(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 109313640 B
(45)授权公告日 2022.03.04

(21)申请号 201780030553.3
(22)申请日 2017.03.24
(65)同一申请的已公布的文献号
申请公布号 CN 109313640 A
(43)申请公布日 2019.02.05
(30)优先权数据
2016901204 2016.03.31 AU
2016202911 2016.05.05 AU
(85)PCT国际申请进入国家阶段日
2018.11.16
(86)PCT国际申请的申请数据
PCT/AU2017/050265 2017.03.24
(87)PCT国际申请的公布数据
WO2017/165914 EN 2017.10.05

(73)专利权人 慧咨环球有限公司
地址 澳大利亚新南威尔士
(72)发明人 布雷特·安东尼·希勒
(74)专利代理机构 中科专利商标代理有限责任公司 11021
代理人 潘军

(51)Int.Cl.
G06F 16/2453(2019.01)
(56)对比文件
CN 104915450 A,2015.09.16
CN 104573092 A,2015.04.29
CN 103902537 A,2014.07.02
CN 102163223 A,2011.08.24
审查员 杨琦

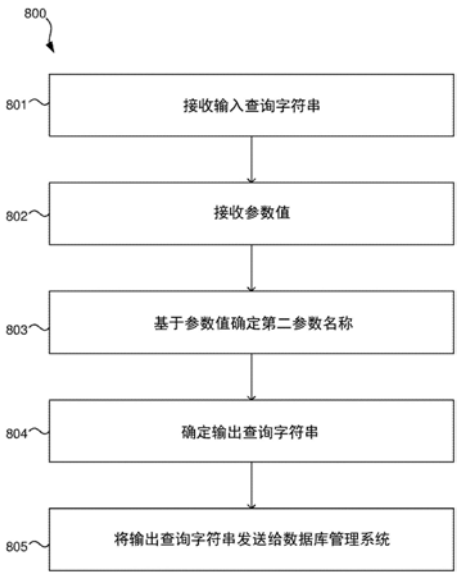
权利要求书3页 说明书15页 附图10页

(54)发明名称

用于数据库优化的方法和系统

(57)摘要

本公开涉及用于数据库优化的方法和系统，具体涉及提高数据库查询的性能。代理服务器接收输入查询字符串和查询字符串中第一参数名称的参数值。代理服务器确定基于所述参数值的并与第一参数名称不同的第二参数名称。然后，代理服务器基于输入查询字符串确定输出查询字符串。输出查询字符串包括具有字段名称和第二字段值的过滤子句，所述输出查询字符串的第二字段值基于第二参数名称。代理服务器最终将输出查询字符串发送给数据库管理系统，以使数据库管理系统使用基于输出查询字符串中的第二参数名称的执行计划来执行数据库查询。



CN 109313640 B

1.一种用于查询数据库的方法,所述方法包括:

接收输入查询字符串,所述输入查询字符串包括具有字段名称和第一字段值的过滤子句,所述第一字段值指示第一参数名称;

接收所述第一参数名称的参数值;以及

向数据库管理系统发送输出查询字符串和所述参数值,以使所述数据库管理系统执行数据库查询;

其特征在于,所述方法进一步包括:

确定基于所述参数值的并与所述第一参数名称不同的第二参数名称;

基于所述输入查询字符串确定所述输出查询字符串,所述输出查询字符串包括具有所述字段名称和第二字段值的过滤子句,所述输出查询字符串的所述第二字段值基于所述第二参数名称;以及

将所述输出查询字符串和所述参数值发送给所述数据库管理系统,以使所述数据库管理系统使用基于所述输出查询字符串中的所述第二参数名称的执行计划来执行数据库查询;以及

将所述第二参数名称重用于后续查询。

2.根据权利要求1所述的方法,其中确定所述输出查询字符串包括:用所确定的第二参数名称替换所述输入查询字符串中的所述第一参数名称。

3.根据权利要求1或2所述的方法,其中所述第二参数名称包括所述第一参数名称以及后缀或前缀,并且确定所述第二参数名称包括确定所述后缀或前缀。

4.根据权利要求1或2所述的方法,其中所述第二参数名称是所述第一参数名称的后缀或前缀,并且确定所述第二参数名称包括确定所述后缀或前缀。

5.根据权利要求1或2所述的方法,其中

所述字段名称指代外部关键字,并且

确定所述第二参数名称包括基于与所述外部关键字相关联的表中的行数来确定所述第二参数名称。

6.根据权利要求5所述的方法,还包括:在确定了与所述外部关键字相关联的表中的行数低于阈值时,确定所述第二参数名称以使得所述第二参数名称对于与所述外部关键字相关联的表中的每一行是唯一的。

7.根据权利要求6所述的方法,其中

所述第一参数名称的参数值包括字符串,并且

确定所述第二参数名称以使得所述第二参数名称对于与所述外部关键字相关联的表中的每一行是唯一的包括:基于所述第一参数名称的参数值,将后缀附加到所述第一参数名称。

8.根据权利要求7所述的方法,其中所述后缀包括所述参数值的前两个或更多个字母。

9.根据权利要求1或2所述的方法,其中

所述方法还包括接收所述字段名称的直方图数据,并且

确定所述第二参数名称包括:基于所述直方图数据确定所述第二参数名称。

10.根据权利要求9所述的方法,其中

基于所述直方图数据确定所述第二参数名称包括:确定所述第二参数名称以使得所述

第二参数名称对于每个直方图阶梯是唯一的。

11.根据权利要求9所述的方法,其中基于所述直方图数据确定所述第二参数名称包括:基于关于直方图阶梯的多个预定义条件确定所述第二参数名称,使得对于每个预定义条件而言,所述第二参数名称对于直方图阶梯中满足该条件的所有参数值都是相同的。

12.根据权利要求11所述的方法,其中确定所述第二参数名称包括:在确定了所述参数值处于具有低于预定阈值的频率值的直方图阶梯中时,使用预定义参数名称作为所述第二参数名称。

13.根据权利要求1或2所述的方法,其中

所述参数值与日期和/或时间有关,

所述方法还包括:基于所述日期和/或时间确定时间段的长度,并且

确定所述第二参数名称包括:基于所述时间段的长度确定所述第二参数名称。

14.根据权利要求13所述的方法,其中确定所述第二参数名称包括:基于关于所述时间段的长度的多个预定义条件来确定所述第二参数名称,使得对于每个预定义条件而言,所述第二参数名称对于所述时间段的长度满足该条件的所有参数值而言是相同的。

15.根据权利要求1或2所述的方法,其中确定所述第二参数名称包括确定所述字段名称是否与过滤后索引有关,并且在确定了所述字段名称与过滤后索引有关时,使用所述参数值作为所述第二参数名称。

16.一种用于查询数据库的计算机系统,所述计算机系统包括:

输入端口,

用于接收输入查询字符串,所述输入查询字符串包括具有字段名称和第一字段值的过滤子句,所述第一字段值指示第一参数名称;并且

用于接收所述第一参数名称的参数值;

处理器,用于基于所述输入查询字符串确定输出查询字符串;以及

输出端口,用于向数据库管理系统发送所述输出查询字符串和所述参数值,以使所述数据库管理系统执行数据库查询;

其他特征在于:

所述处理器还被配置为确定基于所述参数值的并与所述第一参数名称不同的第二参数名称;

所述输出查询字符串包括具有所述字段名称和第二字段值的过滤子句,所述输出查询字符串的所述第二字段值基于所述第二参数名称;

所述输出端口还被配置为将所述输出查询字符串和所述参数值发送给所述数据库管理系统,以使所述数据库管理系统使用基于所述输出查询字符串中的所述第二参数名称的执行计划来执行数据库查询,以及

所述处理器还被配置为将所述第二参数名称重用于后续查询。

17.一种数据库代理系统,所述代理系统包括:

输入端,所述输入端连接到客户端计算机用来从所述客户端计算机接收输入查询字符串,所述输入查询字符串包括具有字段名称和第一字段值的过滤子句,所述第一字段值指示第一参数名称,并且用来接收所述第一参数名称的参数值;

处理器,用于基于所述输入查询字符串确定输出查询字符串;以及

输出端口，用于向数据库管理系统发送所述输出查询字符串和所述参数值，以使所述数据库管理系统执行数据库查询；

其他特征在于：

所述处理器还被配置为确定基于所述参数值的并与所述第一参数名称不同的第二参数名称；

所述输出查询字符串包括具有所述字段名称和第二字段值的过滤子句，所述输出查询字符串的所述第二字段值基于所述第二参数名称；

所述输出端口连接到所述数据库管理系统，以将所述输出查询字符串和所述参数值发送给所述数据库管理系统，以使所述数据库管理系统使用基于所述输出查询字符串中的所述第二参数名称的执行计划来执行数据库查询；以及

所述处理器还被配置为将所述第二参数名称重用于后续查询。

18.一种数据库管理系统，所述数据库管理系统包括：

输入端，所述输入端被连接用来接收输入查询字符串，所述输入查询字符串包括具有字段名称和第一字段值的过滤子句，所述第一字段值指示第一参数名称，并且用来接收所述第一参数名称的参数值；

处理器，用于基于所述输入查询字符串确定输出查询字符串；以及

数据库引擎，用于使用基于所述输出查询字符串中的第二参数名称的执行计划来执行数据库查询，

其特征在于：

所述处理器还被配置为确定基于所述参数值的并与所述第一参数名称不同的所述第二参数名称；

所述输出查询字符串包括具有所述字段名称和第二字段值的过滤子句，所述输出查询字符串的所述第二字段值基于所述第二参数名称；以及

所述处理器还被配置为将所述第二参数名称重用于后续查询。

用于数据库优化的方法和系统

[0001] 相关申请的交叉引用

[0002] 本申请要求于2016年3月31日提交的澳大利亚临时申请2016901204的优先权，其内容通过引用的方式并入本文中。

[0003] 本申请还要求于2016年5月5日提交的澳大利亚完整申请2016202911的优先权，其内容通过引用的方式并入本文中。

技术领域

[0004] 本公开涉及提高数据库查询的性能的方法和系统。

背景技术

[0005] 对于大多数公司而言，按照某种方式或形式对数据进行管理是非常重要的。举例来说，货运公司会跟踪货物，比如，货物的发货地、目的地和当前位置以及海关通关状态。在物流行业，每天产生的数据相当得大。

[0006] 图1示出了采用笔101和纸102的方法，其中物流经理以表格形式手工保存不同产品的书面记录。纸102包括多列，所述多列包括产品(product)列103、发货地(origin)列104、目的地(destination)列105和位置(location)列106。物流经理能够正确地填写城市名称，并且在产品从发货地运往目的地时更新当前位置列106中的城市名称。

[0007] 一旦数据发生快速变化并且需要监控大量产品，这种采用笔和纸的方法在实际使用中很快就会遇到局限性。实现更高效且有效的数据管理的解决方案是求助于计算机化系统。

[0008] 图2示出了计算机化的数据管理系统，该计算机化的数据管理系统包括显示器201、输入设备202(如键盘)、处理器203和硬盘204。硬盘204存储文本文件205。文本文件205包含排成行的人类可读字符，这样，每个产品构成一行，如图2所示。用分号对列进行分隔。表格结构基本上与图1中的相同，但却是存储在计算机存储器上，而不是纸上。

[0009] 计算机化允许使用程序代码来存储和访问数据。具体地，开发者可以创建出用于访问数据的程序代码，这种程序代码查询文本文件205中的数据，以便返回满足特定标准的行数。例如，程序代码可以通过如下方式来计算出有多少产品当前位于“Los Angeles(洛杉矶)”：遍历所有行，通过分号分割每一行，并且如果第四分割元素是“Los Angeles”，则使计数器递增。例如：

[0010] line＝getline(datafile)；

[0011] if(line.split(“;”)[3]＝＝“Los Angeles”){count++}；

[0012] 虽然基于文本文件205的解决方案相比起使用纸102的解决方案来说带来了显著的优点，但是也存在着局限性，造成这些局限性的部分原因在于处理器203存储和访问文本文件205的方式。此外，当每天接收海量数据时，需要数十个表，甚至有时候需要数百个表，而且这些表经常还互相引用。存储和搜索策略对于优化进程中的CPU时间来说非常重要。

