在VO size方面，我们方案的验证对象大小约为vChain的1/5左右，约为SEBDB的1/3左右，在验证时间方面，我们方案的验证时间比vChian有了大幅度的减少，与加入了成熟数据库的SEBDB相比，我们的时间比SEBDB有了略微的减少。实验证明，对于关键字搜索，我们系统的VO大小比另外两个方案更小，同时对VO具有更短的验证时间。

**4.5 本章小结**

本章针对区块链数据共享系统中高效检索技术方案进行了实验设计和结果分析，首先介绍了实验的环境配置和使用的数据集，随后对于单链检索和跨链检索两个模块分别进行了性能测试，对测试结果进行了分析，验证了我们系统的可行性。为了体现我们系统的优越性，本章还将我们的方案与其他类似方案进行了对比，实验结果表明，相较于其他方案，我们的方案在检索效率和成本开销方面更具优势。

**第五章 总结与展望**

**5.1 工作总结**

区块链由于其安全可信、不可篡改等特性，目前常常被应用于存储大量有价值的数据，而区块链的去中心化和共识机制可以保证这些有价值数据的安全共享。

但是，对于越来越多的链上数据，目前没有一种高效的检索方案来使得用户快速获取到自己感兴趣的数据。为了提升区块链数据共享系统的检索效率，本文在广泛阅读相关文献的基础上，提出并设计实现了一种针对区块链数据共享系统的高效检索技术，可以在保证区块链高可靠性的前提下对其数据共享系统中存储的数据进行对数时间复杂度的检索，兼顾了安全性和检索效率。本文主要工作内容总结如下：

（1）提出了一种去中心化语义提取算法，以在链下数据和链上数据之间建立语义映射关系，使链上搜索的信任可以传递到链下的数据上，同时为链上索引结构的构建提供支撑。

（2）分别为单链检索和跨链检索设计了两个不同的检索模块，单链检索方面，在链上通过矿工的共识构建了一个名为默克尔语义搜索树的索引结构，结合了默克尔树，帕特丽夏树，B+树，哈希指针，AC自动机等技术，利用该索引结构可以对之前提取的链下数据的语义信息和元数据进行多功能的复杂搜索操作，包括多关键字搜索，范围搜索，模糊查询等。跨链检索方面，对不同链上的数据进行格式整合，通过委员会抽取算法在每条链上抽取跨链委员会，委员会之间组成一条知识链，通过共识来构建一个全局的跨链知识图谱。

（3）为区块链系统中的轻节点设计了一种代理查询和结果验证的方案，轻节点只需要连接全节点进行代理查询同时在本地即可对代理查询的结果进行验证。

最后，我们对本文所设计的检索方案进行了实现，并在真实的区块链网络中进行了实验测试，通过性能测试和对比测试的实验结果来看，我们的方案具有较好的性能表现。

**5.2 未来展望**

虽然本文提出的方案在检索效率和检索能力上均有不错的表现，但随着信息技术和区块链的发展，本方案可能不能很好地适用于未来数据更海量化、应用领域更复杂化的检索需求。从目前来看还有以下几点提升空间：

（1）首先，本文所设计的检索系统默认对用户所有上传的数据都进行处理，虽然有索引结构的平衡操作等方式来降低无用信息的干扰度，但仍会降低系统的效率。因此，未来的一个研究方向是在将用户的信息提交到区块链之前对用户上传的数据进行分析和判断，通过节点间投票等方式来决定是否可以上传到链上。

（2）第二点，本文设计的索引结构并没有考虑到对数据的删除操作，因此所有的数据只能追加而不能被删除掉，这与区块链的特性相关，但对索引结构的设计可以在保证区块链特性的情况下进行优化，使其添加删除数据功能，这也是一个比较大的挑战。

（3）最后，随着5G、物联网、边缘计算、人工智能等技术的发展，它们均会对数据的安全可靠以及高效检索产生需求，如何将我们的方案与这些技术相融合是我们未来需要研究的问题。

参考文献

[1] Zhou J, Tang F, Zhu H, et al. Distributed Data Vending on Blockchain[J]. 2018.

[2] Sicilia M A, Sánchezalonso S, Garcíabarriocanal E. Sharing Linked Open Data over Peer-to-Peer Distributed File Systems: The Case of IPFS[C]// 2016.

[3] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Consulted, 2008.

[4]王千阁,何蒲,聂铁铮, et al. 区块链系统的数据存储与查询技术综述[J]. 计算机科学, 45(12):19-25.

[5] T.T. A. Dinh, R. Liu, M. Zhang, et al., “Untangling blockchain: A data processing view of blockchain systems,” TKDE, pp. 1366–1385, 2018.

