(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 112363979 B
(45)授权公告日 2023.08.04

(21)申请号 202010984750.2

(22)申请日 2020.09.18

(65)同一申请的已公布的文献号
申请公布号 CN 112363979 A

(43)申请公布日 2021.02.12

(73)专利权人 杭州欧若数网科技有限公司
地址 311100 浙江省杭州市余杭区五常街道文一西路998号海创园18幢509室

(72)发明人 陈勃胜 陈恒

(74)专利代理机构 杭州创智卓英知识产权代理事务所(普通合伙) 33324
专利代理师 张超

(51)Int.Cl.
G06F 16/13(2019.01)
G06F 16/901(2019.01)

(56)对比文件
CN 104615677 A,2015.05.13
CN 110633378 A,2019.12.31
US 10242065 B1,2019.03.26
US 10698955 B1,2020.06.30
US 2017255709 A1,2017.09.07
US 2018144061 A1,2018.05.24
CN 110309334 A,2019.10.08
CN 110659292 A,2020.01.07
CN 111639082 A,2020.09.08
US 2014136555 A1,2014.05.15
王文龙;李建中.一种有效的在不确定图数据库中挖掘频繁子图模式的MUSIC算法.智能计算机与应用.2013,(第05期),全文.

审查员 夏容

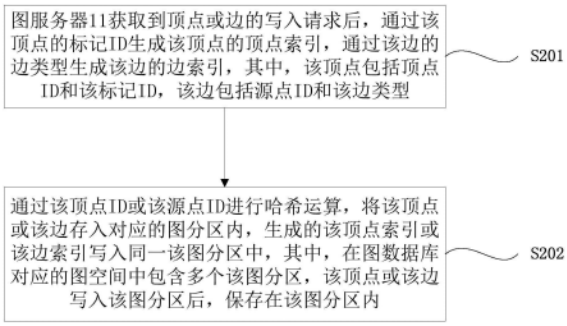
权利要求书2页 说明书8页 附图3页

(54)发明名称

一种基于图数据库的分布式索引方法和系统

(57)摘要

本申请涉及一种基于图数据库的分布式索引方法和系统，该方法包括获取到顶点或边的写入请求后，通过该顶点的标记ID生成该顶点的顶点索引，通过该边的边类型生成该边的边索引，其中，该顶点包括顶点ID和该标记ID，该边包括源点ID和该边类型；通过该顶点ID或该源点ID进行哈希运算，将该顶点或该边存入对应的图分区内，生成的该顶点索引或该边索引写入同一该图分区中，其中，在图数据库对应的图空间中包含多个该图分区，该顶点或该边写入该图分区后，保存在该图分区内，解决了对Nebula Graph的索引查询效率不高，查询产生的不必要的网络开销较高的问题，使用户可以快速地对Nebula Graph中的顶点和边进行查询。



CN 112363979 B

1.一种基于图数据库的分布式索引方法,其特征在于,所述方法包括:

获取到顶点或边的写入请求后,通过所述顶点的标记ID生成所述顶点的顶点索引,通过所述边的边类型生成所述边的边索引,其中,所述顶点包括顶点ID和所述标记ID,所述边包括源点ID和所述边类型;

通过所述顶点ID或所述源点ID进行哈希运算,将所述顶点或所述边存入对应的图分区内,生成的所述顶点索引或所述边索引写入同一所述图分区中,其中,在图数据库对应的图空间中包含多个所述图分区,所述顶点或所述边写入所述图分区后,保存在所述图分区内,其中,生成所述顶点的顶点索引或生成所述边的边索引包括:所述顶点索引或所述边索引存入的所述图分区确定图分区ID;所述顶点索引在标记模型的基础上,由所述标记ID创建得到索引ID;所述边索引在边模型的基础上,由所述边类型创建得到索引ID,其中,所述顶点索引或所述边索引包括:所述图分区ID、数据类型、所述索引ID和属性。

2.根据权利要求1所述的方法,其特征在于,所述顶点或所述边写入所述图分区后,所述方法包括:

获取到API客户端发送的查询请求消息,调用元数据服务器验证所述请求有效性,通过存储客户端将所述查询请求消息发送到所有存储服务器中的图分区,根据所述查询请求消息对所述顶点索引或所述边索引并行索引扫描,其中,通过所述图分区ID和所述标记ID或通过所述图分区ID和所述边类型明确所述索引扫描的查询范围;

获得所述存储服务器查询得到的结果集,汇总所述结果集返回给所述API客户端。

3.根据权利要求2所述的方法,其特征在于,所述根据所述查询请求消息对所述顶点索引或所述边索引并行索引扫描包括:

对所述顶点索引或所述边索引的并发索引扫描出错后,生成错误码,通过所述错误码定位到失败的所述图分区。

4.根据权利要求1所述的方法,其特征在于,所述顶点或所述边写入所述图分区后,所述方法包括:

获取到所述顶点或所述边的更新请求后,通过对所述顶点ID或所述源点ID进行哈希计算,确定更新目标的所述图分区;

通过元数据服务器获取所述图分区的列表,所述图分区将所述顶点、所所述顶点的数据、所述顶点索引或所述边的数据、所述边索引并行更新。

5.根据权利要求1所述的方法,其特征在于,所述顶点或所述边写入所述图分区后,所述方法包括:

获取到删除请求后,对所述顶点ID或所述源点ID进行哈希计算,确定写入目标的所述图分区;

在所述图分区内一并放着所述顶点的数据、所述顶点索引或所述边的数据、所述边索引,将所述图分区里的数据和索引加入到删除列表中一起删除。

6.一种基于图数据库的分布式索引系统,其特征在于,所述系统包括图服务器、元数据服务器、存储服务器:

所述图服务器获取到顶点或边的写入请求后,通过所述顶点的标记ID生成所述顶点的顶点索引,通过所述边的边类型生成所述边的边索引,其中,所述顶点包括顶点ID和所述标记ID,所述边包括源点ID和所述边类型;

所述图服务器通过所述顶点ID或所述源点ID进行哈希运算，将所述顶点或所述边存入对应的图分区内，所述顶点或所述边写入所述图分区后，生成的所述顶点索引或所述边索引写入同一所述图分区中，其中，在图数据库对应的图空间中包含多个所述图分区，所述顶点或所述边写入所述图分区后，保存在所述图分区内，其中，生成所述顶点的顶点索引或生成所述边的边索引包括：所述顶点索引或所述边索引存入的所述图分区确定图分区ID；所述顶点索引在标记模型的基础上，由所述标记ID创建得到索引ID；所述边索引在边模型的基础上，由所述边类型创建得到索引ID，其中，所述顶点索引或所述边索引包括：所述图分区ID、数据类型、所述索引ID和属性。