[0013] 图3更详细地示出了硬盘204。具体地，硬盘204包括转盘301以及承载读取和写入

头303的臂302。头303将数据写入扇区，比如，转盘301上的示例性扇区304。扇区304的大小由硬盘制造商确定，并且可以是512字节或4千字节。这意味着转盘301上由文本文件205占据的空间量是扇区大小的倍数，并且多个文件不会共享一个扇区。随着文本文件205的增大，它可能不再适合于一个扇区，磁盘管理器将第二扇区分配给文本文件205。第二扇区可能不与第一扇区304直接相邻，这被称为碎片化。因此，当读取整个文件时，臂302在分配给文本文件205的不同扇区之间移动读取头303。由于此操作涉及到机械移动并且臂302的速度受到限制，因此，整个文件的读取可能明显减慢。这是在读取整个文本文件205时(比如，在计算当前位于Los Angeles的所有产品时)存在的特定问题。在多个用户同时查询同一文本文件205并且读取头303在扇区之间更频繁移动的情况下，该问题会进一步被加剧。这样，响应时间可能会受到影响。

[0014] 使用文件来存储数据的另一个缺点在于：用于查询数据的程序是复杂的且容易出错。具体而言就是，难以创建提供可靠结果且不浪费CPU时间的对数据文件组合的复合查询。

[0015] 为了解决这些复杂性和功能局限性的问题，可以使用数据库管理系统(DBMS)。例如，SQL数据库(比如，Oracle的数据库如服务(DAaaS)、MySQL和Microsoft SQL服务器)可被用于存储和访问海量数据。在数据库术语中，文本文件205的每一行可以被称为一条记录，而文本文件205的所有记录被称为表。“产品”、“发货地”、“目的地”和“位置”列被称为字段。但是，术语“行”和“列”同样可以用于数据库。数据库通常托管许多不同的表，这些表常常互相引用。

[0016] DBMS将文本文件205中的数据的行分成每个大小为8KB的片段或子集。在转盘301上的存储这些子集中的每一个子集的存储空间被称为页。八个页合在一起构成了区(extent)。这意味着512KB的扇区保存8个区中的64个页，并且这8个区通常是作为连续数据存储在扇区中，以减少臂302的移动，进而缩短延迟。当DBMS从页读取行时，DBMS将整个页加载到易失性存储器(如RAM)上的缓存中。然后，对同一页中的行的进一步查询可以使用该缓存版本。由于RAM不具有移动臂302或其他机械部件，而是直接通过位线和地址线寻址数据，因此，RAM缓存的使用极大地缩短了访问时间。

[0017] 当处理器203创建新行时，DBMS可以将该行存储在最后一页的末尾。也就是说，DBMS不会根据数据来更改转盘301上的数据的顺序。由于此举产生了没有顺序的表，因此，它被称为堆(heap)，如图2所示。在堆中，大多数查询访问每一行，并因此从硬盘204检索整个表。访问每行花费大量的系统时间。

[0018] 图4示出了有序表400，有序表400也被称为聚簇索引，当表较大时，当经常从表中查询数据范围时以及当数据经常按排列顺序返回时，采用这种聚簇索引，而不是堆。需要注意的是，图4中显示行的顺序也代表了行在转盘301上的存储顺序。对产品名称(比如，以“L”开头的所有行)的查询可以在不从转盘301检索所有行的情况下执行。处理器203可以检索处于中间的或大致处于中间的行，并确定该行是否以字母表中“L”之前或之后的字母开始。例如，如果行以“T”开始，则处理器203继而仅考虑行集的前半部分。处理器203检索处于前半部分的中间的行，并检查起始字母是在字母表中的“L”之前还是之后，依此类推。尤其是对于大型表而言，这样做可以快速地缩小搜索空间并显著提高性能。此外，一旦处理器203找到以“L”开始的一行，则以“L”开始的任何其他行与该第一个定位的行直接相邻。这样，处

理器203不会检索除直接相邻的那些行之外的其他行。需要注意的是，这是简化后的解释，而实际数据库使用的是更复杂的数据结构，比如B树(B-tree)。然而，相同的概念是适用的，唯一的区别在于：树的叶子表示行，而处理器203可以通过忽略与标准不匹配的那些节点下方的整个子树来找到索引中的行。

[0019] 图5a示出了使用“ID”列501生成的聚簇索引500的另一示例。这在检索给定了特定ID的行时特别有效。返回参考图4，需要注意的是，表仅具有一个聚簇索引，这是因为该表一次只能以一种方式进行排序。因此，图4中的聚簇索引允许高效地搜索产品字段，但不能高效地搜索其他字段。为了解决这个问题，DBMS可以创建非聚簇索引。

[0020] 图5b示出了作为附加数据存储在转盘301上的非聚簇索引510。与聚簇索引相反的是，非聚簇索引是一种特殊类型的索引，其中索引的逻辑顺序与行在盘上的物理排序顺序不匹配。非聚簇索引510基本上是包括一个字段511(本示例中为“发货地”字段)和对图5a中原始表的引用512(在这种实例下为“ID”)的表。非聚簇索引510按发货地511进行排序。处理器203现在可以应用如上所述的相同搜索策略(即执行计划)来例如计算出发自New York(纽约)的所有产品。如果总计数是唯一请求的输出，则处理器203不从转盘301检索除非聚簇索引510(其可能会适合一个页)之外的任何数据。因此，查询速度是非常快的。如果请求了来自匹配行的其他数据，则处理器203可以检索如ID字段512所指示的行，如针对New York示例中是行“1”、“2”和“6”。可以针对多个相应的列/字段创建多个非聚簇索引，以用于存储索引数据的附加存储如图5b所示为代价来提高速度。

[0021] 图6a和图6b示出了图6a中的第一表600和图6b中的第二表610的示例，第二表610被第一表600引用。在此，第一表600中的城市名称被指向第二表610的标识符替换。这样做实现了如下优点：如果更改了一个城市的名称，那么仅需要更改第二表610中的一个条目以反映出这种更改。此外，存储所占用的空间更高效，这是因为每个城市的字符串仅存储一次。分割成多个表遵循了范式数据库的设计概念。第一表600中的城市标识符被称为外部关键字，因为它们指向了其他表(即第二表610)中的条目。例如，为了执行查询以计算出当前位于Los Angeles的所有产品，处理器203使用城市标识符作为交叉引用将第一表600与第二表610连接起来。

[0022] 为了简化说明，返回参考图5a，如参考图2中的堆结构描述的通过检索所有行来查询数据库被称为“表扫描”。相对地，如参考图4、图5a和图5b描述的通关使用索引来查询数据库被称为“索引查找”。一般而言，索引查找在执行方面比表扫描更快，其原因在于：处理器203不检索整个表。然而，在大多数行与搜索标准匹配的情况下，处理器203无论如何都会检索这些大多数行，这意味着表扫描实际上是更加高效的操作。尤其是在记住检索全体页时，其使得例如如果50％的行与标准匹配，则处理器203可能会检索整个表。

[0023] 扫描与查找之间的不同性能说明了处理器203可以在这两个选项之间做出决策以优化性能。需要注意的是，上述示例是进行了简化的，并且选项的数量通常远远大于二。特别是在如图6a和图6b所示的对多个表进行了组合的情况下，选项的数量可以呈指数级地增长。将针对查询的不同方面的不同选项加以组合以实现期望的结果被称为如上所述的执行计划。对于数据库而言，拥有600个表或更多的表，每个表有数百万个条目，每周数百万个查询，这种情况并不少见。

[0024] 处理器203从多个执行计划中选出在执行时间和存储要求方面估计成本最小的一

个执行计划。从计算方面来看，选出最佳执行计划可能成本昂贵，这是因为要评估许多不同的选项和组合。因此，处理器203旨在尽可能避免创建新的执行计划。如上所述，处理器203将生成的执行计划存储在计划缓存中，使得后续查询可以重用该执行计划。这样做产生了如下的问题：在选择存储执行计划以避免优化延迟与创建对于特定查询而言是最佳的新执行计划之间做出决策。另一个约束条件是执行计划的不受控制的产生导致了“计划膨胀”，“计划膨胀”指的是占据大量数据库RAM缓存的大量计划，因此，这些数据库RAM缓存不能用于对数据库行进行缓存。针对计划膨胀问题的一种解决方案是执行定期的计划冲刷(flush)，其从缓存中删除掉所有计划。由于重新创建计划通常使用大量的计算能力，因此计划冲刷通常是不切实际的，并且可能会在重新创建期间导致延长的响应时间。

[0025] 在查询包含参数(比如，SELECT*FROM products WHERE origin＝@origin_param)的情况下，DBMS通常会创建一个执行计划，然后每次针对参数@origin_param的不同值执行相同的查询时重用相同的执行计划。这样做对于@origin_param的不同值的频率彼此之间明显不同的大型数据库而言通常产生次优结果。对于具有接近1000个交叉引用的表的数据库(每个表中有数百万个条目，每周有数百万个查询)，此问题变得尤为严重。

[0026] 对本说明书中包括的文献、动作、材料、装置、物品等的任何论述不应被视为承认了：任何或所有这些事项构成了现有技术基础的一部分或者是在本申请的每项权利要求的优先权日之前存在的本公开相关领域内的公知常识。

[0027] 在整个说明书中，词语“包括”，或者诸如“包含”或“具有”之类的变型将被理解为暗示了包括所陈述的元素、整数或步骤，或者元素、整数或步骤的组合，但是没有排除任何其他元素、整数或步骤，或者元素、整数或步骤的组合。

发明内容

[0028] 再次参考图2，处理器203优选地从多个执行计划中选出在执行时间和存储要求方面估计成本最小的一个执行计划。然而从计算方面来看，选出最佳执行计划可能成本昂贵，这是因为要评估许多不同的选项和组合。这样做产生了如下的问题：在选择存储的执行计划以避免优化延迟与创建对于特定查询而言是最佳的新执行计划之间做出决策。但是，SQL服务器将从计划缓存中排除掉不常用的计划。结果就是频繁生成新计划，这以CPU使用为代价。

[0029] 另一个约束条件是执行计划的不受控制的产生导致了“计划膨胀”，“计划膨胀”指的是占据大量数据库RAM缓存的大量计划，因此这些数据库RAM缓存不能用于对数据库记录进行缓存。针对计划膨胀问题的一种解决方案是执行定期的计划冲刷，其从缓存中删除掉所有计划。由于重新创建计划通常使用大量计算能力，因此，计划冲刷通常是不切实际的，并且可能会在重新创建期间导致延长的响应时间。重新创建大量计划被称为“计划风暴”。在某种情况下(如实时交易)，花费几分钟从计划风暴中恢复可能都是灾难性的。

[0030] 虽然SQL Server确实重用了计划，但是例如在查询包含参数(比如，SELECT*IN products WHERE origin＝@origin_param)的情况下，DBMS通常会创建一个执行计划，然后每次针对参数@origin_param的不同值执行相同的查询时重用相同的执行计划。这种方法对于大型数据库(其中@origin_param的不同值的频率彼此之间明显不同)而言通常产生次优结果。例如，当针对代表了表的相当大部分的标准执行搜索时，将会使用诸如表扫描之类

的计划。如果执行的第一次搜索针对的是美国，那么后续请求(比如，FJ-fiji)将导致表扫描，尽管它们并不是最佳的。相反，如果第一次搜索的是FJ-，则将会执行索引查找。于是，后来的搜索(如美国)将是非常低效的对大量数据的索引查找。对于具有接近1000个交叉引用的表的数据库(每个表中有数百万个条目，每周有数百万个查询)，此问题变得尤为严重。

[0031] 为了避免浪费的计划膨胀，避免可能是灾难性的计划风暴并且由此节省系统资源，公开了用于改变查询以使其在考虑所存储的数据的统计的情况下生成计划的方法和系统。所公开的用于改变查询的方法和系统使得能够以下述方式生成计划：使得计划不是搜索整个日期数据库，而是搜索所请求的受限位置。这样，在将查询呈现给(在查询与服务器之间的代理系统中的)SQL服务器之前，基于以统计方式确定的要存储的内容和实际请求的内容，生成所生成的计划。通过将参数更改为有用的内容(比如，在这种情况下，是受限位置)，该计划将被重用，这是因为可能会重复使用受限位置。