[6] Lozi J P, David F, Thomas G, et al. Fast and Portable Locking for Multicore Architectures[J]. 2016, 33(4):1-62.

[7] Androulaki, Elli, Barger, Artem, Bortnikov, Vita et al. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains[J].

[8] 王守道, 蒋玉明, 胡大裟. 基于区块链的智能合约压缩存储方法[J]. 现代计算机, 2019(9):42-46.

[9]BigChain DB [EB/OL]. https://www.bigchaindb.com

[10] Tien Tuan Anh Dinh, Ji Wang, Gang Chen. BLOCKBENCH: A Framework for Analyzing Private Blockchains[C]// the 2017 ACM International Conference. ACM, 2017.

[11] Yang Li, Kai Zheng, Ying Yan, et al. EtherQL: A Query Layer for Blockchain System[C]// International Conference on Database Systems for Advanced Applications. 2017.

[12] El-Hindi M, Binnig C, Arasu A, et al. BlockchainDB: a shared database on blockchains[J]. Proceedings of the VLDB Endowment, 2019, 12(11): 1597-1609.

[13] Xu Y, Zhao S, Kong L, et al. ECBC: A high performance educational certificate blockchain with efficient query[C]//International Colloquium on Theoretical Aspects of Computing. Springer, Cham, 2017: 288-304.

[14] Ruan P, Chen G, Dinh T T A, et al. Fine-grained, secure and efficient data provenance on blockchain systems[J]. Proceedings of the VLDB Endowment, 2019, 12(9): 975-988.

[15] Xu C, Zhang C, Xu J. vChain: Enabling verifiable boolean range queries over blockchain databases[C]//Proceedings of the 2019 International Conference on Management of Data. ACM, 2019: 141-158.

[16] Zhu Y, Zhang Z, Jin C, et al. SEBDB: Semantics Empowered BlockChain DataBase[C]//2019 IEEE 35th International Conference on Data Engineering (ICDE). IEEE, 2019: 1820-1831.

[17] O'Dwyer K J, Malone D. Bitcoin mining and its energy footprint[J]. 2014.

[19]Pouwelse J, Garbacki P, Epema D, et al. The bittorrent p2p file-sharing system: Measurements and analysis[C]//International Workshop on Peer-to-Peer Systems. Springer, Berlin, Heidelberg, 2005: 205-216.

[20]Benet J. Ipfs-content addressed, versioned, p2p file system[J]. arXiv preprint arXiv:1407.3561, 2014.

[21] Arulmozhi S J, Praveenkumar K, Vinayagamoorthi G. BITCOIN IN INDIA: A DEEP DOWN SUMMARY[J]. Advance and Innovative Research, 2019: 28.

[22] Benet J, Greco N. Filecoin: A decentralized storage network[J]. Protoc. Labs, 2018.

[23]Wood G. Ethereum: A secure decentralised generalised transaction ledger[J]. Ethereum project yellow paper, 2014, 151(2014): 1-32.

[24] Bast H, Buchhold B. An index for efficient semantic full-text search[C]//Proceedings of the 22nd ACM international conference on Information & Knowledge Management. ACM, 2013: 369-378.

[25] Van den Bosch A, Bogers T, De Kunder M. Estimating search engine index size variability: a 9-year longitudinal study[J]. Scientometrics, 2016, 107(2): 839-856.

[26] Ramos J. Using tf-idf to determine word relevance in document queries[C]//Proceedings of the first instructional conference on machine learning. 2003, 242: 133-142.

[27] Buckley C, Lewit A F. Optimization of inverted vector searches[C]//Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval. 1985: 97-110.

[28] 于莉. 经典信息检索模型的分类比较[J]. 软件, 2011, 32(3):32-34.

[29] BootWiki.B+树[EB/OL].https://www.bootwiki.com/datastructure/data-structure-b-plus-tree.html,2018.

[30] Page L, Brin S, Motwani R, et al. The PageRank citation ranking: Bringing order to the web[R]. Stanford InfoLab, 1999.

[31] Deng L, Chen H, Zeng J, et al. Research on cross-chain technology based on sidechain and hash-locking[C]//International Conference on Edge Computing. Springer, Cham, 2018: 144-151.

[32] 任惠. 基于区块链的语义知识共享机制[D].

[33] RDF Working Group.RDF[EB/OL].https://www.w3.org/RDF/,2014-2-25.

[34] 杭婷婷,冯钧,陆佳民.知识图谱构建技术:分类、调查和未来方向[J].计算机科学,2021,48(02):175-189.

[35] Eberhardt J, Heiss J. Off-chaining models and approaches to off-chain computations[C]//Proceedings of the 2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers. ACM, 2018: 7-12.

[36] Sunday D M. A very fast substring search algorithm[J]. Communications of the ACM, 1990, 33(8): 132-142.