7.根据权利要求6所述的系统，其特征在于，所述系统包括：

所述图服务器获取到API客户端发送的查询请求消息，调用元数据服务器验证所述请求有效性，通过存储客户端将所述请求发送到所有存储服务器中的图分区进行对所述顶点索引或所述边索引的并行索引扫描，其中，通过所述图分区ID和所述标记ID或通过所述图分区ID和所述边类型明确所述索引扫描的查询范围；

所述图服务器获得所述存储服务器查询得到的结果集，汇总所述结果集返回给所述API客户端。

8.一种电子装置，包括存储器和处理器，其特征在于，所述存储器中存储有可运行计算机程序，所述计算机程序可执行权利要求1至5中任一项所述的基于图数据库的分布式索引方法。

9.一种存储介质，其特征在于，所述存储介质中存储有计算机程序，所述计算机程序可执行权利要求1至5中任一项所述的基于图数据库的分布式索引方法。

一种基于图数据库的分布式索引方法和系统

技术领域

[0001] 本申请涉及计算机领域,特别涉及一种基于图数据库的分布式索引方法和系统。

背景技术

[0002] 随着零售、金融、电商、互联网、物联网等行业的兴起,基础数据量成几何状增长,为了将日益增长的庞大数据量组织成一个关系网,传统的关系数据库已经很难应对了;由此业界上出现了一批专门针对关系网数据存储、计算的数据库--图数据库;在海量关系数据中的检索效率是每个图数据库必须要面对的问题,图数据库索引的实现有效提高了数据检索效率。

[0003] 在相关技术中,比较有代表性的图数据库是Nebula Graph、Neo4j和JanusGraph等,Nebula Graph为一个高性能图数据库,可以处理千亿节点万亿条边的海量图数据,同时解决了海量数据存储和分布式并行计算的问题;如今,对于Nebula Graph的索引效果不好,无法快速地对Nebula Graph中的顶点和边进行查询。

[0004] 目前针对相关技术中,对Nebula Graph的索引查询效率不高,查询产生的不必要的网络开销较高的问题,尚未提出有效的解决方案。

发明内容

[0005] 本申请涉及计算机领域,特别涉及一种基于图数据库的分布式索引方法和系统,以至少解决相关技术中对Nebula Graph的索引查询效率不高,查询产生的不必要的网络开销较高的问题。

[0006] 第一方面,本申请实施例提供了一种基于图数据库的分布式索引方法,所述方法包括:获取到顶点或边的写入请求后,通过所述顶点的标记ID生成所述顶点的顶点索引,通过所述边的边类型生成所述边的边索引,其中,所述顶点包括顶点ID和所述标记ID,所述边包括源点ID和所述边类型;通过所述顶点ID或所述源点ID进行哈希运算,将所述顶点或所述边存入对应的图分区内,生成的所述顶点索引或所述边索引写入同一所述图分区中,其中,在图数据库对应的图空间中包含多个所述图分区,所述顶点或所述边写入所述图分区后,保存在所述图分区内。

[0007] 在其中一些实施例中,生成所述顶点的顶点索引或生成所述边的边索引包括:所述顶点索引或所述边索引存入的所述图分区确定图分区ID;所述顶点索引在标记模型的基础上,由所述标记ID创建得到索引ID;所述边索引在边模型的基础上,由所述边类型创建得到索引ID,其中,所述顶点索引或所述边索引包括:所述图分区ID、数据类型、所述索引ID和属性。

[0008] 在其中一些实施例中,所述顶点或所述边写入所述图分区后,所述方法包括:获取到API客户端发送的查询请求消息,调用元数据服务器验证所述请求有效性,通过存储客户端将所述查询请求消息发送到所有存储服务器中的图分区,根据所述查询请求消息对所述顶点索引或所述边索引并行索引扫描,其中,通过所述图分区ID和所述标记ID或通过所述

图分区ID和所述边类型明确所述索引扫描的查询范围;获得所述存储服务器查询得到的结果集,汇总所述结果集返回给所述API客户端。

[0009] 在其中一些实施例中,所述根据所述查询请求消息对所述顶点索引或所述边索引并行索引扫描包括:对所述顶点索引或所述边索引的并发索引扫描出错后,生成错误码,通过所述错误码定位到失败的所述图分区。

[0010] 在其中一些实施例中,所述顶点或所述边写入所述图分区后,所述方法包括:获取到所述顶点或所述边的更新请求后,通过对所述顶点ID或所述源点ID进行哈希计算,确定更新目标的所述图分区;通过元数据服务器获取所述图分区的列表,所述图分区将所述顶点、所所述顶点的数据、所述顶点索引或所述边的数据、所述边索引并行更新。

[0011] 在其中一些实施例中,所述顶点或所述边写入所述图分区后,所述方法包括:获取到删除请求后,对所述顶点ID或所述源点ID进行哈希计算,确定写入目标的所述图分区;在所述图分区内一并放着所述顶点的数据、所述顶点索引或所述边的数据、所述边索引,将所述图分区里的数据和索引加入到删除列表中一起删除。

[0012] 第二方面,本申请实施例提供了一种基于图数据库的分布式索引系统,所述系统包括图服务器、元数据服务器、存储服务器:所述图服务器获取到顶点或边的写入请求后,通过所述顶点的标记ID生成所述顶点的顶点索引,通过所述边的边类型生成所述边的边索引,其中,所述顶点包括顶点ID和所述标记ID,所述边包括源点ID和所述边类型;所述图服务器通过所述顶点ID或所述源点ID进行哈希运算,将所述顶点或所述边存入对应的图分区内,所述顶点或所述边写入所述图分区后,生成的所述顶点索引或所述边索引写入同一所述图分区中,其中,在图数据库对应的图空间中包含多个所述图分区,所述顶点或所述边写入所述图分区后,保存在所述图分区内。