[0032] 定义：

[0033] 数据库是经过组织的数据集合。数据库是模式、表、查询、视图和其他对象的集合。数据库由数据库管理系统(DBMS)进行管理，数据库管理系统是与用户、其他应用程序和数据库本身进行交互以获得数据并分析数据的计算机软件应用程序。

[0034] 表是数据库中以结构化格式保存的相关数据的集合。表由字段(列)和行组成。在关系数据库和平面文件数据库中，表是数据元素(值)的集合，其采用垂直列(可通过列名称或字段名称标识)和水平行构成的模型，单元格是行和列相交的单位。表具有指定数量的列，但却可以具有任意数量的行。每一行由出现在特定列子集中的一个或多个值标识。对行进行唯一标识的列子集称为主关键字。在许多应用程序中，数据库保存着许多表，第一个表可以包含对第二个表中的行的引用。这种引用通常是第二个表的主关键字，于是被称为外部关键字。例如，发货地列中的单元格包含整数，作为对单独的城市名称的表的引用，而不是对城市名称本身的字符串的引用。使用单独的表和外部关键字使得能创建出符合范式的数据库，范式是确保数据库保持良好结构的规则集合。连接(join)是将主表与两个或更多个其他表组合在一起的操作，使得主表中的外部关键字可以是查询，就如同它们被外部关键字指代的其他表中的值替换一样。

[0035] 查询是对数据库的命令的执行。查询可以是执行从数据库检索数据的命令。从数据库检索数据也可以被称为从数据库中选择数据，比如，通过执行SELECT命令来实现。查询通常被表述为遵循查询语言语法的查询字符串或语句。可以在http://www.w3schools.com/sql/上找到示例性语法(SQL语法)的实用参考资料。示例性查询字符串是“SELECT*FROM Customers”，用于从Customers表中检索所有行。

[0036] 变量是在命令执行期间可能会取一个以上的值的对象或数据项。每个变量都具备变量名，变量名可以在命令中用来指代该变量，比如，声明、设置或读取变量。在SQL的示例中，变量被称为@variable_name，其中“variable_name”是该变量的名称。

[0037] 参数类似于变量，两个术语可以互换地使用。在某些上下文中，参数是指用于在查询与执行或生成查询的脚本之间传递数据的对象，而变量是指用于在单个查询中的不同语句之间传递数据的对象。通常情况下，最佳的做法是选择反映出对象是参数还是变量的名称，比如，通过向对象名称添加‘param’或‘var’后缀来实现。执行或生成查询并提供参数值的脚本可以是物流软件程序的一部分，例如，该物流软件程序生成用户界面，以便显示从数

据库选择的数据或者生成每周报告。

[0038] 过滤子句是查询的一部分，其定义了过滤器，由此使得数据库服务器仅返回满足过滤子句的那些行。在SQL的示例中，过滤子句以‘WHERE’开头，使得查询可以是“SELECT* FROM Customers WHERE Country＝′Mexico′”。在本示例中，‘Customers’是表名称，‘Country’是字段名称，而‘Mexico’是过滤子句的字段值。表达式“Country＝′Mexico′”被称为字段名称-值对。字段值可以是变量或参数，比如在“Country＝@countryparam”中就是如此。这种包含参数名称‘countryparam’的表达式也被称为字段名称-值对。通常，字段名称-值对还包括运算符(比如，′＝′)，以将字段名称链接到字段值。其他运算符(比如，‘＜＞’、‘＞’、‘＜’、‘＞＝、‘＜＝、‘BETWEEN’、‘LIKE’和‘IN’)同样是可能的。字面化(literalisation)意味着以参数值替换参数名称。例如，字面化将“@countryparam”替换为“Mexico”。在这个意义上，字面化还基于参数值确定第二参数名称。

[0039] 索引是一种提高对数据库表的数据检索操作速度的数据结构，其代价是用于维护索引数据结构的额外的写入和存储空间。索引用于快速定位数据，无需在每次访问数据库表时搜索数据库表中的每一行。具体而言，索引解决了对硬盘驱动器(HDD)上的实际数据的访问速度慢的问题。DBMS在HDD上最后一行之后存储新行，并且在没有索引的情况下，DBMS将需要从HDD检索所查询的表的所有行，这将导致大量的HDD访问以及关联的较长访问时间。这种用于查询的对全表的检索称为全表扫描，或者简称为扫描。在有索引的情形下，DBMS可以查询索引，索引将提供对匹配数据的引用，这被称为查找。索引占用HDD的空间通常相当得少，因此，检索完整索引以对其进行搜索比检索所有行要快得多。当使用聚簇索引时，DBMS更改行存储在HDD上的物理位置，以反映出索引结构。虽然这样做可以进一步提高性能，但每个表只能使用一个聚簇索引。当使用非聚簇索引时，行的位置保持不变，每个表可以使用多个非聚簇索引。一般情况下，查找比扫描更快，但是却需要额外的用于索引的存储空间和额外的处理器时间来维护索引。

[0040] 过滤后的索引是关于数据库中的行子集创建的索引。换句话说，在创建该索引时，提供了类似于上面WHERE子句的选择标准，使得只有满足WHERE子句的行才被包括在索引中。选择标准也被称为过滤谓词。

[0041] 查询计划(或查询执行计划或执行计划)是用于访问数据库中的行的有序步骤集。DBMS使用查询计划来执行查询。大多数查询可以通过许多不同的查询计划执行。例如，索引查找和全表扫描检索出相同的结果。在查询计划生成期间，DBMS估计哪些步骤将实现查询的最佳执行。例如，DBMS估计索引查找或全表扫描是否会实现对所需行的更快检索。由于查询计划的创建和优化从计算上来看是昂贵的，因此，DBMS可以将所创建的查询计划存储在计划缓存中。当要执行类似的查询时，DBMS对计划缓存进行检查并重用现有计划中的计划，这样通过避免重新创建查询计划来加快查询速度。当在查询中使用变量时，针对具有不同参数值的查询的计划可以是相同的，只要参数名称不变即可。例如，在接收查询“SELECT* FROM Customers WHERE Country＝@countryparam”时，DBMS创建执行计划。每次将要针对@countryparam参数的不同值执行相同的查询时，DBMS就会使用此执行计划。换句话说就是，对于作为@countryparam的值的“Mexico”和“Canada”的查询，DBMS使用相同的执行计划。

[0042] 直方图测量数据集中的每个不同值的出现频率。DBMS可以通过以统计方式对行进行采样或者通过对表中的所有行执行完全扫描来选择列值。如果直方图是根据行的采样集

创建的，那么，行数和不同值的数量的存储总数是估计值，并且可能不是整数。为了创建直方图，DBMS可以对列值进行排序，计算与每个不同列值相匹配的值的数量，然后将列值聚合进最多200个连续的直方图阶梯。每个步骤包括一系列列值，后跟上限列值。范围包括边界值(不包括边界值本身)之间的所有可能的列值。经过排序的列值中的最低值是第一直方图阶梯的上边界值。直方图被存储为直方图数据，直方图数据可以包括文本行，其中每行包括上限列值以及下述量的数值：该范围中行数、相等行的数量、不同范围行的数量和范围行的平均值。

[0043] 一种用于查询数据库的方法，包括：

[0044] 接收输入查询字符串，输入查询字符串包括具有字段名称和第一字段值的过滤子句，第一字段值指示第一参数名称；

[0045] 接收第一参数名称的参数值；

[0046] 确定基于参数值的并与第一参数名称不同的第二参数名称；

[0047] 基于输入查询字符串确定输出查询字符串，输出查询字符串包括具有字段名称和第二字段值的过滤子句，输出查询字符串的第二字段值基于第二参数名称；以及

[0048] 将输出查询字符串发送给数据库管理系统，以使数据库管理系统使用基于输出查询字符串中的第二参数名称的执行计划来执行数据库查询。

[0049] 由于数据库管理系统使用了基于输出查询字符串中的第二参数名称的执行计划，因此，执行计划的生成会受第二参数名称的影响。因此，确定第二参数名称可以在不更改DBMS本身的情况下优化DBMS中执行计划的生成和缓存。此举增加了对DBMS的操作的控制。

[0050] 确定输出查询字符串可以包括：用所确定的第二参数名称替换输入查询字符串中的第一参数名称。

[0051] 第二参数名称可以包括第一参数名称以及后缀或前缀，并且确定第二参数名称可以包括确定后缀或前缀。

[0052] 第二参数名称可以是第一参数名称的后缀或前缀，并且确定第二参数名称可以包括确定后缀或前缀。

[0053] 字段名称可以指代外部关键字，并且确定第二参数名称可以包括基于与外部关键字相关联的表中的行数来确定第二参数名称。

[0054] 该方法可以进一步包括：在确定了与外部关键字相关联的表中的行数低于阈值时，确定第二参数名称以使得第二参数名称对于与外部关键字相关联的表中的每一行是唯一的。

[0055] 第一参数名称的参数值可以包括字符串，并且确定第二参数名称以使得第二参数名称对于与外部关键字相关联的表中的每一行是唯一的可以包括：基于第一参数名称的参数值，将后缀附加到第一参数名称。

[0056] 后缀可以包括参数值的前两个或更多个字母。

[0057] 该方法可以进一步包括接收字段名称的直方图数据，并且确定第二参数名称可以包括基于直方图数据确定第二参数名称。

[0058] 基于直方图数据确定第二参数名称可以包括：确定第二参数名称以使得第二参数名称对于每个直方图阶梯是唯一的。

[0059] 基于直方图数据确定第二参数名称可以包括：基于关于直方图阶梯的多个预定义

条件确定第二参数名称，使得对于每个预定义条件而言，第二参数名称对于直方图阶梯中满足该条件的所有参数值都是相同的。

[0060] 确定第二参数名称可以包括：在确定了参数值处于具有低于预定阈值的频率值的直方图阶梯中时，使用预定义参数名称作为第二参数名称。

[0061] 为针对低频率的值的所有查询分配相同的参数名称是有利的，这是因为针对低频率的一个查询的执行计划可能也会适合于针对低频率的另一个查询。

[0062] 参数值可以与日期和/或时间有关，该方法还可以包括基于日期和/或时间确定时间段的长度，并且确定第二参数名称可以包括基于时间段的长度确定第二参数名称。

[0063] 确定第二参数名称可以包括基于关于时间段的长度的多个预定义条件来确定第二参数名称，使得对于每个预定义条件而言，第二参数名称对于时间段长度满足该条件的所有参数值而言是相同的。

[0064] 确定第二参数名称可以包括确定字段名称是否与过滤后索引有关，并且在确定了字段名称与过滤后索引有关时，使用参数值作为第二参数名称。

[0065] 使用参数值作为第二参数名称被称为字面化。

[0066] 一种用于查询数据库的计算机系统，包括：

[0067] 输入端口，

[0068] 用于接收输入查询字符串，输入查询字符串包括具有字段名称和第一字段值的过滤子句，第一字段值指示第一参数名称；并且

[0069] 用于接收第一参数名称的参数值；

[0070] 处理器，

[0071] 用于确定基于参数值的并与第一参数名称不同的第二参数名称；

[0072] 用于基于输入查询字符串确定输出查询字符串，输出查询字符串包括具有字段名称和第二字段值的过滤子句，输出查询字符串的第二字段值基于第二参数名称；以及

[0073] 输出端口，用于将输出查询字符串发送给数据库管理系统，以使数据库管理系统使用基于输出查询字符串中的第二参数名称的执行计划来执行数据库查询。

[0074] 一种数据库代理系统，包括：

[0075] 输入端，输入端连接到客户端计算机以从客户端计算机接收输入查询字符串，输入查询字符串包括具有字段名称和第一字段值的过滤子句，第一字段值指示第一参数名称，并且