[0013] 在其中一些实施例中,所述图服务器获取到API客户端发送的查询请求消息,调用元数据服务器验证所述请求有效性,通过存储客户端将所述请求发送到所有存储服务器中的图分区进行对所述顶点索引或所述边索引的并行索引扫描,其中,通过所述图分区ID和所述标记ID或通过所述图分区ID和所述边类型明确所述索引扫描的查询范围;所述图服务器获得所述存储服务器查询得到的结果集,汇总所述结果集返回给所述API客户端。

[0014] 第三方面,本申请实施例提供了一种电子装置,包括存储器和处理器,所述存储器中存储有可运行计算机程序所述计算机程序可执行上述的基于图数据库的分布式索引方法。

[0015] 第四方面,本申请实施例提供了一种存储介质,所述存储介质中存储有计算机程序,所述计算机程序可执行上述的基于图数据库的分布式索引方法。

[0016] 相比于相关技术,本申请实施例提供一种基于图数据库的分布式索引方法和系统,该方法包括获取到顶点或边的写入请求后,通过该顶点的标记ID生成该顶点的顶点索引,通过该边的边类型生成该边的边索引,其中,该顶点包括顶点ID和该标记ID,该边包括源点ID和该边类型;通过该顶点ID或该源点ID进行哈希运算,将该顶点或该边存入对应的图分区内,生成的该顶点索引或该边索引写入同一该图分区中,其中,在图数据库对应的图空间中包含多个该图分区,该顶点或该边写入该图分区后,保存在该图分区内,解决了对Nebula Graph的索引查询效率不高,查询产生的不必要的网络开销较高的问题,使用户可以快速地对Nebula Graph中的顶点和边进行查询。

附图说明

[0017] 此处所说明的附图用来提供对本申请的进一步理解，构成本申请的一部分，本申请的示意性实施例及其说明用于解释本申请，并不构成对本申请的不当限定。在附图中：

[0018] 图1是根据本申请实施例的Nebula Graph分布式架构示意图；

[0019] 图2是根据本申请实施例的基于图数据库的分布式索引方法的索引插入流程示；

[0020] 图3是根据本申请实施例的基于图数据库的分布式索引方法的索引查询流程图；

[0021] 图4是根据本申请实施例的基于图数据库的分布式索引方法的索引更新流程图；

[0022] 图5是根据本申请实施例的基于图数据库的分布式索引方法的索引删除流程；

[0023] 图6是根据本申请实施例的电子设备的内部结构示意图。

具体实施方式

[0024] 为了使本申请的目的、技术方案及优点更加清楚明白，以下结合附图及实施例，对本申请进行描述和说明；应当理解，此处所描述的具体实施例仅仅用以解释本申请，并不用于限定本申请；基于本申请提供的实施例，本领域普通技术人员在没有作出创造性劳动的前提下所获得的所有其他实施例，都属于本申请保护的范围。

[0025] 显而易见地，下面描述中的附图仅仅是本申请的一些示例或实施例，对于本领域的普通技术人员而言，在不付出创造性劳动的前提下，还可以根据这些附图将本申请应用于其他类似情景；此外，还可以理解的是，虽然这种开发过程中所作出的努力可能是复杂并且冗长的，然而对于与本申请公开的内容相关的本领域的普通技术人员而言，在本申请揭露的技术内容的基础上进行的一些设计，制造或者生产等变更只是常规的技术手段，不应当理解为本申请公开的内容不充分。

[0026] 在本申请中提及“实施例”意味着，结合实施例描述的特定特征、结构或特性可以包含在本申请的至少一个实施例中；在说明书中的各个位置出现该短语并不一定均是指相同的实施例，也不是与其它实施例互斥的独立的或备选的实施例；本领域普通技术人员显式地和隐式地理解的是，本申请所描述的实施例在不冲突的情况下，可以与其它实施例相结合。

[0027] 除非另作定义，本申请所涉及的技术术语或者科学术语应当为本申请所属技术领域内具有一般技能的人士所理解的通常意义；本申请所涉及的“一”、“一个”、“一种”、“该”等类似词语并不表示数量限制，可表示单数或复数；本申请所涉及的术语“包括”、“包含”、“具有”以及它们任何变形，意图在于覆盖不排他的包含；例如包含了一系列步骤或模块(单元)的过程、方法、系统、产品或设备没有限定于已列出的步骤或单元，而是可以还包括没有列出的步骤或单元，或可以还包括对于这些过程、方法、产品或设备固有的其它步骤或单元；本申请所涉及的“连接”、“相连”、“耦接”等类似的词语并非限定于物理的或者机械的连接，而是可以包括电气的连接，不管是直接的还是间接的；本申请所涉及的“多个”是指两个或两个以上；“和/或”描述关联对象的关联关系，表示可以存在三种关系，例如，“A和/或B”可以表示：单独存在A，同时存在A和B，单独存在B这三种情况；字符“/”一般表示前后关联对象是一种“或”的关系；本申请所涉及的术语“第一”、“第二”、“第三”等仅仅是区别类似的对象，不代表针对对象的特定排序。

[0028] 在介绍具体的实施例之前，先对本发明实施例所涉及的专业术语进行解释：

[0029] Nebula Graph:一个开源的分布式图数据库,本文的分布式索引技术是Nebula Graph的一个重要功能;

[0030] 图(Graph):一个关系网的最小逻辑单元,可以完整的描述这个关系网中的实体、实体间的关系,以及实体上附带的属性、实体间关系附带的属性;

[0031] 图空间(Graph Space):可以理解为关系网的存储单元,在Nebula Graph中,图空间是一个图的逻辑存储单元;

[0032] 图分区(Graph Partition):Nebula Graph的物理存储分区,一个图空间包含多个图分区,每个图分区拥有多个副本,并分布在不同节点上,通过Raft分布式协议保证图分区的一致性;

[0033] 图模型(Graph Schema):是图中的模型统称,定一个实体或关系数据存储的逻辑结构,例如“人”做为一个实体,对“人”的相关属性可以定义为一个图模型,其包含属性有如:姓名、年龄、性别等;

[0034] 标记模型(Tag Schema):是图模型中的一个子类,主要用于对实体的属性定义,标记模型拥有一个全局唯一的ID,称为标记ID(TagID);

[0035] 边模型(Edge Schema):是图模型中的另一个子类,主要用于对实体间关系的属性定义,例如实体“人”之间的关系可以定义为一个边模型,其属性有:关系(亲友、同事、同学)、交往时间、共同爱好等,边模型同样拥有全局唯一的ID,称为边类型(EdgeType);

[0036] 索引模型(Index Schema):是基于标记模型或边模型的索引模型,包含标记模型或边模型的一个或多个属性,其属性顺序有严格的要求,不同的属性顺序会导致不同的索引存储,索引模型有唯一的ID,称为索引ID(IndexID);

[0037] 顶点(Vertex):一个实体的统称;

[0038] 边(Edge):实体间关系的统称;

[0039] 属性(Property):顶点或边上附带的属性,其数据类型可以为INT、BOOL、STRING或DOUBLE等。

[0040] 在一个实施例中,本申请实施例提供了一种基于图数据库的分布式索引系统,图1是根据本申请实施例的Nebula Graph分布式架构示意图,如图1所示,Nebula Graph的集群架构与用户客户端10(User client)连接,其中,Nebula Graph的集群架构主要分为三层:图服务器层11(Graph Server),元数据服务器层(Meta Server)12,存储服务器层13(Storage Server),以上每层都可以进行分布式部署。

[0041] 图服务器层11主要功能是语法、语义的解析和执行控制等功能;元数据服务器层12主要是对图空间、图分区、图模型的元数据信息的存储,对集群节点信息的存储和集群节点结构的控制等。存储服务器层13主要负责数据和索引的存储、检索等功能,存储服务器层13中包含多个逻辑的图分区,不同的图分区被划分到了不同的存储服务器层13中,通过Raft分布式协议保证图分区副本之间的数据强一致性。

[0042] 在一个实施例中,本申请实施例提供了一种基于图数据库的分布式索引方法,图2是根据本申请实施例的基于图数据库的分布式索引方法的索引插入流程图,如图2所示,包括如下步骤:

[0043] S201,图服务器11获取到顶点或边的写入请求后,通过该顶点的标记ID生成该顶点的顶点索引,通过该边的边类型生成该边的边索引,其中,该顶点包括顶点ID和该标记

ID,该边包括源点ID和该边类型;

[0044] S202,通过该顶点ID或该源点ID进行哈希运算,将该顶点或该边存入对应的图分区内,生成的该顶点索引或该边索引写入同一该图分区中,其中,在图数据库对应的图空间中包含多个该图分区,该顶点或该边写入该图分区后,保存在该图分区内。

[0045] 通过上述步骤S201至S202,相比于现有技术中的对Nebula Graph的索引查询效率不高,查询产生的不必要的网络开销较高的问题,该技术方案通过生成顶点或边的索引并将顶点或边与对应的索引插入到同一个一个图分区里来解决,最后该方案解决了上述问题,使查询时各个所有存储服务器13中的图分区并行查询,提高了Nebula Graph的索引查询效率。

[0046] 在一个实施例中,图3是根据本申请实施例的基于图数据库的分布式索引方法的索引查询流程图,如图3所示,包括如下步骤:

[0047] S301,该顶点或该边写入该图分区后,图服务器11获取到用户客户端10发送的查询请求消息,调用元数据服务器12验证该请求有效性;

[0048] S302,通过存储客户端13将该查询请求消息发送到所有存储服务器13中的图分区,根据该查询请求消息对该顶点索引或该边索引并行索引扫描,其中,通过该图分区ID和该标记ID或通过该图分区ID和该边类型明确该索引扫描的查询范围;

[0049] S303,获得该存储服务器13查询得到的结果集,汇总该结果集返回给该用户客户端10。

[0050] 在一个实施例中,图4是根据本申请实施例的基于图数据库的分布式索引方法的索引更新流程图,如图4所示,包括如下步骤:

[0051] S401,该顶点或该边写入该图分区后,图服务器11获取到该顶点或该边的更新请求后,通过对该顶点ID或该源点ID进行哈希计算,确定更新目标的该图分区;

[0052] S402,通过元数据服务器12获取该图分区的列表,该图分区将该顶点、所该顶点的数据、该顶点索引或该边的数据、该边索引并行更新。

[0053] 在一个实施例中,图5是根据本申请实施例的基于图数据库的分布式索引方法的索引删除流程图,如图5所示,包括如下步骤:

[0054] S501,该顶点或该边写入该图分区后,图服务器11获取到删除请求后,对该顶点ID或该源点ID进行哈希计算,确定写入目标的该图分区;

[0055] S502,在该图分区内一并放着该顶点的数据、该顶点索引或该边的数据、该边索引,将该图分区里的数据和索引加入到删除列表中一起删除。

[0056] 在一个实施例中,顶点存储的键包括图分布区ID(Partition ID)、顶点ID(VertexID)和标记ID(TagID),其中,图分布区ID由3byte的ID和1byte的键类型(KeyType)组成,键类型是对数据类型的划分,例如数据、索引等;图服务器11可以通过对顶点ID的哈希运算,将顶点哈希分布存储到不同的图分区中;一个顶点可以附加多个标记模型,标记ID可以区分不同标记模型的顶点。

[0057] 边存储的键包括图分布区ID(Graph Partition ID)、源点ID(Source VertexID)、边类型(EdgeType)、边权重(EdgeRanking)和目的顶点ID(Destination VertexID),其中,图分布区ID由3byte的ID和1byte的键类型组成,键类型是对数据类型的划分,例如数据、索引等;图服务器11可以对源点ID进行哈希运算,把边分布存储到不同的图分区中,因为源点

的哈希值和边的哈希值相同,所以源点和边将被存储在同一个图分区;一个边可以附加多个边模型,边类型可以区分不同边模型;边权重为边的权重,由用户输入决定;目的顶点ID为边上的目的顶点ID,目的顶点可能会被划分到另外的图分区中。

[0058] 索引的分布式存储是在顶点和边存储的基础上实现的,该顶点索引或该边索引包括:该图分区ID(Graph Partition ID)、该索引ID(IndexID)和属性(Properties);其中,图分布区ID由3byte的ID和1byte的数据类型(DataType)组成,数据类型是对数据类型的划分,此处是索引(Index)类型;索引ID为4byte,索引ID用以区分不同顶点或边的索引项,因为索引是在标记模型或边模型的基础上创建,所以根据索引ID也可以计算出标记ID或边类型;属性为存储的属性值,因为底层基于KV系统存储,底层KV系统可以对属性值进行排序。

[0059] 表1

[0060]

Vertex	Partition ID	Key Type	IndexID	Prop
	3Byte	1Byte	4Byte	0Byte
Vertex 1	1	Type::Index	110	
Vertex 2	2	Type::Index	110	