[0076] 用于接收第一参数名称的参数值；

[0077] 处理器，

[0078] 用于确定基于参数值的并与第一参数名称不同的第二参数名称；

[0079] 用于基于输入查询字符串确定输出查询字符串，输出查询字符串包括具有字段名称和第二字段值的过滤子句，输出查询字符串的第二字段值基于第二参数名称；以及

[0080] 输出端口，输出端口连接到数据库管理系统，以将输出查询字符串发送给数据库管理系统，以使数据库管理系统使用基于输出查询字符串中的第二参数名称的执行计划来执行数据库查询。

[0081] 一种数据库管理系统，包括：

[0082] 输入端，输入端连接来接收输入查询字符串，输入查询字符串包括具有字段名称

和第一字段值的过滤子句,第一字段值指示第一参数名称,并且

[0083] 用于接收第一参数名称的参数值;

[0084] 处理器,

[0085] 用于确定基于参数值的并与第一参数名称不同的第二参数名称;

[0086] 用于基于输入查询字符串确定输出查询字符串,输出查询字符串包括具有字段名称和第二字段值的过滤子句,输出查询字符串的第二字段值基于第二参数名称;以及

[0087] 数据库引擎,用于使用基于输出查询字符串中的第二参数名称的执行计划来执行数据库查询。

[0088] 一种用于查询数据库的方法,包括:

[0089] 接收SELECT<fieldnames>FROM<table names>WHERE<fieldname><operator>@<firstparametername>形式的输入查询字符串;

[0090] 接收@<firstparametername>的参数值;

[0091] 确定基于参数值的并与<firstparametername>不同的<secondparametername>;

[0092] 确定SELECT<fieldnames>FROM<table names>WHERE<fieldname><operator>@<secondparametername>形式的输出查询字符串;以及

[0093] 将输出查询字符串发送给数据库管理系统,以使数据库管理系统使用基于输出查询字符串中的第二参数名称的执行计划来执行数据库查询,

[0094] 其中,<>括号内的术语由在实际实现中采用的术语进行替换,并且在执行该方法期间是不可变的。

[0095] 在适当的情况下,方法、计算机可读介质或计算机系统的任何方面的所述可选特征同样适用于在此也进行描述的其他方面。

附图说明

[0096] 图1示出了用于管理物流数据的采用笔和纸的方法。

[0097] 图2示出了计算机化数据管理系统。

[0098] 图3示出了在图2的计算机化数据管理系统中使用的硬盘。

[0099] 图4示出了有序表,其也被称为聚簇索引。

[0100] 图5a示出了聚簇索引的另一示例。

[0101] 图5b示出了非聚簇索引。

[0102] 图6a和图6b分别示出了第一表和第二表的示例。

[0103] 将参考以下内容描述示例:

[0104] 图7示出了用于查询数据库的计算机系统。

[0105] 图8示出了用于查询数据库的方法。

[0106] 图9示出了图8中的方法的示例性数据。

[0107] 图10示出了用于确定基于参数值的第二参数名称的流程图。

[0108] 图11示出了数据库表的直方图。

[0109] 图12示出了数据库管理系统。

具体实施方式

[0110] 如上所述，所公开的用于改变查询的方法和系统使得能够以下述方式生成计划：使得计划搜索所请求的受限时间段，而不是搜索整个日期数据库。这样，在将查询呈现给(在查询与服务器之间的代理系统中的)SQL服务器之前，基于以统计方式确定的要存储的内容和实际请求的内容，生成所生成的计划。通过将参数更改为有用的内容(比如，受限位置或受限日期范围)，该计划将被重用，这是因为可能会重复使用受限位置或受限日期范围。

[0111] 所公开的方法和系统由计算机系统执行。图7示出了用于查询数据库的计算机系统700。计算机系统700包括处理器701，处理器701连接到程序存储器702、数据存储器703、输入端口704和输出端口705。输入端口704通信地耦合到(比如，通过互联网)由用户707操作的客户端计算机706。例如，用户707操作物流软件程序，客户端计算机706生成包括数据字段的用户界面，这些数据字段将由用户707所请求的物流数据进行填充。客户端计算机706可以生成作为可点击列表的国家列表，并将该列表呈现给用户707。然后，客户端计算机706接收标识“Mexico”的用户输入，例如，作为用户希望看到客户数据的国家。响应于接收到该用户输入，客户端计算机706选择参数化数据库查询字符串“SELECT*FROM CustomersWHERECountry＝@countryparam”，并将该查询字符串与“@countryparam”参数的参数值“Mexico”一起发送给计算机系统100。

[0112] 输出端口705通信地耦合到(比如，通过互联网)数据库服务器708(比如，Microsoft SQL服务器)，数据库服务器708又连接到数据库存储区709。处理器701修改查询字符串(将在下文描述)并将经过修改的查询字符串发送给数据库服务器708。数据库服务器708根据从计算机系统100接收的查询字符串执行查询。执行查询包括选择或生成查询的执行计划。

[0113] 程序存储器702是非暂时性计算机可读介质，比如，硬盘驱动器、固态盘硬或CD-ROM。软件(即，存储在程序存储器702上的可执行程序)使得处理器701执行图8中的方法，即，处理器701接收带有参数名称的输入查询字符串以及参数值，将参数名称替换为基于参数值的字符串，并将所得到的输出查询字符串发送给数据库服务器。处理器701可以将输出和/或输入查询字符串存储在数据存储器703上，比如存储在RAM或处理器寄存器上。

[0114] 处理器701可以从数据存储器703以及从输入端口704接收数据，比如，查询字符串。在一个示例中，处理器701经由输入端口704从客户端计算机706接收查询字符串，比如，通过使用根据IEEE 802.11的Wi-Fi网络来接收。Wi-Fi网络可以是分散式自组织网络，这样便不需要专用管理基础设施(如路由器)，或者Wi-Fi网络可以是具有管理网络的路由器或接入点的集中式网络。

[0115] 尽管输入端口704和输出端口705被示为不同的实体，但是应该理解，可以使用任何类型的数据端口来接收数据，比如，网络连接、存储器接口、处理器701的芯片封装引脚或逻辑端口(比如，存储在程序存储器702上并由处理器701执行的IP套接字或功能参数)。这些参数可以存储在数据存储器703上，并且可以在源代码中按值或按引用(即作为指针)进行处理。

[0116] 处理器701可以通过所有这些接口接收数据，其包括易失性存储器(比如，缓存或RAM)或非易失性存储器(比如，光盘驱动器、硬盘驱动器、存储服务器或云存储装置)的存储

器访问。计算机系统700还可以在云计算环境内实现，比如，托管动态数量的虚拟机的互连服务器的管理组。

[0117] 在一个示例中，计算机系统700作为代理服务器操作。这意味着如果客户端计算机706直接连接到数据库服务器708，则到客户端计算机706的接口与数据库服务器将提供的接口相同。换句话说，客户端计算机706不知道并且没有确定代理服务器700实际上不是数据库服务器708。与直接连接的主要区别在于代理服务器700通过修改查询字符串来实现的性能提高，这种修改是通过基于那些参数的值来替换参数名称而实现的。这还意味着输入端口704和输出端口705可以是双向全双工输入/输出通信端口。也就是说，输入端口704和输出端口705还将查询数据库所产生的数据库行传回客户端计算机706，比如，使用SQL协议来回传。例如，代理服务器700可以处理数据并将各个行转换为JSON或XML格式。此外，输入端口704和输出端口705二者可以由单个物理LAN接口表示，从而向客户端计算机706和数据库服务器708发送数据以及从客户端计算机706和数据库服务器708接收数据。

[0118] 应当理解，在任何接收步骤之前，处理器701可以确定或计算稍后接收的数据。例如，处理器701确定查询字符串并将该查询字符串存储在数据存储器703中，比如，存储在RAM或处理器寄存器中。然后，处理器701从数据存储器703请求数据，比如，通过提供读取信号以及存储器地址来请求。数据存储器703提供作为物理位线上的电压信号的数据，并且处理器701经由存储器接口接收查询字符串。例如，客户端计算机706发送查询字符串的标识符，而不是查询字符串本身，处理器701使用该标识符作为关键字从数据存储器703中检索查询字符串。应当理解的是，在整个本公开中除非另有说明，否则字符串、节点、边缘、图表、解、变量、参数、执行计划等指的是物理存储在数据存储器703上或数据库管理系统上的或由处理器701处理的数据结构。

[0119] 图8示出了由处理器701执行的用于查询数据库的方法800。图9示出了方法800的示例性数据900，其中附图标记彼此相关，这样附图标记801与901相关，802与902有关，依次类推。图8应被理解为软件程序的蓝图，并且可以逐步地实现，这样图8中的每个步骤由编程语言(如C++或Java)中的函数表示。随后，将得到的源代码编译并存储为程序存储器702上的计算机可执行指令。

[0120] 方法800开始于处理器701接收801输入查询串901。输入查询串901包括具有字段名称907、运算符910和第一字段值908的过滤子句906，第一字段值908表示第一参数名称909。在本示例中，第一字段值908是以“@”符号为前缀的第一参数名称909。

[0121] 处理器701还接收802第一参数名称的参数值902，这可以在输入查询串901的接收步骤801之前、之后或同时进行。处理器701然后确定803基于参数值902的并与第一参数名称909不同的第二参数名称903。在本示例中，处理器701使用参数值(“Mexico”)(在本实例中，它是字符串)的前三个字母('mex')，并使用那些字母作为第一参数名称909的后缀。虽然这里的示例涉及后缀，但应注意，同样可以使用前缀以及基于参数值的对参数名称的任何其他修改。处理器701确定第二参数名称的方式可以取决于由字段名称907标识的字段的数据类型，以及其他因素，这将在下面更加详细地说明。

[0122] 接下来，处理器701基于输入查询字符串901确定804输出查询字符串904，输出查询字符串901包括具有字段名称907和第二字段值912的过滤子句911。输出查询字符串904的第二字段值912基于第二参数名称903。在本实例中，处理器701用“@”符号作为第二参数

名称903的前缀，并将结果而不是第一字段值908放在运算符910后面。换句话说，处理器701用基于参数值的第二参数名称903替换了第一参数名称909。

[0123] 最后，处理器701将输出查询字符串905发送805给数据库管理系统。这使得数据库管理系统使用基于输出查询字符串904中的第二参数名称903的执行计划来执行数据库查询。

[0124] 图10示出了流程图1000，流程图1000更详细地示出了确定803基于参数值的第二参数名称的步骤。流程图1000可以作为存储在程序存储器702上的程序代码中的多个嵌套的if-then-else块。需要注意的是，流程图1000中的判定按特定顺序排列，应该理解的是，判定的其他顺序同样可以适用。

[0125] 处理器701首先确定1001参数是否指代过滤后索引。对于此判定，处理器701可以从数据库服务器708请求包括过滤后索引的列表的元数据。例如，处理器701访问SQL表对象“sys.indexes”，其中“has_filter”列具有值“1”。如果列名称位于列表上(即，参数指代过滤谓词(filter predicate))，则处理器701将参数字面化1002，这意味着处理器701用参数值替换参数名称。这使得数据库服务器708为参数的每个不同值生成新的执行计划，这样做使得计划的数量明显增多，但是在过滤谓词的不同值的数量以及因此的执行计划的数量有限的情况下，这样做可能是切合实际的。

[0126] 如果参数不是指过滤谓词，则处理器701确定1003参数是否指代外部关键字(foreign key)。处理器701可以首先识别查询字符串中的字段名称。在图6a和图6b的示例以及查询“SELECT*FROM Shipments WHERE Origin＝@origin_param”中，处理器701将运算符之前(也就是‘＝’之前)的单词“Origin”标识为字段名称。然后，处理器701通过运行sys.foreign_keys命令来重试DBMS元数据。之后，处理器701可以将字段名称与元数据中的外部关键字进行比较，并且如果存在着匹配，则该参数指代外部关键字。在本实例中，元数据表明，图6a中的第一表600中的“Origin”字段是图6b中的第二表610的外部关键字。

[0127] 如果处理器701确定了参数名称指代外部关键字，则处理器701确定1004外部关键字所指代的表的行数。在以上示例中，处理器701确定第二表610中的行数。处理器701可以执行包括对数据库的COUNT语句的查询，或者从数据库服务器708请求包含每个表中的行数的元数据。在行数不发生明显改变的应用中，处理器701可以将该值存储在数据存储器703上，以供进一步与其他查询一起使用。

[0128] 然后，处理器701确定与外部关键字相关联的行数(即，父表中的行数)是否超过1004阈值，比如，大于2000行或大于5000行。如果行数超过了阈值，则处理器701不更改参数名称1005。该判定的合理性在于：对于大量行而言，更改参数可能会产生大量执行计划并引发计划膨胀。相反，数据库服务器708使用单个计划来查询可能具有数百万行的表中的任何一行。

[0129] 如果外部关键字的表中的行数低于阈值(步骤1004)，或者如果参数根本不是指代外部关键字(步骤1003)，则处理器701更改1006参数名称。在一个示例中，处理器701对外部关键字参数进行字面化，这意味着新的参数名称对于外部关键字表中的每一行来说是唯一的。例如，图6b中的表只有13个没有重复城市名称的条目，这意味着对参数进行字面化会带来对每一行而言唯一的名称，并且只产生13个执行计划。处理器701可以用参数值作为参数名称的后缀，比如，从cityname_para”到cityname_param_london”，或者可以仅使用前三个

字母来确定“cityname_param_lon”。

[0130] 为了决定如何更改参数名称,处理器701确定数据类型1007。处理器701可以再次请求包括数据库的列/字段的数据类型的元数据。例如,处理器701可以执行命令“SELECT TABLE_NAME,COLUMN_NAME,DATA_TYPEFROM Database.INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME＝′Shipments′AND COLUMN_NAME＝cityname′。如果数据类型是STRING(字符串),则处理器701确定1008参数值是否具有小于阈值数量的字符,比如,小于5个字符或小于3个字符。如果参数名称确实具有小于阈值数量的字符,则处理器701将参数值作为参数名称的后缀1009。通过这样做,处理器701可以检查不在参数名称中使用的任何非法字符,比如,撇号或斜杠。如果处理器701检测到非法字符,则处理器701计算参数值的哈希值并使用该哈希值作为后缀,或者甚至可以使用哈希值作为参数名称。

[0131] 返回步骤1008,如果参数值具有不小于阈值数量的字符,则处理器701截断1010参数值,以获得具有参数的阈值数量的长度的后缀。在另一示例中,处理器701基于直方图确定新参数名称,处理器701可以通过对在过滤子句中提供的字段名称运行“DBCC SHOW_STATISTICS”(“Shipments.cityname”,“AK_cityname_rowguid)WITH HISTOGRAM”命令而从数据库服务器708获得该直方图。换言之,直方图提供了指定列中的值的频率。直方图数据可以包括200个元素的列表,例如下述格式的列表

[0132] RANGE_HI_KEY|RANGE_ROWS|EQ_ROWS|DISTINCT_RANGE_ROWS|AVG_RANGE_ROWS

[0133] 并且,两个示例性条目是:

[0134] USBOS 99 72 40 2.475

[0135] USBUF 49 19 22 2.227273

[0136] 以上示例涉及货物表中的机场列,其中前面的字母是国家代码,后面的字母是机场代码。这就意味着,第二条线涉及爱德华·劳伦斯·洛根将军国际机场(波士顿)与布法罗尼亚加拉国际机场(前缀为“美国”)之间按字母顺序排列的所有机场。

[0137] 图11以图形方式示出了使用RANGE ROWS值作为y轴1101的直方图。处理器701现在确定参数名称,使得在直方图中具有相似值的参数值也具有相同的参数名称。这就意味着,将使用相同的执行计划来执行对相似直方图值的参数值的查询。更具体地,处理器701可以在直方图中的y轴上确定所接收参数值的范围。在图11的示例中,五个范围1102到1106被定义成关于直方图阶梯的多个预定义条件。当处理器701接收参数值(比如,伯灵顿国际机场的“USBTV”)时,处理器701确定该范围与直方图1100中的y值49相关。因此,该参数值落在范围1104内。换句话说,处理器701确定参数值满足处于范围1104内这一预定义条件。每个范围1102到1106可以被分配给一个预定义参数名称,比如,针对范围1102的“airport_param_1”、针对范围1103的“airport_param_2”、针对范围1104的“airport_param_3”、针对范围1105的“airport_param_4”和针对范围1106的“airport_param_5”。所以,在本示例中,新的参数名称是“airport_param_3”。在另一示例中,在三个范围的示例中参数名称的后缀是“_large”、“_medium”和“_small”。目的在于:对于每个预定义条件(即对于每个范围),参数名称对于直方图阶梯中满足此条件(即,位于范围内)的所有参数值而言是相同的。

[0138] 这种基于直方图的方法在可能的参数值的数量大于数据库服务器708所提供的直方图阶梯的数量(对于Microsoft SQL服务器,该数量当前为200)的情况下可能是特别有用的。因此,每个直方图阶梯都有一个唯一的参数名称。对于从字母顺序上来看接近的参数值

也共享类似频率的情形，这种方法可能也是有用的。例如，美国的大多数机场可能具有类似的特征，因此，当机场参数名称以国家代码开头时，同一国家内的机场可能位于相同的直方图括号中。此外，在使用具有多个参数名称的多个字段的情况下，直方图方法可以减少执行计划的数量。如果三个字段各自具有100个不同的值(比如，100个不同的截断式三字母值)，那么，从组合上来看，就有100^3＝1000000个不同的组合，也就是创建和存储1000000个不同的执行计划。通过仅使用五个直方图范围，此数量减少到了5^3＝125个执行计划。

[0139] 返回到确定参数值的数据类型的步骤1007，如果数据类型是DATE(YYYY-MM-DD)，则处理器701通过参数值与当前日期之间的比较来确定日期范围1011。例如，许多用户查询在上周发生的物流数据。过滤子句可以是“WHERE order_date BETWEEN CURDATE()AND@date_param”，其中CURDATE()的示例值为“2000-01-01”，参数值为“2000-01-08”。处理器701分析该过滤子句并提取出七天的时间间隔。因此，处理器701为参数名称‘date_param’加入后缀‘7’或‘七’或‘一周’。实际上，只要处理器701保留了后缀与时间段之间关系的记录，那么，后缀可以是任意的。这样，处理器701将为查询上周或任何其他的一周的时间段的所有查询确定相同的参数名称。因此，数据库服务器708为所有这些查询创建一次执行计划，并将该执行计划重复用于一周时间段的所有未来查询。类似地，处理器701可以确定针对查询上个月的所有查询的参数名称。不同参数名称的数量可包括但不限于：

[0140] date_param_1week:1周时间段

[0141] date_param_2week:2周时间段

[0142] date_param_3week:3周时间段

[0143] date_param_1month:1个月时间段

[0144] date_param_2month:2个月时间段

[0145] date_param_3month:3个月时间段(季度)

[0146] date_param_6month:6个月时间段(半年)

[0147] date_param_1year:1年时间段

[0148] 对于不属于这些日期范围的所有其他查询，处理器701可以保持参数名称不变。因而，数据库服务器708借助于为原始参数名称“date_param”创建的相同执行计划来执行这些不太常见的查询。还可以使用与参数名称和时间段有关的其他策略。例如，时间段可以用小时、分钟或秒而不是天数来定义，以便执行对时间要求更严格的操作。

[0149] 再次返回到数据类型选择1007，当处理器701确定数据类型是布尔值时，处理器701可以针对每个值进行参数化1012。换句话说，处理器701确定参数名称的两种不同可能性中的一种。例如，字段名称可能是“已付款”，而布尔值“真”表示货物已全额付清。当处理器701接收到具有“WHERE paid＝@paid_param”过滤子句的查询并且参数值为“真”时，处理器701将新参数名称确定为“paid_param_true”。反之亦然，在参数值为假的情况下，处理器701确定参数名称“paid_param_false”。如上所述，两个参数名称可以是任意的，只要它们不同且处理器701记录了该关系即可。例如，针对真值的参数名称可以是“51vq7Zm4h5”，而针对假值的参数名称可以是“KKG9Y4bUQ4”(这些是完全随机的字符串)。关键点在于：根据相同的逻辑来映射后续的查询。也就是说，处理器701不是每次都使参数名称随机化，而是仅进行一次，然后再将该值重用于后续查询。这也适用于包括字符串和数据范围的其他示例：处理器701为参数值满足相同测试的输入查询确定相同的参数名称。虽然对于数据库服

务器708而言参数的实际名称基本上是无关的，但是可以通过将参数值或参数值的一部分用作后缀来促进调试和监测。

[0150] 这样，数据库服务器708根据paid_param参数的值创建并使用两个不同的执行计划，这会使性能得到提高。当付费货物的数量与未付货物的数量之间存在较大差异时，这种性能尤为突出。也就是说，对于严重偏斜的分布，优势效果才最大。

[0151] 图12示出了数据库管理系统1200。数据库管理系统1200包括输入端1201、处理器1202和数据库引擎1203。输入端1201被连接来从客户端计算机706接收输入查询字符串。输入查询字符串包括带有字段名称和第一字段值的过滤子句。第一字段值指示第一参数名称。输入端1201还接收第一参数名称的参数值。

[0152] 处理器1202连接到程序存储器1204和数据存储器1205，并执行存储在程序存储器1204上的程序代码。在这种意义上，处理器1202确定基于参数值的并与第一参数名称不同的第二参数名称。处理器1202还基于输入查询字符串确定输出查询字符串。输出查询字符串包括带有字段名称和第二字段值的过滤子句。输出查询字符串的第二字段值基于第二参数名称。

[0153] 数据库引擎1203可以是在处理器1202上运行的单独进程，或者可以由不同的处理器核心、不同的处理器芯片、相同计算机硬件所托管的不同虚拟机或者不同的专用计算机系统来执行。数据库引擎1203使用基于输出查询字符串中的第二参数名称的执行计划在数据存储区1206上执行数据库查询。处理器1202可以遵循与参考图8、图9、图10和图11描述的相同的步骤。如上所述，由于数据库引擎使用基于第二参数名称的执行计划，因此，执行计划的使用可以通过基于参数值动态地确定第二参数名称来加以控制。这样做开启了基于参数值优化计划使用的可能性，这对于现有系统来说是不可能实现的。

[0154] 本领域技术人员将理解的是，在不脱离权利要求限定的范围的前提下，可以对具体实施例作出许多变化和/或修改。

[0155] 应当理解，可以使用各种技术来实现本公开的技术。例如，本文描述的方法可以通过驻留在合适的计算机可读介质上的一系列计算机可执行指令来实现。合适的计算机可读介质可以包括易失性(例如，RAM)和/或非易失性(例如，ROM、磁盘)存储器、载波和传输介质。示例性载波可以采用电气信号、电磁信号或光学信号的形式，这些信号沿着本地网络或诸如互联网之类的公共可访问网络传送数字数据流。

[0156] 还应该理解的是，除非从以下讨论中明确地说明，否则应当理解的是，在整个说明书中，利用诸如“估计”或“处理”或“估算”或“计算”或“优化”或“确定”或“显示”或“最大化”等术语的论述是指计算机系统或类似电子计算设备的动作和过程，所述计算机系统或类似电子计算设备对表示为计算机系统的寄存器和存储器内的物理(电子)量的数据进行处理并将其转换成类似地表示为计算机系统存储器或寄存器或其他此类信息存储、传输或显示设备内的物理量的其他数据。

[0157] 因此，本发明的实施例在所有方面都被认为是说明性的，而非限制性的。

[0158] 本领域技术人员将理解的是，在不脱离本公开的广泛一般性范围的前提下，可以对上述实施例作出许多变化和/或修改。因此，本发明的实施例在所有方面都被认为是说明性的，而非限制性的。