[0061] 表2

[0062]

Edge	Partition ID	Key Type	Index ID	Prop
	3Byte	1Byte	4Byte	0Byte
Edge 1	1	Type::Index	210	
Edge 2	2	Type::Index	210	

[0063] 下面结合具体应用场景对本发明中存储情况进行详细说明,我们创建一个图空间,包含2个图分区,其中,图空间命名为space,图分区ID分别为1和2;

[0064] 创建一个标记模型,标记模型中仅有一列tag_col,string类型,其中,模型命名为tag,标记ID为100;创建一个边模型,边模型中仅有一列edge_col,string类型,其中,模型命名为edge,边类型为200;

[0065] 基于tag创建标记索引(Tag Index),包含tag中的列tag_col,其中,标记索引命名为tag_index,标记索引ID为110;基于edge创建边索引(Edge Index),包含edge中的列edge_col,其中,边索引命名为edge_index,边索引ID为210;

[0066] 为了说明分布式存储,插入2个顶点,被分别划分到图分区1和图分区2中,插入2个顶点的同时,2个顶点会分别生成1个索引行,其中属性默认为空;顶点1(Vertex 1)ID为1000,顶点2(Vertex 2)ID为1001,此时,索引行Key的结构如表1所示;

[0067] 插入2个边,随着源点ID分别被划分到图分区1和图分区2中,插入2个边的同时,2个边会分别生产1个索引行,并随同边分别被划分到图分区1和图分区2中,其中边权重默认为0,属性默认为空;边1(Edge 1)为从顶点1到顶点2的边,边2(Edge 2)为从顶点2到顶点1的边,此时,索引行Key的结构如表2所示。

[0068] 在一个实施例中,分布式索引查询逻辑为:客户端发送请求到图服务器11,图服务器首先调用元数据服务器12验证请求的有效性,并对查询请求进行优化,优化后的执行计划将通过存储客户端发送到所有存储服务器13中的图分区进行并行的索引扫描;根据索引键(IndexKey)的结构可知,通过图分区ID和标记ID或边类型已经明确了索引扫描的查询范围,仅针对本次请求的图分区进行扫描;因为属性经过底层KV系统的排序,经过优化的执行

计划可以高效的从有序的属性中找出符合条件的结果;经过各存储服务器13的并发查询,统一降结果集返回给存储服务器13;存储服务器13汇总结果集后返回给客户端;自此,索引的分布式查询完成;另外,如果在索引扫描的的请求中,出现索引所包含字段之外的字段,此时通过索引键可以识别出顶点中的顶点ID,或边中的源点ID,此时可以在同一个图分区中查出顶点或边源点的属性。

[0069] 在一个实施例中,在并行扫描的过程中,不可避免的存在硬件或网络的异常导致某个图分区执行失败,当某个图分区执行失败后,图服务器11可以通过错误码具体定位到失败的图分区和失败原因;具体对整体结果集的处理方式由图服务器11决定。

[0070] 在一个实施例中,分布式索引查询插入为:索引的插入操作依赖于顶点或边的插入操作,当插入顶点或边时,会自动生成对应于顶点数据(VertexData)或边数据(EdgeData)的顶点索引或边索引;通过对顶点ID的哈希计算,确定将要写入的目标图分区;图服务器11通过元数据服务器12获取目标图分区的列表,将写入请求发送到对应的图分区中;由此可知,存储服务器13中的图分区可以并行执行写入操作;每个图分区可以通过原子操作写入数据和索引;另外,对多个图分区并发写入操作的一致性处理由事务机制来控制;对于upsert的处理,索引的主要逻辑是查询旧索引->如果存在则删除旧索引->插入新的索引,所有操作需要保证原子性。

[0071] 在一个实施例中,分布式索引查询更新为:索引的更新操作依赖于顶点或边的更新操作;当更新顶点或边时,通过对顶点的顶点ID或边的源点ID进行哈希计算,确定要包含更新数据和索引的目标图分区;图服务器11将更新请求发送到目标图分区进行更新操作,因为目标图分区可能分布于多个不同的存储服务器13中,因此分布于不同的存储服务器13中的图分区可以执行并行操作。

[0072] 在一个实施例中,分布式索引查询删除为:当删除顶点或边时,其对应的顶点索引或边索引也会被一并删除,顶点或边和其对应的索引被存储到了相同的图分区中,所以在删除过程中可以通过原子操作将数据和索引一起删除;在删除的过程中,首先通过顶点ID或边类型查到符合删除条件的数据,通过这些查到的数据,可以构造出将要删除的目标索引键,将索引键加入到删除列表中进行原子删除操作。

[0073] 在一个实施例中,通过对顶点ID进行哈希计算,将顶点或边均衡分布到所有的图分区中;因为索引与其对应的顶点或边存储在同一个图分区中,所以索引数据也相应的进行了均衡分布;由此可以避免数据倾斜的问题。

[0074] 在一个实施例中,索引查询往往不会针对某个顶点或边,多数时间会检索符合判断条件的任何顶点或边,因此在分布式索引并发查询时,可以对所有的图分区进行并发查询;有效提高了检索效率。

[0075] 在一个实施例中,当通过索引的分布式并发查询出符合条件的顶点或边后,往往会通过这些顶点或边进行进一步的计算,因为顶点和边的索引被分配到了相同的图分区中,在做对顶点或边做进一步计算时在当前图分区中执行即可;避免了跨存储服务器13或跨图分区计算所产生的不必要的网络开销或其他资源开销。

[0076] 在一个实施例中,图6是根据本申请实施例的电子设备的内部结构示意图,如图6所示,提供了一种电子设备,该电子设备可以是服务器,其内部结构图可以如图6所示。该电子设备包括通过系统总线连接的处理器、存储器、网络接口和数据库。其中,该电子设备的

处理器用于提供计算和控制能力。该电子设备的存储器包括非易失性存储介质、内存储器。该非易失性存储介质存储有操作系统、计算机程序和数据库。该内存储器为非易失性存储介质中的操作系统和计算机程序的运行提供环境。该电子设备的数据库用于存储数据。该电子设备的网络接口用于与外部的终端通过网络连接通信。该计算机程序被处理器执行时以实现一种基于图数据库的分布式索引方法。

[0077] 本领域技术人员可以理解,图6中示出的结构,仅仅是与本申请方案相关的部分结构的框图,并不构成对本申请方案所应用于其上的电子设备的限定,具体的电子设备可以包括比图中所示更多或更少的部件,或者组合某些部件,或者具有不同的部件布置。

[0078] 本领域普通技术人员可以理解实现上述实施例方法中的全部或部分流程,是可以通过计算机程序来指令相关的硬件来完成,该计算机程序可存储于一非易失性计算机可读取存储介质中,该计算机程序在执行时,可包括如上述各方法的实施例的流程;其中,本申请所提供的各实施例中所使用的对存储器、存储、数据库或其它介质的任何引用,均可包括非易失性和/或易失性存储器;非易失性存储器可包括只读存储器(ROM)、可编程ROM(PROM)、电可编程ROM(EPROM)、电可擦除可编程ROM(EEPROM)或闪存;易失性存储器可包括随机存取存储器(RAM)或者外部高速缓冲存储器;作为说明而非局限,RAM以多种形式可得,诸如静态RAM(SRAM)、动态RAM(DRAM)、同步DRAM(SDRAM)、双数据率SDRAM(DDRSDRAM)、增强型SDRAM(ESDRAM)、同步链路(Synchlink)DRAM(SLDRAM)、存储器总线(Rambus)直接RAM(RDRAM)、直接存储器总线动态RAM(DRDRAM)、以及存储器总线动态RAM(RDRAM)等。

[0079] 以上实施例仅表达了本申请的几种实施方式,其描述较为具体和详细,但并不能因此而理解为对发明专利范围的限制。应当指出的是,对于本领域的普通技术人员来说,在不脱离本申请构思的前提下,还可以做出若干变形和改进,这些都属于本申请的保护范围。因此,本申请专利的保护范围应以所附权利要求为准。

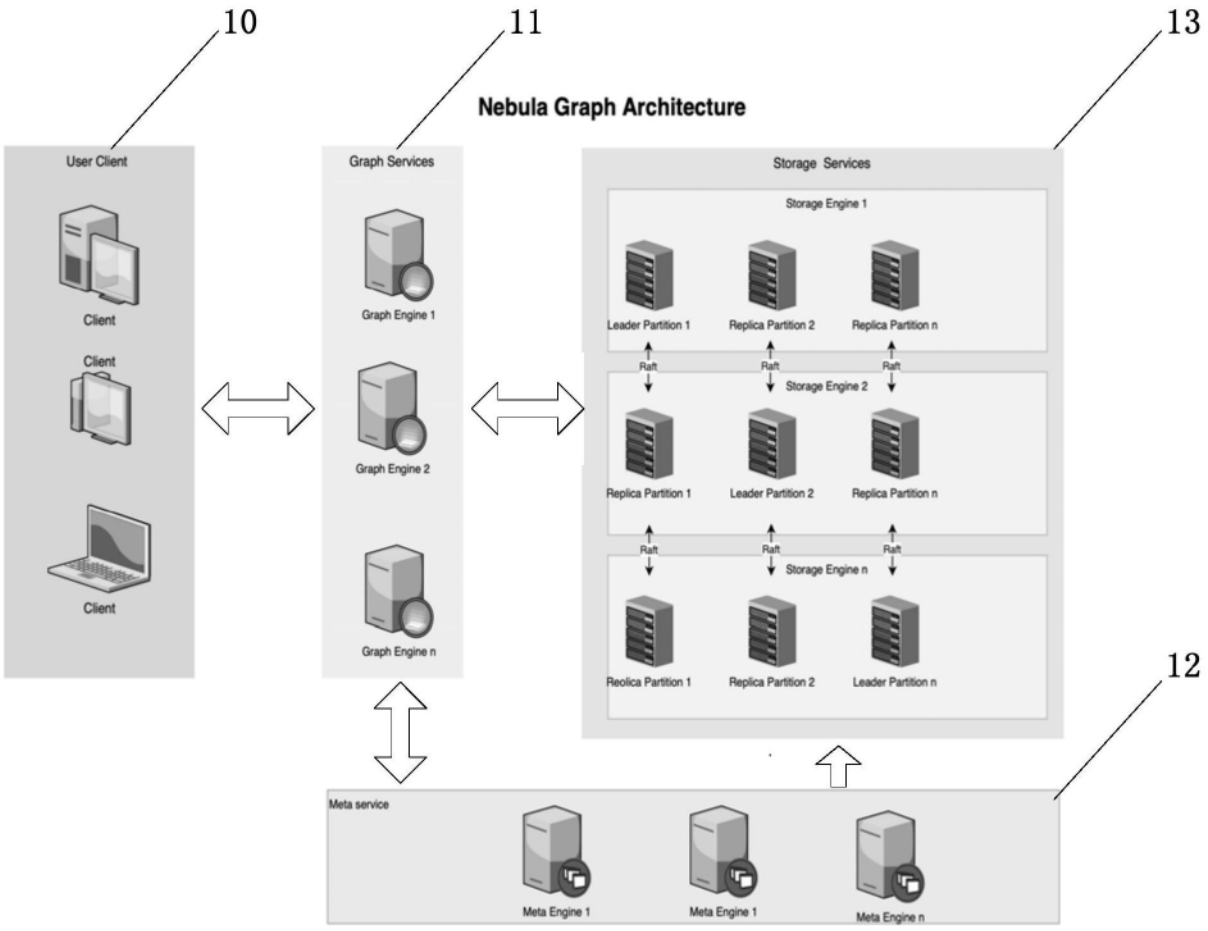

10
11
13
12
Nebula Graph Architecture
User Client
Client
Client
Client
Graph Services
Graph Engine 1
Graph Engine 2
Graph Engine n
Storage Services
Storage Engine 1
Leader Partition 1
Replica Partition 2
Replica Partition n
Raft
Storage Engine 2
Replica Partition 1
Leader Partition 2
Replica Partition n
Storage Engine n
Reolica Partition 1
Replica Partition 2
Leader Partition n
Meta service
Meta Engine 1
Meta Engine 1
Meta Engine n