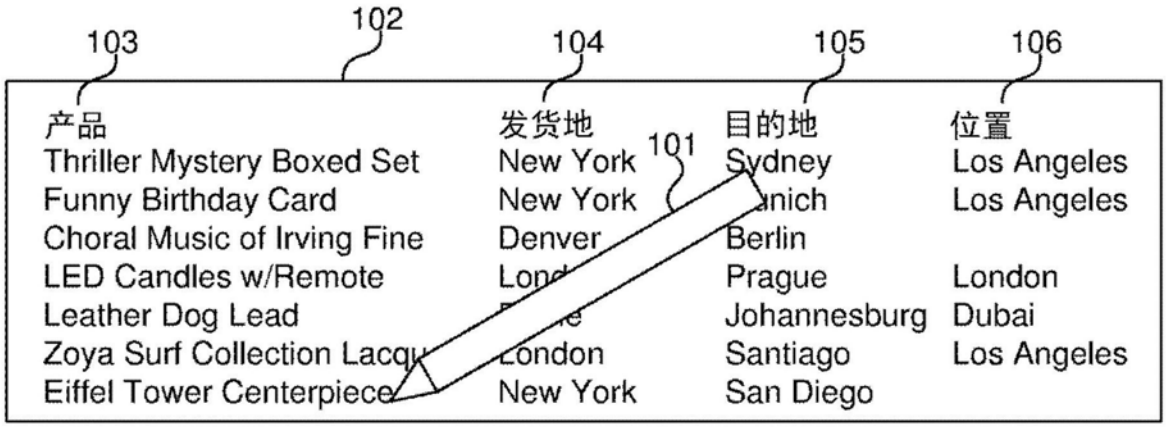

103
102
104
105
106
101
产品
发货地
目的地
位置
Thriller Mystery Boxed Set
New York
Sydney
Los Angeles
Funny Birthday Card
New York
Los Angeles
Choral Music of Irving Fine
Denver
Berlin
LED Candles w/Remote
Prague
London
Leather Dog Lead
Johannesburg
Dubai
Zoya Surf Collection Lacqu
London
Santiago
Los Angeles
Eiffel Tower Centerpiece
New York
San Diego

图1

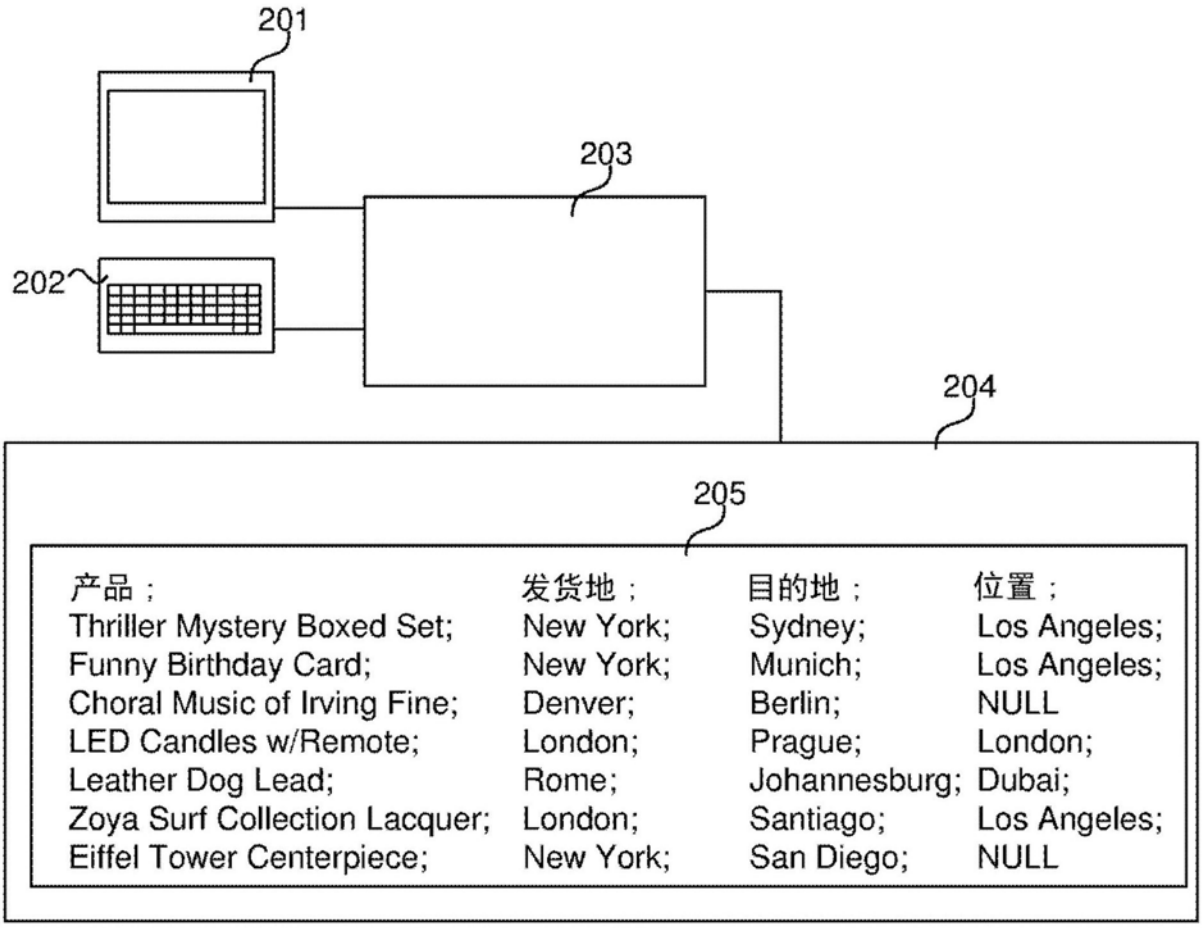

201
203
202
204
205
产品；
发货地；
目的地；
位置；
Thriller Mystery Boxed Set; New York; Sydney; Los Angeles;
Funny Birthday Card; New York; Munich; Los Angeles;
Choral Music of Irving Fine; Denver; Berlin; NULL
LED Candles w/Remote; London; Prague; London;
Leather Dog Lead; Rome; Johannesburg; Dubai;
Zoya Surf Collection Lacquer; London; Santiago; Los Angeles;
Eiffel Tower Centerpiece; New York; San Diego; NULL

图2

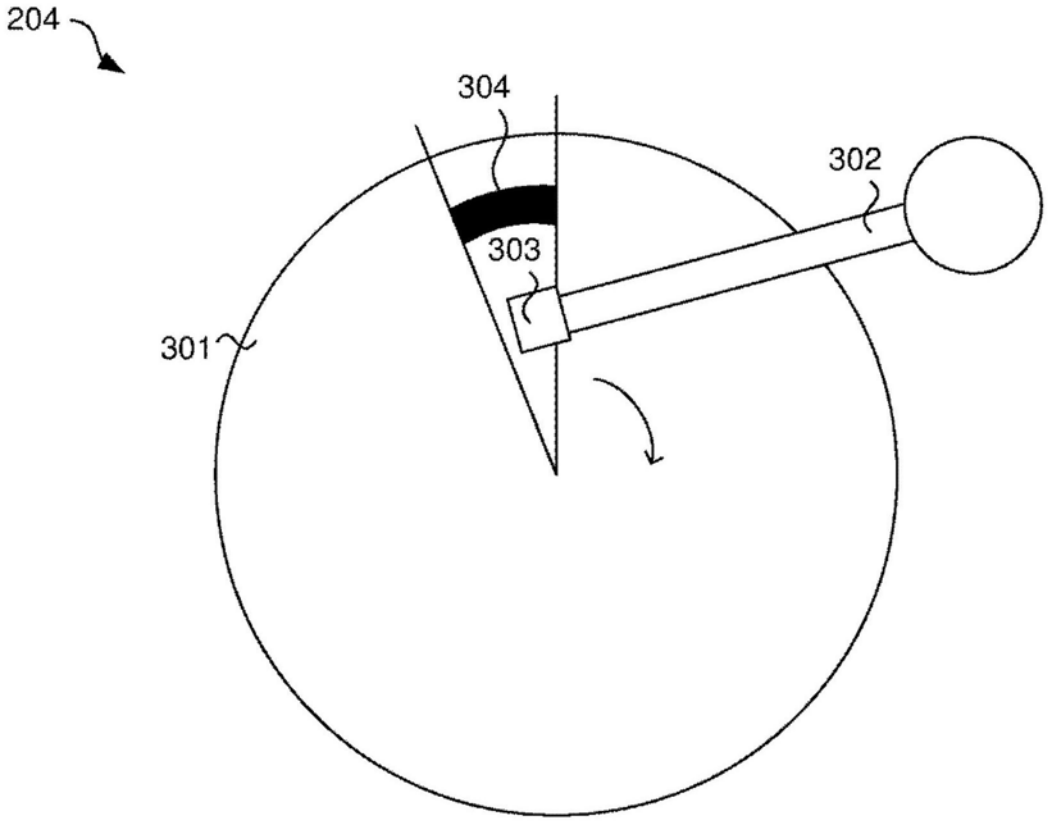
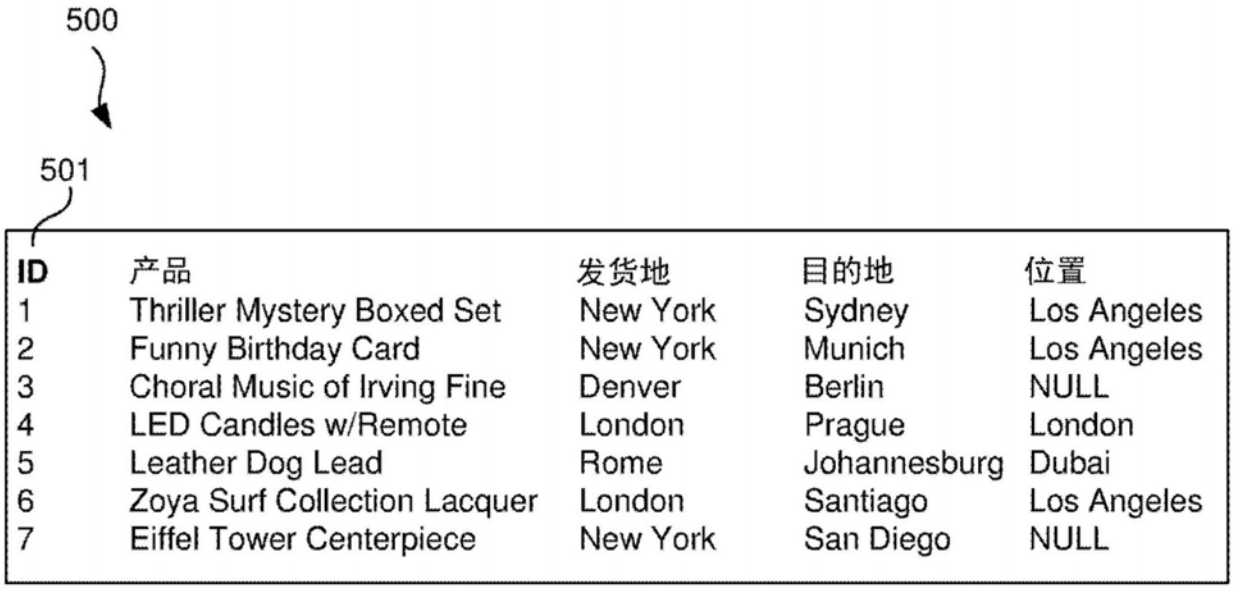


图3

400

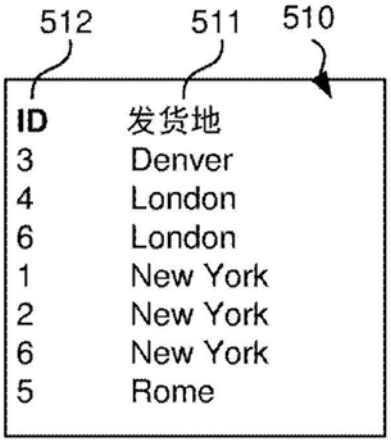
产品	发货地	目的地	位置
Choral Music of Irving Fine	Denver	Berlin	NULL
Eiffel Tower Centerpiece	New York	San Diego	NULL
Funny Birthday Card	New York	Munich	Los Angeles
LED Candles w/Remote	London	Prague	London
Leather Dog Lead	Rome	Johannesburg	Dubai
Thriller Mystery Boxed Set	New York	Sydney	Los Angeles
Zoya Surf Collection Lacquer	London	Santiago	Los Angeles