图1

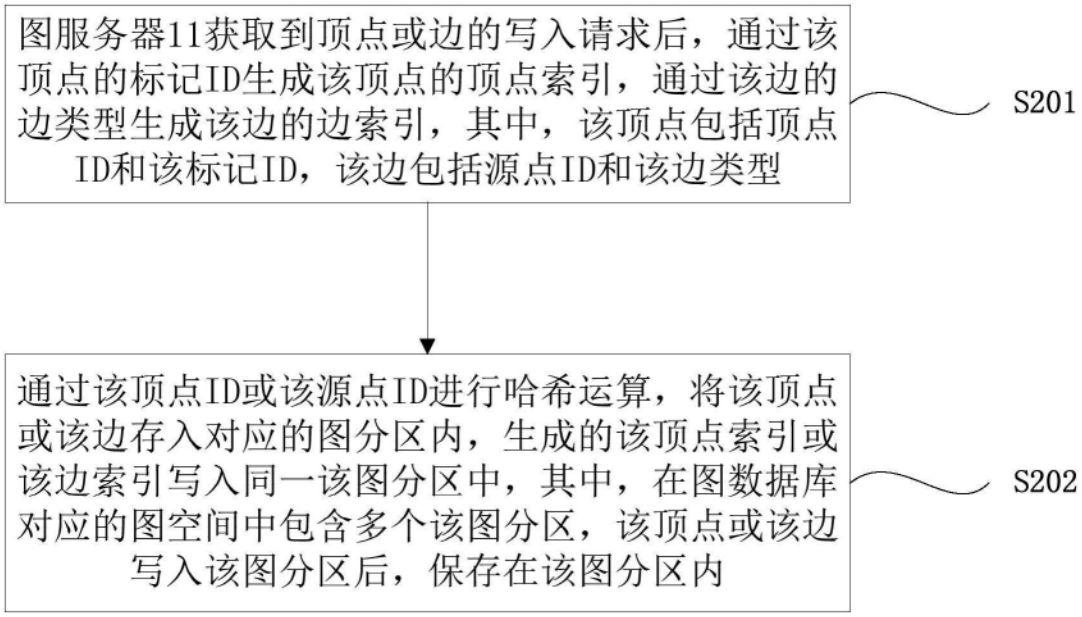

图服务器11获取到顶点或边的写入请求后，通过该顶点的标记ID生成该顶点的顶点索引，通过该边的边类型生成该边的边索引，其中，该顶点包括顶点ID和该标记ID，该边包括源点ID和该边类型
S201
通过该顶点ID或该源点ID进行哈希运算，将该顶点或该边存入对应的图分区内，生成的该顶点索引或该边索引写入同一该图分区中，其中，在图数据库对应的图空间中包含多个该图分区，该顶点或该边写入该图分区后，保存在该图分区内
S202

图2

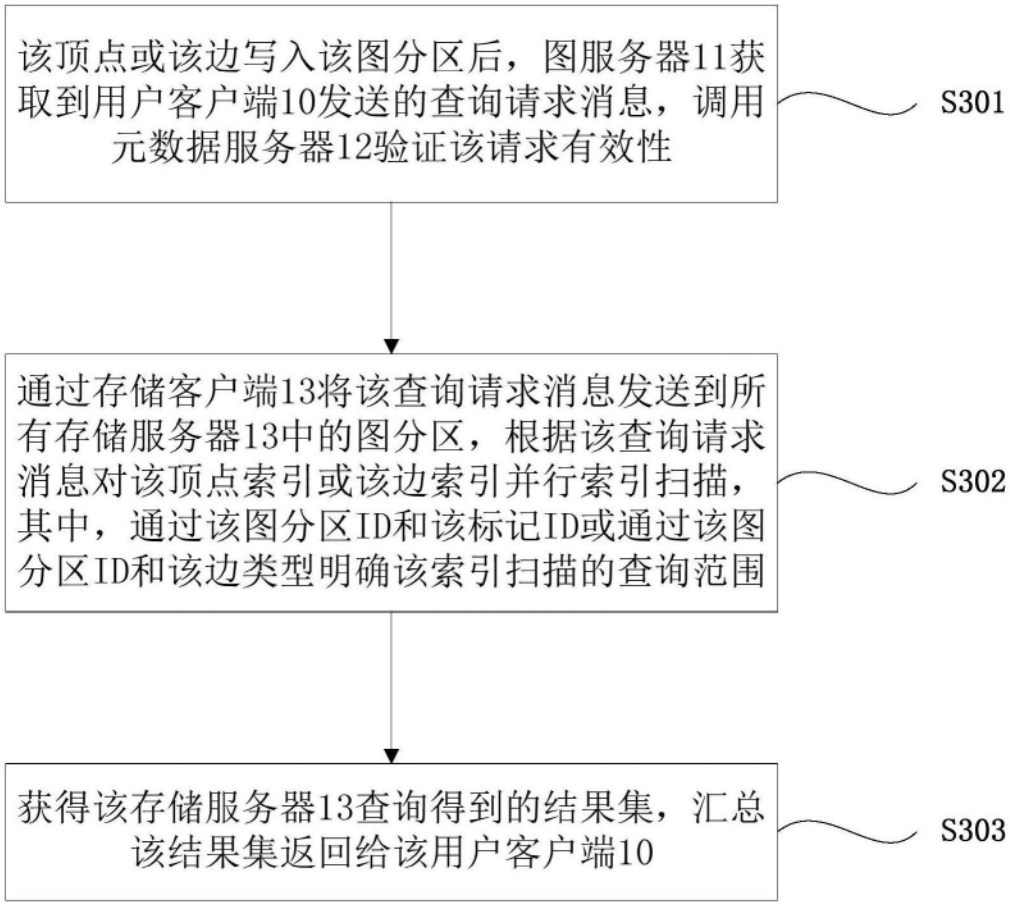

该顶点或该边写入该图分区后，图服务器11获取到用户客户端10发送的查询请求消息，调用元数据服务器12验证该请求有效性
S301
通过存储客户端13将该查询请求消息发送到所有存储服务器13中的图分区，根据该查询请求消息对该顶点索引或该边索引并行索引扫描，其中，通过该图分区ID和该标记ID或通过该图分区ID和该边类型明确该索引扫描的查询范围
S302
获得该存储服务器13查询得到的结果集，汇总该结果集返回给该用户客户端10
S303