图4



ID	产品	发货地	目的地	位置
1	Thriller Mystery Boxed Set	New York	Sydney	Los Angeles
2	Funny Birthday Card	New York	Munich	Los Angeles
3	Choral Music of Irving Fine	Denver	Berlin	NULL
4	LED Candles w/Remote	London	Prague	London
5	Leather Dog Lead	Rome	Johannesburg	Dubai
6	Zoya Surf Collection Lacquer	London	Santiago	Los Angeles
7	Eiffel Tower Centerpiece	New York	San Diego	NULL

图5a



ID	发货地
3	Denver
4	London
6	London
1	New York
2	New York
6	New York
5	Rome

图5b

600

ID	产品	发货地	目的地	位置
1	Thriller Mystery Boxed Set	8	13	6
2	Funny Birthday Card	8	7	6
3	Choral Music of Irving Fine	2	1	NULL
4	LED Candles w/Remote	5	9	5
5	Leather Dog Lead	10	4	3
6	Zoya Surf Collection Lacquer	5	11	6
7	Eiffel Tower Centerpiece	8	12	NULL

图6a

610

ID	城市名称
1	Berlin
2	Denver
3	Dubai
4	Johannesburg
5	London
6	Los Angeles
7	Munich
8	New York
9	Prague
10	Rome
11	Santiago
12	San Diego
13	Sydney

图6b

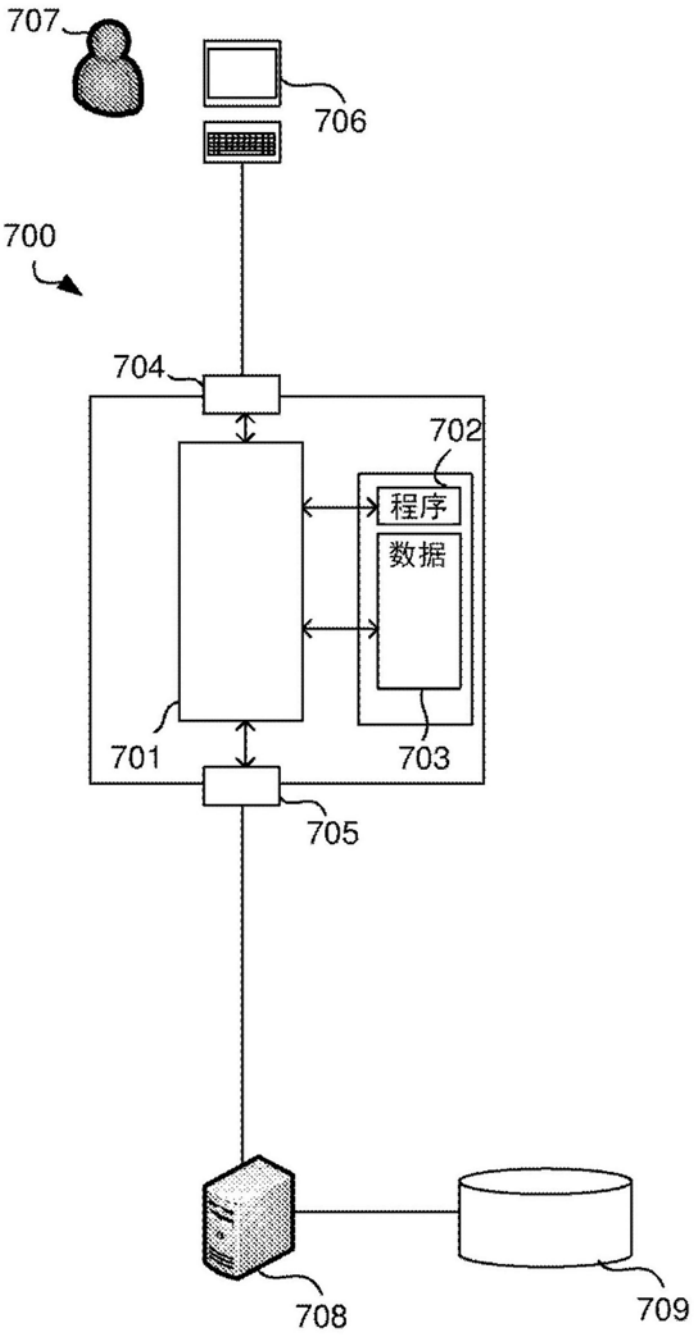


图7

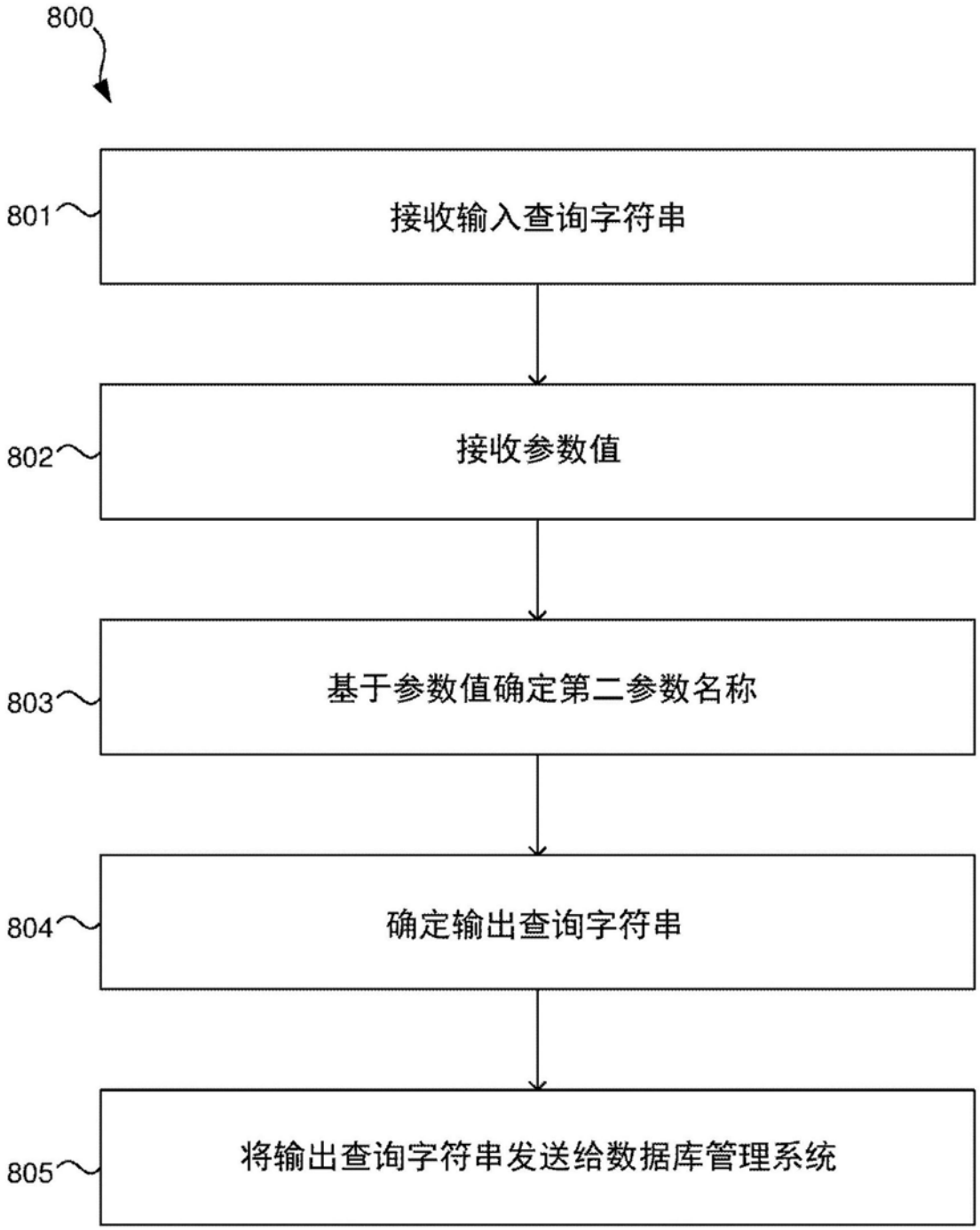

800
801
接收输入查询字符串
802
接收参数值
803
基于参数值确定第二参数名称
804
确定输出查询字符串
805
将输出查询字符串发送给数据库管理系统

图8

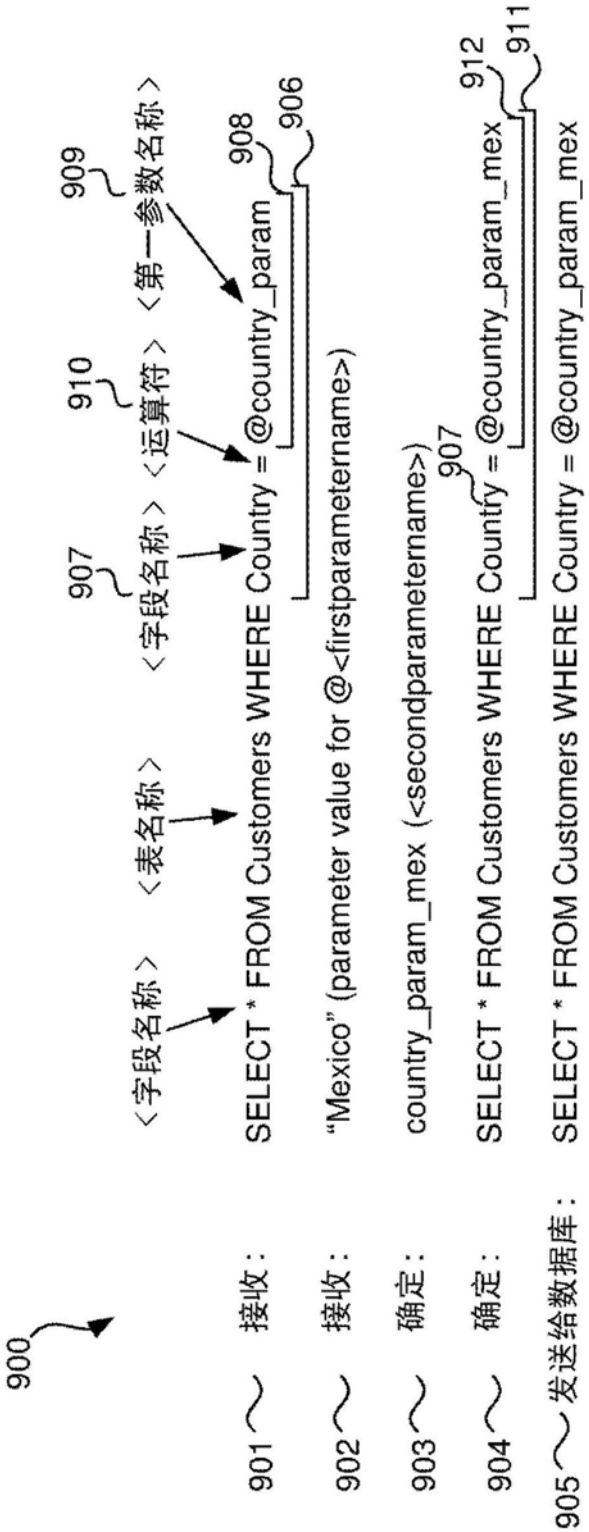

900
901 接收：〈字段名称〉〈表名称〉〈字段名称〉〈运算符〉〈第一参数名称〉
SELECT * FROM Customers WHERE Country = @country_param
907
910
909
908
906
902 接收："Mexico" (parameter value for @<firstparametername>)
903 确定：country_param_mex (<secondparametername>)
907
904 确定：SELECT * FROM Customers WHERE Country = @country_param_mex
912
911
905 发送给数据库：SELECT * FROM Customers WHERE Country = @country_param_mex