图3

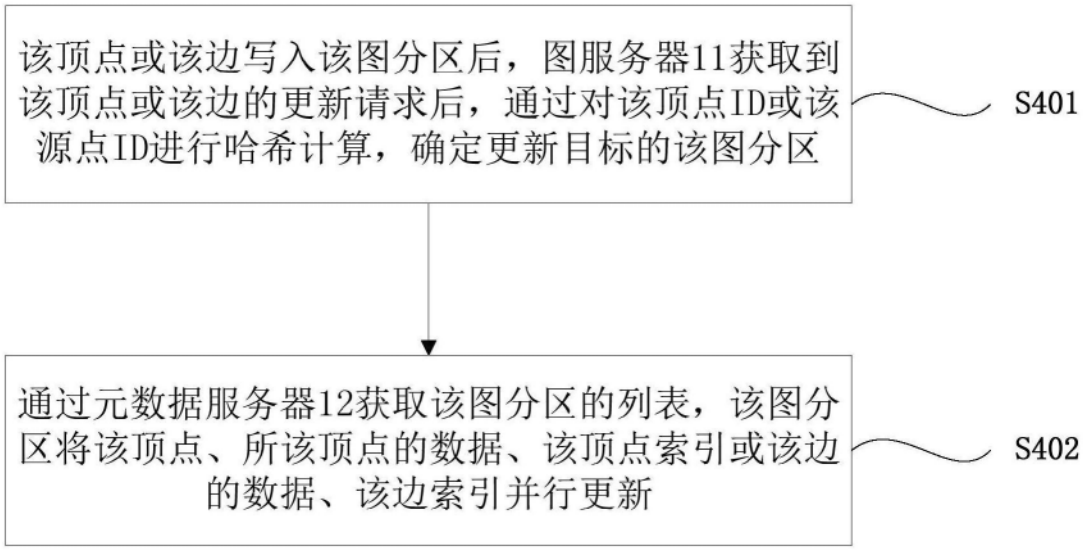

该顶点或该边写入该图分区后，图服务器11获取到该顶点或该边的更新请求后，通过对该顶点ID或该源点ID进行哈希计算，确定更新目标的该图分区
S401
通过元数据服务器12获取该图分区的列表，该图分区将该顶点、所该顶点的数据、该顶点索引或该边的数据、该边索引并行更新
S402

图4

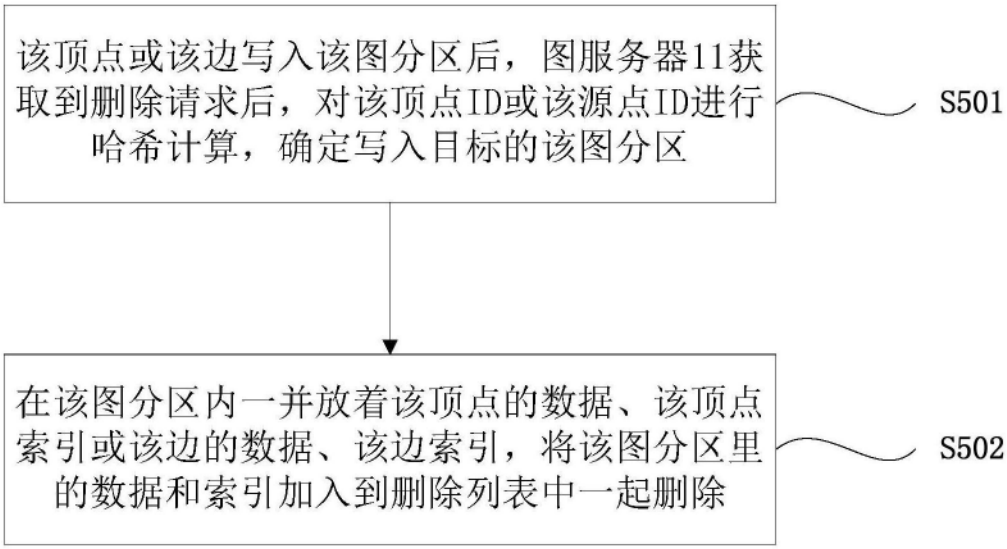

该顶点或该边写入该图分区后，图服务器11获取到删除请求后，对该顶点ID或该源点ID进行哈希计算，确定写入目标的该图分区
S501
在该图分区内一并放着该顶点的数据、该顶点索引或该边的数据、该边索引，将该图分区里的数据和索引加入到删除列表中一起删除
S502

图5

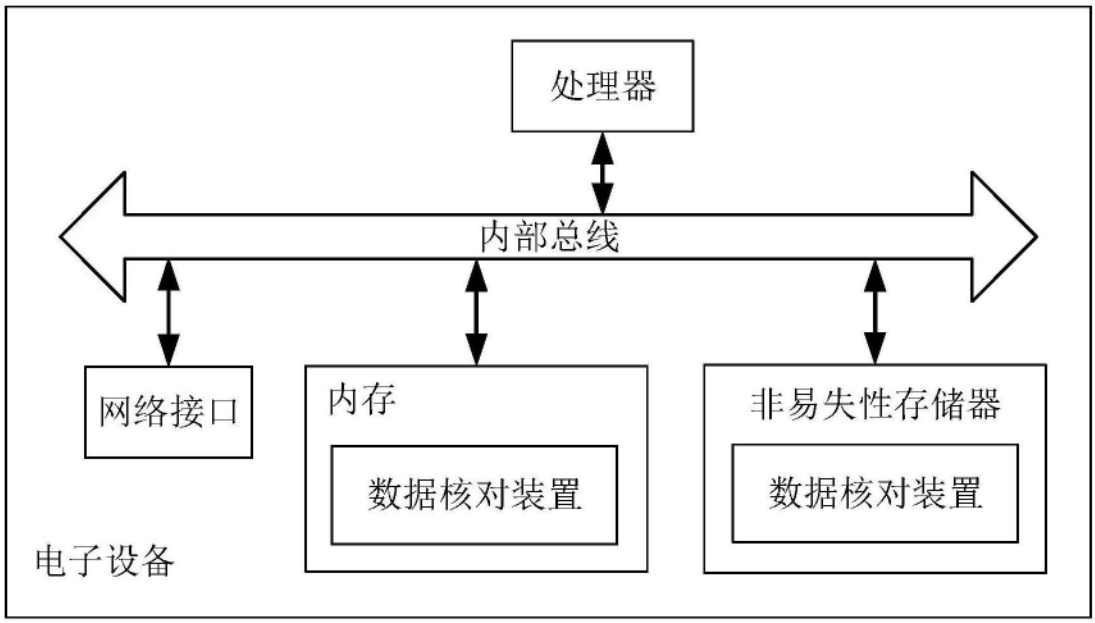

处理器
内部总线
网络接口
内存
数据核对装置
非易失性存储器
数据核对装置
电子设备

图6