图9

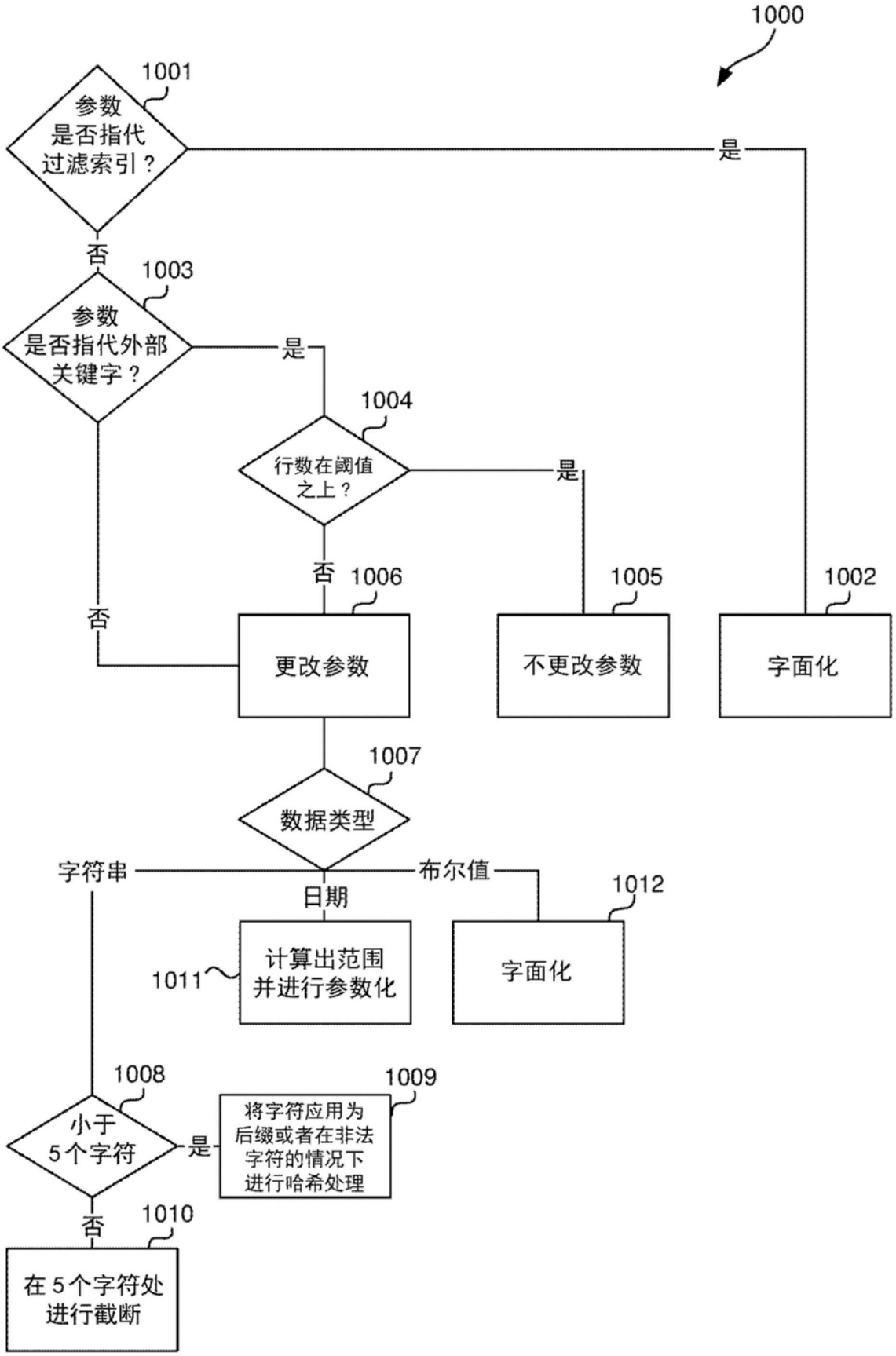

1000
1001
参数
是否指代
过滤索引？
是
否
1003
参数
是否指代外部
关键字？
是
1004
行数在阈值
之上？
是
否
1006
1005
1002
否
更改参数
不更改参数
字面化
1007
数据类型
字符串
布尔值
日期
1012
1011
计算出范围
并进行参数化
字面化
1008
1009
小于
5个字符
是
将字符应用为
后缀或者在非法
字符的情况下
进行哈希处理
否
1010
在5个字符处
进行截断

图10

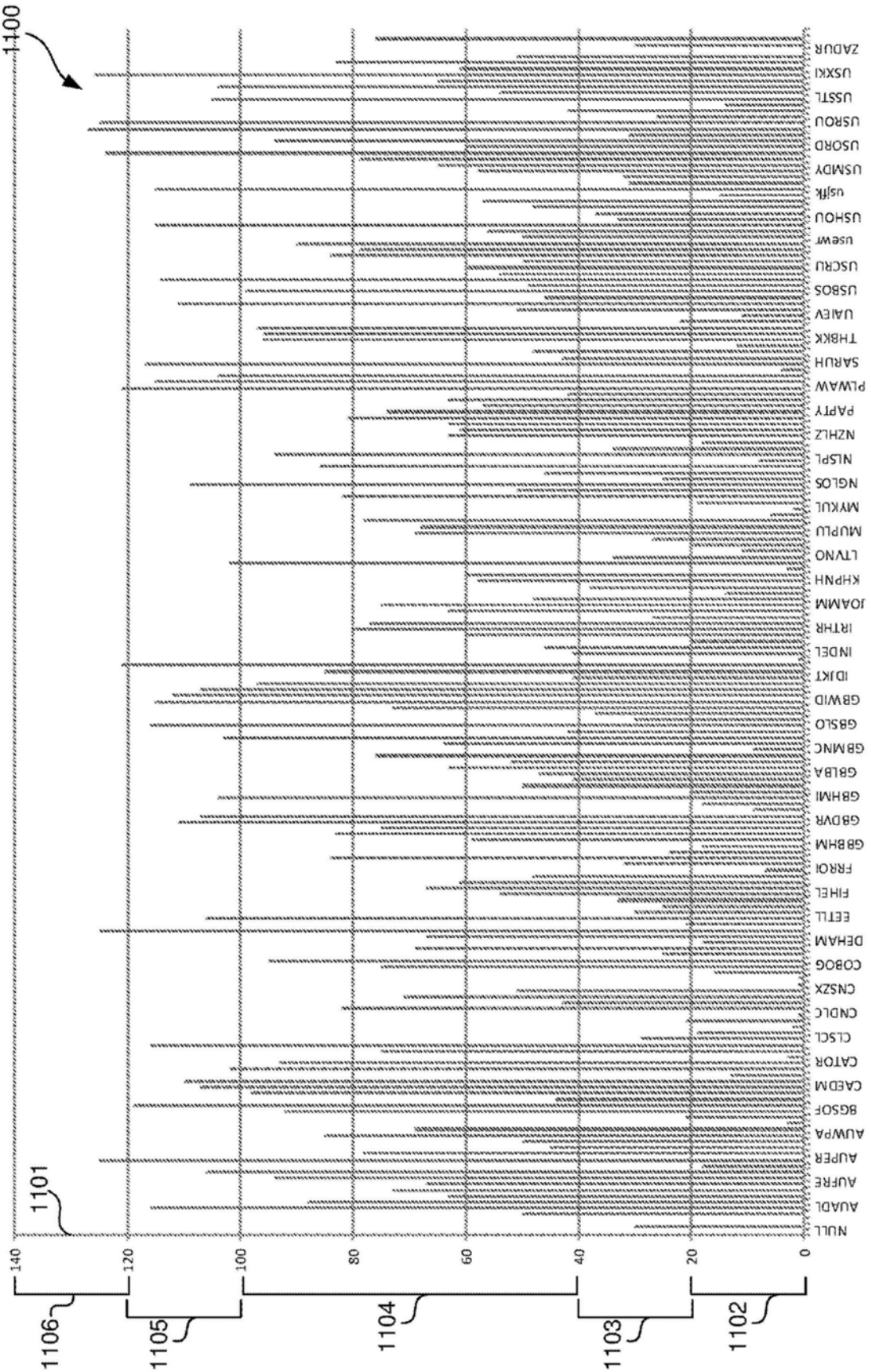

1100
1101
1102
1103
1104
1105
1106
0
20
40
60
80
100
120
140
NULL
AUADL
AUFRE
AUPER
AUWPA
BGSOF
CAEDM
CATOR
CLSCL
CNDLC
CNSZX
COBOG
DEHAM
EETLL
FIHEL
FRROI
GBBHM
GBDVR
GBHMI
GBLBA
GBMNC
GBSLO
GBWID
IDJKT
INDEL
IRTHR
JOAMM
KHPNH
LTVNO
MUPLU
MYKUL
NGLOS
NLSPL
NZHLZ
PAPTY
PLWAW
SARUH
THBKK
UAIEV
USBOS
USCRU
usewr
USHOU
usjfk
USMDY
USORD
USROU
USSTL
USXKI
ZADUR

图11

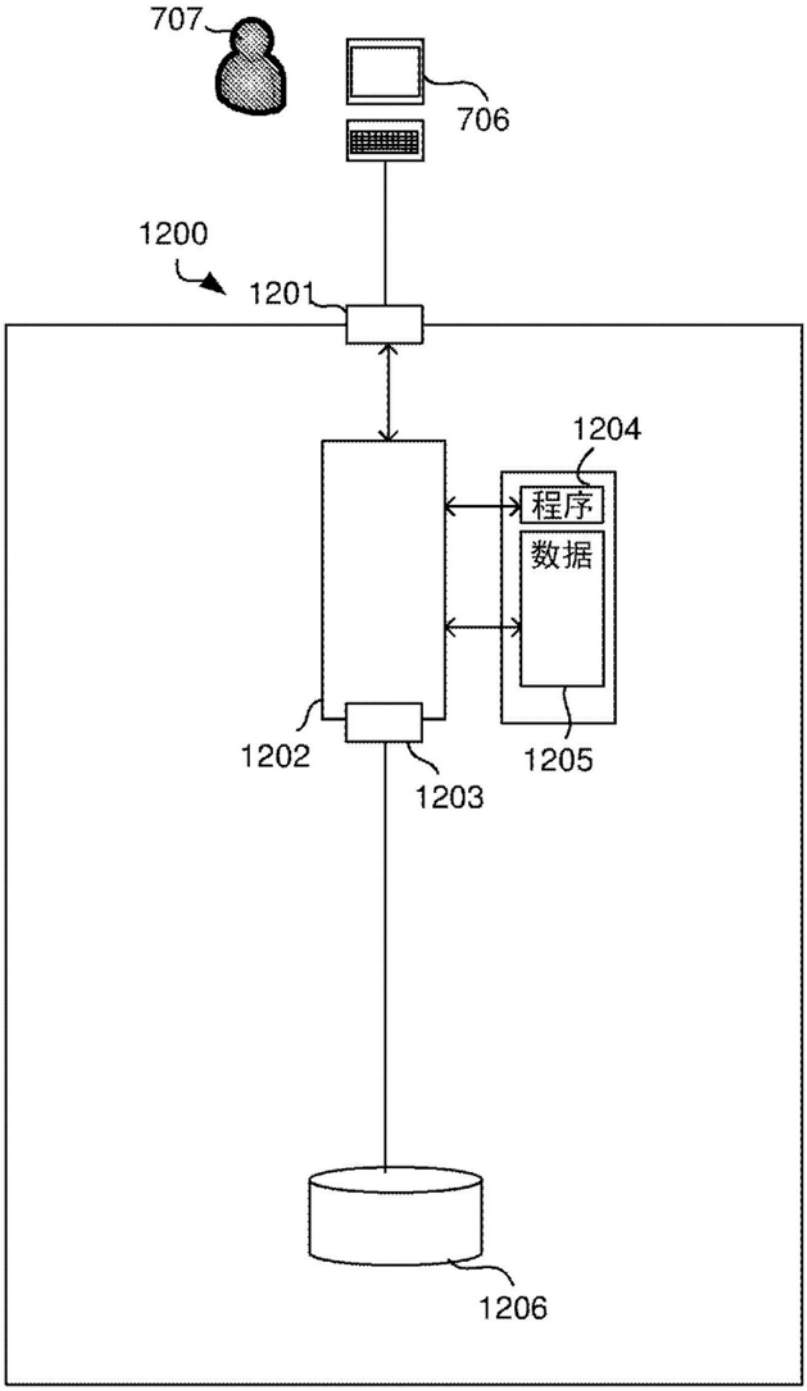


图12